SYBASE®

Performances et optimisation
Volume 1 : Concepts de base

Adaptive Server Enterprise

12.5

Réf. du document : 37924-01-1250-01

Dernière mise à jour : juillet 2001

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608, Etats-Unis d'Amérique.

Table des matières

Préface ..... **xv**

CHAPITRE 1 **Présentation** **1**
- Bonnes performances 1
  - Temps de réponse 1
  - Débit 2
  - Conception orientée performances 2
- Optimisation des performances 2
  - Niveaux d'optimisation 3
- Identification des limites du système 9
- Définition des objectifs 9
- Analyse des performances 10
  - Formes normales 11
  - Verrouillage 12
  - Considérations spéciales 12

CHAPITRE 2 **Réseaux et performances** **15**
- Introduction 15
- Problèmes potentiels de performances 16
  - Questions fondamentales sur les performances réseau 16
  - Récapitulatif des techniques 17
  - Utilisation de sp_sysmon lors de la modification de la configuration réseau 17
- Utilisation du réseau par Adaptive Server 17
- Modification de la taille d'un paquet réseau 18
- Paquets de grande taille et paquets ayant une taille par défaut pour les connexions utilisateur 18
  - Importance du nombre de paquets 19
  - Outils d'évaluation avec Adaptive Server 20
  - Outils d'évaluation en dehors d'Adaptive Server 20
  - Techniques de réduction du trafic réseau fournies par le serveur 21
- Incidence d'autres activités du serveur 22
  - Un seul utilisateur et plusieurs utilisateurs 22

Amélioration des performances réseau 23
Séparation des utilisateurs intensifs d'un réseau 23
Définition de tcp no delay sur les réseaux TCP 24
Configuration de plusieurs récepteurs de réseau..... 24

CHAPITRE 3 Utilisation des moteurs et des CPU 25
Concepts généraux 25
Traitement des requêtes clientes dans Adaptive Server..... 26
Mise en oeuvre de tâche cliente..... 27
Modèle de traitement SMP (une CPU)..... 29
Planification des moteurs pour la CPU..... 29
Planification des tâches pour le moteur 30
Planification de la tâche en exécution..... 32
Modèle de processus SMP d'Adaptive Server..... 35
Planification des moteurs pour les CPU..... 35
Planification des tâches d'Adaptive Server sur des moteurs... 36
Moteurs réseau multiples 36
Propriétés des tâches et files d'attente d'exécution 37
Exemple de traitement 37
Amélioration de l'utilisation CPU par la tâche housekeeper..... 39
Effets pervers de la tâche housekeeper..... 39
Configuration de la tâche housekeeper..... 40
Evaluation de l'utilisation CPU 42
Machines monoprocesseur 42
Détermination de la configuration des moteurs supplémentaires 43
Mise hors ligne des moteurs 44
Activation de la spécialisation de moteur par CPU 45
Instructions pour la conception d'applications multiprocesseur 47

CHAPITRE 4 Répartition des ressources des moteurs 49
Algorithme de répartition optimale des ressources moteur..... 49
Recommandations sur les algorithmes 52
Analyse et planification d'un environnement 53
Tests comparatifs..... 55
Définition des objectifs 56
Analyse et optimisation des résultats..... 56
Contrôle régulier de l'environnement..... 57
Gestion de la priorité d'accès aux ressources..... 57
Types de classe d'exécution 58
Classes d'exécution prédéfinies..... 58
Classe d'exécution définie par l'utilisateur..... 59
Attributs des classes d'exécution 60
Priorité initiale..... 60

Tranche de temps 61
Spécialisation des tâches affectées aux moteurs 62
Définition des attributs des classes d'exécution 63
Affectation des classes d'exécution 63
Groupes de moteurs et spécialisation des tâches affectées aux moteurs 64
Incidence des liaisons des classes d'exécution sur la programmation 66
Définition d'attributs pour une seule session 69
Informations 69
Règles de définition de la priorité et de la portée 69
Objets et classes d'exécution variés 70
Résolution d'un conflit de priorité 72
Exemples : définition de la priorité 73
Exemple illustrant les règles de priorité 75
Planification 76
Configuration 77
Caractéristiques d'exécution 77
Eléments à prendre en compte pour la répartition des ressources moteur 78
Applications clientes OLTP et DSS 78
Logins Adaptive Server : utilisateurs prioritaires 80
Procédure enregistrée "points de contention" 80

CHAPITRE 5 Gestion de l'emplacement physique des données 81
Optimisation des performances grâce au positionnement des objets 81
Symptômes d'un mauvais positionnement des objets 83
Problèmes sous-jacents 83
Utilisation de sp_sysmon lors du changement d'emplacement des données 84
Terminologie et concepts 84
Recommandations pour améliorer les performances des E/S 85
Répartition des données sur plusieurs disques afin d'éviter les conflits d'E/S 86
Isolation des E/S réalisées au niveau du serveur, des E/S sur les bases de données 87
Journaux de transactions sur un disque séparé 88
Mise en miroir d'un device sur un disque séparé 89
Création d'objets sur des segments 90
Utilisation de segments 91
Séparation des tables et des index 92
Répartition de tables volumineuses sur plusieurs devices 92
Déplacement du stockage de texte vers un device distinct 92

Partitionnement des tables pour les performances 93
Transparence 94
Tables partitionnées et traitement des requêtes en parallèle.. 95
Amélioration des performances d'insertion avec les partitions 96
Restrictions sur les tables partionnées..... 97
Paramètres de configuration pour les partitions..... 97
Répartition des partitions sur les devices..... 97
Planification d'espace relative aux tables partitionnées 98
Tables en lecture seule 99
Tables principalement en lecture 99
Tables avec modification de données de façon aléatoire 100
Commandes servant au partitionnement des tables..... 101
Syntaxe alter table...partition..... 102
Syntaxe alter table...unpartition..... 102
Modification du nombre de partitions 103
Répartition uniforme des données sur les partitions 103
Utilisation de bcp parallèle pour copier des données dans les partitions..... 106
Informations sur les partitions 107
Utilisation de bcp pour équilibrer les partitions..... 108
Contrôle de la distribution des données sur les devices avec sp_helpsegment..... 110
Mise à jour des statistiques de partition 112
Etapes de partitionnement des tables..... 113
Sauvegarde de la base de données après le partitionnement des tables..... 114
La table n'existe pas..... 114
La table existe déjà dans un autre emplacement de la base de données 116
La table existe dans le segment..... 117
Procédures spéciales pour les situations difficiles 122
Index clusterisés sur des tables de grande taille..... 122
Alternative pour les index clusterisés 124
Problèmes relatifs aux devices saturés des tables partitionnées. 127
Ajout de disques en cas de saturation des devices 127
Ajout de disques en cas de devices presque saturés 129
Problèmes de maintenance et tables partitionnées 130
Contrôles de maintenance périodiques portant sur les tables partitionnées..... 131

CHAPITRE 6 Conception de bases de données..... 133
Conception de base 133
Conception physique des bases de données pour Adaptive Server..... 134

Tailles des pages logiques 134
Normalisation 135
Niveaux de normalisation 135
Avantages de la normalisation 135
Première forme normale 136
Deuxième forme normale 138
Troisième forme normale 139
Dénormalisation pour l'optimisation des performances 140
Risques 141
Données pour la dénormalisation 143
Techniques 143
Fractionnement de tables 146
Gestion des données dénormalisées 149
Utilisation de triggers 149
Utilisation de la logique de l'application 150
Mise à jour par batch 150

CHAPITRE 7 Stockage de données 151
Gains de performances grâce à l'optimisation des requêtes 151
Traitement des requêtes et lectures de pages 152
Adaptive Server pages 154
En-têtes de page et tailles de page 154
Tailles variables des pages logiques 155
Pages de données et d'index 156
Pages d'objets volumineux (LOB) 156
Pages gérant l'allocation de l'espace 158
Pages de la table d'allocation globale 158
Pages d'allocation 159
Pages de la table d'allocation d'objets 159
Gestion du stockage des objets par les pages d'OAM et les pages d'allocation 160
L'allocation de pages permet de ne pas séparer les pages d'un objet 161
Table sysindexes et accès aux données 161
Surcoût lié à l'espace disque 162
Nombre de colonnes et taille 162
Nombre de lignes par page de données 167
Nombres maximaux 168
Données en vrac : tables sans index clusterisé 168
Plans de verrouillage et différences entre les tables sans index 169
Opérations de sélection sur les tables sans index 170
Insertion de données dans une table APL sans index 170
Insertion de données dans une table DOL sans index 172

Suppression de données d'une table sans index 173
Mise à jour de données dans une table sans index 174
Opérations d'E/S d'Adaptive Server sur les tables sans index 175
Prélecture séquentielle (E/S étendues) 176
Caches et liaisons d'objets 176
Données sans index, E/S et stratégie sur les caches 177
Opérations de sélection et mise en mémoire cache 179
Modification de données et mémoire cache 180
Prélecture asynchrone et E/S de tables sans index 181
Tables sans index : avantages et inconvénients 183
Gestion des tables sans index 183
Méthodes 184
Journal de transactions : table spéciale sans index 185

CHAPITRE 8 Indexation pour l'optimisation des performances 187
Incidence des index sur les performances 187
Détection des problèmes d'indexation 188
Symptômes d'une mauvaise indexation 189
Limites des index et ressources requises 192
Choix des index 193
Clés d'index et clés logiques 194
Recommandations relatives aux index clusterisés 194
Choix des index clusterisés 195
Candidats en tant qu'index non clusterisés 196
Autres recommandations concernant les index 196
Choix des index non clusterisés 198
Choix des index composés 199
Ordre des clés et performances dans les index composés 199
Avantages et inconvénients des index composés 201
Techniques de choix d'index 202
Choix d'un index pour une requête à intervalle 202
Ajout d'une requête ponctuelle avec des critères d'indexation différents 203
Index et gestion des statistiques 205
Suppression des index qui limitent les performances 206
Choix des propriétés de gestion de l'espace des index 206
Conseils supplémentaires d'indexation 207
Création de colonnes artificielles 207
Entrées d'index courtes pour réduire l'overhead 207
Suppression et reconstruction des index 208

CHAPITRE 9 Fonctionnement des index 209
Types d'index 210
Pages d'index 211

Taille des index 212
Index clusterisés dans des tables APL 213
Index clusterisés et sélections..... 213
Index clusterisés et insertions 215
Fractionnement des pages pleines 216
Fractionnement des pages d'index..... 218
Incidence des performances sur le fractionnement de page. 218
Pages d'overflow 219
Index clusterisés et suppressions 220
Index non clusterisés 222
Pages feuille revisitées..... 222
Structure des index non clusterisés 223
Index non clusterisés et sélections..... 224
Performances des index non clusterisés..... 225
Index non clusterisés et insertions 226
Index non clusterisés et suppressions 227
Index clusterisés sur les tables DOL 228
Couverture par index..... 229
Couverture par index avec correspondance 229
Couverture par index sans correspondance 231
Index et mise en mémoire cache 232
Utilisation de caches distincts pour les pages de données et d'index..... 232
Nombre de cycles effectués par l'index dans le cache..... 233

CHAPITRE 10 Configuration et optimisation des verrouillages 235
Verrouillage et performances 235
Utilisation de sp_sysmon et sp_object_stats..... 236
Réduction des conflits de verrouillage..... 237
Autres instructions relatives au verrouillage..... 240
Configuration des verrous et des seuils de conversion des verrous..... 241
Configuration du nombre maximal de verrous sur Adaptive Server..... 242
Configuration de la table de hachage de verrou 244
Définition des seuils de conversion des verrous 244
Choix du plan de verrouillage d'une table 251
Analyse des applications existantes..... 252
Choix d'un plan de verrouillage sur la base des statistiques relatives aux conflits 252
Contrôle et gestion des tables après la conversion..... 254
Applications peu susceptibles de bénéficier d'un verrouillage des données seules..... 255

CHAPITRE 11 Utilisation des commandes de verrouillage..... 257
Spécification du plan de verrouillage d'une table..... 257
Définition d'un schéma de verrouillage sur l'ensemble du serveur..... 258
Spécification d'un plan de verrouillage avec create table..... 258
Changement de plan de verrouillage à l'aide de alter table .. 259
Changement de plan de verrouillage – avant et après..... 260
Coût relatif au passage du verrouillage de toutes les pages au verrouillage des données seules ou inversement 261
Performances du tri durant l'exécution de alter table 262
Spécification d'un plan de verrouillage à l'aide de select into 263
Contrôle des niveaux d'isolement 264
Définition des niveaux d'isolement pour une session..... 264
Syntaxe des options de verrouillage au niveau requête et au niveau table 265
Utilisation de holdlock, noholdlock ou shared 265
Utilisation de la clause at isolation 266
Définition de verrous plus contraignants 267
Définition de verrous moins contraignants 268
Verrouillage readpast..... 269
Curseurs et verrouillage..... 269
Utilisation du mot-clé shared..... 271
Commandes supplémentaires de verrouillage..... 272
Commande lock table..... 272
Temporisations de verrous..... 273

CHAPITRE 12 Informations sur les verrous 275
Outils de verrouillage 275
Informations relatives aux processus bloqués 275
Affichage des verrous..... 276
Affichage des verrous..... 279
Blocage dans une famille lors des fusions de buffers réseau 280
Interblocages et concurrence d'accès..... 280
Interblocages au niveau serveur et au niveau application 280
Interblocages de tâches serveur 281
Interblocages et requêtes parallèles 282
Consignation des informations d'interblocage dans le journal d'erreurs..... 284
Comment éviter les interblocages..... 285
Identification des tables pour lesquelles la concurrence d'accès est problématique..... 287
Informations sur la gestion des verrous 289

CHAPITRE 13 Définition des propriétés de gestion de l'espace 291
Réduction des opérations de maintenance des index 291
Avantages de fillfactor 292
Inconvénients de fillfactor 293
Définition des valeurs de fillfactor 294
exemples de fillfactor 294
Utilisation des options sorted_data et fillfactor 298
Réduction de la redirection de lignes 298
Valeurs par défaut, minimale et maximale de exp_row_size 299
Spécification d'une taille de ligne prévue avec create table .. 300
Ajout ou modification d'une taille de ligne prévue 301
Définition d'une taille de ligne prévue par défaut au niveau serveur 301
Affichage de la taille de ligne prévue pour une table 302
Choix d'une taille de ligne prévue pour une table 302
Conversion de max_rows_per_page en exp_row_size 304
Contrôle et gestion des tables qui utilisent la taille de ligne prévue 304
Réservation d'espace pour les lignes redirigées et les insertions 305
Opérations d'allocation d'extent et reservepagegap 306
Spécification d'un intervalle de page réservée avec create table 307
Spécification d'un intervalle de page réservée avec create index 308
Modification de reservepagegap 308
Exemples de reservepagegap 309
Choix d'une valeur pour reservepagegap 310
Contrôle des valeurs de reservepagegap 311
Options reservepagegap et sorted_data dans create index .. 311
Utilisation de max_rows_per_page dans des tables APL 314
Réduction des conflits de verrouillage 315
Index et max_rows_per_page 315
select into et max_rows_per_page 316
Application de max_rows_per_page aux données existantes 316

CHAPITRE 14 Utilisation et performances de la mémoire 317
Incidence de la mémoire sur les performances 317
Quantité de mémoire à configurer 318
Caches sur Adaptive Server 321
Cache des procédures 322
Informations sur la taille du cache des procédures 323
Dimensionnement du cache des procédures 323
Evaluation de la taille d'une procédure stockée 324

Cache de données 325
Cache par défaut au moment de l'installation 325
Vieillissement des pages dans le cache de données..... 325
Impact du cache de données sur les extractions 327
Impact des modifications de données sur le cache..... 328
Performances du cache de données..... 328
Test des performances du cache de données 328
Configuration du cache de données pour améliorer les performances 330
Commandes de configuration des caches de données nommés..... 333
Optimisation des caches nommés 333
Objectifs de la configuration des caches..... 334
Collecte de données, planification et mise en oeuvre..... 335
Evaluation des besoins en cache..... 337
E/S étendues et performances..... 337
Réduction des conflits de verrous avec les partitions de caches 339
Stratégies et principes de remplacement dans le cache..... 340
Recommandations relatives au cache de données nommé 343
Définition de la taille des caches pour des objets spécifiques, pour tempdb et pour les journaux de transaction 345
Détermination de la taille des zones de données en fonction des plans d'exécution de requête et des E/S 349
Configuration de la taille de vidage des buffers 352
Overhead de la configuration des zones et de la liaison d'objets 353
Maintien des performances du cache de données pour les E/S étendues..... 354
Diagnostic d'un nombre d'E/S excessifs 355
Utilisation de sp_sysmon pour vérifier les performances des E/S étendues 356
Vitesse de reprise 356
Optimisation de l'intervalle de reprise..... 357
Influence de la tâche "housekeeper" sur le temps de reprise 358
Audit et performances 358
Définition de la taille de la file d'attente d'audit..... 358
Recommandations concernant les performances des audits 359

CHAPITRE 15 Détermination de la taille des tables et des index..... 361
Pourquoi la taille des objets est importante pour l'optimisation des requêtes..... 361
Outils pour la détermination de la taille des tables et des index .. 363

Conséquences des modifications de données sur la taille des objets 363
Utilisation de optdiag pour afficher la taille des objets 364
Avantages de optdiag 365
Inconvénients de optdiag 365
Utilisation de la procédure sp_spaceused pour afficher la taille d'un objet 365
Avantages de sp_spaceused 367
Inconvénients de sp_spaceused 367
Evaluation de la taille des objets à l'aide de sp_estspace 367
Avantages de sp_estspace 369
Inconvénients de sp_estspace 369
Evaluation de la taille des objets à l'aide des formules de calcul. 369
Facteurs pouvant modifier la taille de stockage 370
Tailles de stockage des différents types de données 371
Tables et index utilisés dans les formules 372
Calcul des tailles des tables et des index clusterisés pour les tables APL 373
Calcul des tailles des tables DOL 379
Autres facteurs influant sur la taille des objets 384
Lignes très courtes 386
Pages LOB 387
Avantages de l'utilisation de formules pour l'évaluation de la taille des objets 388
Inconvénients de l'utilisation de formules pour l'évaluation de la taille des objets 388

CHAPITRE 16 Activités de maintenance et performances 389
Exécution de reorg sur des tables et des index 390
Création et maintenance d'index 390
Configuration d'Adaptive Server pour accélérer le tri 391
Sauvegarde d'une base de données après la création d'un index 391
Création d'un index sur des données triées 391
Gestion des index et des statistiques sur les colonnes 393
Reconstruction des index 394
Création ou modification d'une base de données 396
Sauvegarde et restauration 397
Sauvegardes locales 398
Sauvegardes à distance 398
Sauvegardes en ligne 398
Utilisation de seuils pour éviter le manque d'espace de journalisation 398
Striping de sauvegarde 399

Diminution du temps de restauration..... 399
Ordre de restauration 399
Bulkcopy..... 399
Bulkcopy parallèle 400
Batchs et bulkcopy 401
Bulkcopy en mode lent..... 401
Amélioration des performances de bulkcopy 401
Remplacement de données dans une table volumineuse..... 402
Ajout d'un grand volume de données dans une table 402
Utilisation de partitions et de processus de bulkcopy multiples 403
Conséquences sur les autres utilisateurs..... 403
Vérificateur de cohérence de base de données..... 403
Utilisation de dbcc tune (cleanup) 404
Détermination de l'espace disponible pour des activités de maintenance..... 404
Généralités sur les besoins en espace 405
Outils servant à vérifier l'utilisation de l'espace et sa disponibilité..... 405
Estimation de l'impact lié aux propriétés de gestion de l'espace 408
Insuffisance d'espace libre 409

Préface

A qui s'adresse ce manuel ?

Ce manuel s'adresse aux administrateurs et aux concepteurs de bases de données, aux développeurs et aux administrateurs système.

Remarque Vous pouvez utiliser votre propre base de données pour tester les modifications et les requêtes. Prenez un cliché de la base de données en question et configurez-la sur une machine de test.

Comment utiliser ce manuel ?

Ce manuel vous aidera normalement à optimiser ou à améliorer les performances sur Adaptive Server ainsi qu'à résoudre d'éventuels problèmes. Le document *Performances et optimisation* est divisé en trois volumes :

- Volume 1 - *Concepts de base*
- Volume 2 - *Optimisation de requêtes avec les plans abstraits*
- Volume 3 - *Outils d'administration et de suivi des performances*

Sont traités les thèmes suivants :

Volume 1 : Concepts de base

Le chapitre 1, "Présentation", décrit les principaux composants à analyser lors de la recherche de performances.

Le chapitre 2, "Réseaux et performances", contient une brève description des bases de données relationnelles et d'une conception performante des bases de données.

Le chapitre 3, "Utilisation des moteurs et des CPU", décrit comment les processus clients sont répartis sur les moteurs dans Adaptive Server.

Le chapitre 4, "Répartition des ressources des moteurs", explique comment définir des priorités pour l'exécution de certaines applications.

Le chapitre 5, "Gestion de l'emplacement physique des données", décrit comment utiliser les segments et les partitions pour gérer l'emplacement physique des données sur les devices de stockage.

Le chapitre 6, "Conception de bases de données", contient une brève description des bases de données relationnelles et d'une conception performante des bases de données.

Le chapitre 7, "Stockage de données", décrit les types de page d'Adaptive Server, la manière dont les données sont stockées dans ces pages et la manière dont les requêtes sont exécutées sur les tables sans index.

Le chapitre 8, "Indexation pour l'optimisation des performances", fournit des instructions et des exemples pour le choix des index.

Le chapitre 9, "Fonctionnement des index", fournit des informations sur la manière d'utiliser les index pour résoudre des requêtes.

Le chapitre 10, "Configuration et optimisation des verrouillages", décrit l'impact du verrouillage sur les performances ; il présente aussi les outils permettant d'analyser les problèmes de verrouillage et de configurer les verrous.

Le chapitre 11, "Utilisation des commandes de verrouillage", décrit les commandes qui définissent des plans de verrouillage des tables et contrôlent les niveaux d'isolement et autres modes de verrouillage lors du traitement des requêtes.

Le chapitre 12, "Informations sur les verrous", décrit les procédures système qui génèrent des rapports sur les verrous et les conflits de verrouillage.

Le chapitre 13, "Définition des propriétés de gestion de l'espace", décrit comment définir les propriétés de gestion de l'espace dans les tables en vue d'améliorer les performances et de réduire la fréquence des opérations de maintenance sur les tables et les index.

Le chapitre 14, "Utilisation et performances de la mémoire", décrit la manière dont Adaptive Server utilise la mémoire pour les caches de procédures et de données.

Le chapitre 15, "Détermination de la taille des tables et des index", décrit différentes méthodes pour déterminer la taille actuelle des objets d'une base de données et pour estimer leur taille future.

Le chapitre 16, "Activités de maintenance et performances", décrit l'impact des activités de maintenance sur les performances et comment optimiser les performances à l'aide de certaines opérations, telles que la recréation d'index.

Volume 2 : Optimisation de requêtes avec les plans abstraits

Le chapitre 17, "Optimiseur Adaptive Server", détaille le processus d'optimisation des requêtes et explique comment sont utilisées les statistiques en vue de rechercher les arguments et les jointures pour les requêtes.

Le chapitre 18, "Outils d'optimisation avancés", décrit les outils avancés pour l'optimisation des performances des requêtes.

Le chapitre 19, "Outils d'optimisation de la requête", contient une présentation des outils d'optimisation des requêtes et une description de leur mode d'interaction.

Le chapitre 20, "Méthodes d'accès et évaluation des coûts de requête sur une seule table", décrit la manière dont Adaptive Server accède aux tables des requêtes qui ne se réfèrent qu'à une seule table et comment il évalue les coûts associés aux différentes méthodes d'accès.

Le chapitre 21, "Méthodes d'accès et coût des requêtes pour des jointures et des sous-requêtes", décrit comment Adaptive Server accède aux tables pendant les jointures et les sous-requêtes et comment il détermine les coûts.

Le chapitre 22, "Traitement des requêtes parallèles", présente les concepts et les ressources nécessaires pour le traitement des requêtes parallèles.

Le chapitre 23, "Optimisation des requêtes parallèles", traite en détail de l'optimisation des requêtes parallèles.

Le chapitre 24, "Tri parallèle", traite de l'utilisation du tri parallèle pour les requêtes et la création d'index.

Le chapitre 25, "Optimisation de la prélecture asynchrone", décrit comment l'option prefetch asynchrone améliore les performances des requêtes qui exécutent des opérations d'E/S disque d'envergure.

Le chapitre 26, "Performances de tempdb", souligne l'importance de la base de données temporaire, *tempdb,* et fournit des indications pour améliorer ses performances.

Le chapitre 27, "Curseurs et performances", traite des questions relatives aux performances avec curseurs.

Le chapitre 28, "Introduction aux plans abstraits", contient une présentation des plans abstraits et de la manière dont ils peuvent être utilisés pour résoudre des problèmes d'optimisation des requêtes.

Le chapitre 29, "Guide des plans abstraits pour les requêtes", contient une introduction à la rédaction des plans abstraits pour des types de requêtes spécifiques ainsi qu'à l'utilisation des plans abstraits pour détecter les changements apportés à l'optimisation des requêtes et liés à des modifications de la configuration ou du système.

Le chapitre 30, "Création et utilisation des plans abstraits", décrit les commandes nécessaires pour sauvegarder et utiliser des plans abstraits.

Le chapitre 31, "Gestion des plans abstraits avec des procédures système", décrit les procédures système qui gèrent les plans abstraits et les groupes de plans abstraits.

Le chapitre 32, "Référence de langage des plans abstraits", décrit le langage des plans abstraits.

Volume 3 : Outils d'administration et de suivi des performances

Le chapitre 33, "Utilisation des statistiques pour améliorer les performances", explique comment utiliser la commande update statistics pour créer et mettre à jour des statistiques.

Le chapitre 34, "Utilisation des commandes set statistics", décrit les commandes qui fournissent des informations concernant l'exécution.

Le chapitre 35, "Utilisation de set showplan", offre des exemples de messages showplan.

Le chapitre 36, "Affichage des tables de statistiques avec optdiag", décrit les tables qui stockent les statistiques et les résultats de la commande optdiag qui affiche les statistiques utilisées par l'optimiseur de requêtes.

Le chapitre 37, "Optimisation avec dbcc traceon", explique comment utiliser les commandes dbcc traceon pour analyser les problèmes d'optimisation de requêtes.

Le chapitre 38, "Contrôle des performances avec sp_sysmon", explique comment utiliser une procédure système afin de contrôler les performances d'Adaptive Server.

Index

L'index complet de l'ensemble des trois volumes se trouve à la fin du *Volume 3 : Outils d'administration et de suivi des performances*.

Documentation

La documentation Sybase Adaptive Server Enterprise comprend les manuels suivants :

- Les Notes de mise à jour pour votre plate-forme contiennent les informations de dernière minute qui ne figurent pas dans les manuels.

  Une version plus récente des Notes de mise à jour est disponible sur le Web. Pour rechercher des informations ultérieures à la commercialisation du CD-ROM du produit, consultez le site Sybase Technical Library.
- Le *Guide d'installation* d'Adaptive Server pour votre plate-forme décrit les procédures d'installation, de mise à niveau et de configuration pour tous les produits Adaptive Server et Sybase associés.
- Le *Manuel de configuration d'Adaptive Server Enterprise* pour votre plate-forme fournit des instructions sur les tâches de configuration particulières pour Adaptive Server.
- Le document *Sybase Adaptive Sever Enterprise - Nouvelles fonctionnalités* décrit les nouvelles caractéristiques de la version 12.5 d'Adaptive Server, les modifications apportées au système pour leur prise en charge et les modifications susceptibles d'avoir des conséquences sur vos applications existantes.
- Le *Guide de l'utilisateur Transact-SQL* présente Transact-SQL, version enrichie du langage de base de données relationnelle de Sybase. Ce manuel sert de référence pour les utilisateurs qui découvrent le système de gestion de bases de données. Il décrit également les bases de données exemple pubs2 et pubs3.
- Le *Guide d'administration système* fournit des informations détaillées sur l'administration des serveurs et des bases de données. Ce manuel contient des instructions relatives à la gestion des ressources physiques, de la sécurité et des bases de données système et utilisateur, ainsi qu'au paramétrage de la conversion de caractères, la langue et l'ordre de tri.
- Le *Manuel de référence* contient des informations détaillées sur toutes les commandes, fonctions, procédures et types de données Transact-SQL. Ce manuel fournit également la liste des mots réservés Transact-SQL et les définitions des tables système.
- Le document *Performances et optimisation* explique comment optimiser les performances d'Adaptive Server. Il contient des informations sur les aspects de la conception des bases de données qui conditionnent les performances, sur l'optimisation des requêtes, sur le paramétrage d'Adaptive Server pour des bases de données volumineuses, sur la configuration des caches et des disques et sur l'impact du verrouillage et des curseurs sur les performances.

- Le manuel *Utilitaires* décrit les utilitaires d'Adaptive Server tels qu'isql et bcp, qui sont exécutés au niveau du système d'exploitation.
- Le *Guide de référence rapide* est un petit fascicule qui répertorie les noms et syntaxes des commandes, fonctions, procédures système, procédures système étendues, types de données et utilitaires. Il est uniquement disponible sur papier.
- Le *Diagramme des tables système* est un poster qui illustre les tables système selon le modèle entités-relations. Il est uniquement disponible sur papier.
- Les documents *Error Messages et Troubleshooting Guide* expliquent comment résoudre les conditions d'erreur les plus courantes et donnent les solutions aux problèmes système souvent rencontrés par les utilisateurs.
- Le *Guide de l'utilisateur de Component Integration Services* explique comment utiliser la fonctionnalité Component Integration Services d'Adaptive Server pour se connecter à des bases de données distantes Sybase et non Sybase.
- Le document *Java dans Adaptive Server Enterprise* explique comment installer et utiliser les classes Java en tant que types de données, fonctions et procédures stockées dans la base de données Adaptive Server.
- Le document *Utilisation de Sybase Failover en environnement haute disponibilité* traite de l'utilisation de Sybase Failover pour configurer Adaptive Server comme serveur compagnon dans un environnement haute disponibilité.
- Le document *Utilisation des fonctionnalités DTM* traite de la configuration et de l'utilisation des fonctionnalités DTM d'Adaptive Server ainsi que de la résolution des éventuels problèmes dans les environnements de traitement des transactions distribuées.
- Le *Guide de l'utilisateur d'EJB Server* explique comment utiliser EJB Server pour déployer et exécuter Enterprise JavaBeans dans Adaptive Server.
- Le document *XA Interface Integration Guide for CICS, Encina, and TUXEDO* fournit des instructions sur l'utilisation de l'interface DTM XA de Sybase avec les gestionnaires de transactions X/Open XA.
- Le *Glossaire* définit les termes techniques utilisés dans la documentation Adaptive Server.
- Le document *Sybase jConnect for JDBC Programmer's Reference* décrit le produit jConnect for JDBC et explique comment l'utiliser pour accéder aux données stockées dans des systèmes de gestion de bases de données relationnelles.

- Le document *Full-Text Search Specialty Data Store - Guide de l'utilisateur* explique comment utiliser la fonction Full-Text Search avec Verity afin d'effectuer des recherches dans les données d'Adaptive Server Enterprise.
- Le *Guide de l'utilisateur de Monitor Historical Server* explique comment utiliser Historical Server afin d'obtenir des statistiques de performances de SQL Server et Adaptive Server.
- Le *Guide de l'utilisateur de Monitor Server* explique comment utiliser Monitor Server afin d'obtenir des statistiques de performances de SQL Server et Adaptive Server.
- Le document *Monitor Client Library Programmer's Guide* explique comment écrire des applications Monitor Client Library accédant aux données de performances d'Adaptive Server.

Autres sources d'information

Utilisez le CD Sybase Technical Library ainsi que le site Web Technical Library Product Manuals pour obtenir davantage d'informations sur les produits :

- Le CD-ROM Technical Library, qui contient des manuels sur les produits et des documents techniques, est livré avec le logiciel. L'explorateur DynaText (téléchargeable à partir du site Product Manuals (http://www.sybase.com/detail/1,3693,1010661,00.html)) vous permet d'accéder aux informations techniques sur les produits dans un format facile à utiliser.

  Pour plus d'informations sur l'installation et le démarrage de la Technical Library, reportez-vous au document *Technical Library Installation Guide*.

- Le site Web Technical Library Product Manuals est une version HTML du CD Technical Library, à laquelle vous pouvez accéder à l'aide d'un navigateur Web standard. Outre les manuels sur les produits, vous trouverez également des liens vers le site Web Technical Documents (anciennement Tech Info Library), la page Solved Cases et des forums Sybase/Powersoft.

  Pour accéder à Technical Library Product Manuals, rendez-vous sur le site Product Manuals (http://www.sybase.com/support/manuals/).

Certifications Sybase sur le Web

La documentation technique disponible sur le site Web de Sybase est fréquemment mise à jour.

❖ **Pour accéder aux informations les plus récentes sur les certifications de produit :**

1. Cliquez sur Technical Documents (http://www.sybase.com/support/techdocs/).
2. Sélectionnez des produits dans la barre de navigation située à gauche.
3. Sélectionnez un nom de produit dans la liste des produits.
4. Sélectionnez le filtre Certification Report, choisissez une période de temps dans la liste Time Frame, puis cliquez sur Go.
5. Cliquez sur le titre du rapport de certification que vous voulez consulter.

❖ **Pour accéder aux informations les plus récentes sur les recueils de correctifs de bugs et les mises à jour**

1. Cliquez sur Technical Documents (http://www.sybase.com/support/techdocs/).
2. Sélectionnez EBFs/Updates. Saisissez vos nom d'utilisateur et mot de passe si vous y êtes invité (pour les comptes Web existants) ou créez un compte (service gratuit).
3. Sélectionnez une période de temps dans la liste Time Frame, puis cliquez sur Go.
4. Sélectionnez un produit.
5. Cliquez sur l'EBF ou la mise à jour souhaité.

❖ **Pour créer une vue personnalisée du site Web de Sybase (y compris des pages de support technique)**

Créez un profil MySybase. MySybase est un service gratuit qui vous permet de créer une vue personnalisée des pages Web de Sybase.

1. Cliquez sur Technical Documents (http://www.sybase.com/support/techdocs/).
2. Cliquez sur MySybase, puis créez un profil MySybase.

Conventions

Les conventions typographiques utilisées dans ce manuel sont les suivantes :

Présentation des instructions SQL

SQL est un langage à structure non imposée. Il n'existe aucune règle quant au nombre de mots qui peuvent être placés sur une ligne ou à l'emplacement des fins de ligne. Cependant, pour une meilleure lisibilité, toutes les instructions syntaxiques et tous les exemples de ce manuel sont présentés de sorte que chaque clause d'une instruction commence sur une nouvelle ligne. Les clauses composées de plusieurs parties s'étendent sur les lignes suivantes, qui apparaissent en retrait.

Conventions typographiques et syntaxiques

Les conventions typographiques et syntaxiques utilisées dans ce manuel sont répertoriées dans le tableau 1 :

Tableau 1 : Conventions typographiques et syntaxiques de ce manuel

Elément	Exemple
Les noms de commande, d'option de commande, d'utilitaire, de drapeau d'utilitaire et autres mots-clés apparaissent en gras.	select sp_configure
Les noms de base de données, les types de données, les noms de fichier et de chemin d'accès figurent en *italique*.	base de données master

Les variables (termes se substituant aux valeurs à insérer) apparaissent en *italiques*.	`select`
	nom_colonne
	
	`from`
	nom_table
	
	`where`
	conditions_recherche

Les parenthèses font partie de la commande et doivent être saisies.	`compute`
	`agrégat_ligne`
	`(`
	`nom_colonne`
	`)`

Les accolades indiquent que vous devez choisir au moins une des options énumérées. Ne tapez pas d'accolade.	`{espèces, chèque, crédit}`
Les crochets indiquent que les options citées sont facultatives. Ne tapez pas de crochet.	`[anchois]`
La barre verticale indique que vous ne pouvez choisir qu'une seule des options énumérées.	`{mourir_debout | vivre_à_genoux | vivre_debout}`

Elément	Exemple
La virgule vous permet de choisir autant d'options que vous le souhaitez, à condition de les séparer par une virgule dans la commande.	`[fromage, avocat, sauce]`
Les points de suspension (...) indiquent que vous pouvez *répéter* le dernier élément autant de fois que vous le souhaitez.	`acheter article = prix [comptant | chèque | carte] [, article = prix [comptant | chèque | carte]]...` Vous devez acheter au moins un article et indiquer son prix. Vous pouvez choisir un mode de paiement : l'un des éléments entre crochets. Vous pouvez également décider d'acheter des articles supplémentaires : autant d'articles que vous le souhaitez. Pour chaque article acheté, indiquez son nom, son prix et (éventuellement) un mode de paiement.

- Les instructions syntaxiques (affichage de la syntaxe et de toutes les options d'une commande) apparaissent comme suit :

```
sp_dropdevice [nom_device]
```

  ou, dans le cas d'une commande comportant davantage d'options :

```
select nom_colonne
    from nom_table
    where conditions_recherche
```

Dans les instructions syntaxiques, les mots-clés (commandes) figurent en caractères standard et les identificateurs figurent en minuscules : caractères standard pour les mots-clés, en italique pour les mots fournis par l'utilisateur.

- Les exemples de résultats apparaissent à l'écran de la façon suivante :

```
0736 New Age Books Boston MA
0877 Binnet & Hardley Washington DC
1389 Algodata Infosystems Berkeley CA
```

Casse

Dans ce manuel, la plupart des exemples sont en minuscules. Cependant, vous pouvez ignorer la distinction entre majuscules et minuscules lorsque vous tapez des mots-clés Transact-SQL. Par exemple, SELECT, Select et select sont équivalents. Notez que la distinction majuscules/minuscules faite par Adaptive Server pour les objets de base de données, tels que des noms de table, dépend de l'ordre de tri défini dans Adaptive Server. Vous pouvez modifier ce paramètre pour les jeux de caractères à octet simple en reconfigurant l'ordre de tri d'Adaptive Server.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Expressions

Les instructions syntaxiques Adaptive Server utilisent les types d'expression suivants :

Tableau 2 : Types d'expressions utilisées dans les instructions syntaxiques

Utilisation	Définition
expression	Peut inclure des constantes, des littéraux, des fonctions, des identificateurs de colonne, des variables ou des paramètres.
expression logique	Expression renvoyant une valeur TRUE (vrai), FALSE (faux) ou UNKNOWN (inconnu).
expression constante	Expression renvoyant toujours la même valeur, telle que "5+3" ou "ABCDE".
expression_flottante	Toute expression à virgule flottante ou expression convertie implicitement en une valeur flottante.
expression_entier	Toute expression entière ou expression convertie implicitement en une valeur entière.
expression_numérique	Toute expression numérique renvoyant une valeur unique.
expression_caractère	Toute expression renvoyant une valeur unique de type caractère.
expression_binaire	Toute expression renvoyant une valeur unique de type *binary* ou *varbinary*.

Exemples

La plupart des exemples de ce manuel proviennent d'une base de données appelée pubtune. Le schéma de cette base de données est le même que celui de la base de données pubs2 mais les tables utilisées dans les exemples contiennent plus de lignes : la table titles en comporte 5000, la table authors 5000 et la table titleauthor 6250. Divers index sont générés pour illustrer différentes fonctionnalités dans de nombreux exemples et ces index sont décrits dans le texte.

La base de données pubtune n'est pas fournie avec Adaptive Server. Etant donné que de nombreux exemples illustrent les résultats de commandes telles que set showplan et set statistics io, l'exécution des requêtes décrites dans ce manuel sur les tables pubs2 n'aboutira pas aux mêmes résultats d'E/S et, dans la plupart des cas, n'aboutira pas aux plans de requête présentés ici.

Si vous avez besoin d'aide

Pour chaque installation Sybase faisant l'objet d'un contrat de support, une ou plusieurs personnes désignées sont autorisées à contacter le Support Technique de Sybase. Si vous avez des questions ou besoin d'aide pendant l'installation, demandez à la personne désignée de contacter le Support Technique de Sybase ou la filiale Sybase la plus proche.

CHAPITRE 1

Présentation

Ce chapitre est une introduction à l'optimisation des performances des bases de données.

Sujet	Page
Bonnes performances	1
Optimisation des performances	2
Identification des limites du système	9
Définition des objectifs	9
Analyse des performances	10

Bonnes performances

Les performances mesurent l'efficacité d'une ou plusieurs applications s'exécutant dans un même environnement. Elles s'expriment généralement en termes de *temps de réponse* et en *débit*.

Temps de réponse

Le temps de réponse est le temps nécessaire à l'exécution d'une tâche particulière. Pour réduire ce temps de réponse, vous pouvez :

- réduire les conflits et les temps d'attente, notamment les temps d'attente d'E/S disque,
- utiliser des composants plus rapides,
- limiter la durée d'utilisation des ressources.

Dans certains cas, Adaptive Server est optimisé pour réduire le temps de réponse initial, c'est-à-dire le temps nécessaire pour renvoyer la première ligne à l'utilisateur.

Cela s'avère particulièrement utile dans les applications où un utilisateur peut être amené à extraire plusieurs lignes avec une requête, pour ensuite les parcourir lentement avec un outil frontal.

Débit

Le débit désigne le volume de travail effectué pendant une période déterminée. Il existe deux conceptions du débit :

- pour une transaction particulière, par exemple 5 transactions UpdateTitle par minute ; ou
- pour l'ensemble d'Adaptive Server, par exemple 50 ou 500 transactions sur l'ensemble du serveur par minute.

On exprime généralement le débit en transactions par seconde (tps), mais on peut également le mesurer en minutes, heures, jours, etc.

Conception orientée performances

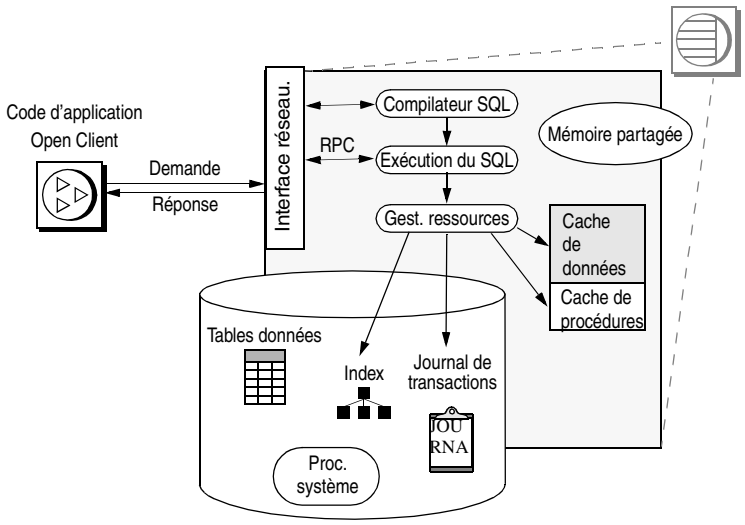
En termes de performances, la majeure partie des gains sont le résultat d'une bonne conception de la base de données, d'une analyse approfondie des requêtes et d'une indexation appropriée. C'est en apportant une attention toute particulière à la conception de la base de données et en apprenant à travailler avec l'optimiseur de requêtes d'Adaptive Server dans le développement de vos applications que vous obtiendrez les meilleurs gains de performances.

D'autres observations, telles que l'analyse du matériel et du réseau, peuvent vous permettre d'identifier des goulets d'étranglement dans votre système.

Optimisation des performances

L'optimisation consiste à améliorer les performances. Pour identifier les problèmes de performances au niveau de chaque couche, vous pouvez utiliser un modèle de système comprenant Adaptive Server et son environnement.

Figure 1-1 : Modèle de système Adaptive Server



L'objectif principal de l'optimisation consiste à réduire les conflits relatifs aux ressources du système. En effet, les conflits d'utilisation de ressources telles que les caches de données et de procédures, les verrous d'attente sur les ressources système et le processeur, augmentent avec le nombre d'utilisateurs. Les risques de conflits de verrouillage de pages de données connaissent également une progression.

Niveaux d'optimisation

Adaptive Server et son environnement, de même que les applications, peuvent être divisés en composants, ou couches d'optimisation, de façon à isoler certains composants du système à analyser. Très souvent, deux ou plusieurs couches doivent être optimisées de façon à travailler ensemble da façon optimale.

Dans certains cas, le fait de supprimer un goulot d'étranglement au niveau d'une couche révèle d'autres problèmes. Inversement, et c'est là une perspective plus optimiste, la résolution d'un problème permet parfois d'en éliminer plusieurs autres.

Par exemple, si le taux d'E/S physiques est élevé pour les requêtes et que vous ajoutez de la mémoire pour améliorer le temps de réponse et augmenter le taux de présence dans le cache, vous résolvez ainsi également des problèmes de conflits de disque.

Les informations suivantes concernent les couches d'optimisation pour Adaptative Server.

Couche applicative

La plupart des gains de performances découlent de l'optimisation des requêtes, fondée sur une conception optimale de la base de données. Ce manuel traite essentiellement du fonctionnement interne d'Adaptive Server et des techniques et outils de traitement des requêtes ayant le but de maintenir des performances élevées.

Les principaux points à prendre en considération sont les suivants :

- Les applications décisionnelles (DSS) et transactionnelles (OLTP) impliquent des stratégies de performances différentes.
- La conception des transactions peut réduire les performances, dans la mesure où les transactions longues maintiennent longtemps leurs verrous et limitent ainsi l'accès des autres utilisateurs aux données.
- L'intégrité relationnelle nécessite des jointures pour la modification des données.
- L'indexation pour la gestion des sélections allonge les temps de modification des données.
- Les audits effectués à des fins de sécurité peuvent nuire aux performances.

Pour pallier ces inconvénients, vous pouvez notamment :

- Utiliser le traitement à distance ou le traitement répliqué pour éviter d'effectuer des traitements décisionnels sur les machines OLTP.
- Utilisation des procédures enregistrées pour réduire le temps de compilation et l'utilisation du réseau.
- Définition du niveau de verrouillage minimal répondant aux besoins de l'application.

Couche base de données

Les applications partagent les ressources (disques, journal de transactions, cache de données) au niveau de la couche base de données.

Une base de données peut contenir 2^31 (2 147 483 648) pages logiques. Ces pages logiques sont divisées entre les différents devices, jusqu'à la limite disponible sur chaque device. Par conséquent, la taille maximale d'une base de données dépend du nombre et de la taille des devices disponibles.

L'overhead ou surcoût est l'espace réservé au serveur et non disponible pour une quelconque base de données de l'utilisateur. Il comprend :

- la taille de la base de données master,
- plus la taille de la base de données modèle,
- plus la taille de tempdb,
- (12.0 et suivantes) plus la taille de sybsystemdb,
- plus 8 ko pour la zone de configuration du serveur.

Les principaux points à prendre en considération sont les suivants :

- Développement d'une stratégie de sauvegarde et de restauration.
- Répartition des données sur les devices.
- Audits : pour préserver les performances, procédez uniquement aux audits nécessaires.
- Planification des activités de maintenance pouvant diminuer les performances et interdire aux utilisateurs l'accès aux tables.

Pour pallier ces inconvénients, vous pouvez notamment :

- Définir des seuils au niveau du journal de transactions pour automatiser les sauvegardes des journaux et éviter la saturation de la mémoire.
- Définir des seuils pour contrôler l'espace dans les segments de données.
- Définir des partitions pour accélérer le chargement des données.
- Installer les objets sur les devices de façon à éviter les conflits de disque ou à tirer profit du parallélisme d'E/S.
- Mettre en mémoire cache les tables et index importants de façon à garantir leur disponibilité.

Couche Adaptive Server

Couche serveur : la couche serveur regroupe de nombreuses ressources partagées, telles que les caches de données et le cache de procédures, les verrous et les CPU.

Les principaux points à prendre en considération pour la couche d'Adaptive Server sont les suivants :

- Les types d'application supportés : application OLTP ou DSS, ou une combinaison des deux.
- Le nombre d'utilisateurs supportés peut influer sur les choix en matière d'optimisation, puisque les risques de conflits de ressources augmentent avec le nombre d'utilisateurs.
- La charge du réseau.
- Replication Server®, ou tout autre traitement réparti, peut être un problème lorsque le nombre d'utilisateurs et le taux de transactions atteignent des niveaux élevés.

Pour pallier ces inconvénients, vous pouvez notamment :

- Optimiser la mémoire (le paramètre de configuration le plus critique) et d'autres paramètres.
- Opter pour un traitement client ou serveur : une partie du traitement ne peut-elle pas être assurée par le client ?
- Configurer les tailles des caches et des E/S.
- Ajouter des CPU.
- Programmer les traitements en batch et la génération de rapports en dehors des heures ouvrées.
- Reconfigurer certains paramètres pour réorganiser la charge de travail.
- Déterminer s'il est possible de transférer l'application DSS sur un autre Adaptive Server.

Couche devices

Cette couche concerne le disque et les contrôleurs qui stockent les données. Adaptive Server peut gérer jusqu'à 256 devices.

Les principaux points à prendre en considération sont les suivants :

- Le device master, les devices contenant la base de données utilisateur ou les journaux de bases de données seront-ils dupliqués en miroir ?
- Comment sont répartis les bases de données système, les bases de données utilisateur et les journaux de bases de données sur les devices ?
- Des partitions sont-elles nécessaires pour améliorer les performances des requêtes parallèles ou des insertions sur les tables sans index ?

Pour pallier ces inconvénients, vous pouvez notamment :

- Utiliser des devices et des contrôleurs de taille moyenne, ce qui peut vous offrir un meilleur débit d'E/S qu'un petit nombre de devices plus grands.
- Répartir les bases de données, tables et index pour créer une charge d'E/S uniforme sur les devices.
- Utiliser des segments et des partitions pour améliorer les performances d'E/S sur les tables volumineuses utilisées dans des requêtes parallèles.

Couche réseau

Cette couche contient le ou les réseaux connectant les utilisateurs à Adaptive Server.

Pratiquement tous les utilisateurs d'Adaptive Server accèdent à leurs données par l'intermédiaire du réseau. Les principaux points à prendre en considération sont les suivants :

- L'importance du trafic.
- Les goulets d'étranglement.
- La vitesse.

Pour pallier ces inconvénients, vous pouvez notamment :

- Configurer la taille des paquets en fonction des besoins de l'application.
- Configurer les sous-réseaux.

- Isoler les utilisations lourdes du réseau.
- Passer à une capacité de réseau supérieure.
- Adopter une configuration autorisant plusieurs moteurs de réseau.
- Concevoir les applications de façon à limiter le trafic nécessaire sur le réseau.

Couche matériel

La couche matériel comprend les CPU disponibles.

Les principaux points à prendre en considération sont les suivants :

- Débit des CPU.
- Accès aux disques : contrôleurs et disques.
- Sauvegarde des disques.
- Utilisation de la mémoire.

Pour pallier ces inconvénients, vous pouvez notamment :

- Ajouter des CPU pour répondre à la charge de travail.
- Configurer les tâches d'intendance (Housekeeper) de façon à améliorer l'utilisation de la CPU.
- Suivre les consignes de conception d'applications multiprocesseur afin de réduire les conflits.
- Configurer plusieurs caches de données.

Couche système d'exploitation

En théorie, Adaptive Server doit être la principale application présente sur une machine et doit partager la CPU, la mémoire et les autres ressources uniquement avec le système d'exploitation et d'autres logiciels Sybase tels que Backup Server™ et Adaptive Server Monitor™.

Les principaux points à prendre en considération sont les suivants :

- les systèmes de fichiers disponibles pour Adaptive Server ;
- la gestion de la mémoire : évaluation précise de l'overhead au niveau du système d'exploitation et de l'utilisation de la mémoire par d'autres programmes ;
- CPU disponibles et CPU allouées à Adaptive Server.

Vous disposez des techniques suivantes :

- Interface réseau.
- Opter pour des fichiers ou des partitions de disque.
- Augmenter la taille de la mémoire.
- Transférer les opérations clientes et les traitements en batch sur d'autres machines.
- Utiliser plusieurs CPU pour Adaptive Server.

Identification des limites du système

Il existe des limites à l'augmentation des performances. En effet, les caractéristiques physiques de la CPU, les sous-systèmes de disques et les réseaux imposent certaines limites. Il est possible d'en dépasser certaines en ajoutant de la mémoire, en utilisant des disques à accès rapide, en optant pour des réseaux à largeur de bande supérieure ou encore en augmentant le nombre des CPU.

Pour un ensemble de composants donné, toute requête nécessite un temps de réponse minimal. En outre, selon les limites du système, les sous-systèmes physiques saturent au-delà d'un certain niveau.

Définition des objectifs

Pour nombre de systèmes, les spécifications de performances développées dès les premières phases du cycle de vie d'une application définissent les temps de réponse des différents types de requête, ainsi que le débit prévu pour l'ensemble du système.

Analyse des performances

En cas de problèmes de performances, vous devez en identifier les causes et déterminer vos objectifs en ce qui concerne leur résolution. Pour l'analyse, procédez comme suit :

1 Réunissez les données relatives aux performances afin d'établir des critères de mesure. Par exemple, vous pouvez utiliser un ou plusieurs des outils suivants :

   - Tests de performances développés en interne ou tests standard disponibles sur le marché.
   - Procédure système sp_sysmon, qui contrôle les performances d'Adaptive Server et renvoie des statistiques sur le fonctionnement du système Adaptive Server.

     Reportez-vous au document *Performances et optimisation : Outils pour les statistiques de performances* pour toute information sur l'utilisation de sp_sysmon.
   - Adaptive Server Monitor, qui fournit des outils graphiques pour l'optimisation des performances et des informations de niveau objet sur les E/S et les verrous.
   - Tout autre outil approprié.

2 Analysez les données pour comprendre le fonctionnement du système et détecter les problèmes de performances. Etablissez une liste de questions et essayez d'y répondre afin d'analyser l'environnement de votre Adaptive Server. Voici quelques questions qu'il est bon de se poser :

   - Quels sont les symptômes ?
   - Quels sont les composants du modèle de système concernés ?
   - Tous les utilisateurs sont-ils touchés, ou seulement les utilisateurs de certaines applications ?
   - S'agit-il d'un problème récurrent ou occasionnel ?

3 Définissez les objectifs de performances requis par le système :

   - Quelle est la fréquence d'exécution de la requête ?
   - Quel est le temps de réponse visé ?

4 Définissez l'environnement Adaptive Server : déterminez la configuration et les limites de chaque couche.

5 Analysez la conception de l'application : en examinant notamment les tables, les index, les transactions.

6 Formulez une hypothèse quant aux causes possibles du problème et aux solutions envisageables, en prenant comme référence les données de performances.

7 Testez cette hypothèse en mettant en œuvre les solutions à partir de la dernière étape :

- Réglez les paramètres de configuration.
- Procédez à une nouvelle conception des tables.
- Ajoutez des ressources mémoire ou modifiez l'allocation de la mémoire disponible.

8 Utilisez les tests de l'étape 1 pour mesurer les effets de l'optimisation. En général, vous devrez vous y prendre à plusieurs reprises avant d'atteindre les objectifs d'optimisation fixés.

   Si la procédure de l'étape 7 ne permet pas d'atteindre les objectifs définis à l'étape 3 ou si les réglages effectués génèrent de nouveaux problèmes, reprenez l'analyse à partir de l'étape 2. Vous serez peut-être amené à revoir les objectifs de performances requis par le système.

9 Si le test donne des résultats satisfaisants, appliquez la solution à votre environnement de développement.

Formes normales

Generalement, plusieurs techniques sont utilisées pour réorganiser une base de données de façon à réduire sa taille et éviter des incohérences et redondances, par exemple les formes normales.

Le fait d'utiliser les différents niveaux de formes normales permet d'organiser les informations de manière à faciliter une maintenance, un stockage et une mise à jour efficace. Ceci simplifie la gestion des requêtes et des mises à jour, y compris la sécurité et l'intégrité de la base de données. Toutefois, cette normalisation crée généralement un plus grand nombre de tables, ce qui peut, à son tour, augmenter la taille de la base de données.

Les administrateurs des bases de données doivent décider des différentes techniques à utiliser pour leur environnement.

Pour vous guider la définition des bases de données, consultez le document *Manuel de référence Adaptive Server*.

Verrouillage

Pour protéger les tables, les pages ou les lignes de données utilisées par des transactions actives, Adaptive Server les verrouille. Cette fonctionnalité est indispensable en environnement multi-utilisateur où les accès simultanés aux données sont fréquents.

Toutefois, le verrouillage réduit les performances lorsque les verrous posés par un processus interdisent à un autre processus d'accéder aux données. Ce dernier est alors mis en veille en attendant que le verrou qui le bloque soit libéré. On parle dans ce cas de *conflit de verrouillage*.

Par ailleurs, les interblocages posent des problèmes de performances plus graves. Un **interblocage** se produit lorsque deux processus ont verrouillé une page ou une table et que chacun tente de poser un verrou sur l'objet verrouillé par l'autre. La transaction du processus avec le temps CPU le plus faible est interrompue et toutes les opérations qu'elle a effectuées sont annulées.

Pour limiter les conflits de verrouillage et éviter les interblocages, il est indispensable de connaître les types de verrou d'Adaptive Server et de comprendre leur fonctionnement.

Pour plus d'informations sur le verrouillage, reportez-vous au document *Guide d'administration système*.

Le verrouillage dans un but de performances est décrit au chapitre 10, "Configuration et optimisation des verrouillages", au chapitre 11, "Utilisation des commandes de verrouillage", et au chapitre 12, "Informations sur les verrous".

Considérations spéciales

Les bases de données sont allouées parmi les devices en fragments appelés "sections de disque", où chaque section de disque est representée par une entrée dans master.dbo.sysusages. Chaque section de disque :

- représente un fragment contigu d'un device et peut avoir la taille maximale du device ;
- est un multiple pair de 256 pages logiques.

Un device peut être divisé entre un grand nombre de bases de données différentes. Beaucoup de fragments d'un device peuvent être incorporés dans une même base de données sous forme de sections de disque différentes.

Il n'existe pas de limite en pratique au nombre de sections de disque dans une base de données, à part le fait que la mémoire configurée pour Adaptive Server doit être suffisante pour contenir la représentation en mémoire correspondante.

Etant donné que les sections de disque sont des multiples de 256 pages logiques, certaines parties des devices ayant des tailles non communes peuvent rester inutilisées. Par exemple, si un device a 83 Mo alors que le serveur utilise des pages de 16 ko, 256 pages logiques occupent 256 * 16 ko = 4 Mo. Les 3 Mo finaux de ce dispositif ne seront utilisés par aucune base de données car c'est un espace insuffisant pour contenir un groupe de 256 pages logiques.

Le device master réserve ses premiers 8 ko en tant que zone de configuration. Par conséquent, pour éviter de gâcher de l'espace, un device master de taille correcte doit comprendre un nombre pair de 256 pages logiques *plus* 8 ko.

CHAPITRE 2

Réseaux et performances

Ce chapitre décrit le rôle joué par le réseau au niveau des performances des applications utilisant Adaptive Server.

Sujet	Page
Introduction	15
Problèmes potentiels de performances	16
Utilisation du réseau par Adaptive Server	17
Modification de la taille d'un paquet réseau	18
Techniques de réduction du trafic réseau fournies par le serveur	21
Incidence d'autres activités du serveur	22

Introduction

Généralement, l'administrateur système est le premier à identifier un problème de réseau ou de performances, dont voici des exemple :

- Les temps de réponse des processus varient de façon importante, sans raison apparente.
- Les requêtes renvoyant un grand nombre de lignes prennent plus de temps que prévu.
- Les traitements du système d'exploitation sont ralentis pendant les temps normaux de traitement Adaptive Server.
- Les traitements Adaptive Server sont ralentis pendant certaines périodes de traitement du système d'exploitation.
- Il semble qu'un processus client en particulier ralentisse tous les autres processus.

Problèmes potentiels de performances

Voici quelques-uns des problèmes sous-jacents pouvant être provoqués par les réseaux :

- Adaptive Server utilise mal les services réseau.
- Les limites physiques du réseau ont été atteintes.
- Les processus extraient des valeurs de données non nécessaires, augmentant inutilement le trafic réseau.
- Les processus ouvrent et ferment trop souvent des sessions, ce qui accroît la charge du réseau.
- Les processus soumettent régulièrement la même transaction SQL, ce qui entraîne un trafic réseau excessif et redondant.
- La mémoire réseau allouée à Adaptive Server est insuffisante.
- Adaptive ServerLa taille des paquets réseau d'Adaptive Server est insuffisante pour supporter le type de traitement requis par certains clients.

Questions fondamentales sur les performances réseau

Lorsque vous examinez des problèmes qui semblent liés au réseau, posez-vous les questions suivantes :

- Quels processus extraient généralement un volume élevé de données ?
- Y a-t-il souvent des erreurs réseau ?
- Quelles sont les performances globales du réseau ?
- Quel est le total des transactions effectuées à l'aide de SQL et des procédures stockées ?
- Le protocole de "commit à deux phases" est-il utilisé par un grand nombre de processus ?
- Des services de réplication sont-ils assurés sur le réseau ?
- Quel pourcentage du réseau le système d'exploitation utilise-t-il ?

Récapitulatif des techniques

Lorsque vous avez rassemblé les données voulues, vous pouvez utiliser plusieurs techniques susceptibles d'améliorer les performances du réseau. Ces techniques sont les suivantes :

- utilisation de petits paquets pour la plupart des activités de base de données ;
- utilisation d'une plus grande taille de paquet pour des tâches effectuant d'importants transferts de données ;
- utilisation de procédures stockées pour la réduction du trafic total ;
- filtrage de données pour éviter les transferts importants ;
- séparation des gros utilisateurs réseau et des utilisateurs ordinaires ;
- utilisation des procédures de contrôle client pour les cas spéciaux.

Utilisation de *sp_sysmon* lors de la modification de la configuration réseau

Utilisez sp_sysmon lorsque vous modifiez la configuration réseau pour connaître l'incidence des modifications sur les performances. Utilisez Adaptive Server Monitor pour localiser un conflit réseau relatif à un objet de base de données.

Pour plus d'informations sur l'utilisation de sp_sysmon, reportez-vous au document *Performances et optimisation : Outils de contrôle et d'analyse des performances*.

Utilisation du réseau par Adaptive Server

Toutes les communications réseau client/serveur s'effectuent au moyen de paquets sur un réseau. Les paquets comportent un en-tête, des informations relatives à l'acheminement des données et les données proprement dites.

Adaptive Server a été l'un des premiers systèmes de base de données conçu sur une architecture réseau de type client/serveur. Les clients ouvrent une session sur le serveur. La connexion envoie les demandes du client et les réponses du serveur. Les applications peuvent ouvrir simultanément autant de sessions que nécessaires pour exécuter la tâche requise.

Le protocole utilisé entre le client et le serveur, appelé Tabular Data Stream™ (TDS), constitue la base de communication de tous les produits Sybase.

Modification de la taille d'un paquet réseau

Par défaut, toutes les connexions à Adaptive Server utilisent une taille de paquet de 512 octets. Cela fonctionne bien pour les clients qui émettent des requêtes courtes et qui reçoivent de petits jeux de résultats. Toutefois, certaines applications ont avantage à utiliser une taille de paquet plus grande.

Généralement, une application transactionnelle (OLTP) envoie et reçoit un grand nombre de paquets contenant très peu de données. Une instruction d'insertion ou de mise à jour type peut ne représenter que 100 ou 200 octets. Une extraction de données, même dans le cas d'une jointure de plusieurs tables, peut ne récupérer qu'une ou deux lignes de données et ne pas remplir complètement un paquet. Les applications utilisant des procédures stockées et des curseurs adressent et reçoivent en général de petits paquets.

Les applications décisionnelles (DSS) comportent souvent de longs traitements en batch Transact-SQL et renvoient des jeux de résultats plus importants.

En environnement OLTP ou DSS, il est parfois préférable d'augmenter la taille des paquets pour certaines tâches spéciales telles que le chargement de données en batch ou le traitement de données de type text.

Pour plus d'informations sur la modification des paramètres de configuration suivants, reportez-vous au *Guide d'administration système* :

- default network packet size (taille par défaut des paquets réseau), si la plupart de vos applications effectuent de longues opérations de lecture et d'écriture ;
- max network packet size (taille maximale des paquets réseau) et additional network memory (mémoire réseau additionnelle), qui fournissent une quantité supplémentaire de mémoire pour les connexions de paquets de grande taille.

Seul l'administrateur système peut modifier ces paramètres de configuration.

Paquets de grande taille et paquets ayant une taille par défaut pour les connexions utilisateur

Adaptive Server réserve suffisamment d'espace pour permettre à toutes les connexions configurées par l'utilisateur de se connecter avec la taille de paquet par défaut. Les paquets réseau de grande taille ne peuvent pas utiliser cet espace. Les connexions qui utilisent la taille par défaut des paquets réseau ont toujours trois buffers réservés à la connexion.

Les connexions qui nécessitent des paquets de grande taille doivent acquérir l'espace correspondant à leurs buffers d'E/S réseau dans la zone additional network memory. S'il n'y a pas suffisamment d'espace dans cette zone pour allouer trois buffers en fonction des paquets de grande taille, les connexions utilisent la taille de paquet par défaut.

Importance du nombre de paquets

Généralement, le nombre de paquets transférés est plus important que la taille des paquets. Les performances "réseau" comprennent également le temps requis par le processeur et le système d'exploitation pour traiter un paquet réseau. Cet overhead par paquet réduit le plus les performances. Des paquets plus gros réduisent le coût total de l'overhead et augmentent le débit physique, à condition que le volume des données à envoyer soit suffisant.

L'augmentation de la taille des paquets peut accélérer le traitement des sources de transfert suivantes :

- Bulkcopy
- commandes readtext et writetext
- instructions select générant des jeux de résultats importants

Il existe toujours un point au-delà duquel l'augmentation de la taille des paquets n'améliore plus les performances et risque de les diminuer, parce que les paquets ne sont pas toujours pleins. Bien qu'il soit possible de calculer ce seuil, il est plus courant d'essayer plusieurs tailles de paquets pour le déterminer de manière empirique. Si vous effectuez ces expérimentations sur une période de test significative, vous pouvez déterminer une taille de paquets convenant à de nombreux processus. Toutefois, comme la taille des paquets peut être adaptée à chaque connexion, il est souvent intéressant de procéder à des tests par processus.

Les résultats peuvent varier de façon significative d'une application à l'autre. L'opération de bulkcopy peut très bien fonctionner pour une taille de paquet, tandis que d'importantes extractions de données de type image s'effectuent mieux avec une autre taille de paquet.

Si les tests montrent que certaines applications peuvent obtenir de meilleures performances avec des tailles de paquet importantes mais que la plupart des applications adressent et reçoivent des paquets de petite taille, les clients doivent utiliser la taille de paquet la plus grande.

Outils d'évaluation avec Adaptive Server

La procédure système sp_monitor fournit des informations sur l'activité des paquets. Elle n'affiche que les résultats associés aux paquets :

```
...
packets received packets sent packet err
---------------- ------------ ----------
10866(10580)     19991(19748) 0(0)
...
```

Vous pouvez utiliser également ces variables globales :

- @@*pack_sent* : nombre de paquets envoyés par Adaptive Server,
- @@*pack_received* : nombre de paquets reçus,
- @@*packet_errors* : nombre d'erreurs.

Ces instructions SQL montrent comment les compteurs peuvent être utilisés :

```
select "before" = @@pack_sent
select * from titles
select "after" = @@pack_sent
```

sp_monitor et les variables globales affichent l'activité des paquets pour tous les utilisateurs depuis le dernier redémarrage d'Adaptive Server.

Reportez-vous au document *Performances et optimisation : outils pour les statistiques de performances* pour toute information sur l'utilisation de sp_monitor et de ces variables globales.

Outils d'évaluation en dehors d'Adaptive Server

Les commandes du système d'exploitation fournissent également des informations sur les transferts de paquets. Reportez-vous à la documentation relative à votre système d'exploitation pour plus de détails sur ces commandes.

Techniques de réduction du trafic réseau fournies par le serveur

L'utilisation de procédures stockées, de vues et de triggers peut réduire le trafic réseau. Ces outils Transact-SQL peuvent socker de grandes quantités de codes sur le serveur, de façon à ce que seules les commandes courtes soient envoyées sur le réseau. Si vos applications envoient des batchs importants de commandes Transact-SQL à Adaptive Server, leur conversion en procédures stockées peut réduire le trafic réseau.

- procédure stockée

  Les applications qui envoient des traitements batch Transact-SQL de grande taille peuvent réduire la charge qu'elles font peser sur le réseau si le SQL est converti en procédures stockées. Les vues peuvent également contribuer à réduire le trafic réseau.

  Vous pouvez réduire l'overhead du réseau en désactivant les paquets "doneinproc".

  Pour plus d'informations à ce sujet, reportez-vous à la section "Réduction du surcoût des paquets", page 1077.

- Demande des seules informations nécessaires

  Les applications ne doivent demander que les lignes et les colonnes dont elles ont besoin, filtrant ainsi autant de données que possible au niveau du serveur afin de réduire le nombre de paquets devant être envoyés. Souvent, cela permet également de réduire le nombre d'E/S sur le disque.

- Transferts importants

  Les transferts importants provoquent simultanément la baisse du débit total et l'augmentation des temps de réponse moyens. Les transferts importants doivent être effectués, dans la mesure du possible, en dehors des heures de travail. Si vous effectuez couramment des transferts volumineux, envisagez l'acquisition d'un matériel réseau adapté à de tels transferts. Le tableau 2-1 présente les caractéristiques de certains types de réseau.

Tableau 2-1 : Options de réseau

Type	Caractéristiques
Anneau à jeton	Le système d'anneau à jeton donne de meilleurs résultats qu'Ethernet pour des utilisations intensives.
Fibre optique	La fibre optique fournit une très grande largeur de bande mais elle est généralement trop coûteuse pour pouvoir être utilisée à travers tout un réseau.
Réseau séparé	Un réseau séparé peut gérer le trafic réseau entre les stations de travail présentant le plus grand volume et Adaptive Server.

- Surcharge du réseau

  La fréquence de l'encombrement des réseaux augmente avec le nombre d'ordinateurs, d'imprimantes et de périphériques équipés pour les réseaux. Les gestionnaires de réseau détectent rarement un problème avant que les utilisateurs de bases de données ne contactent leur administrateur système.

  Coopérez avec vos gestionnaires de réseau local et fournissez-leur des estimations de vos besoins en matière de réseau lorsqu'ils envisagent d'ajouter des ressources. En outre, surveillez le réseau et tâchez de prévoir les problèmes résultant de l'ajout récent de matériels ou des besoins des applications.

Incidence d'autres activités du serveur

Vous devez être conscient de l'impact de certaines activités du serveur et de l'activité de maintenance sur le réseau, en particulier pour les activités suivantes :

- protocole "commit à deux phases",
- duplication,
- sauvegardes.

Ces activités, et plus particulièrement les réplications et le protocole "commit à deux phases", impliquent la communication par réseau. Les systèmes sur lequels ces activités sont fréquentes risquent de subir des problèmes réseau. Par conséquent, leur nombre doit être limité au strict minimum. Tâchez d'effectuer les sauvegardes pendant les périodes de faible activité réseau.

Un seul utilisateur et plusieurs utilisateurs

Vous devez tenir compte de la présence d'autres utilisateurs avant de tenter de résoudre un problème de base de données, en particulier si ces utilisateurs emploient le même réseau.

Comme la plupart des réseaux ne peuvent transférer qu'un seul paquet à la fois, de nombreux utilisateurs risquent d'être retardés lors d'un transfert important. Un tel retard peut accroître la durée de maintien des verrous, ce qui provoque encore plus de retards.

Lorsque les temps de réponse sont "anormalement" longs et que les tests normaux n'indiquent pas de problème, ceci peut être dû à la présence d'autres utilisateurs sur le réseau. Dans ce cas, demandez à l'utilisateur quand le processus a été exécuté, si le système d'exploitation était lent, si d'autres utilisateurs effectuaient également des transferts importants, etc.

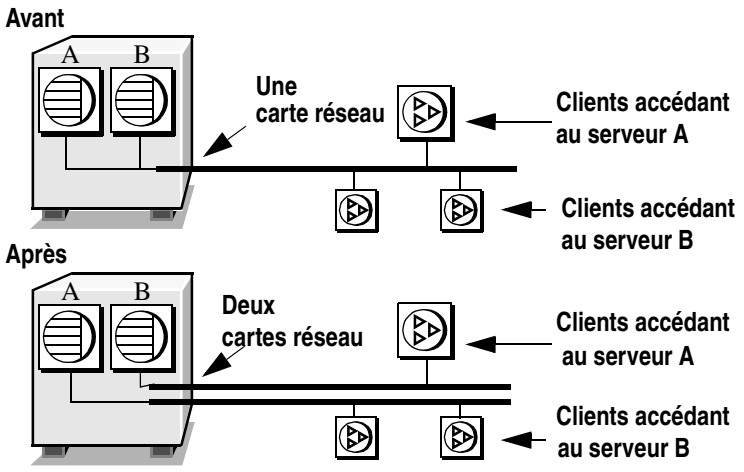
De manière générale, tenez compte des effets liés à la présence de plusieurs utilisateurs, tels que le retard causé par la transaction longue, avant d'approfondir la recherche dans le système de base de données pour résoudre un problème de temps de réponse anormal.

Amélioration des performances réseau

Séparation des utilisateurs intensifs d'un réseau

Séparez les utilisateurs réseau intensifs des utilisateurs réseau ordinaires en les plaçant sur un réseau distinct, comme représenté à la figure 2-1.

Figure 2-1 : Séparation des utilisateurs intensifs d'un réseau



Dans la partie "Avant" de l'illustration, les clients accédant à deux Adaptive Server utilisent la même carte réseau. Les clients accédant aux serveurs A et B doivent entrer en concurrence sur le réseau jusqu'au niveau de la carte réseau.

Dans la partie "Après" de l'illustration, les clients qui accèdent au serveur A utilisent une carte réseau et les clients accédant au serveur B en utilisent une autre.

Définition de *tcp no delay* sur les réseaux TCP

Par défaut, le paramètre de configuration tcp no delay est désactivé, ce qui signifie que le réseau traite les paquets en batch. De cette manière, l'envoi sur le réseau de paquets presque complets est un peu retardé.

Cette technique améliore les performances réseau en environnement d'émulation de terminal, mais ralentit celles des applications d'Adaptive Server qui envoient et reçoivent de petits batchs. Pour désactiver le traitement des paquets en batch, l'administrateur système peut attribuer la valeur 1 au paramètre de configuration tcp no delay.

Configuration de plusieurs récepteurs de réseau

Utilisez deux récepteurs (ou plus) pour chaque Adaptive Server. Pour associer le logiciel frontal à l'un des ports réseau configurés, il suffit de définir la variable d'environnement DSQUERY.

L'utilisation de plusieurs ports réseau répartit la charge réseau et élimine ou réduit les goulets d'étranglement tout en augmentant le débit d'Adaptive Server.

Pour plus d'informations sur la consignation de plusieurs récepteurs de réseau, reportez-vous au manuel de configuration Adaptive Server pour votre plate-forme.

CHAPITRE 3

Utilisation des moteurs et des CPU

L'architecture multitâche d'Adaptive Server est conçue pour offrir des performances élevées sur des systèmes monoprocesseur et multiprocesseur. Ce chapitre décrit comment Adaptive Server utilise les moteurs et les processeurs (CPU) pour satisfaire les requêtes clientes et gérer les opérations internes. Il présente l'utilisation des ressources CPU dans Adaptive Server, décrit le modèle SMP (multitraitement symétrique) d'Adaptive Server et illustre la planification des tâches à l'aide d'un exemple de traitement.

Ce chapitre fournit également des instructions sur la conception d'applications multiprocesseur et indique comment évaluer et optimiser les fonctions associées aux CPU et aux moteurs.

Sujet	Page
Concepts généraux	25
Modèle de traitement SMP (une CPU)	29
Modèle de processus SMP d'Adaptive Server	35
Amélioration de l'utilisation CPU par la tâche housekeeper	39
Evaluation de l'utilisation CPU	42
Activation de la spécialisation de moteur par CPU	45
Instructions pour la conception d'applications multiprocesseur	47

Concepts généraux

Cette section présente la façon dont Adaptive Server traite les requêtes clientes. Elle contient également des informations sur les tâches et procédures associées.

A l'instar d'un système d'exploitation, une base de données relationnelle doit être en mesure de répondre aux requêtes de nombreux utilisateurs concurrents. Adaptive Server repose sur une architecture multitâche à traitement simple qui lui permet de gérer des milliers de connexions clientes et de multiples requêtes clientes concurrentes sans surcharger le système d'exploitation.

Dans un système comptant plusieurs CPU, vous pouvez accroître les performances en configurant Adaptive Server pour qu'il s'exécute à l'aide de plusieurs *moteurs*. Chaque moteur représente un processus de système d'exploitation offrant des performances élevées lorsque vous configurez un moteur par CPU.

Tous les moteurs fonctionnent d'égal à égal et communiquent par le biais de la mémoire partagée du fait qu'ils agissent sur des bases de données utilisateur communes et des structures internes, telles que les caches de données et les chaînes de verrous. Les moteurs d'Adaptive Server sont conçus pour satisfaire les requêtes clientes. Ils exécutent toutes les fonctions de base de données, notamment la recherche dans les caches de données, la soumission des demandes de lecture et d'écriture d'E/S disque, les demandes de pose et de libération de verrou, la mise à jour et la consignation.

Adaptive Server gère la répartition des ressources CPU entre les moteurs qui traitent les requêtes clientes. Il gère également les services système (tels que le verrouillage de base de données, les E/S disque et les E/S réseau) qui ont une incidence sur les ressources de traitement.

Traitement des requêtes clientes dans Adaptive Server

Adaptive Server crée une nouvelle *tâche client* pour chaque nouvelle connexion. Il répond à une demande client par les étapes suivantes :

1 Le programme client établit une connexion réseau par socket à Adaptive Server.

2 Adaptive Server attribue une tâche parmi la zone de tâches allouées au démarrage. La tâche est identifiée par un identificateur de processus d'Adaptive Server, ou spid, qui figure dans la table système sysprocesses.

3 Adaptive Server transmet le contexte de la requête cliente, notamment les informations telles que les autorisations et la base de données courante, à la tâche.

4 Adaptive Server analyse, optimise et compile la requête.

5 Si l'exécution des requêtes en parallèle est activée, Adaptive Server alloue des sous-tâches afin de faciliter cette exécution. Ces sous-tâches sont appelées *processus de travail* et sont expliquées à la section "Adaptive Server Modèle de processus de travail", page 551.

6 Adaptive Server exécute la tâche. Si la requête a été exécutée en parallèle, la tâche fusionne les résultats des sous-tâches.

7 La tâche renvoie les résultats au client, à l'aide de paquets TDS.

Pour chaque nouvelle connexion utilisateur, Adaptive Server alloue une zone de stockage de données individuelle, une pile dédiée et d'autres structures de données internes.

Il utilise la pile pour suivre l'état de chaque tâche cliente lors du traitement et les mécanismes de synchronisation tels que la mise en file d'attente, le verrouillage, les sémaphores et les verrous d'attente pour garantir qu'une seule tâche accède à un moment donné à toute structure de données modifiables. Ces mécanismes s'avèrent nécessaires car Adaptive Server traite simultanément de nombreuses requêtes. Sans ces mécanismes, si deux requêtes ou plus venaient à accéder aux mêmes données, l'intégrité des données serait sacrifiée.

Les structures de données requièrent des ressources mémoire et système minimales pour l'overhead de changement de contexte. Certaines de ces structures de données ont une orientation connexion et contiennent des informations statiques sur le client.

D'autres sont orientées commandes. Par exemple, lorsqu'un client envoie une commande à Adaptive Server, le plan de requête exécutable est stocké dans une structure de données interne.

Mise en oeuvre de tâche cliente

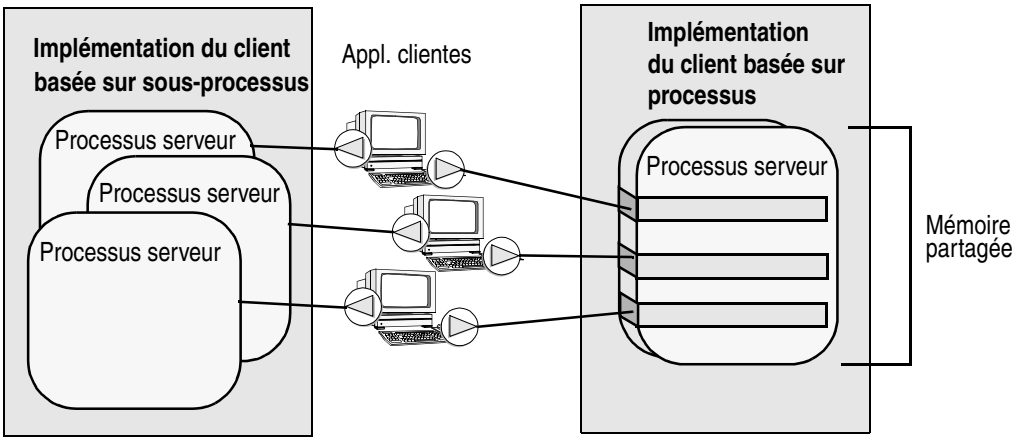
Les tâches clientes d'Adaptive Server sont mises en oeuvre comme des sous-processus ou "processus allégés" car ceux-ci n'utilisent qu'une petite partie des ressources requises par un processus.

Plusieurs processus s'exécutant simultanément requièrent davantage de mémoire et de temps CPU que des sous-processus multiples. En outre, les processus nécessitent plus de ressources de système d'exploitation pour changer de contexte (partage de temps) d'un processus au suivant.

L'utilisation des sous-processus élimine la plupart de l'overhead relatif à la pagination, au changement de contexte, au verrouillage et à d'autres fonctions du système d'exploitation associées à une architecture de processus par connexion. Une fois lancés, les sous-processus ne requièrent aucune ressource de système d'exploitation et peuvent partager beaucoup de structures et de ressources système.

La figure 3-1 illustre la différence entre les ressources système requises par des connexions clientes mises en oeuvre comme processus et des connexions clientes implémentées comme sous-processus. Les sous-processus existent et fonctionnent au sein d'une seule instance du processus de programme d'exécution et de son espace d'adresse dans la mémoire partagée.

Figure 3-1 : Architecture des processus et des sous-processus



Pour offrir à Adaptive Server la puissance de calcul maximum, seuls les processus essentiels non liés à Adaptive Server s'exécutent sur la machine de base de données.

Modèle de traitement SMP (une CPU)

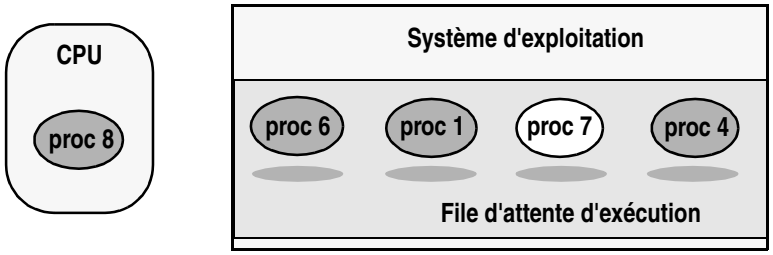
Dans un système monoprocesseur, Adaptive Server s'exécute comme un processus unique et partage le temps CPU avec d'autres processus, selon la planification du système d'exploitation. Cette section est une présentation générale de la façon dont Adaptive Server à une seule CPU utilise la CPU pour traiter les demandes client.

La section "Modèle de processus SMP d'Adaptive Server", page 35 développe la façon dont Adaptive Server à plusieurs CPUs traite les demandes client.

Planification des moteurs pour la CPU

La figure 3-2 montre une file d'attente d'exécution dans un environnement monoprocesseur dans lequel le processus 8 (proc 8) s'exécute sur la CPU pendant que les processus 6, 1, 7 et 4 attendent le temps CPU dans la file d'attente d'exécution du système d'exploitation. Le processus 7 est un processus d'Adaptive Server ; les autres processus sont des processus du système d'exploitation.

Figure 3-2 : Processus en attente dans la file d'attente d'exécution dans un environnement monoprocesseur

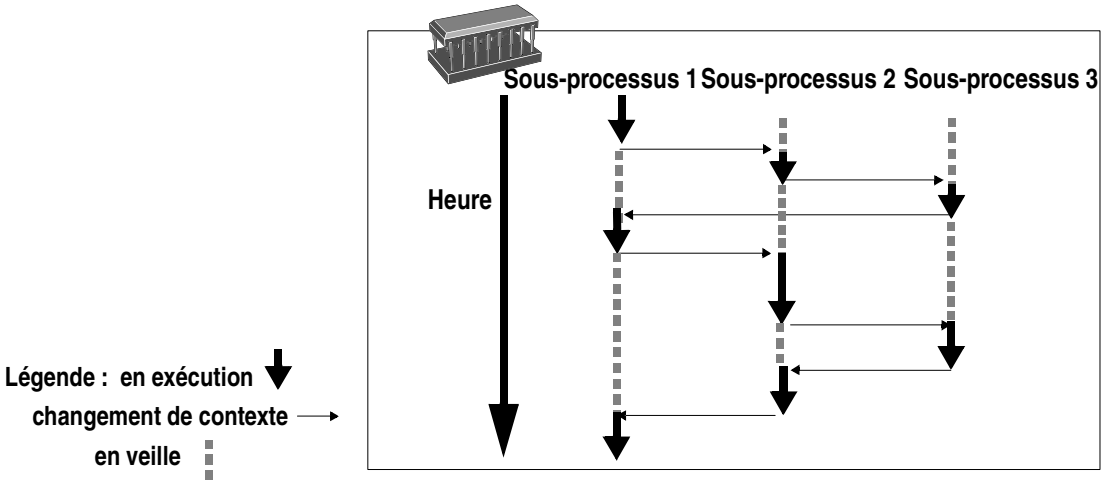


Dans un environnement multitâche, plusieurs processus ou sous-processus s'exécutent simultanément et partagent en alternance les ressources CPU.

La figure 3-3 illustre trois sous-processus dans un environnement multi-tâches. Les sous-processus sont représentés par des flèches noires épaisses pointant vers le bas. Ils se partagent une seule CPU en se connectant et en se déconnectant au moteur. Lorsque la flèche devient pleine (à proximité de la pointe), cela signifie que les sous-processus utilisent du temps CPU. S'ils sont dans la file d'attente avant d'être exécutés ou s'ils sont en veille jusqu'à l'obtention de ressources, ils sont représentés par des lignes en pointillé.

Il convient de noter qu'un seul processus s'exécute à un moment donné. Les autres sont en veille lors des diverses étapes du déroulement.

Figure 3-3 : Traitement multitâche



Planification des tâches pour le moteur

La figure 3-4 illustre les tâches (ou processus de production) en attente d'un moteur d'Adaptive Server dans un environnement monoprocesseur. Cette figure montre le passage d'Adaptive Server dans le contexte du système d'exploitation (voir figure 3-2, page 29) au traitement des tâches internes dans Adaptive Server. Adaptive Server, et non le système d'exploitation, planifie de façon dynamique les tâches clientes depuis la file d'attente d'exécution d'Adaptive Server pour le moteur. Lorsque le moteur achève le traitement d'une tâche, il exécute la tâche en tête de la file d'attente.

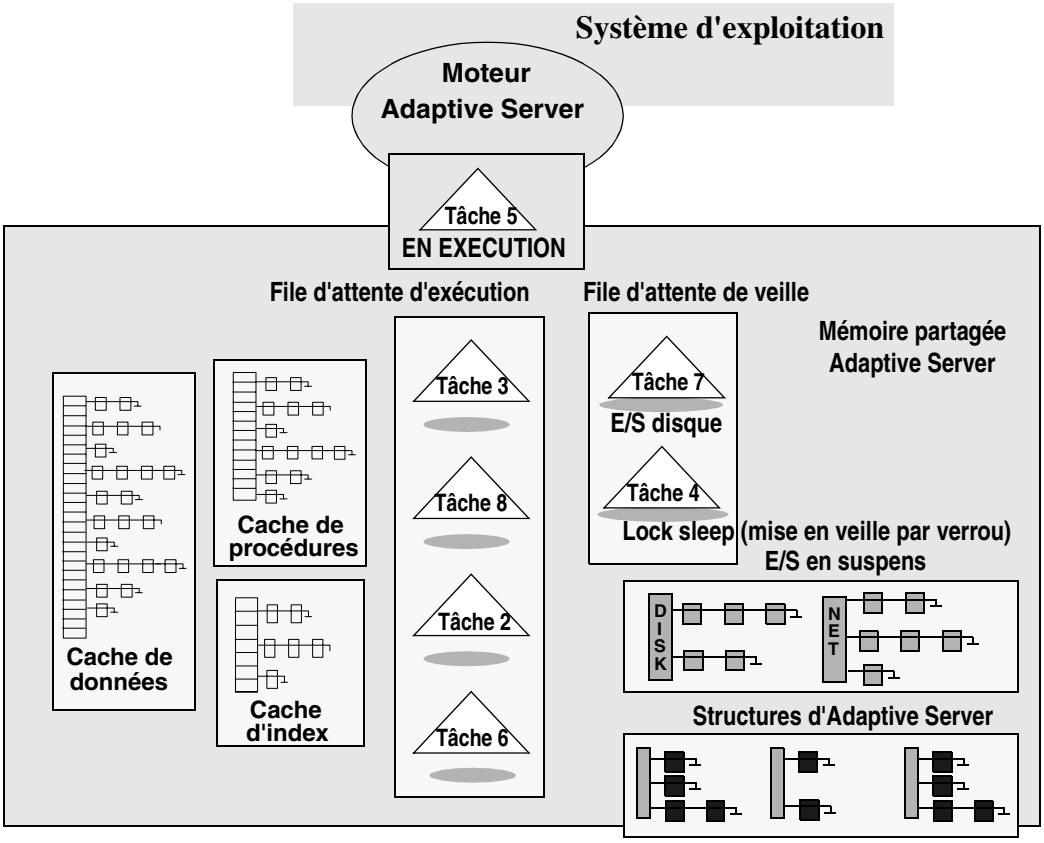
Une fois qu'une tâche s'exécute sur le moteur, celui-ci continue de la traiter jusqu'à ce qu'un des événements suivants se produise :

- La tâche a besoin d'une ressource telle qu'une page non disponible verrouillée par une autre tâche ou doit exécuter une tâche lente telle que des E/S disque ou des E/S réseau. La tâche se met en veille en attendant la ressource.
- La tâche s'exécute pour un temps défini et atteint un point d'arrêt. La tâche rend alors la main au moteur et le processus suivant dans la file d'attente s'exécute. La section "Planification du temps de traitement des tâches clientes", page 33 en détaille le fonctionnement.

Lorsque vous exécutez sp_who sur un système monoprocesseur à l'aide de plusieurs tâches actives, le résultat de la commande sp_who n'indique qu'une seule tâche "en cours d'exécution", la tâche sp_who elle-même. Toutes les autres tâches de la file d'attente sont "en attente d'exécution". Le résultat de sp_who indique également pourquoi les tâches sont en veille.

La figure 3-4 illustre aussi la file d'attente de veille avec deux tâches en veille ainsi que d'autres objets dans la mémoire partagée. Les tâches sont mises en veille lorsqu'elles attendent des ressources ou les résultats d'une opération d'E/S disque.

Figure 3-4 : Tâches en attente du moteur d'Adaptive Server



Planification de la tâche en exécution

Le planificateur gère le temps de traitement des tâches clientes et de la gestion interne des tâches.

Planification du temps de traitement des tâches clientes

Le paramètre de configuration time slice empêche les tâches en cours d'exécution de monopoliser les moteurs. Le scheduler autorise l'exécution d'une tâche sur un moteur d'Adaptive Server pour un temps maximum égal à la combinaison des valeurs time slice et cpu grace time avec les valeurs par défaut pour time slice (100 millisecondes, soit 1/10 de seconde ou une impulsion d'horloge) et cpu grace time (500 impulsions d'horloge, soit 50 secondes).

Le planificateur Adaptive Server ne force pas les tâches à libérer un moteur Adaptive Server. Les tâches rendent volontairement la main au moteur à un *point d'arrêt*, qui correspond au temps pendant lequel la tâche n'occupe pas une ressource vitale telle qu'un verrou d'attente.

Chaque fois que la tâche atteint un point d'arrêt, elle vérifie si la valeur attribuée au paramètre time slice a été dépassée. Si ce n'est pas le cas, la tâche continue à s'exécuter. Si le temps d'exécution dépasse la valeur de time slice, la tâche rend volontairement la main au moteur dans l'intervalle cpu grace time et la tâche suivante dans la file d'attente s'exécute.

La valeur par défaut du paramètre time slice correspond à 100 millisecondes. Rares sont les raisons de le modifier. La valeur par défaut du paramètre cpu grace time est 500 impulsions d'horloge. Si la valeur définie pour time slice est trop faible, il est possible qu'un moteur passe trop de temps entre une tâche et l'autre, ce qui tend à allonger le temps de réponse.

Si la valeur définie pour time slice est trop élevée, il est probable que les processus qui consomment beaucoup de temps CPU monopolisent la CPU, ce qui peut augmenter le temps de réponse des tâches courtes. Si vos applications rencontrent des erreurs de tranche de temps, ajustez le paramètre cpu grace time et **non** time slice.

Pour plus d'informations, reportez-vous au chapitre 4, "Répartition des ressources des moteurs".

Utilisez sp_sysmon pour déterminer le nombre de fois que les tâches rendent volontairement le contrôle.

Si vous souhaitez augmenter le temps d'exécution des applications fortement consommatrices de CPU sur un moteur avant qu'elles ne rendent le contrôle, vous pouvez affecter des attributs d'exécution à des logins, des applications ou des procédures stockées spécifiques.

Si la tâche doit rendre la main au moteur avant de satisfaire la requête cliente, elle se place à la fin de la file d'attente d'exécution sauf si aucune autre tâche n'y figure. Si aucune tâche ne figure dans la file d'attente d'exécution lorsque la tâche atteint le point d'arrêt dans le délai imparti (grace time), Adaptive Server octroie à la tâche un nouvel intervalle de traitement.

Si aucune autre tâche ne figure dans la file d'attente d'exécution et que le moteur dispose encore de temps CPU, Adaptive Server continue d'octroyer des intervalles de tranche de temps à la tâche jusqu'à ce qu'elle s'achève.

Normalement, les tâches rendent la main au moteur à des points d'arrêt avant l'achèvement de l'intervalle cpu grace time. Il est possible qu'une tâche n'atteigne pas de point d'arrêt et qu'elle dépasse l'intervalle time slice. Lorsque l'intervalle cpu grace time s'achève, Adaptive Server termine la tâche avec une erreur de tranche de temps. Si vous recevez une erreur de ce type, tentez de doubler la valeur du paramètre cpu grace time. Si le problème persiste, contactez le Support Technique de Sybase.

Maintien de la disponibilité CPU pendant le temps d'inactivité

Si Adaptive Server n'a pas de tâche à exécuter, il boucle (conserve la CPU) et recherche des tâches exécutables. Le paramètre de configuration runnable process search count contrôle le nombre de fois qu'Adaptive Server boucle.

Avec la valeur par défaut 2000, Adaptive Server boucle 2000 fois tout en recherchant des requêtes clientes entrantes, des E/S disque achevées et de nouvelles tâches dans la file d'attente d'exécution. Si aucune activité ne se produit pendant la durée de runnable process search count, Adaptive Server libère la CPU pour le système d'exploitation.

La valeur par défaut du paramètre runnable process search count fournit en général un temps de réponse correct, si le système d'exploitation n'exécute pas d'autres clients que ceux d'Adaptive Server.

Utilisez la procédure sp_sysmon pour déterminer la manière dont le paramètre runnable process search count conditionne l'utilisation des cycles CPU par Adaptive Server, les monopolisations de moteurs et les contrôles réseau bloquants.

Reportez-vous au document *Performances et optimisation : Outils pour le contrôle et l'analyse des performances* à l'aide de sp_sysmon.

Modèle de processus SMP d'Adaptive Server

La mise en oeuvre de SMP (Multitraitement symétrique) d'Adaptive Server confère les avantages des performances de l'architecture multitâche d'Adaptive Server aux systèmes multiprocesseur. Dans l'environnement SMP, plusieurs CPU coopèrent pour effectuer le travail plus vite qu'un seul processeur.

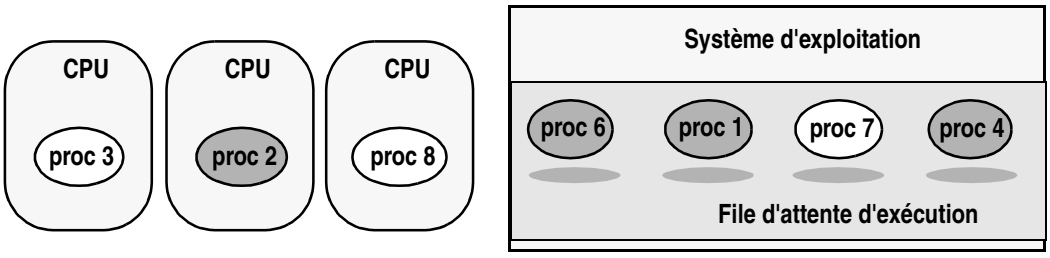
SMP est conçu pour des machines offrant les caractéristiques suivantes :

- un système d'exploitation avec multiprocesseurs symétriques,
- une mémoire partagée sur un bus commun,
- de deux à 128 processeurs,
- un débit très élevé.

Planification des moteurs pour les CPU

Dans un système équipé de plusieurs CPU, des processus multiples s'exécutent simultanément. La figure 3-5 représente les moteurs d'Adaptive Server par des ovales non grisés, en attente de temps de traitement sur l'une des trois CPU, dans la file d'attente d'exécution du système d'exploitation. Elle montre deux moteurs d'Adaptive Server, proc 3 et proc 8, qui sont traités simultanément.

Figure 3-5 : Processus en attente de plusieurs CPU dans la file d'attente d'exécution du SE



L'aspect *symétrique* de SMP est un manque de spécialisation entre les processus et les CPU : les processus ne sont pas liés à une CPU spécifique. Sans la spécialisation CPU, le système d'exploitation planifie des moteurs vers les CPU de la même façon qu'il lance les processus non Adaptive Server vers les CPU. Si un moteur d'Adaptive Server ne trouve pas de tâches exécutables, il peut libérer la CPU pour le système d'exploitation ou continuer à chercher une tâche à exécuter en bouclant un certain nombre de fois défini par le paramètre de configuration runnable process search count.

Planification des tâches d'Adaptive Server sur des moteurs

La planification des tâches d'Adaptive Server sur des moteurs dans l'environnement SMP est similaire à la planification des tâches en environnement monoprocesseur, telle qu'elle est décrite à la section "Planification des tâches pour le moteur", page 30. La différence est que, dans l'environnement SMP :

- Chaque moteur a une file d'exécution. Les tâches sont spécialisées par moteur. Lorsqu'une tâche s'exécute sur un moteur, un lien de spécialisation s'établit avec celui-ci. Si une tâche rend le contrôle puis se remet en file d'attente, elle sera placée dans la file d'exécution du moteur pour lequel elle a une spécialisation.
- Tous les moteurs peuvent traiter les tâches figurant dans la file d'exécution globale (à moins que la gestion de processus logique n'ait servi à associer la tâche à un moteur ou à un groupe de moteurs particulier).

Moteurs réseau multiples

Chaque moteur Adaptive Server gère les E/S réseau pour ses propres connexions. Les moteurs sont numérotés dans l'ordre, en commençant par le moteur 0.

Lorsqu'un utilisateur se connecte à Adaptive Server, la tâche est associée, selon un mode attribution circulaire, à l'un des moteurs qui deviendra dès lors son *moteur réseau*. Ce moteur gère la connexion de façon à établir la taille du paquet, la langue, le jeu de caractères et les autres paramètres de la connexion. Toutes les E/S réseau sont gérées par ce moteur jusqu'à ce que la tâche soit terminée.

Propriétés des tâches et files d'attente d'exécution

A certains moments, Adaptive Server augmente la priorité des tâches, notamment si elles bloquent une ressource importante ou si elles ont dû attendre une ressource. De plus, la gestion de processus logique vous permet d'affecter des priorités aux logins, aux procédures ou aux applications en utilisant sp_bindexeclass et des procédures système associées.

Pour plus d'informations sur l'optimisation des performances et les priorités des tâches, reportez-vous au chapitre 4, "Répartition des ressources des moteurs".

A chaque tâche est affectée une priorité ; celle-ci peut changer au cours de la durée de vie de la tâche. Lorsqu'un moteur cherche une tâche à exécuter, il explore en premier sa propre file d'attente prioritaire, puis la file d'exécution globale ayant la priorité la plus haute.

S'il ne trouve pas de tâches à priorité élevée, il cherche celles de priorité moyenne, puis basse. S'il ne trouve aucune tâche dans ses propres files d'attente d'exécution ou dans les files globales, il peut explorer les files d'exécution d'un autre moteur et s'approprier une tâche d'un autre moteur. Cette combinaison de priorités, de files locales et globales et la possibilité de transférer des tâches entre les moteurs permettent d'équilibrer la charge de travail.

Les tâches figurant dans la file d'attente globale ou dans la file d'un moteur sont toutes en attente d'exécution. Les résultats de la procédure système sp_who donne la liste des tâches "exécutable" lorsqu'elles sont dans une file d'attente d'exécution.

Exemple de traitement

Les étapes suivantes décrivent la planification d'une tâche dans un environnement SMP. Le cycle d'exécution pour des systèmes monoprocesseur est très similaire. Un système monoprocesseur gère le changement des tâches en mettant en veille les tâches en attente d'E/S réseau ou disque et en contrôlant en même temps les files d'attente.

1 Affectation d'un moteur réseau durant la connexion

   Lorsqu'une connexion est établie à Adaptive Server, elle est associée à un moteur qui traitera ses E/S réseau. Ce moteur gère ensuite la connexion.

Le moteur attribue une structure de tâche et définit la taille de paquet, la langue, le jeu de caractères, et les autres paramètres de la connexion. Une tâche est en veille lorsqu'elle attend que le client envoie une requête.

2 Vérification des requêtes clientes

Un autre moteur vérifie les requêtes clientes entrantes, une à chaque impulsion d'horloge.

Lorsque ce moteur trouve une commande (ou requête) provenant de la connexion d'une tâche, il active la tâche et la place à la fin de sa file d'attente d'exécution.

3 Réponse à une requête cliente

Lorsqu'une tâche se trouve en tête dans la file, le moteur effectue l'analyse, la compilation et commence à exécuter les phases définies dans le plan d'exécution correspondant.

4 Exécution des E/S réseau

Si la tâche doit accéder à une page verrouillée par un autre utilisateur, elle se met en veille en attendant que la page soit disponible. Au bout d'un certain temps, la tâche acquiert une priorité plus élevée et passe dans la file d'exécution globale afin qu'un moteur puisse l'exécuter.

5 Exécution des E/S réseau

Lorsque la tâche doit renvoyer des résultats à l'utilisateur, le moteur qui l'exécute émet une demande d'E/S réseau et place les tâches en veille sur une écriture réseau.

Le moteur effectue une vérification à chaque impulsion d'horloge pour déterminer si l'E/S réseau est terminée. Lorsque c'est le cas, la tâche est placée dans la file d'exécution du moteur avec lequel elle possède un lien de spécialisation, ou dans la file d'exécution globale.

Amélioration de l'utilisation CPU par la tâche housekeeper

Quand Adaptive Server n'a pas de tâche utilisateur à traiter, une tâche housekeeper commence automatiquement à écrire des buffers de pages de données modifiées sur le disque. Comme ces écritures sont effectuées pendant les cycles d'inactivité du serveur, elles sont appelées *écritures libres*. Elle se traduisent par une meilleure utilisation de la CPU et par une réduction du nombre de vidages de buffers lors du traitement des transactions. Elles réduisent également le nombre et la durée des pointes de point de reprise (nombre de fois où le processus de point de reprise provoque une hausse nette et brève des écritures sur disque).

Le housekeeper est la poubelle. Il nettoie les données qui ont été effacées logiquement et remet à zéro les lignes de façon à créer de l'espace dans les tables.

Effets pervers de la tâche housekeeper

Si la tâche housekeeper peut vider toutes les zones de buffer actives dans l'ensemble des caches configurés, elle active la tâche du point de reprise.

Cette tâche détermine si elle peut effectuer le point de reprise dans la base de données. Le cas échéant, elle écrit un enregistrement qui consigne le point de reprise et indique que toutes les pages de données modifiées ont été écrites sur le disque. Les points de reprise supplémentaires résultant du processus housekeeper peuvent accélérer la reprise de la base de données.

Dans des applications modifiant régulièrement la même page de base de données, la tâche housekeeper peut être à l'origine de certaines écritures de base de données qui ne s'avèrent pas nécessaires. Bien que ces écritures soient effectuées pendant les cycles d'inactivité du serveur, elles peuvent être inacceptables sur des systèmes dont les disques sont surchargés.

Configuration de la tâche housekeeper

Les administrateurs système peuvent utiliser le paramètre de configuration housekeeper free write percent pour contrôler les effets pervers de la tâche housekeeper. Ce paramètre spécifie le pourcentage maximum d'augmentation d'écritures de bases de données résultant des écritures effectuées par la tâche du gestionnaire de tâches. L'intervalle des valeurs se situe entre 0 et 100.

Par défaut, le paramètre housekeeper free write percent a la valeur 1. Cela permet à la tâche du gestionnaire de tâches de continuer à vider les buffers tant que l'augmentation d'écritures de bases de données ne dépasse pas 1 %. Grâce au paramètre configuré par défaut, le travail effectué par la tâche housekeeper optimise les performances et accélère la reprise sur la plupart des systèmes. Cependant, l'attribution d'une valeur trop élevée au paramètre housekeeper free write percent risque de provoquer une diminution des performances. Si vous voulez augmenter cette valeur, ne l'augmentez que de 1 ou 2 % à la fois.

L'option deviochar de dbcc tune contrôle la taille des batchs que le gestionnaire de tâches peut écrire sur le disque à un moment donné.

Pour plus d'informations, reportez-vous à la section "Augmentation de la limite de la taille des batch pour la tâche housekeeper", page 1067.

Modification du pourcentage d'augmentation des écritures

Utilisez la procédure système sp_configure pour modifier le pourcentage d'augmentation des écritures de base de données résultant du processus housekeeper, comme suit :

```
sp_configure "housekeeper free write percent", value
```

Par exemple, lancez la commande suivante pour interrompre l'exécution de la tâche du gestionnaire de tâches lorsque la fréquence des écritures de base de données dépasse la normale de 2 % :

```
sp_configure "housekeeper free write percent", 2
```

Désactivation de la tâche housekeeper

Vous pouvez désactiver la tâche housekeeper afin d'établir un environnement contrôlé dans lequel seules les tâches utilisateur spécifiées s'exécutent. Pour désactiver la tâche housekeeper, définissez la valeur du paramètre housekeeper free write percent à 0 (zéro) :

```
sp_configure "housekeeper free write percent", 0
```

Avertissement ! Outre le vidage des buffers, le processus housekeeper transfère régulièrement les statistiques dans les tables système. Ces statistiques sont utilisées pour l'optimisation des requêtes et, si elles sont erronées, elles peuvent réduire considérablement les performances. Ne définissez pas housekeeper free write percent à 0 sur un système où les commandes de modification de données risquent d'avoir une incidence sur le nombre de lignes et de pages dans les tables et les index.

Fonctionnement en continu de la tâche housekeeper

Pour permettre à la tâche housekeeper de fonctionner en continu, même en présence de cycles CPU d'inactivité et quel que soit le pourcentage d'augmentation des écritures de base de données, définissez la valeur du paramètre housekeeper free write percent à 100 :

```
sp_configure "housekeeper free write percent", 100
```

La section "Gestion de la restauration", page 1065 de sp_sysmon fournit des informations sur le point de reprise pour vous aider à déterminer l'efficacité du gestionnaire de tâches.

Evaluation de l'utilisation CPU

Cette section décrit comment évaluer l'utilisation CPU sur des machines monoprocesseur et multiprocesseur.

Machines monoprocesseur

Il n'existe aucune correspondance entre les données fournies par votre système d'exploitation sur l'utilisation CPU et les informations "Taux d'occupation CPU" internes à Adaptive Server. Il est normal pour un Adaptive Server d'indiquer une utilisation CPU très élevée lors de l'exécution d'une tâche liée aux E/S.

Un moteur de base de données multitâche n'est pas autorisé à se bloquer sur les E/S. Lors de l'exécution des E/S disque asynchrones, Adaptive Server gère d'autres tâches utilisateur en attente de traitement. S'il n'a aucune tâche à exécuter, il boucle en attendant l'achèvement des E/S disque asynchrones. Cette boucle d'attente de faible priorité peut se traduire par une utilisation CPU très élevée qui, compte tenu de la faible priorité, n'a guère d'incidence.

Utilisation de *sp_monitor* pour évaluer l'occupation de la CPU

Utilisez sp_monitor pour afficher, en pourcentage, le nombre de fois où Adaptive Server utilise la CPU pendant un intervalle de temps :

```
last_run                    current_run                 seconds
--------------------------- --------------------------  -----------
Jul 28 1999  5:25PM         Jul 28 1999  5:31PM         360

cpu_busy                io_busy                idle
----------------------- ---------------------- -----------------
5531(359)-99%           0(0)-0%                178302(0)-0%

packets_received        packets_sent           packet_errors
----------------------- ---------------------- ------------------
57650(3599)             60893(7252)            0(0)

total_read        total_write      total_errors    connections
----------------- ---------------- --------------- --------------
190284(14095)     160023(6396)     0(0)            178(1)
```

Pour plus d'informations sur sp_monitor, reportez-vous au document *Manuel de référence d'Adaptive Server*.

Utilisation de *sp_sysmon* pour l'évaluation de l'utilisation de la CPU

sp_sysmon fournit des informations plus détaillées que sp_monitor. La section "Kernel Utilization" des résultats de sp_sysmon affiche le taux d'occupation du moteur pendant l'exécution de l'intervalle échantillon. Le pourcentage de ce résultat se base sur le moment où la CPU a été allouée à Adaptive Server ; ce n'est pas un pourcentage de l'intervalle d'échantillon total.

La section "CPU Yields by Engine" affiche des informations sur le nombre de fois où le moteur rend la main au système d'exploitation pendant l'intervalle échantillon.

Pour plus d'informations sur sp_sysmon, reportez-vous au chapitre 38, "Contrôle des performances avec sp_sysmon".

Commandes du système d'exploitation et utilisation CPU

Les commandes du système d'exploitation permettant d'afficher l'utilisation CPU sont décrites dans les guides d'installation et de configuration d'Adaptive Server.

Si les outils de votre système d'exploitation indiquent que l'utilisation CPU dépasse 85 % la plupart du temps, envisagez un environnement multiprocesseur ou déchargez une partie des tâches sur un autre Adaptive Server.

Détermination de la configuration des moteurs supplémentaires

Pour déterminer si vous devez ajouter des moteurs supplémentaires, vous devez tenir compte des facteurs suivants :

- la charge des moteurs existants,
- les conflits de ressources tels que les verrous de tables, les disques et les verrous d'attente de caches,
- temps de réponse.

Si la charge des moteurs existants dépasse 80%, l'ajout d'un moteur pourrait améliorer le temps de réponse, sauf si le conflit de ressources est élevé ou si le moteur supplémentaire provoque le conflit.

Avant de configurer des moteurs supplémentaires, utilisez sp_sysmon pour établir une référence. Reportez-vous au résultat sp_sysmon des sections suivantes dans le chapitre 38, "Contrôle des performances avec sp_sysmon".

Plus spécifiquement, étudiez les lignes ou les sections figurant dans le résultat susceptibles d'indiquer des points de contention.

- "Conflits de verrouillage logique", page 992.
- "Conflits de verrouillage d'adresse", page 993.
- "Requêtes de sémaphores de l'ULC", page 1021.
- "Requêtes de sémaphores de journal", page 1021.
- "Allocations au journal de transactions", page 1023.
- "Récapitulatif des verrous", page 1039.
- "Conflits de verrouillage d'attente de hachage", page 1036.
- "E/S retardées par", page 1070.

Après avoir augmenté le nombre de moteurs, exécutez de nouveau sp_sysmon dans les mêmes conditions de charge et vérifiez la section "Engine Busy Utilization" du résultat ainsi que les éventuels points de contention répertoriés ci-dessus.

Mise hors ligne des moteurs

dbcc (engine) permet de mettre des moteurs hors ligne. Respectez la syntaxe suivant :

```
dbcc engine(offline, [enginenum])

dbcc engine("online")
```

Si *enginenum* n'est pas spécifié, le moteur qui a le numéro le plus élevé est mis hors ligne. Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Activation de la spécialisation de moteur par CPU

Par défaut, il n'existe aucune spécialisation entre les CPU et les moteurs dans Adaptive Server. En établissant la spécialisation de moteur par CPU, il est possible que vous notiez une légère amélioration des performances dans des environnements à débit élevé.

Tous les systèmes d'exploitation ne supportent pas la spécialisation CPU. La commande dbcc tune est ignorée sur des systèmes qui ne prennent pas en charge la spécialisation de moteur par CPU. La commande dbcc tune doit être lancée à chaque redémarrage d'Adaptive Server. Chaque fois que la spécialisation CPU est activée ou désactivée, Adaptive Server imprime un message dans le journal d'erreurs indiquant les numéros de moteurs et de CPU concernés :

```
Engine 1, cpu affinity set to cpu 4.
Engine 1, cpu affinity removed.
```

Respectez la syntaxe suivante :

dbcc tune(cpuaffinity, *cpu_initiale* [, on | off])

cpu_initiale spécifie la CPU à laquelle le moteur 0 doit être lié. Le moteur 1 est lié à la CPU numérotée (*cpu_initiale* + 1). La formule pour déterminer la liaison au moteur *n* est la suivante :

```
((cpu_initiale + n) % nombre_de_cpus
```

L'intervalle des numéros de CPU se situe entre 0 (zéro) et le nombre de CPU moins 1.

Sur une machine comptant 4 CPU (numérotées de 0 à 3) et un Adaptive Server équipé de 4 moteurs, la commande suivante :

```
dbcc tune(cpuaffinity, 2, "on")
```

La commande donne le résultat :

Moteur	CPU	
0	2	(numéro *cpu_initiale* spécifié)
1	3	
2	0	
3	1	

Sur la même machine, avec un Adaptive Server équipé de 3 moteurs, la même commande a pour résultat la spécialisation suivante :

Moteur	CPU
0	2
1	3
2	0

Dans cet exemple, la CPU 1 ne sera pas utilisée par Adaptive Server.

Pour désactiver la spécialisation CPU, utilisez –1 à la place de *cpu_initiale* et off dans la configuration, comme indiqué :

```
dbcc tune(cpuaffinity, -1, "off")
```

Vous pouvez activer la spécialisation CPU sans modifier la valeur de *cpu_initiale* en utilisant –1 et on dans la configuration, comme suit :

```
dbcc tune(cpuaffinity, -1, "on")
```

La valeur par défaut de *cpu_initiale* est 1 si la spécialisation CPU n'a pas préalablement été définie.

Pour spécifier une nouvelle valeur pour *cpu_initiale* sans modifier la configuration on/off, utilisez la commande suivante :

```
dbcc tune (cpuaffinity, cpu_initiale)
```

Si la spécialisation CPU est activée (on) et que la nouvelle valeur de *cpu_initiale* diffère de la précédente, Adaptive Server modifie la spécialisation de chaque moteur.

Si la spécialisation CPU est désactivée (off), Adaptive Server note la nouvelle valeur de *cpu_initiale*, et la nouvelle spécialisation devient effective à la prochaine activation de la spécialisation CPU.

Pour consulter la valeur courante et savoir si la spécialisation est activée, utilisez la commande suivante :

```
dbcc tune(cpuaffinity, -1)
```

Cette commande imprime uniquement les paramètres courants dans le journal d'erreurs et ne modifie pas la spécialisation ni les paramètres.

Instructions pour la conception d'applications multiprocesseur

Si vous déplacez des applications d'un environnement monoprocesseur vers un environnement SMP, vous devez prendre en compte les informations fournies dans cette section.

Compte tenu du débit accru des Adaptive Server multiprocesseur, il est probable que plusieurs processus tentent d'accéder simultanément à la même page de données. Il est particulièrement important de suivre les instructions de conception de base de données ci-après pour éviter le conflit. Les sections suivantes détaillent certaines instructions à prendre en compte pour la conception d'applications dans un environnement SMP.

- Index multiples

  Le débit accru d'un environnement SMP peut augmenter le conflit de verrous lors de la modification de tables présentant plusieurs index. Il est conseillé de ne créer que deux ou trois index sur des tables fréquemment modifiées.

  Pour plus d'informations sur les effets de la maintenance des index sur la performance, reportez-vous à la section "Gestion des index", page 1023.

- Gestion des disques

  Il est possible que la puissance de calcul supplémentaire de SMP augmente les demandes sur les disques. Par conséquent, il est recommandé de répartir les données sur plusieurs devices pour les bases de données très utilisées.

  Pour plus d'informations sur les résultats de sp_sysmon relatifs à l'utilisation des disques, reportez-vous à la section "Gestion des E/S disque", page 1068.

- Ajustement de fillfactor dans les commandes create index

  Il est possible que vous deviez ajuster le paramètre fillfactor dans les commandes create index. Parce que le débit est accru sur les multiprocesseurs, l'attribution d'une valeur inférieure à fillfactor peut temporairement réduire le conflit sur les pages de données et d'index.

- Longueur de la transaction

  Il est probable que les transactions incluant beaucoup d'instructions ou dont la durée d'exécution est longue augmentent le conflit de verrous. Essayez de réduire les transactions autant que possible et évitez de poser des verrous (surtout de type exclusif ou de mise à jour) pendant l'interaction utilisateur.

- Tables temporaires

  Les tables temporaires (tables figurant dans tempdb) ne provoquent pas de conflit car elles sont associées à des utilisateurs individuels et ne sont pas partagées. Toutefois, si plusieurs processus utilisateur ont recours à tempdb pour des objets temporaires, il peut exister certains conflits sur les tables système résidant dans tempdb.

  Pour plus d'informations sur les moyens de réduire les conflits, reportez-vous à la section "Tables temporaires et verrouillage", page 683.

CHAPITRE 4

Répartition des ressources des moteurs

Ce chapitre explique comment affecter des attributs d'exécution, comment Adaptive Server interprète les combinaisons d'attributs d'exécution et comment prévoir les incidences de l'affectation d'attributs d'exécution sur le système.

Pour bien comprendre ce chapitre, il est indispensable de comprendre comment Adaptive Server utilise les ressources de la CPU.

Pour plus d'informations à ce sujet, reportez-vous au chapitre 3, "Utilisation des moteurs et des CPU".

Sujet	Page
Algorithme de répartition optimale des ressources moteur	49
Gestion de la priorité d'accès aux ressources	57
Types de classe d'exécution	58
Définition des attributs des classes d'exécution	63
Règles de définition de la priorité et de la portée	69
Exemple illustrant les règles de priorité	75
Eléments à prendre en compte pour la répartition des ressources moteur	78

Algorithme de répartition optimale des ressources moteur

Cette section présente une méthode d'optimisation efficace au niveau des tâches.

Les interactions entre les objets d'exécution dans un environnement Adaptive Server sont complexes. En outre, les environnements sont tous différents : Chacun comprend sa propre combinaison d'applications clientes, de logins et de procédures stockées et se caractérise par l'interdépendance de ces entités.

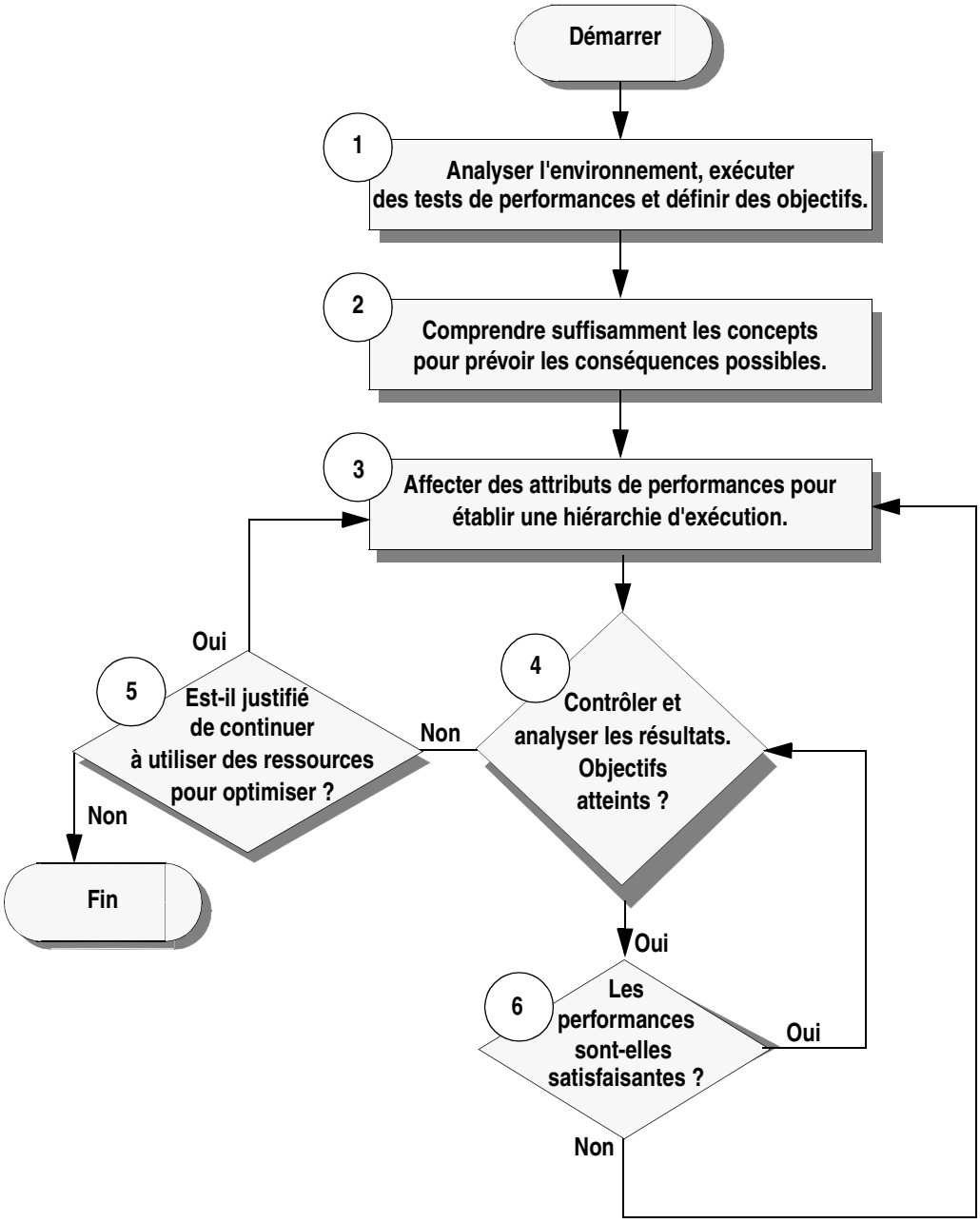
L'application d'une priorité d'exécution sans que l'environnement et les éventuelles conséquences n'aient été préalablement analysés peut produire des résultats inattendus, voire négatifs.

Supposons que vous ayez identifié un objet d'exécution prioritaire et que vous souhaitiez augmenter la valeur de ses attributs pour améliorer les performances de manière permanente ou pour une session déterminée. Si l'objet d'exécution accède aux mêmes tables qu'un ou plusieurs autres objets d'exécution, une priorité d'exécution plus élevée peut se traduire par une dégradation des performances, en raison des goulets d'étranglement qui existent entre les tâches à différents niveaux de priorité.

Etant donné la spécificité de chaque environnement Adaptive Server, il est impossible de mettre au point une procédure exhaustive permettant d'affecter à tous les systèmes une priorité d'exécution cohérente. Toutefois, cette section fournit des instructions avec des étapes progressives pour aborder les problèmes qui se rapportent à chacune d'entre elles.

Les étapes qui constituent l'affectation d'attributs d'exécution sont illustrées dans la figure 4-1. Chaque étape est reprise en détail dans les instructions qui suivent.

Figure 4-1 : Processus d'attribution de priorité d'exécution



Recommandations sur les algorithmes

1 Analysez l'environnement Adaptive Server.

Reportez-vous à la section "Analyse et planification d'un environnement", page 53 pour plus de détails.

- Analysez le comportement des objets d'exécution et classez-les de la manière la plus précise possible.
- Analysez les interdépendances et les interactions entre les objets d'exécution.
- Effectuez des tests comparatifs. Ils vous serviront de référence pour effectuer des comparaisons après avoir défini la priorité.
- Pensez à la manière dont vous pourriez répartir le traitement dans un environnement multiprocesseur.
- Identifiez les objets d'exécution prioritaires pour lesquels vous souhaitez améliorer les performances.
- Identifiez les objets non prioritaires pour lesquels vous pouvez réduire les performances.
- Définissez un ensemble d'objectifs quantifiables en termes de performances pour les objets d'exécution définis lors des deux étapes précédentes.

2 Analysez les conséquences de l'utilisation des classes d'exécution.

Reportez-vous à la section "Attributs des classes d'exécution", page 60 pour plus de détails.

- Analysez les concepts fondamentaux relatifs à l'affectation des classes d'exécution.
- Décidez si vous souhaitez créer une ou plusieurs classes d'exécution définies par l'utilisateur.
- Analysez les incidences des affectations de niveaux de classe différents.

3 Affectez les classes d'exécution et tous les attributs indépendants de spécialisation des moteurs.

4 Après avoir procédé à des affectations de priorité d'exécution, analysez l'environnement Adaptive Server actif.

Reportez-vous à la section "Analyse et optimisation des résultats", page 56 pour plus de détails.

- Lancez les tests comparatifs que vous avez utilisés à l'étape 1 et comparez les résultats.
- Si les résultats ne sont pas conformes à vos prévisions, analysez attentivement les interactions entre les objets d'exécution, comme indiqué à l'étape 1.
- Recherchez les éventuelles références que vous n'avez pas prises en compte.

5 Affinez les résultats en répétant les opérations décrites aux étapes 3 et 4, autant de fois que nécessaire.

6 Contrôlez régulièrement l'environnement.

Analyse et planification d'un environnement

Cette section reprend le contenu de l'étape 1 décrite dans l'"Algorithme de répartition optimale des ressources moteur", page 49.

L'analyse et la planification englobent les opérations suivantes :

- Analyse de l'environnement
- Exécution de tests comparatifs à utiliser comme référence
- Définition des objectifs, en termes de performances

Analyse

L'incidence de l'affectation des attributs d'exécution sur les performances d'un objet d'exécution dépend des caractéristiques et des interactions de l'objet avec d'autres objets figurant dans l'environnement Adaptive Server. Il est indispensable d'analyser et de connaître en détail l'environnement Adaptive Server pour pouvoir prendre les décisions appropriées et atteindre les objectifs.

Plan de démarrage

L'analyse comprend les deux phases suivantes :

- Phase 1 : analysez le comportement de chaque objet d'exécution.
- Phase 2 : utilisez les résultats de l'analyse pour prévoir les interactions entre des objets d'exécution dans le système Adaptive Server.

En premier lieu, dressez la liste des objets d'exécution pouvant être traités dans l'environnement. Ensuite, classez les objets d'exécution et les caractéristiques correspondantes. Classez-les par ordre d'importance. Attribuez à chacun l'une des définitions suivantes :

- objet d'exécution prioritaire qui requiert un temps de réponse optimal,
- objet d'exécution d'importance moyenne,
- objet d'exécution non prioritaire qui ne requiert pas un temps de réponse rapide.

Exemple : phase 1 – Analyse d'un objet d'exécution

En général, les applications sont dépendantes/indépendantes, consommatrices d'E/S ou consommatrices de CPU. Par exemple, essayez de classer chaque objet dans l'une de ces catégories. Il vous faudra probablement tenir compte d'éléments supplémentaires propres à l'environnement pour approfondir votre analyse.

Dépendance et indépendance

Deux objets ou plus traités sur le même moteur Adaptive Server sont *dépendants* lorsqu'ils utilisent des ressources communes.

Applications dépendantes	
Incidence de l'affectation d'attributs	L'affectation d'une priorité élevée à des applications dépendantes peut se traduire par une dégradation des performances.
Exemple	Une application non prioritaire est sur le point de libérer une ressource et se trouve bloquée par une application prioritaire dont le traitement commence. Si une deuxième application prioritaire tente d'accéder à la ressource bloquée, son traitement est également interrompu.

Lorsque les applications dans l'environnement Adaptive Server utilisent des ressources différentes, elles sont *indépendantes*.

Applications indépendantes	
Incidence de l'affectation d'attributs	Vous pouvez améliorer les performances en affectant des attributs d'exécution prioritaire à une application indépendante.
Exemple	Les opérations distinctes et simultanées sur des tables figurant dans des bases de données différentes sont indépendantes. Deux opérations sont également indépendantes si l'une est fortement consommatrice de calculs et l'autre fortement consommatrice d'E/S.

Objets d'exécution fortement consommateurs d'E/S et de CPU

Lorsqu'un objet d'exécution est fortement consommateur d'E/S, il est préférable de lui affecter des attributs EC1 et, dans le même temps, d'affecter des attributs EC3 aux objets d'exécution fortement consommateurs de calculs. Cette méthode est efficace, car les objets consommateurs d'E/S cessent d'utiliser le temps CPU tant que les E/S ne sont pas terminées.

Lorsque les tâches d'Adaptive Server fortement consommatrices d'E/S sont prioritaires, Adaptive Server fait en sorte qu'elles soient exécutables dès que les E/S sont terminées. Lorsque les E/S sont prioritaires, la CPU peut prendre correctement en charge les deux types d'application et de login.

Exemple : phase 2 – Analyse de l'environnement dans sa totalité

La phase 1 était consacrée aux objets d'exécution. Dans la phase 2, réfléchissez aux interactions entre les applications.

En général, une application a des comportements différents selon les cas ; elle peut être alternativement dépendante et indépendante, fortement consommatrice d'E/S et fortement consommatrice de CPU. Il est donc difficile de prévoir les interactions entre les applications, mais il est toutefois possible de dégager des tendances.

Vous devez trier les résultats de l'analyse, de manière à identifier correctement les objets, les uns par rapport aux autres. Par exemple, vous pouvez créer un tableau pour identifier les objets et leurs comportements.

Les outils de contrôle d'Adaptive Server permettent de bien comprendre les incidences des objets d'exécution sur l'environnement.

Tests comparatifs

Vous devez procéder à des tests comparatifs avant d'affecter des attributs d'exécution pour pouvoir utiliser les résultats comme référence après avoir effectué des réglages.

Il existe deux outils permettant de comprendre le fonctionnement du système et des applications :

- Adaptive Server Monitor fournit un ensemble complet de statistiques sur les performances. Présentées sous forme graphique, elles permettent d'identifier les problèmes en matière de performances.

- sp_sysmon est une procédure système permettant de contrôler les performances du système pendant une durée déterminée et d'imprimer un rapport contenant du texte ASCII.

  Pour plus d'informations sur l'utilisation de sp_sysmon, reportez-vous au document *Performances et optimisation : Outils de contrôle et d'analyse des performances.* Reportez-vous en particulier à la section "Gestion des applications", page 999.

Définition des objectifs

Définissez un ensemble d'objectifs quantifiables en termes de performances. Ces objectifs doivent correspondre à des chiffres spécifiques basés sur les résultats des tests comparatifs et sur vos prévisions en matière d'amélioration des performances. Vous devez vous référer à ces objectifs lorsque vous affectez des attributs d'exécution.

Analyse et optimisation des résultats

Vous trouverez ci-après quelques suggestions pour analyser l'environnement Adaptive Server après avoir défini l'ordre d'exécution :

1 Effectuez les mêmes tests comparatifs que précédemment avant d'affecter des attributs d'exécution et comparez les résultats avec les résultats de référence. Pour plus d'informations, reportez-vous à la section "Analyse et planification d'un environnement", page 53.

2 A l'aide d'Adaptive Server Monitor ou de sp_sysmon, assurez-vous que les ressources sont correctement réparties entre tous les moteurs disponibles. Vérifiez le contenu de la section "Kernel Utilization" ("Utilisation du kernel") dans le rapport de sp_sysmon.

  Voir également "Gestion des applications", page 999.

3 Si les résultats ne sont pas conformes à vos prévisions, analysez plus attentivement les interactions entre les objets d'exécution.

  Conformément à la section "Analyse et planification d'un environnement", page 53. recherchez les éventuelles prévisions inadéquates et les références que vous n'avez pas prises en compte.

4 Réglez les attributs de performances.

5 Affinez les résultats en répétant ces opérations autant de fois que nécessaire.

Contrôle régulier de l'environnement

Adaptive Server a plusieurs procédures enregistrées, par exemple sp_sysmon, optdiag, sp_spaceused, qui sont utilisées pour contrôler les performances et qui donneront des informations valides sur l'état du système.

Reportez-vous au document *Performances et optimisation : Outils de contrôle et d'analyse des performances* pour obtenir des informations sur le contrôle du système.

Gestion de la priorité d'accès aux ressources

La plupart des techniques d'optimisation des performances vous donnent le contrôle soit au niveau du système soit à un niveau spécifique de requête. Adaptive Server vous donne également le contrôle sur les performances de tâches en exécution simultanée.

A moins de disposer de ressources illimitées, la nécessité de contrôler au niveau des tâches est supérieure en exécution parallèle car la concurrence est plus intense pour des ressources limitées.

Vous pouvez utiliser des procédures système pour affecter des *attributs d'exécution* qui indiquent quelles sont les tâches pouvant bénéficier d'un accès prioritaire aux ressources. Le gestionnaire de processus logique (Logical Process Manager) utilise les attributs d'exécution lorsqu'il place des tâches dans l'une des trois files d'attente prioritaires.

Les attributs d'exécution ont également une influence sur la durée pendant laquelle un processus peut utiliser un moteur chaque fois qu'il s'exécute. En pratique, l'affectation d'attributs d'exécution permet d'indiquer à Adaptive Server comment il doit répartir les ressources des moteurs entre les applications clientes, les logins et les procédures stockées dans un environnement mixte.

Chaque application cliente ou chaque login peut lancer de nombreuses tâches Adaptive Server. Dans un environnement ne comportant qu'une seule application, vous pouvez répartir les ressources au niveau du login et de la tâche pour améliorer les performances des applications et des connexions ou sessions sélectionnées. Dans un environnement comportant plusieurs applications, la répartition des ressources peut permettre d'améliorer les performances des applications et des connexions ou sessions sélectionnées.

Avertissement ! Soyez prudent lorsque vous affectez des attributs d'exécution.

Toute modification arbitraire des attributs d'exécution d'une application cliente, d'un login ou d'une procédure stockée peut avoir des répercussions négatives sur les performances.

Types de classe d'exécution

Une *classe d'exécution* est une combinaison spécifique d'attributs d'exécution permettant de définir des valeurs pour la priorité des tâches, les tranches de temps et la spécialisation des tâches affectées aux moteurs. Vous pouvez lier une classe d'exécution à un ou plusieurs *objets d'exécution*, c'est-à-dire des applications clientes, des logins et des procédures stockées.

Il existe deux types de classe d'exécution : *prédéfinie* et *définie par l'utilisateur*. Adaptive Server intègre trois classes d'exécution prédéfinies. Vous pouvez créer des classes d'exécution définies par l'utilisateur en combinant des attributs d'exécution.

Classes d'exécution prédéfinies

Adaptive Server intègre les classes d'exécution prédéfinies suivantes :

- EC1 – comprend les attributs préférentiels.
- EC2 – comprend les valeurs moyennes d'attributs.
- EC3 – comprend les valeurs d'attributs peu utilisées.

Les objets associés à la classe EC2 ont une priorité d'accès moyenne aux ressources moteur. Si un objet d'exécution est associé à la classe EC1, Adaptive Server considère qu'il est prioritaire et tente de lui donner accès aux ressources moteur.

Les objets d'exécution associés à la classe EC3 ont la priorité la plus faible et n'ont accès aux ressources que lorsque le traitement des objets d'exécution EC1 et EC2 est terminé. Par défaut, les objets d'exécution sont dotés des attributs EC2.

Pour remplacer la classe d'exécution par défaut EC2 d'un objet, utilisez la procédure sp_bindexeclass, décrite à la section "Affectation des classes d'exécution", page 63.

Classe d'exécution définie par l'utilisateur

Outre les classes d'exécution prédéfinies, vous pouvez définir vos propres classes. Voici une liste des raisons justifiant ce choix :

- EC1, EC2, et EC3 ne prennent pas en charge toutes les combinaisons d'attributs qui pourraient être utiles.
- L'association d'objets d'exécution à un groupe de moteurs déterminé peut permettre d'améliorer les performances.

La procédure système sp_addexeclass permet de créer une classe d'exécution avec le nom et les attributs de votre choix. Par exemple, l'instruction suivante définit une classe d'exécution appelée DS avec une valeur de priorité faible (low) pour qu'elle puisse être exécutée sur n'importe quel moteur :

```
sp_addexeclass DS, LOW, 0, ANYENGINE
```

Vous pouvez associer une classe d'exécution définie par l'utilisateur à un objet d'exécution à l'aide de sp_bindexeclass, comme pour une classe d'exécution prédéfinie.

Attributs des classes d'exécution

Chaque classe d'exécution prédéfinie ou définie par l'utilisateur se compose d'une combinaison de trois attributs : priorité initiale, tranche de temps et spécialisation de moteur. Ces attributs définissent les caractéristiques des performances lors de l'exécution.

Les attributs des classes d'exécution prédéfinies EC1, EC2 et EC3 sont invariables, comme indiqué dans le tableau 4-1. Vous devez définir la combinaison des valeurs d'attributs pour les classes d'exécution lorsque vous les créez, à l'aide de la procédure système sp_addexeclass.

Tableau 4-1 : Combinaison d'attributs invariables pour des classes d'exécution prédéfinies

Niveau de classe d'exécution	Priorité initialeattribut*	Tranche de tempsattribut **	Spécialisation de moteur attribut ***
EC1	Elevée	Tranche de temps > t	Aucun
EC2	Moyenne	Tranche de temps = t	Aucun
EC3	Faible	Tranche de temps < t	Moteur ayant le numéro d'ID le plus élevé

Pour plus d'informations, reportez-vous aux sections "Priorité initiale", page 60, "Tranche de temps", page 61 et "Spécialisation des tâches affectées aux moteurs", page 62.

Par défaut, une tâche sur Adaptive Server fonctionne avec les mêmes attributs que ceux de la classe EC2 : sa priorité initiale est moyenne, l'intervalle de temps est défini à une impulsion et elle peut fonctionner sur tous les moteurs.

Priorité initiale

La priorité initiale est celle qui s'applique à une tâche lorsqu'elle est créée. Les valeurs possibles sont "élevée", "moyenne" et "faible". Il existe une file d'attente d'exécution pour chaque priorité et pour chaque moteur et la file globale d'exécution contient également une file d'attente pour chaque priorité.

Lorsqu'un moteur cherche une tâche à exécuter, il explore en premier sa propre file d'attente prioritaire, puis la file d'exécution globale ayant la priorité la plus haute, puis sa propre file d'attente de priorité moyenne et ainsi de suite. Le résultat est que les tâches exécutables des files d'attente à haute priorité sont planifiées plus rapidement dans les moteurs que les tâches des autres files.

Lors de l'exécution, Adaptive Server peut temporairement modifier la priorité d'une tâche. Dans ce cas, la nouvelle priorité peut être supérieure ou égale à la priorité initiale, mais elle ne peut en aucun cas être inférieure.

Lorsque vous créez une classe d'exécution définie par l'utilisateur, vous pouvez affecter à la tâche les valeurs élevée, moyenne ou basse.

Tranche de temps

Adaptive Server gère plusieurs processus simultanés en alternant de l'un à l'autre, c'est-à-dire en permettant l'exécution d'un processus pendant une durée déterminée (tranche de temps) avant de lancer le suivant.

Dans le tableau 4-1, page 60, l'attribut de la tranche de temps varie pour chaque classe d'exécution prédéfinie. EC1 a la tranche de temps la plus longue, EC3 la plus courte et EC2 a une tranche de temps comprise entre celles de EC1 et EC3.

Plus précisément, le temps pendant laquelle la tâche peut s'exécuter dépend de la valeur du paramètre de configuration time slice décrit dans "Planification du temps de traitement des tâches clientes", page 33. Avec les valeurs par défaut des paramètres de configuration, les objets d'exécution EC1 peuvent s'exécuter deux fois plus longtemps que la valeur spécifiée par la tranche de temps ; la tranche de temps d'un objet d'exécution EC2 est équivalente à la valeur configurée ; un objet d'exécution EC3 rend le contrôle au premier point d'arrêt qu'il rencontre et souvent il ne s'exécute pas pendant toute une tranche de temps.

Si les tâches ne rendent pas le contrôle au moteur pour d'autres raisons (par exemple la nécessité d'effectuer une E/S ou le fait d'être bloquée par un verrou), les clients EC1 s'exécutent plus longtemps et monopolisent le moteur plus souvent pendant la durée d'exécution d'une tâche. Les objets d'exécution EC3 s'exécutent pendant de très courtes périodes lorsqu'ils ont accès au moteur, de sorte qu'ils rendent plus souvent le contrôle durant une tâche. Les tâches EC2 se trouvent entre EC1 et EC3 en termes de temps d'exécution et de passage de contrôle.

Actuellement, vous ne pouvez pas affecter des tranches de temps lorsque vous créez des classes d'exécution avec la procédure sp_addexeclass. Adaptive Server affecte les valeurs de tranche de temps à EC1, EC2 et EC3 pour des tâches de priorité élevé, moyenne et basse, respectivement.

Spécialisation des tâches affectées aux moteurs

Dans un environnement supportant plusieurs moteurs, n'importe quel moteur peut traiter la prochaine tâche de la file d'exécution globale. L'attribut de spécialisation de moteur permet d'affecter une tâche à un moteur ou à un groupe de moteurs. Il existe deux méthodes permettant d'utiliser la spécialisation :

- Associez des objets d'exécution non prioritaires à un groupe de moteurs déterminé pour limiter leur traitement à un sous-ensemble du nombre total de moteurs. Cette méthode permet de restreindre la disponibilité du processeur pour ces objets. Les objets d'exécution dont la priorité est plus élevée peuvent alors être traités surAdaptive Server n'importe quel moteur avec des performances accrues, car ils ont accès à des ressources qui sont interdites aux objets dont la priorité est inférieure.
- Associez les objets d'exécution prioritaires à un groupe de moteurs déterminé auquel les objets non prioritaires n'ont pas accès. Cette méthode permet de garantir l'accès des objets prioritaires à un niveau de traitement déterminé.

EC1 et EC2 ne permettent pas de définir la spécialisation de moteur pour l'objet d'exécution ; cependant, EC3 définit la spécialisation du moteur d'Adaptive Server à l'aide du plus grand numéro de moteur de la configuration actuelle.

Avec la procédure sp_addengine, vous pouvez créer des groupes de moteurs et, avec sp_addexeclass, vous liez les objets d'exécution à un de ces groupes. Si vous ne souhaitez pas définir une spécialisation de moteur pour une classe d'exécution utilisateur, le paramètre de groupe de moteurs ANYENGINE permet à la tâche de s'exécuter sur n'importe quel moteur.

Remarque L'attribut de spécialisation de moteur n'est pas utilisé avec les procédures stockées.

Définition des attributs des classes d'exécution

Vous pouvez appliquer et gérer l'ordre d'exécution des applications clientes, des logins et des procédures stockées à l'aide des cinq procédures système décrites ci-dessous.

Tableau 4-2 : Procédures système pour gérer la priorité d'exécution des objets

Catégorie	Description	procédures système
Classe d'exécution définie par l'utilisateur	Créer et supprimer une classe définie par l'utilisateur dotée d'attributs personnalisés ou modifier les attributs d'une classe existante.	• sp_addexeclass • sp_dropexeclass
Liaison d'une classe d'exécution	Créer et supprimer des liaisons de classes prédéfinies ou définies par l'utilisateur avec des applications clientes et avec des logins.	• sp_bindexeclass • sp_unbindexeclass
Uniquement pour la session ("on the fly")	Définir et supprimer les attributs d'une session active.	• sp_setpsexe • sp_clearpsexe
Moteurs	Ajouter ou supprimer des moteurs dans des groupes de moteurs ; créer ou supprimer des groupes de moteurs.	• sp_addengine • sp_dropengine
Rapports	Rapports sur les affectations aux groupes de moteurs, sur les liaisons d'applications et sur les attributs des classes d'exécution.	• sp_showcontrolinfo • sp_showexeclass • sp_showpsexe

Pour plus d'informations sur les procédures système décrites dans le tableau 4-2, consultez le *Adaptive Server - Manuel de référence.*

Affectation des classes d'exécution

L'exemple ci-dessous décrit l'attribution d'une priorité d'accès aux ressources à un objet d'exécution en l'associant à la classe EC1. Dans ce cas, l'objet d'exécution est une combinaison d'application et de login.

La syntaxe de sp_bindexeclass est la suivante :

sp_bindexeclass *nom_objet*, *type_objet*,
portée, *nom_classe*

Supposons que le login "sa" doive obtenir des résultats provenant d'isql le plus vite possible. Vous pouvez indiquer à Adaptive Server qu'il doit donner la priorité d'exécution au login "sa" lorsqu'il exécute isql, en lançant la procédure sp_bindexeclass avec la classe d'exécution prioritaire EC1. Exemple :

```
sp_bindexeclass sa, LG, isql, EC1
```

Cette instruction indique que la tâche du login "sa" doit être exécutée avec les attributs de la classe EC1 chaque fois qu'un login (LG) appelé "sa" exécute l'application isql. Adaptive Server optimise le temps de réponse pour le login "sa" de la façon suivante :

- en plaçant la tâche dans la file d'attente d'exécution qui a une priorité élevée, afin qu'elle soit plus rapidement associée à un moteur,
- en autorisant une exécution qui dure plus longtemps que ne l'indique la tranche de temps, afin que la tâche effectue plus de travail pendant le temps qu'elle a accès au moteur.

Groupes de moteurs et spécialisation des tâches affectées aux moteurs

Les étapes ci-après illustrent l'utilisation de procédures système pour créer un groupe de moteurs associé à une classe d'exécution définie par l'utilisateur et la liaison de cette classe à des sessions utilisateur. Dans l'exemple suivant, le serveur est utilisé par le personnel du support technique, qui doit répondre aussi vite que possible aux clients, et par des gestionnaires qui généralement compilent des rapports et acceptent des temps de réponse plus longs.

Dans cet exemple, les procédures sp_addengine et sp_addexeclass sont utilisées.

Avec sp_addengine, vous pouvez créer des groupes de moteurs ou ajouter des moteurs aux groupes existants. Respectez la syntaxe suivante :

sp_addengine *numéro_moteur*, *groupe_moteur*

Définissez les attributs de la classe d'exécution utilisateur avec sp_addexeclass. Respectez la syntaxe suivante :

sp_addexeclass *nom_classe*, *priorité_initiale*,
tranche_temps, *groupe_moteur*

Les étapes de cette opération sont les suivantes :

1 Créez un groupe de moteurs avec sp_addengine. Cette instruction crée un groupe appelé DS_GROUP, comprenant le moteur 3 :

```
sp_addengine 3, DS_GROUP
```

Pour agrandir le groupe de telle sorte qu'il comprenne également les moteurs 4 et 5, vous devez exécuter sp_addengine deux fois :

```
sp_addengine 4, DS_GROUP
sp_addengine 5, DS_GROUP
```

2 Créez une classe d'exécution définie par l'utilisateur et associez-la au groupe de moteurs DS_GROUP en utilisant la procédure système sp_addexeclass.

   Cette instruction définit une nouvelle classe d'exécution appelée DS avec le priorité "LOW" et l'associe au groupe de moteurs DS_GROUP :

```
sp_addexeclass DS, LOW, 0, DS_GROUP
```

3 A l'aide de sp_bindexeclass, associez les objets d'exécution non prioritaires à cette nouvelle classe.

   Par exemple, vous pouvez lier les logins "mgr1", "mgr2" et "mgr3" à la classe d'exécution DS en utilisant sp_bindexeclass à trois reprises :

```
sp_bindexeclass mgr1, LG, NULL, DS
sp_bindexeclass mgr2, LG, NULL, DS
sp_bindexeclass mgr3, LG, NULL, DS
```

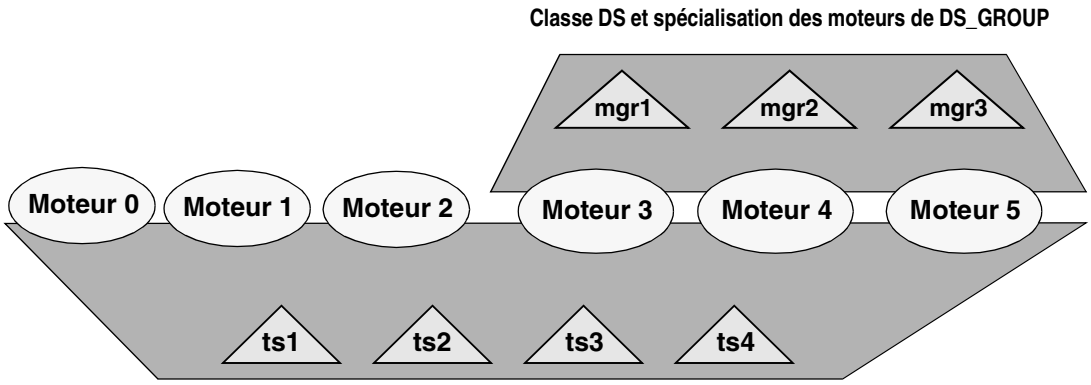
   Le second paramètre, "LG", indique que le premier paramètre est un nom de login. Le paramètre NULL indique que l'association est valide pour n'importe quelle application susceptible d'être exécutée par le login. Le paramètre DS indique que le login est associé à la classe d'exécution DS.

Le résultat de cet exemple est que le groupe du support technique (qui n'est pas lié à un groupe de moteurs) peut accéder à des ressources de traitement plus rapidement que les gestionnaires.

La figure 4-2 illustre les associations dans ce cas de figure :

- Les logins "mgr1", "mgr2" et "mgr3" sont associés au groupe de moteurs DS comprenant les moteurs 3, 4 et 5.
- Les logins "ts1", "ts2", "ts3", et "ts4" peuvent utiliser les six moteurs Adaptive Server.

Figure 4-2 : Exemple de spécialisation de moteur



Incidence des liaisons des classes d'exécution sur la programmation

Vous pouvez utiliser une gestion de processus logiques pour accroître la priorité de logins ou d'applications spécifiques, ou de logins spécifiques exécutant des applications spécifiques. Cet exemple examine :

- Une application order_entry, de type OLTP, qui enregistre les commandes client.
- Une application sales_report, qui prépare des rapports divers. Certains gestionnaires l'exécutent avec les valeurs par défaut, mais d'autres exécutent les rapports avec une priorité plus basse.
- D'autres utilisateurs, qui exécutent différentes applications avec les niveaux de priorité par défaut (sans affectation de classes d'exécution ou de priorités).

Liaisons d'une classe d'exécution

L'instruction suivante lie order_entry aux attributs EC1, en donnant la priorité la plus élevée aux tâches qui l'exécutent :

```
sp_bindexeclass order_entry, AP, NULL, EC1
```

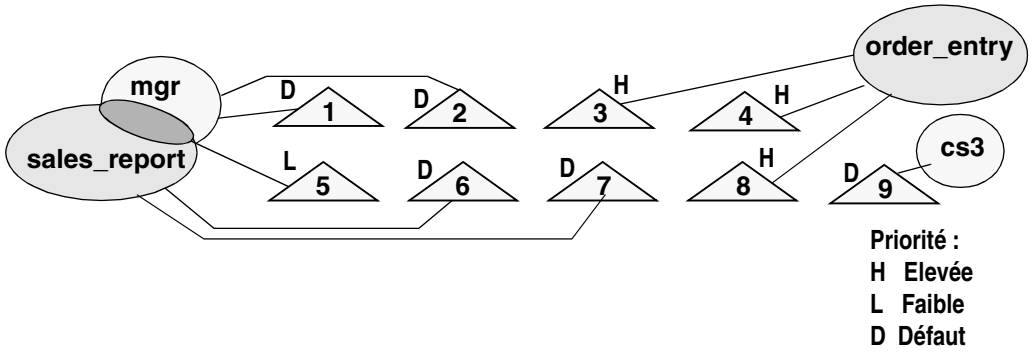
L'instruction sp_bindexeclass suivante spécifie EC3 lorsque "mgr" exécute l'application sales_report :

```
sp_bindexeclass mgr, LG, sales_report, EC3
```

Cette tâche ne peut s'exécuter que lorsque des tâches associées aux attributs de EC1 et EC2 sont inactives ou en veille.

La figure 4-3 montre quatre objets qui exécutent des tâches. Plusieurs utilisateurs exécutent les applications order_entry et sales_report. Deux autres logins sont actifs, "mgr" (connecté une fois en utilisant l'application sales_report et deux en utilisant isql) et "cs3" (qui n'utilise pas les applications affectées).

Figure 4-3 : Objets d'exécution et tâches correspondantes



Lorsque le login "mgr" utilise isql (tâches 1 et 2), la tâche s'exécute avec les attributs par défaut. Mais lorsque "mgr" utilise sales_report, la tâche s'exécute au niveau EC3. D'autres gestionnaires qui exécutent l'application sales_report (tâches 6 et 7) utilisent les attributs par défaut. Toutes les tâches qui exécutent order_entry disposent de la priorité élevée, avec les attributs EC1 (tâches 3, 4 et 8). "cs3" s'exécutent avec les attributs par défaut.

La spécialisation de moteur peut affecter la planification

Chaque classe d'exécution est associée à une priorité différente :

- Les tâches affectées à EC1 sont placées dans la file d'attente d'exécution dotée de la priorité élevée.

- Les tâches affectées à EC2 sont placées dans la file d'attente d'exécution dotée d'une priorité moyenne.
- Les tâches affectées à EC3 sont placées dans la file d'attente d'exécution dotée de la priorité la plus faible.

Lorsqu'un moteur cherche une tâche à exécuter, il vérifie d'abord dans sa file d'exécution prioritaire, puis dans la file d'exécution globale à priorité élevée. S'il ne trouve pas de tâches à priorité élevée, il cherche celles de priorité moyenne dans sa propre file d'attente d'exécution, puis dans la file d'exécution globale et enfin les tâches à basse priorité.

Que se passe-t-il lorsqu'une tâche est associée à un moteur déterminé ? Supposons que la tâche 7 de la figure 4-3, page 67, qui a une priorité élevée dans la file d'exécution globale, soit associée à une classe d'exécution définie par l'utilisateur avec une priorité élevée et une spécialisation attribuée au moteur 2. Celui-ci a actuellement des tâches à priorité élevée en attente et il exécute une autre tâche.

Si le moteur 1 n'a aucune tâche de priorité élevée en attente lorsqu'il a terminé de traiter la tâche 8 de la figure 4-3, page 67 il vérifie le contenu de la file globale, mais ne peut pas traiter la tâche 7 du fait qu'elle est liée à un moteur. Le moteur 1 cherche alors dans sa file d'attente les tâches de priorité moyenne et exécute la tâche 15. Bien que l'administrateur système ait attribué la classe d'exécution prioritaire EC1 à l'application, la spécialisation du moteur abaisse temporairement le niveau de priorité de la tâche 7 au-dessous de celui d'une tâche associée à la classe EC2.

Cette conséquence peut s'avérer indésirable, à moins que ce ne soit l'effet recherché. Il est possible de définir une spécialisation de moteur et des classes d'exécution de telle sorte que la priorité des tâches ne soit pas celle que vous souhaitez obtenir. Vous pouvez aussi réaliser des affectations de telle sorte que les tâches à priorité basse ne s'exécutent jamais ou, si elles s'exécutent, attendent très longtemps, ce qui constitue une autre raison pour planifier et tester soigneusement vos décisions lorsque vous spécifiez des classes d'exécution et des spécialisations de moteur.

Définition d'attributs pour une seule session

Vous pouvez modifier temporairement n'importe quelle valeur d'attribut pour une session active, à l'aide de sp_setpsexe.

La modification des attributs n'est autorisée que pour un spid déterminé et ne s'applique que pendant la session, indépendamment de la manière dont celle-ci s'achève. Le paramétrage des attributs à l'aide de sp_setpsexe ne modifie pas la définition des classes d'exécution d'un autre processus et ne s'applique pas lorsque vous souhaitez réutiliser le processus actif.

Pour effacer les attributs définis pour une session, utilisez sp_clearpsexe.

Informations

Adaptive Server enregistre les informations sur les affectations des classes d'exécution dans les tables système sysattributes et sysprocesses. De même, il supporte plusieurs procédures système permettant d'identifier les affectations.

La procédure système sp_showcontrolinfo permet d'afficher les informations relatives aux objets d'exécution liés à des classes d'exécution, aux moteurs Adaptive Server d'un groupe de moteurs et aux liaisons d'attributs au niveau de la session. Si vous ne spécifiez pas de paramètres, sp_showcontrolinfo affiche l'ensemble des liaisons et la composition de tous les groupes de moteurs.

sp_showexeclass affiche les valeurs d'attributs d'une ou de toutes les classes d'exécution.

Vous pouvez également utiliser sp_showpsexe pour afficher les attributs de tous les processus en cours d'exécution.

Règles de définition de la priorité et de la portée

La définition de l'ordre d'exécution applicable à deux objets ou plus peut s'avérer compliquée. Que se passe-t-il lorsque la combinaison d'objets d'exécution interdépendants et de divers attributs d'exécution ne permet pas d'obtenir un ordre d'exécution précis ?

Par exemple, une application cliente EC3 fait appel à une procédure EC1 enregistrée en mémoire. Les deux objets d'exécution prennent-ils les attributs EC3, les attributs EC1 ou les attributs EC2 ?

Il est important de bien comprendre comment Adaptive Server définit la priorité d'exécution pour optimiser l'affectation des classes d'exécution. Deux règles fondamentales, la *règle de priorité* et la *règle de portée* peuvent aider à déterminer l'ordre d'exécution.

Objets et classes d'exécution variés

Adaptive Server utilise des règles de *priorité* et de *portée* pour définir la configuration à appliquer en évitant les conflits.

Vous devez utiliser les règles dans l'ordre suivant :

1 Utilisez la règle de priorité lorsque le processus comprend plusieurs types d'objet d'exécution.

2 Utilisez la règle de portée lorsque plusieurs définitions de classe d'exécution sont associées au même objet d'exécution.

Règle de priorité

La règle de priorité permet de définir la priorité lorsqu'un objet d'exécution appartenant à une classe d'exécution fait appel à un objet d'exécution appartenant à une autre classe d'exécution.

La règle de priorité précise que la classe d'exécution d'une procédure enregistrée remplace celle d'un login, laquelle remplace à son tour celle d'une application cliente.

Lorsqu'une procédure stockée comporte plus de classes d'exécution prioritaires que le processus de l'application cliente qui la sollicite, le niveau de priorité de l'application cliente est temporairement élevé à celui de la procédure enregistrée pendant la durée d'exécution. Cette règle s'applique également aux procédures enregistrées imbriquées.

Remarque *Exception à la règle de priorité :* lorsqu'un objet d'exécution appelle une procédure stockée dotée d'une classe d'exécution dont la priorité est inférieure, la priorité de l'objet n'est pas abaissée au niveau inférieur.

Exemple de règle de priorité

Cet exemple illustre l'emploi de la règle de priorité. Supposons un login EC2, une application cliente EC3 et une procédure stockée EC1.

Les attributs du login remplacent ceux de l'application cliente et le login devient donc prioritaire. Si la procédure enregistrée est dotée d'une priorité initiale supérieure à celle du login, le niveau de la priorité initiale du processus Adaptive Server qui exécute la procédure enregistrée est temporairement augmenté pendant la durée d'exécution. La figure 4-4 montre l'application de la règle de priorité.

Figure 4-4 : Application de la règle de priorité

La procédure enregistrée s'exécute avec *EC2*

Que se passe-t-il si un login EC2 appelle une application cliente EC1 et si celle-ci appelle à son tour une procédure enregistrée EC3 ? La procédure enregistrée est alors exécutée avec les attributs de la classe EC2 car la classe d'exécution d'un login est toujours prioritaire sur celle d'une application cliente.

Règle de portée

Outre les attributs d'exécution d'un objet, vous pouvez définir sa portée lorsque vous utilisez la procédure système sp_bindexeclass. La portée définit les entités pour lesquelles les liaisons des classes d'exécution s'appliquent. Respectez la syntaxe suivante :

sp_bindexeclass *nom_objet*, *type_objet*,
portée, *nom_classe*

Par exemple, vous pouvez préciser qu'une application cliente isql doit être exécutée avec des attributs EC1, mais uniquement lorsqu'elle est exécutée par un login "sa". Cette instruction définit le login "sa" comme la portée de la liaison de EC1 à l'application cliente isql :

```
sp_bindexeclass isql, AP, sa, EC1
```

Inversement, vous pouvez préciser que le login "sa" doit être exécuté avec des attributs EC1, mais uniquement lorsqu'il exécute des applications clientes isql. Dans ce cas, la portée de la liaison d'EC1 au login "sa" est l'application cliente isql :

```
sp_bindexeclass sa, LG, isql, EC1
```

Les attributs d'exécution de l'objet s'appliquent à toutes les interactions lorsque la portée est NULL.

Lorsqu'une application cliente n'a pas de portée, les attributs d'exécution qui lui sont rattachés s'appliquent à n'importe quel login qui la sollicite.

Lorsqu'un login n'a pas de portée, les attributs s'appliquent au login pour tous les processus qu'il lance.

La commande suivante indique que les applications Transact-SQL sont exécutées avec des attributs EC3 pour tous les logins faisant appel à isql, sauf lorsqu'ils sont associés à une classe d'exécution de niveau supérieur :

```
sp_bindexeclass isql, AP, NULL, EC3
```

Si, en même temps que la règle de priorité, vous utilisez les liaisons ci-dessus qui octroie à l'utilisateur "sa" de l'application isql des attributs d'exécution EC1, une requête isql depuis le login "sa" sera exécutée avec des attributs EC1. Les autres processus exécutant des requêtes isql depuis des logins autres que "sa" seront exécutés avec des attributs EC3.

La règle de portée indique que si plusieurs niveaux de classe d'exécution sont affectés à une application cliente, un login ou une procédure enregistrée, celui dont la portée est la plus réduite est prioritaire. La règle de portée permet d'obtenir le même résultat si vous utilisez la commande suivante :

```
sp_bindexeclass isql, AP, sa, EC1
```

Résolution d'un conflit de priorité

Adaptive Server utilise les règles ci-après pour résoudre les conflits de priorité, lorsque plusieurs objets et plusieurs classes d'exécution ont la même portée.

- Les valeurs par défaut ci-dessous sont affectées aux objets d'exécution qui ne sont pas associés à une classe d'exécution déterminée :

Type d'entité	Nom de l'attribut	Valeur par défaut
Application cliente	Classe d'exécution	EC2
Login	Classe d'exécution	EC2
Procédure enregistrée	Classe d'exécution	EC2

- Lorsqu'une classe d'exécution est affectée à un objet d'exécution, elle a une priorité supérieure à celle qui est définie par défaut. (Un objet EC3 auquel une classe est affectée est prioritaire sur un objet EC2 sans affectation.)
- Si une application cliente et un login ont des classes d'exécution différentes, l'exécution du login est prioritaire sur celle de l'application cliente (règle de priorité).
- Si une procédure enregistrée et une application cliente ou un login sont associés à des classes d'exécution différentes, Adaptive Server utilise celui qui est associé à la classe d'exécution la plus élevée pour définir la priorité lorsqu'il exécute la procédure enregistrée (règle de priorité).
- Si plusieurs définitions s'appliquent au même objet d'exécution, celle dont la portée est la plus réduite a la priorité la plus élevée (règle de portée). Par exemple, la première instruction indique que le login "sa" exécutant isql est prioritaire sur les logins "sa" qui exécutent toutes les autres tâches :

```
sp_bindexeclass sa, LG, isql, EC1
sp_bindexeclass sa, LG, NULL, EC2
```

Exemples : définition de la priorité

Chacune des rangées du tableau 4-3 comprend une combinaison d'objets d'exécution et les attributs en conflit correspondants.

Les colonnes figurant dans la rubrique "Attributs des classes d'exécution" comportent les valeurs des classes d'exécution affectées à l'application "AP" appartenant au login "LG".

Les autres colonnes décrivent comment Adaptive Server résout les conflits de priorité.

Tableau 4-3 : Valeurs d'attributs conflictuelles et valeurs affectées par Adaptive Server

Attributs des classes d'exécution			Valeurs affectées par Adaptive Server		
Application (AP)	**Login (LG)**	**Procédure enregistrée (sp_ec)**	**Application**	**Priorité initiale du login**	**Priorité initiale de la procédure enregistrée**
EC1	EC2	EC1 (EC3)	EC2	Moyenne	Elevée (Moyenne)
EC1	EC3	EC1 (EC2)	EC3	Faible	Elevée (Moyenne)
EC2	EC1	EC2 (EC3)	EC1	Elevée	Elevée (Elevée)
EC2	EC3	EC1 (EC2)	EC3	Faible	Elevée (Moyenne)
EC3	EC1	EC2 (EC3)	EC1	Elevée	Elevée (Elevée)
EC3	EC2	EC1 (EC3)	EC2	Moyenne	Elevée (Moyenne)

Pour tester vos connaissances, entraînez-vous avec les colonnes "Valeurs affectées par Adaptive Server" dans le tableau 4-3. La description ci-dessous illustre le contenu de la première rangée :

- Colonne 1 : application cliente AP associée à la classe EC1.
- Colonne 2 : login "LG" associé à la classe EC2.
- Colonne 3 : procédure enregistrée sp_ec, associée à la classe EC1.

Au moment de l'exécution :

- Colonne 4 : la tâche associée au login "LG", qui exécute l'application cliente AP, utilise des attributs EC2, car la classe d'un login est toujours prioritaire sur celle d'une application (règle de priorité).
- Colonne 5 : la valeur figurant dans la colonne 5 suppose une priorité initiale moyenne pour le login.
- Colonne 6 : le niveau de priorité moyen de la procédure stockée sp_ec passe au niveau de priorité élevé (car il s'agit de EC1).

  Si on affecte la classe EC3 à la procédure enregistrée (comme indiqué entre parenthèses dans la colonne 3), celle-ci a alors une priorité d'exécution moyenne (comme indiqué entre parenthèses dans la colonne 6), car Adaptive Server utilise la priorité d'exécution la plus élevée de l'application ou du login et de la procédure enregistrée.

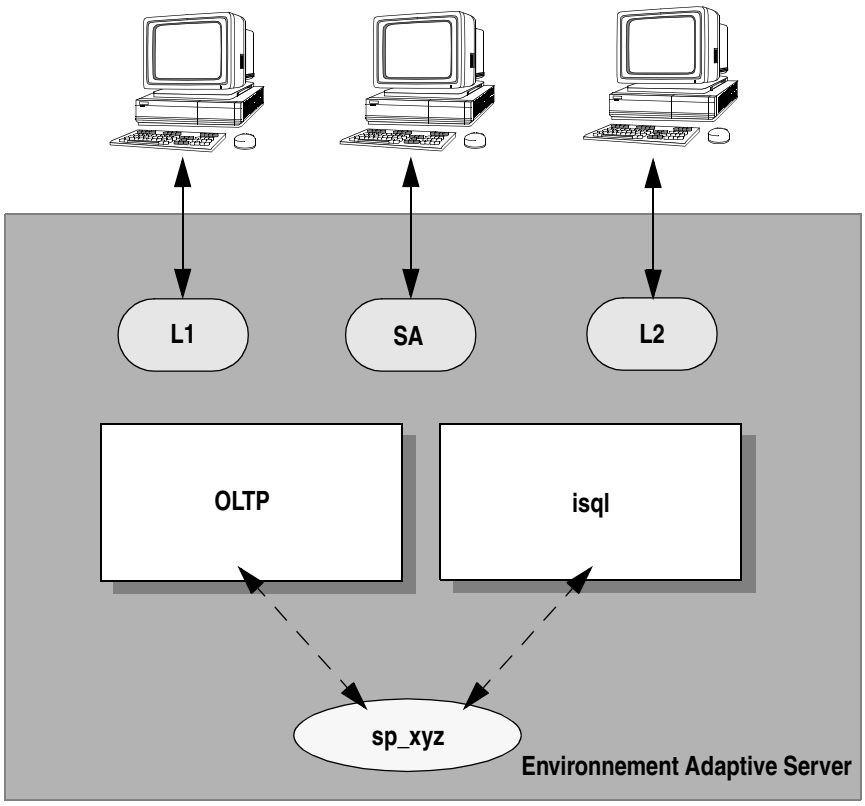
Exemple illustrant les règles de priorité

Cette section présente un exemple qui explique comment Adaptive Server interprète les attributs des classes d'exécution.

La figure 4-5 illustre deux applications clientes, OLTP et isql, ainsi que trois logins Adaptive Server, "L1", "sa" et "L2".

sp_xyz est la procédure enregistrée nécessaire pour lancer les applications OLTP et isql.

Figure 4-5 : Résolution de conflits



Le reste de cette section est consacré à une méthode permettant d'appliquer les instructions décrites dans la section Recommandations sur les algorithmes.

Planification

L'administrateur système effectue une analyse, comme indiqué aux étapes 1 et 2 de l'algorithme dans la section "Algorithme de répartition optimale des ressources moteur", page 49 et met au point le plan hiérarchique suivant :

- L'application OLTP est une application EC1 et l'application isql relève de la classe EC3.
- Le login "L1" peut exécuter des applications clientes différentes à des moments différents et n'a aucune obligation, en termes de performances.
- Le login "L2" correspond à un utilisateur non prioritaire et doit toujours être utilisé avec des performances inférieures.
- Le login "sa" doit toujours être exécuté comme un utilisateur prioritaire.
- La procédure enregistrée sp_xyz doit toujours être exécutée avec des performances élevées. L'application cliente isql peut exécuter la procédure enregistrée ; c'est pourquoi l'attribution de caractéristiques de performances élevées à sp_xyz peut permettre d'éviter les éventuels goulots d'étranglement dans le chemin d'accès à l'application cliente OLTP.

Le tableau 4-4 décrit l'analyse et indique la classe d'exécution que l'administrateur système doit affecter. Notez bien que la granularité de l'optimisation s'affine à mesure que vous descendez dans le tableau. Les applications ont la granularité ou la portée la plus grande. La procédure enregistrée a la granularité la plus fine ou la portée la plus réduite.

Tableau 4-4 : Exemple d'analyse d'un environnement Adaptive Server

Identificateur	Interactions et commentaires	Classe d'exécution
OLTP	• Mêmes tables de données qu'isql • Prioritaire	EC1
isql	• Mêmes tables de données qu'OLTP • Priorité faible	EC3
L1	• Aucune affectation de priorité	Aucun
sa	• Prioritaire	EC1
L2	• Non prioritaire	EC3
sp_xyz	• Eviter les "points de contention"	EC1

Configuration

L'administrateur système exécute les procédures système suivantes pour affecter les classes d'exécution (algorithme, étape 3) :

```
sp_bindexeclass OLTP, AP, NULL, EC1
sp_bindexeclass ISQL, AP, NULL, EC3
sp_bindexeclass L2, LG, NULL, EC3
sp_bindexeclass sa, LG, NULL, EC1
sp_bindexeclass SP_XYZ, PR, sp_owner, EC1
```

Caractéristiques d'exécution

Les cas de figure ci-après peuvent survenir dans un environnement Adaptive Server avec la configuration décrite dans l'exemple :

1 Un client se connecte à Adaptive Server en tant que "L1" en utilisant OLTP.

- Adaptive Server considère qu'OLTP est associé à EC1.
- "L1" n'a pas de classe d'exécution et Adaptive Server lui affecte la classe par défaut EC2. "L1" prend les caractéristiques définies par EC1 lorsqu'il appelle OLTP.
- Si "L1" exécute la procédure enregistrée sp_xyz, sa priorité ne change pas pendant l'exécution de sp_xyz. Tout au long de l'exécution, "L1" a les attributs EC1.

2 Un client se connecte à Adaptive Server en tant que "L1" en utilisant isql.

- Etant donné qu'isql est associé à EC3 et que la classe d'exécution de "L1" n'est pas définie, "L1" est exécuté avec les caractéristiques EC3. Cela signifie que sa priorité est faible et qu'il est affecté au moteur dont le numéro est le plus élevé (à condition qu'il y ait plusieurs moteurs).
- Lorsque "L1" exécute sp_xyz, son niveau de priorité est augmenté, car la procédure enregistrée est associée à la classe EC1.

3 Un client se connecte à Adaptive Server en tant que "sa" en utilisant isql.

   - Adaptive Server identifie la classe d'exécution d'isql et de "sa", en utilisant la règle de priorité. Il exécute l'instance de l'administrateur système d'isql avec des attributs EC1. Lorsque l'administrateur système exécute sp_xyz, la priorité ne varie pas.

4 Un client se connecte à Adaptive Server en tant que "L2" en utilisant isql.

   - L'application et le login étant associés l'une et l'autre à EC3, il n'y a pas de conflit. "L2" exécute sp_xyz au niveau de priorité élevée.

Eléments à prendre en compte pour la répartition des ressources moteur

L'affectation au jugé de classes d'exécution produit rarement les effets escomptés. Certaines conditions permettent d'obtenir de meilleures performances pour chaque type d'objet d'exécution. Le tableau 4-5 indique lorsqu'il est avantageux d'affecter une priorité d'exécution.

Tableau 4-5 : Utilité de l'affectation d'une priorité d'exécution

Exécution objet	Description
Application cliente	Il y a peu de conflits entre les applications clientes pour accéder aux ressources non CPU.
Login Adaptive Server	Un login doit avoir la priorité sur les autres logins pour accéder aux ressources CPU.
Procédure enregistrée	Il existe des "points de contention" clairement définis dans la procédure enregistrée.

Il est plus efficace d'abaisser la classe d'exécution des objets non prioritaires que d'augmenter celle d'un objet prioritaire. Les sections suivantes fournissent des indications plus précises pour améliorer les performances des différents types d'objets d'exécution.

Applications clientes OLTP et DSS

Il est parfois très utile d'affecter la priorité d'exécution maximale à des applications clientes lorsqu'il existe peu de conflits pour accéder aux ressources non CPU.

Supposons qu'une application OLTP et une application DSS soient exécutées en même temps et que vous souhaitiez sacrifier les performances de la seconde pour accélérer l'exécution de la première. Vous pouvez affecter des attributs d'exécution non prioritaire à l'application DSS pour qu'elle n'accède à la CPU qu'après l'exécution des tâches de l'application OLTP.

Applications clientes indépendantes

Le conflit de verrouillage entre applications ne pose pas de problème lorsqu'une application indépendante utilise des tables qui ne sont utilisées par aucune autre application sur le système.

Si la priorité d'exécution est affectée à ce type d'application et si elle lance une tâche exécutable, celle-ci sera placée en premier dans la file d'attente pour accéder au temps CPU.

Applications consommatrices d'E/S

Si une application prioritaire est fortement consommatrice d'E/S et si d'autres applications sont fortement consommatrices de calculs, celles-ci monopolisent les ressources CPU disponibles.

En revanche, un traitement fortement consommateur d'E/S cesse d'utiliser la CPU lorsqu'il effectue une opération E/S. L'affectation d'une classe d'exécution non prioritaire à une application fortement consommatrice de calculs permet à Adaptive Server de traiter plus rapidement les E/S.

Applications hautement prioritaires

S'il se trouve un ou deux objets d'exécution prioritaires parmi des objets d'exécution non prioritaires, essayez de définir une spécialisation de moteur avec un groupe de moteurs affecté aux applications non prioritaires. Vous pouvez de cette façon améliorer le débit des applications prioritaires.

Logins Adaptive Server : utilisateurs prioritaires

Si vous affectez des attributs d'exécution prioritaire à un utilisateur tout en conservant les attributs par défaut relatifs aux autres utilisateurs, Adaptive Server tente de traiter en priorité toutes les tâches associées à l'utilisateur prioritaire.

Procédure enregistrée "points de contention"

Les problèmes de performances en rapport avec les procédures stockées surviennent lorsqu'une procédure enregistrée est très sollicitée par une ou plusieurs applications. Dans ce cas, la procédure enregistrée s'apparente à un *point de contention* dans le chemin d'accès à une application.

En général, la priorité d'exécution dont disposent les applications qui exécutent la procédure enregistrée est moyenne ou faible ; l'affectation d'attributs d'exécution prioritaire supplémentaires à la procédure enregistrée doit donc permettre d'améliorer les performances de l'application qui la sollicite.

CHAPITRE 5

Gestion de l'emplacement physique des données

Ce chapitre indique comment optimiser les performances en positionnant les tables et les index.

Sujet	Page
Optimisation des performances grâce au positionnement des objets	81
Terminologie et concepts	84
Recommandations pour améliorer les performances des E/S	85
Utilisation du mode séquentiel	90
Création d'objets sur des segments	90
Partitionnement des tables pour les performances	93
Planification d'espace relative aux tables partitionnées	98
Commandes servant au partitionnement des tables	101
Etapes de partitionnement des tables	113
Procédures spéciales pour les situations difficiles	122
Problèmes de maintenance et tables partitionnées	130

Optimisation des performances grâce au positionnement des objets

Adaptive Server vous permet de contrôler le positionnement des bases de données, des tables et des index sur les différents devices de stockage physique. Cela vous permet d'améliorer les performances en équilibrant les opérations de lecture et d'écriture sur le disque, via de nombreux devices et contrôleurs. Vous pouvez notamment :

- Placer des segments de la base de données sur un ou plusieurs devices spécifiques et stocker le journal de la base de données sur un device physique distinct. Ainsi, les opérations de lecture et d'écriture dans le journal de la base de données ne créent pas d'interférence avec l'accès aux données.

- Répartir les tables volumineuses et souvent utilisées sur plusieurs devices.
- Placer des tables ou des index non clusterisés spécifiques sur des devices particuliers, Vous pouvez ainsi placer une table dans un segment qui s'étend sur plusieurs devices et ses index non clusterisés dans un segment distinct.
- Placer le chaînage des pages d'images et de texte d'une table sur un device différent de celui de la table proprement dite. La table contient un pointeur vers la valeur effective des données dans une structure de base de données distincte, de sorte que chaque accès à une colonne de texte ou d'images nécessite au moins deux E/S.
- Répartir les tables de façon équitable entre différentes partitions, sur des disques physiques distincts, afin d'optimiser les performances des requêtes parallèles.

Dans le cas de systèmes multi-utilisateur et multi-CPU effectuant un grand nombre d'opérations d'E/S sur le disque, veillez tout particulièrement aux problèmes de devices physiques et logiques et à la répartition des E/S entre les devices :

- Prévoyez une répartition équilibrée des objets entre les devices logiques et physiques.
- Faites appel à un nombre suffisant de devices physiques (y compris contrôleurs de disque) pour garantir la largeur de bande physique.
- Utilisez un nombre supérieur de devices logiques pour limiter au maximum les conflits dans les files d'attente d'E/S internes.
- Utilisez un nombre de partitions autorisant les balayages parallèles, afin de répondre aux objectifs de performances des requêtes.
- Tirez profit de la capacité de create database d'effectuer des E/S parallèles sur six devices à la fois, ce qui permet une nette amélioration des performances, autorisant la création de bases de données de plusieurs giga-octets.

Symptômes d'un mauvais positionnement des objets

Si vous rencontrez les difficultés présentées ci-après, vérifiez le positionnement des objets :

- Dans le cas d'un utilisateur unique, les performances sont acceptables mais les temps de réponse augmentent considérablement lorsque plusieurs processus sont exécutés.
- L'accès à un disque mis en miroir prend deux fois plus de temps que s'il ne l'était pas.
- Les performances des requêtes diminuent à mesure que l'activité portant sur les tables système s'intensifie.
- Les activités de maintenance semblent prendre beaucoup de temps.
- Les procédures stockées ralentissent lors de la création de tables temporaires.
- Les performances d'insertion sont médiocres dans les tables souvent utilisées.
- Les requêtes exécutées en parallèle sont lentes, en raison d'une mauvaise répartition des pages de données entre les partitions ou devices ; ou bien elles s'exécutent en série, lorsque le déséquilibre est trop grand.

Problèmes sous-jacents

Si vous rencontrez des difficultés liées à des conflits sur le disque ou à des positionnements d'objet, passez en revue les problèmes sous-jacents énumérés ci-après :

- Les accès aléatoires (E/S pour les données et index) et accès en série (E/S du journal) utilisent les mêmes disques.
- Les processus de base de données et processus du système d'exploitation utilisent les mêmes disques.
- La mise en miroir en série des disques est utilisée pour des raisons fonctionnelles.
- La maintenance des bases de données (journalisation ou audit) intervient sur les mêmes disques que ceux utilisés pour le stockage des données.
- La base tempdb réside sur le même disque que les tables très souvent utilisées.

Utilisation de *sp_sysmon* lors du changement d'emplacement des données

Utilisez sp_sysmon pour déterminer si le positionnement des données sur les devices physiques entraîne des problèmes de performances. Vérifiez intégralement les résultats de sp_sysmon au cours de l'optimisation, afin de voir dans quelle mesure les modifications affectent les différentes catégories de performances.

Pour plus d'informations sur sp_sysmon, reportez-vous au chapitre 38, "Contrôle des performances avec sp_sysmon".

Veillez en particulier aux résultats des discussions :

- conflits d'E/S sur les devices
- tables sans index APL
- Verrous de dernière page sur les tables sans index
- Gestion E/S disque

Adaptive Server Monitor peut également vous aider à détecter les problèmes.

Terminologie et concepts

Il est important de comprendre les distinctions qui sont faites entre les devices logiques (ou devices de base de données) et les devices physiques :

- Le *disque physique* ou *device physique* est le matériel permettant effectivement le stockage des données.
- Un *device de base de données* ou *device logique* est une partie d'un disque physique qui a été initialisée (à l'aide de la commande disk init) en vue de son utilisation par Adaptive Server. Un device de base de données peut être un fichier de système d'exploitation, un disque entier ou une partition de disque.

  Pour plus d'informations sur les contraintes spécifiques du système d'exploitation relatives à l'utilisation du disque et des fichiers, reportez-vous aux guides d'installation et de configuration d'Adaptive Server.

- Un *segment* est une collection nommée de devices de base de données utilisés par une base. Les devices de base de données qui constituent un segment peuvent être placés sur des disques physiques distincts.
- Une *partition* est un bloc de stockage pour une table. Le partitionnement d'une table la divise de façon à permettre l'accès à plusieurs tâches à la fois. Lorsque des tables partitionnées sont placées dans des segments comportant un nombre correspondant de devices, chaque partition commence sur un device de base de données distinct.

Pour plus d'informations sur les devices, reportez-vous à sp_helpdevice ; pour plus d'informations sur les segments, reportez-vous à sp_helpsegment ; pour plus d'informations sur les partitions, reportez-vous à sp_helpartition.

Recommandations pour améliorer les performances des E/S

Pour améliorer les performances des E/S sur Adaptive Server, observez les recommandations ci-après :

- répartition des données sur plusieurs disques afin d'éviter les conflits d'E/S ;
- isolation des E/S réalisées au niveau du serveur, des E/S sur les bases de données ;
- dans le cas de bases de données fréquemment mises à jour, séparation du stockage des données de celui du journal ;
- séparation des E/S disque aléatoires des E/S disque séquentielles ;
- mise en miroir des devices sur des disques physiques distincts ;
- partitionnement des tables en fonction du nombre de devices physiques du segment.

Répartition des données sur plusieurs disques afin d'éviter les conflits d'E/S

La répartition du stockage des données sur plusieurs disques et plusieurs contrôleurs de disque permet d'éviter les goulots d'étranglement :

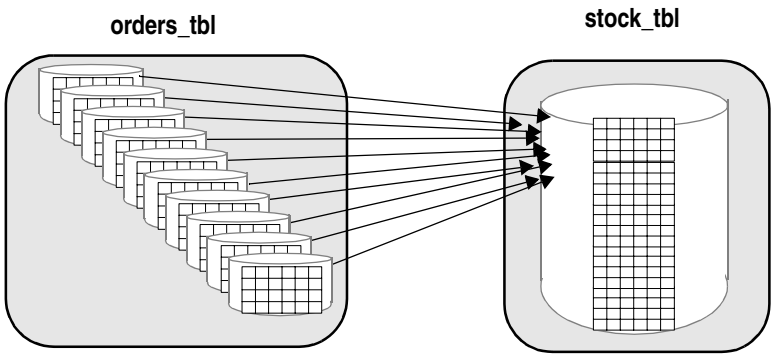
- Installez les bases de données dont vous exigez des performances particulières sur des devices séparés. Dans la mesure du possible, utilisez également des contrôleurs différents de ceux des autres bases de données. Utilisez les segments et les partitions en fonction des besoins des tables les plus importantes et des requêtes parallèles, respectivement.
- Placez les tables les plus souvent utilisées sur des disques distincts.
- Placez les tables fréquemment jointes sur des disques distincts.
- Utilisez les segments pour placer les tables et les index sur leurs propres disques.

Prévention des conflits physiques dans les requêtes parallèle de jointure

L'exemple de la figure 5-1 illustre la jointure de deux tables, orders_tbl et stock_tbl. Dix processus de travail sont disponibles : la table orders_tbl possède dix partitions réparties sur dix devices physiques différents et elle est la table externe de la jointure ; la table stock_tbl n'est pas partitionnée. Les processus de travail ne rencontreront pas de problème de conflit d'accès sur la table orders_tbl mais chaque processus de travail doit balayer la table stock_tbl. Il pourrait y avoir un conflit d'E/S physiques si la table entière ne tient pas dans le cache. Dans le pire des cas, dix processus de travail tentent d'accéder au device physique sur lequel réside la table stock_tbl. Vous pouvez éviter les conflits d'E/S physiques en créant un cache nommé qui contient l'intégralité de la table stock_tbl.

Il existe une autre méthode pour réduire ou éliminer les conflits d'E/S physiques. Elle consiste à partitionner à la fois orders_tbl et stock_tbl et à répartir ces partitions sur des devices physiques distincts.

Figure 5-1 : Jointure de tables placées sur des devices physiques distincts



Isolation des E/S réalisées au niveau du serveur, des E/S sur les bases de données

Placez les bases de données système pour lesquelles le nombre d'E/S est élevé sur des disques physiques et des contrôleurs distincts de ceux des bases de données de votre application.

Où placer *tempdb*

La base tempdb est automatiquement placée sur le device master. Si elle nécessite davantage d'espace, tempdb peut être étendue à d'autres devices. Si vous prévoyez de travailler relativement souvent avec tempdb, placez-la sur un disque qui ne sert pas pour une autre base de données très utilisée. Utilisez le disque le plus rapide pour tempdb. En effet, cette base, fréquemment utilisée, a des incidences sur tous les processus du serveur.

Sur certains systèmes UNIX, une E/S réalisée sur un fichier du système d'exploitation est considérablement plus rapide qu'une E/S effectuée sur une partition de disque. Après un arrêt du système, comme tempdb est systématiquement recréée et non restaurée, vous pouvez améliorer les performances en plaçant tempdb sur un fichier plutôt que sur une partition de disque. Faites l'essai sur votre propre système.

Pour plus d'informations sur le positionnement et les performances de tempdb, reportez-vous au chapitre 26, "Performances de tempdb".

Positionnement de *sybsecurity*

Lorsque vous utilisez les mécanismes d'audit d'Adaptive Server, le système d'audit procède à de nombreuses opérations d'E/S vers la table sysaudits de la base de données sybsecurity. Si vos applications effectuent de nombreuses tâches d'audit, placez la base sybsecurity sur un disque qui n'est pas utilisé pour les tables exigeant un temps de réponse très court. Un fonctionnement optimal sera obtenu en plaçant sybsecurity sur son propre device.

Utilisez également le gestionnaire de seuils pour gérer l'espace libre et éviter d'interrompre les transactions utilisateur lorsque la base de données d'audit est saturée.

Journaux de transactions sur un disque séparé

Vous pouvez limiter la taille de ces journaux en les plaçant sur un segment séparé et éviter qu'ils entrent en concurrence avec d'autres objets pour disposer de l'espace disque. Le stockage du journal sur un disque physique distinct présente les avantages ci-après :

- Amélioration des performances par la réduction des conflits d'E/S.
- Possibilité de restauration complète en cas de panne du disque dur contenant le device de données.
- Accélération de la restauration, dans la mesure où les requêtes simultanées de prélecture asynchrone peuvent lire à l'avance sur le device du journal et sur celui des données, sans créer de conflit.

Il est courant, mais peu fiable, de placer le journal de transactions sur le même device que les données ; si vous choisissez cette méthode, utilisez l'option with override avec create database et alter database.

Le device du journal peut procéder à d'importantes opérations d'E/S sur les systèmes effectuant de nombreuses mises à jour. Adaptive Server écrit les enregistrements du journal sur le disque lorsque les transactions sont validées (commit) et peut être amené à lire des pages du journal dans la mémoire en cas de mises à jour différées ou d'annulations de transactions (rollback) par exemple.

Si le journal et les données sont placés sur les mêmes devices de base de données, les extents attribués au stockage des pages du journal ne sont pas contigus ; les extents du journal et des données sont mélangés. Lorsque le journal est sur un device qui lui est propre, les extents sont généralement attribués en séquence, ce qui réduit le temps de positionnement des têtes de disque et améliore le taux d'E/S.

Si le journal et les données se trouvent sur des devices distincts, Adaptive Server stocke les enregistrements de journal de chaque utilisateur dans un cache de journaux qui lui est réservé, ce qui réduit les conflits lors de l'écriture de la page du journal en mémoire. Si le journal et les données se trouvent sur les mêmes devices, la mise en buffer dans le cache de journaux de l'utilisateur est désactivée, ce qui pénalise fortement les performances des systèmes SMP.

Si vous avez créé une base de données sans placer son journal sur un device distinct, reportez-vous au *Guide d'administration système.*

Mise en miroir d'un device sur un disque séparé

Si vous placez des données en miroir, faites-le sur un disque physique distinct du device principal. Les pannes de disque dur entraînent souvent la perte ou l'indisponibilité de disques physiques entiers. La mise en miroir sur des disques séparés en réduit les conséquences sur les performances.

Performances et mise en miroir des devices

La mise en miroir est une fonction de sécurité facilement accessible, qui permet à Adaptive Server de dupliquer le contenu de tout un device de base de données.

Pour plus d'informations à ce sujet, reportez-vous au *Guide d'administration système*.

Si vous n'utilisez pas la mise en miroir ou si vous utilisez celle du système d'exploitation, définissez le paramètre de configuration disable disk mirroring à 1. Ceci améliore légèrement les performances.

Cette opération peut ralentir les écritures sur le disque car ces dernières sont enregistrées sur les deux disques, simultanément ou séquentiellement. Les lectures sont toujours effectuées sur le device principal. La mise en miroir n'a aucune incidence sur le temps de lecture des données.

Les devices mis en miroir appliquent l'un des deux modes suivants d'écriture sur le disque :

- Le mode *Nonserial (simultané)* peut augmenter le temps d'écriture des données. En effet, les deux écritures démarrent en même temps et Adaptive Server attend que les deux soient terminées. Le temps requis pour achever les écritures en mode simultané est *max(W1 ,W 2)*, c'est-à-dire le plus long des deux temps d'E/S.
- Le mode *séquentiel (serial)* augmente encore davantage le temps d'écriture des données. Adaptive Server démarre la première écriture et attend qu'elle soit terminée pour lancer la seconde. Le temps nécessaire est égal à *W1+W2*, soit la somme des deux temps d'E/S.

Utilisation du mode séquentiel

Bien que le mode séquentiel réduise les performances, il contribue de façon significative à la fiabilité. Il est le mode par défaut car il constitue une protection en cas de panne survenant pendant les opérations d'écriture.

Comme il prévoit que la première écriture doit être terminée avant que la deuxième ne démarre, une panne ne peut pas affecter les deux disques en même temps. Si vous sélectionnez le mode simultané, vous améliorez les performances mais vous risquez de perdre des données en cas de panne impliquant les deux disques en même temps.

Avertissement ! N'utilisez le mode simultané que si la fiabilité du système de la base de données mise en miroir peut n'être que relative.

Création d'objets sur des segments

Un segment est une étiquette pointant sur un ou plusieurs devices de base de données.

Chaque base de données peut contenir jusqu'à 32 segments, dont les 3 segments créés par le système (system, log segment et default) lors de la création de la base de données. Les segments attribuent une étiquette à l'espace d'un ou de plusieurs devices logiques.

Les tables et index sont stockés sur des segments. Lorsque aucun segment n'est indiqué dans les instructions create table ou create index, les objets sont stockés dans le segment default attribué à la base de données. L'indication d'un nom de segment dans l'une de ces deux commandes crée l'objet sur le segment correspondant. La procédure système sp_placeobject permet de désigner le segment à utiliser pour toutes les opérations ultérieures. Les tables peuvent ainsi être réparties sur plusieurs segments.

Le device doit être initialisé avec disk init par un administrateur système (SA) puis il doit être affecté à la base de données par cet administrateur ou par le propriétaire de la base de données à l'aide des commandes create database et alter database.

Lorsque les devices sont accessibles à la base de données, le propriétaire de la base ou les propriétaires d'objets peuvent créer des segments et placer des objets sur les devices.

Si vous créez un segment, vous pouvez y placer des tables ou des index à l'aide des commandes create table et create index :

```
create table tableA(...) on seg1
create nonclustered index myix on tableB(...)
      on seg2
```

En contrôlant l'emplacement des tables et des index stratégiques, vous pouvez choisir leur répartition sur plusieurs disques.

Utilisation de segments

Les segments peuvent améliorer la capacité de traitement avec les méthodes suivantes :

- répartition des tables volumineuses sur plusieurs disques, y compris les tables partitionnées pour améliorer les performances des requêtes parallèles ;
- séparation des tables et de leurs index non clusterisés pour les placer sur différents disques ;
- positionnement de la chaîne de pages de données de type texte ou image sur un disque distinct de celui de la table elle-même, dans laquelle sont stockés les pointeurs des valeurs texte.

Les segments permettent également de contrôler l'utilisation de l'espace :

- La taille d'une table ne peut jamais dépasser celle du segment qui lui est alloué ; les segments vous permettent donc de limiter la taille des tables.
- Les tables stockées sur un segment ne peuvent pas empiéter sur l'espace alloué aux objets d'un segment différent.
- Le gestionnaire de seuils permet de gérer l'utilisation de l'espace.

Séparation des tables et des index

Utilisez les segments pour isoler les tables sur un ensemble de disques et les index non clusterisés sur un autre. Il est impossible de séparer l'index clusterisé de ses pages de données. Lorsque vous créez un index clusterisé avec la clause on *nom_segment*, la table entière se déplace vers le segment indiqué, sur lequel se construit l'arbre d'index clusterisé.

Vous pouvez améliorer les performances en plaçant les index non clusterisés sur un segment distinct

Répartition de tables volumineuses sur plusieurs devices

Les segments peuvent s'étendre sur plusieurs devices, afin de pouvoir être utilisés pour répartir les données sur plusieurs disques. Dans le cas de tables volumineuses et très souvent utilisées, cette méthode permet d'équilibrer les E/S. Dans le cas de requêtes en parallèle, la création de segments incluant plusieurs devices est capitale pour la parallélisation E/S pendant les balayages de partition.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Déplacement du stockage de texte vers un device distinct

Lorsqu'une table inclut des données du type text, image ou Java hors ligne, la table elle-même contient un pointeur sur la valeur des données. Les données réelles sont stockées séparément, dans une liste liée de pages appelée chaîne LOB (large object).

La lecture ou l'écriture d'une valeur LOB implique au moins deux accès au disque, l'un pour lire ou écrire le pointeur, l'autre pour les lectures ou les écritures des données correspondantes. Si votre application lit ou écrit souvent ces valeurs, vous pouvez améliorer les performances en plaçant la chaîne LOB sur un device physique distinct. Isolez les chaînes de type LOB sur des disques qui ne sont pas sollicités par des accès à des tables ou index dans le cadre d'une autre application.

Lorsque vous créez une table avec des colonnes LOB, Adaptive Server crée une ligne dans sysindexes pour l'objet qui contient les données LOB. Le nom de la table et le préfixe "t" figurent dans la colonne name ; la valeur indid est toujours 255. Même lorsque vous avez plusieurs colonnes LOB dans une même table, il n'existe qu'un objet pour le stockage des données. Par défaut, cet objet est placé sur le même segment que la table.

sp_placeobject vous permet de déplacer les affectations ultérieures des colonnes de type LOB vers un segment distinct.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Partitionnement des tables pour les performances

Le partitionnement d'une table peut améliorer les performances pour plusieurs processus. Il convient de partionner une table pour les raisons suivantes :

- Il permet un accès à chaque partition de la table dans le cas du traitement parallèle des requêtes. Lors d'un balayage basé sur les partitions, chacun des processus de travail lit une partition distincte.
- Le partitionnement permet de charger une table en parallèle avec la copie "bulk copy".

  Pour plus d'informations sur le bcp parallèle, reportez-vous au manuel *Utilitaires*.
- Le partitionnement permet de répartir les E/S d'une table entre plusieurs devices de base de données.
- Le partitionnement fournit plusieurs points d'insertion pour une table sans index.

Le choix des tables à partitionner dépend des problèmes de performances que vous rencontrez et des objectifs de performances fixés pour les requêtes portant sur ces tables.

Les sections ci-après décrivent les commandes à utiliser pour le partitionnement et la gestion des tables et elles décrivent les étapes adaptées à chaque situation.

Pour plus d'informations et des exemples de partitionnement adaptés à des objectifs particuliers de performances, reportez-vous à la section "Recommandations pour la configuration des requêtes parallèles", page 573.

Transparence

Adaptive Server gère les tables partitionnées en informant les utilisateurs et les applications. Les requêtes ou les affichages à l'aide de plusieurs utilitaires ne fait pas de différence entre les tables, partitionnées ou non. Les exceptions sont les suivantes :

- Si les requêtes ne comportent pas la commande order by ou d'autres commandes reposant sur des tris, les données retournées par une requête en parallèle ne sont pas dans le même ordre que celles retournées par les requêtes en série.
- Les commandes dbcc checktable and dbcc checkdb répertorient le nombre de pages de données dans chaque partition.

  Pour plus d'informations sur la commande, dbcc, reportez-vous au *Guide d'administration système*.
- sp_helpartition donne des informations sur les partitions d'une table.
- Le résultat de showplan indique le nombre de processus de travail utilisés par les requêtes en parallèle et le "Scan count" de statistics io affiche le nombre de balayages effectués par les processus de travail.
- Vous pouvez effectuer une copie sur une partition d'une table sans index à l'aide d'une opération bulk copy en parallèle.

Tables partitionnées et traitement des requêtes en parallèle

Le traitement en parallèle sur des tables partitionnées peut améliorer considérablement les performances des requêtes. Les partitions augmentent l'accès simultané des processus de travail. Quand les processus de travail sont suffisants et que la valeur du paramètre de configuration max parallel degree est égale ou supérieure au nombre de partitions, un seul processus de travail balaye chaque partition.

Si les partitions sont réparties sur des disques physiques, la réduction de conflits E/S accélère le traitement en parallèle des requêtes et aboutit à un excellent niveau de parallélisation.

L'optimiseur peut choisir d'utiliser le traitement en parallèle des requêtes, s'il est activé, sur une table partitionnée. L'optimiseur envisage un balayage des partitions en parallèle si la table concernée par la requête est partitionnée tandis qu'il recourt à une lecture indexée en parallèle dans le cas d'un index exploitable.

Pour plus d'informations sur l'optimisation des requêtes parallèles, reportez-vous au chapitre 23, "Optimisation des requêtes parallèles".

Répartition des données sur les partitions

La création d'un index clusterisé sur une table partitionnée permet de répartir uniformément les données de la table. Adaptive Server détermine les intervalles de clés d'index de façon à distribuer également les lignes de la partition. Si le nombre de devices dans le segment est égal ou supérieur au nombre de partitions, un device exclusif, au moins, est attribué par partition.

Si vous créez l'index clusterisé sur une table partitionnée vide, Adaptive Serveraffiche un message vous conseillant de le recréer après avoir chargé les données dans la table car, en attendant, elles seront insérées dans la première partition.

Si vous partitionnez une table qui contient déjà un index clusterisé, toutes les pages seront affectées à la première partition. La commande alter table...partition aboutit et affiche un message. Vous devez supprimer puis recréer l'index pour redistribuer les données.

Amélioration des performances d'insertion avec les partitions

Toutes les commandes d'insertion sur une table APL sans index essayent d'insérer les lignes sur la dernière page de la table. Si plusieurs utilisateurs insèrent des données simultanément, chaque nouvelle transaction d'insertion doit attendre son tour.

Le partitionnement d'une table APL sans index réduit les conflits de la dernière page de la chaîne améliorant ainsi l'insertion simultanée de données.

Dans le cas de tables DOL, Adaptive Server stocke une ou plusieurs indications se référant à une page où une insertion s'est effectuée récemment. Seuls des volumes élevés d'insertions peuvent provoquer des encombrements dans ce type de tables.

Le partitionnement des tables DOL sans index donne plus d'indications permettant de résoudre les encombrements.

Résolution des conflits de page avec le partitionnement

Adaptive Server affecte une transaction d'insertion indifféremment à l'une des partitions de la table sans index. Les insertions concurrentes provoqueront moins de blocages, puisque plusieurs dernières pages sont accessibles.

Sélection de tables sans index à partitionner

Les tables APL sans index recevant de grandes quantités d'insertions simultanées trouveront un avantage à la partition. Il faut des débits très élevés d'insertions pour provoquer de réels blocages dans les tables DOL. Pour vérifier l'avantage du partitionnement pour vos bases de données, recourez aux techniques suivantes :

- Vérifiez la présence de verrous de dernière page dans les tables sans index à l'aide de sp_sysmon.

  Pour plus d'informations, reportez-vous à la section "Gestion des verrous", page 1036.

- Utilisez sp_object_stats pour obtenir des rapports sur les conflits de verrous.

  Pour plus d'informations, reportez-vous à la section "Identification des tables pour lesquelles la concurrence d'accès est problématique", page 287.

Restrictions sur les tables partionnées

Vous ne pouvez pas partitionner des table systèmes de Adaptive Server, ni une table déjà partitionnée. Vous devez supprimer les partitions pour pouvoir exécuter une des commandes Transact-SQL suivantes :

- sp_placeobject
- truncate table
- alter table *nom_table* partition *n*

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

Paramètres de configuration pour les partitions

Si vous avez besoin d'un grand nombre de partitions, vous pouvez modifier les valeurs par défaut des paramètres de configurationpartition groups et partition spinlock ratio.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Répartition des partitions sur les devices

Quand vous émettez une commande alter table...partition, Adaptive Server crée le nombre spécifié de partitions dans la table et les répartit de manière équitable sur les devices de bases de données du segment.

Le tableau 5-1 explique comment Adaptive Server répartit 5 partitions sur 3, 5 et 12 devices, respectivement.

Tableau 5-1 : Allocation de partitions à des segments

	Device (D)- Affectations à un segment avec		
ID de partition	**3 Devices**	**5 Devices**	**12 Devices**
Partition 1	D1	D1	D1, D6, D11
Partition 2	D2	D2	D2, D7, D12
Partition 3	D3	D3	D3, D8, D11
Partition 4	D1	D4	D4, D9, D12
Partition 5	D2	D5	D5, D6, D11

Les performances E/S des requêtes parallèles sont optimales lorsque le nombre de partitions correspond au nombre de devices du segment.

Vous pouvez partitionner des tables contenant des données de type text, image ou Java hors ligne. Toutefois, les colonnes ne sont pas partitionnées (elles restent sur une seule chaîne de pages).

Devices RAID et tables partitionnées

Le tableau 5-1 ainsi que d'autres instructions du chapitre décrivent les devices logiques Adaptive Server qui mappent à un seul device physique.

Un device RAID par bandes peut contenir plusieurs disques physiques tout en ne constituant qu'un seul device logique Adaptive Server. Dans ce cas, vous pouvez utiliser plusieurs partitions sur l'unique device logique pour obtenir d'excellentes performances de la requête en parallèle.

Pour déterminer le nombre optimal de partitions nécessaire à votre combinaison d'applications, commencez avec une partition par device dans la séquence de bandes. Pour contrôler l'utilisation et le temps d'attente, utilisez les utilitaires de votre système d'exploitation (vmstat, sar et iostat sur UNIX ; Performance Monitos sur Windows NT).

Pour vérifier le débit maximal de devices, utilisez select count(*), à l'aide de la clause (index *nom_table*) pour forcer un balayage de table, s'il existe un index non clusterisé. Cette commande s'appuie sur une activité CPU très réduite sans conflit majeur avec les autres ressources.

Planification d'espace relative aux tables partitionnées

Les deux principaux problèmes de la planification d'espace sont les suivants :

- le maintien d'un équilibre des charges sur le disque lors des balayages reposant sur les partitions et d'une parallélisation E/S ;
- le maintien d'index clusterisés requiert environ 120% de l'espace occupé par la table pour supprimer et recréer l'index ou exécuter reorg rebuild.

Vos choix dépendent de :

- la disponibilité des ressources disque pour le stockage des tables
- la nature de votre combinaison d'applications

Vous devez évaluer la fréquence des opérations de maintenance sur vos tables partitionnées : certaines applications nécessitent une recréation d'index fréquente pour maintenir l'équilibre ;

Pour ces applications, la disponibilité d'espace pour recréer un index clusterisé ou exécuter reorg rebuild fournit la méthode la plus rapide et la plus simple. Le segment doit cependant disposer d'environ 120% de l'espace occupé par la table pour la copie de pages de données nécessaire à la création d'index clusterisé.

Pour plus d'informations, reportez-vous à la section "Détermination de l'espace disponible pour des activités de maintenance", page 404.

Les descriptions suivantes de modifications de données en lecture seule, en lecture principale et de façon aléatoire indiquent les problèmes que peuvent provoquer le positionnement d'objets et la gestion de tables partitionnées.

Pour plus d'informations sur les tâches spécifiques nécessaires à la maintenance, reportez-vous à la section "Etapes de partitionnement des tables", page 113.

Tables en lecture seule

Les tables en lecture seule ou qui sont rarement modifiées peuvent remplir tout l'espace disponible sur un segment et n'ont pas besoin de maintenance. Si une table n'a pas besoin d'index clusterisé, vous pouvez utiliser une opération bulk copy en parallèle pour remplir complètement l'espace sur le segment.

Dans le cas contraire, les pages de données de la table peuvent occuper jusqu'à 80 % de l'espace dans le segment. L'arbre de l'index clusterisé nécessite environ 20 % de l'espace occupé par la table.

Cette dimension est fonction de la longueur de la clé. Plusieurs étapes sont nécessaires pour le chargement initial de données dans la table et pour la création d'index clusterisés ; après quoi, la maintenance sera réduite.

Tables principalement en lecture

Les recommandations précédentes concernant les tables en lecture seule s'appliquent également aux tables principalement en lecture qui ne contiennent que peu d'insertions. Les seules exceptions sont les suivantes :

- Si la table contient des insertions et que la clé d'index clusterisé ne distribue pas les allocations d'espace sur les partitions de manière homogène, les disques qui supportent certaines partitions peuvent être saturés et les nouvelles allocations d'extent devront être effectuées sur un autre disque physique. On parle alors d'*extent stealing*.

  S'il s'agit de tables volumineuses réparties sur plusieurs disques, l'affectation de quelques allocations sur d'autres devices ne constitue pas un problème. La commande sp_helpsegment vous permet de détecter les devices qui manquent d'espace disponible et doivent recourir à l'extent stealing tandis que sp_helpartition vous permet de trouver les partitions qui ont un nombre disproportionné de pages.

  Si le déséquilibre des partitions affaiblit le temps de réponse ou l'optimisation des requêtes en parallèle, vous pouvez y remédier à l'aide d'une des méthodes décrites dans la section "Etapes de partitionnement des tables", page 113.

- S'il s'agit d'une table sans index, la nature aléatoire des insertions doit permettre de conserver l'équilibre des partitions.

  Recourez aux opérations en bulk copy. Vous pouvez utiliser une opération bulk copy en parallèle pour envoyer des lignes à la partition la moins complète et corriger ainsi le déséquilibre de la répartition de données. Pour plus d'informations, reportez-vous à la section "Utilisation de bcp pour équilibrer les partitions", page 108.

Tables avec modification de données de façon aléatoire

Les tables à index clusterisés qui sont sujettes à plusieurs insertions, mises à jour et suppressions finissent souvent par contenir des pages de données pleines à environ 70–75 %. Ceci risque d'amoindrir les performances sous les aspects suivants :

- Pour accéder à un nombre donné de lignes, il sera nécessaire de lire davantage de pages, ce qui implique des E/S supplémentaires et une perte d'espace dans le cache de données.
- Sur les tables qui utilisent le verrouillage APL, les performances des E/S étendues et de la prélecture asynchrone sont compromises parce que la chaîne de pages dépasse la taille des extents et des unités d'allocations.

Les buffers introduits par des E/S étendues risquent d'être rejetées du cache avant de terminer la lecture de toutes les pages. Le passage d'une allocation à l'autre tout au long de la chaîne de pages réduit la taille de l'ensemble de pages préalablement lues en prélecture asynchrone.

Dès que la fragmentation affecte les performances de l'application, vous devrez exécuter la maintenance. Si ceci implique la suppression et la recréation de l'index clusterisé, vous aurez besoin de 120 % de l'espace occupé par la table.

Faute d'espace suffisant, la maintenance devient longue et difficile à effectuer. La meilleure solution, souvent la moins onéreuse, consiste à ajouter suffisamment de capacité disque pour permettre la création d'index.

Commandes servant au partitionnement des tables

Création et gestion des tables partitionnées implique l'utilisation des divers types de commande suivants :

- commandes pour partitionner et annuler les partitions de la table,
- commandes pour supprimer et recréer des index clusterisés dans le cadre de la répartition de données sur les partitions et/ou sur des devices physiques sous-jacents,
- commandes de bulk copy parallèle pour charger des données dans des partitions,
- commandes pour afficher des informations relatives à la distribution de données sur les partitions et les device,
- commandes pour mettre à jour les statistiques de partition.

Cette section décrit la syntaxe des commandes que vous utilisez pour créer et mettre à jour les tables partitionnées et en donne des exemples.

Pour les situations nécessitant des combinaisons diverses de ces commandes, reportez-vous à la section "Etapes de partitionnement des tables", page 113.

Pour partitionner et annuler les partitions d'une table, utilisez la commande alter table.

Syntaxe *alter table...partition*

La syntaxe pour l'utilisation de la clause partition de alter table est la suivante :

```
alter table table_name partition n
```

où *nom_table* est le nom de la table et *n* est le nombre de partitions que vous créez.

Toute donnée qui se trouve dans la table avant l'utilisation de alter table reste dans la première partition. Le partitionnement d'une table ne déplace pas les données de la table, qui occupent toujours le même espace sur les devices physiques.

Si vous créez des tables partitionnées pour des requêtes parallèles, vous pouvez redistribuer les données soit par création d'un index clusterisé soit par copie des données à l'extérieur de la table, troncation de la table et puis réintégration des données.

Vous ne pouvez pas utiliser dump database dans une transaction définie par l'utilisateur.

La commande suivante crée 10 partitions pour une table dont le nom est historytab :

```
alter table historytab partition 10
```

Syntaxe *alter table...unpartition*

L'annulation du partitionnement d'une table signifie concaténer les diverses partitions d'une table en une seule partition. L'annulation du partitionnement d'une table ne modifie pas l'emplacement des données.

La syntaxe pour l'utilisation de la clause unpartition de alter table est la suivante :

```
alter table table_name unpartition
```

Par exemple, pour annuler les partitions d'une table appelée historytab, entrez :

```
alter table historytab unpartition
```

Modification du nombre de partitions

Pour modifier le nombre de partitions dans une table, annulez d'abord les partitions de la table à l'aide de alter table...unpartition.

Utilisez ensuite alter table...partition pour spécifier le nouveau nombre de partitions. Cette opération ne déplace pas les données existantes dans la table.

Vous ne pouvez pas utiliser la clause partition avec une table déjà partitionnée.

Par exemple, si une table nommée historytab comporte 10 partitions et que vous souhaitez qu'elle en compte 20, entrez les commandes suivantes :

```
alter table historytab unpartition
alter table historytab partition 20
```

Répartition uniforme des données sur les partitions

La qualité des performances dépend d'une bonne distribution des données sur les partitions d'une table. Pour l'obtenir, les deux principales méthodes sont les suivantes :

- Création d'un index clusterisé sur une table partitionnée. Les données doivent déjà se trouver dans la table.
- Utilisation de l'opération "bulk copy" parallèle, en indiquant les partitions dans lesquelles les données doivent être chargées.

La procédure système sp_helpartition *nom_table* indique le nombre de pages sur chacune des partitions de la table.

Commandes permettant de créer et de supprimer des index clusterisés

Il est possible de créer un index clusterisé à l'aide de la commande create clustered index ou en créant une contrainte de clé primaire ou étrangère à l'aide de alter table...add constraint. Les étapes permettant de supprimer et de re-créer sont légèrement différentes, en fonction de la méthode utilisée pour créer l'index clusterisé existant.

La création d'un index clusterisé sur une table partitionnée impose un tri parallèle. Définissez les paramètres de configuration et les options set ci-après avant d'émettre la commande destinée à créer l'index :

- Affectez aux paramètres number of worker processes et max parallel degree une valeur au moins égale au nombre de partitions de la table, plus 1.
- Exécutez sp_dboption "select into/bulkcopy/pllsort", true et lancez checkpoint dans la base de données.

Pour plus d'informations sur la configuration d'Adaptive Server en vue de l'exécution en parallèle, reportez-vous à la section "Contrôle du degré de parallélisation", page 561.

Pour plus d'informations sur le tri parallèle, reportez-vous au chapitre 24, "Tri parallèle".

Si vos requêtes ne font pas appel à l'index clusterisé, vous pouvez supprimer l'index sans modifier la répartition des données. Même si vous n'envisagez pas de conserver l'index clusterisé, efforcez-vous de le créer sur une clé possédant un nombre élevé de valeurs de données. Ainsi, une colonne telle que "sexe", possédant uniquement des valeurs "M" et "F", ne permettra pas une répartition équitable des pages entre les partitions.

La création d'un index utilisant le tri parallèle est une opération comportant une journalisation minimale et n'est pas restaurable. Vous devez vider la base de données après l'exécution de la commande.

Utilisation de *reorg rebuild* sur des tables verrouillées au niveau des pages de données seulement

La commande reorg rebuild copie des lignes de données d'une table verrouillée au niveau des pages de données seulement dans de nouvelles pages de données. S'il existe un index clusterisé, les lignes sont copiées dans l'ordre de la clé d'index.

L'exécution de reorg rebuild redistribue les données de manière équitable sur les partitions. L'index clusterisé et les index non clusterisés sont recréés. Pour exécuter reorg rebuild sur la table, indiquez uniquement le nom de la table :

```
reorg rebuild titles
```

Utilisation de *drop index* et de*create clustered index*

Si l'index de la table a été créé à l'aide de create index, procédez comme suit :

1 Supprimez l'index :

```
drop index huge_tab.cix
```

2 Créez l'index clusterisé, en indiquant le segment :

```
create clustered index cix
    on huge_tab(key_col)
    on big_demo_seg
```

Utilisation de contraintes et de la commande *alter table*

Si l'index de la table a été créé à l'aide d'une contrainte, procédez comme suit pour recréer un index clusterisé :

1 Supprimez la contrainte :

```
alter table huge_tab drop constraint prim_key
```

2 Recréez la contrainte en régénérant l'index :

```
alter table huge_tab add constraint prim_key
    primary key clustered (key_col)
    on big_demo_seg
```

Recommandations particulières concernant les tables partitionnées et les index clusterisés

La création d'un index clusterisé sur une table partitionnée est la seule méthode permettant de redistribuer les données entre les partitions sans devoir recharger ces dernières en les copiant hors de la table puis en les y réintégrant.

Lorsque vous utilisez des tables partitionnées et des index clusterisés, deux aspects sont à prendre en compte :

- N'oubliez pas que les données d'un index clusterisé "suivent" l'index et que, si vous ne spécifiez pas de segment dans les commandes create index ou alter table, le segment default est utilisé comme segment cible.
- La clause with sorted_data permet d'éviter le tri et la copie de données lors de la création d'un index clusterisé. Cela représente un gain de temps lorsque les données se trouvent déjà dans un ordre de clé clusterisé. Toutefois, lorsque vous avez besoin de créer un index clusterisé pour équilibrer la charge des données sur les partitions, n'utilisez pas la clause sorted_data.

  Pour plus de détails sur les options, reportez-vous à la section "Création d'un index sur des données triées", page 391.

Utilisation de *bcp* parallèle pour copier des données dans les partitions

Le chargement de données dans une table partitionnée à l'aide de la commande bcp parallèle vous permet de diriger les données vers une partition spécifique de la table.

- Avant l'exécution de la copie "bulk copy" parallèle, la table doit être placée dans le segment et partitionnée.
- Vous devez supprimer tous les index, de façon à éviter d'éventuelles défaillances dues à des interblocages d'index.
- Exécutez alter table...disable trigger de façon à utiliser une copie "bulk copy" rapide, comportant une journalisation minimale, au lieu de la copie "bulk copy" lente qui est intégralement enregistrée dans le journal.
- Il peut également être intéressant d'activer l'option trunc log on chkpt de la base de données pour éviter la saturation du journal en cas de chargements importants.
- Les commandes du système d'exploitation peuvent être utilisées pour fractionner le fichier en fichiers distincts puis copier chacun d'eux ou utiliser les drapeaux de ligne de commande -F (première ligne) et -L (dernière ligne) pour bcp.

Quelle que soit la méthode choisie, assurez-vous que le nombre de lignes envoyées à chaque partition est toujours à peu près le même.

Exemple d'utilisation de fichiers distincts :

```
bcp mydb..huge_tab:1 in bigfile1
bcp mydb..huge_tab:2 in bigfile2
...
bcp mydb..huge_tab:10 in bigfile10
```

Cet exemple utilise les arguments de commande de la première ligne et de la dernière ligne pour un seul et même fichier :

```
bcp mydb..huge_tab:1 in bigfile -F1 -L100000
bcp mydb..huge_tab:2 in bigfile -F100001 -L200000
...
bcp mydb..huge_tab:10 in bigfile -F900001 -L1000000
```

Si l'espace est suffisant pour permettre le fractionnement du fichier en plusieurs fichiers, la copie de ces fichiers distincts est généralement beaucoup plus rapide que l'utilisation des arguments de commande de la première ligne et de la dernière ligne car bcp doit analyser chaque ligne du fichier d'entrée lorsqu'il utilise -F et -L. Cette analyse peut être très longue et annuler presque tous les avantages du traitement parallèle.

Copie parallèle et verrous

Si vous lancez plusieurs sessions bcp parallèles, Adaptive Server risque de manquer de verrous.

Lorsque vous effectuez une copie dans une table, bcp pose un verrou d'intention exclusif sur celle-ci et des verrous de page ou de ligne, selon le plan de verrouillage en vigueur. Si vous copiez dans des tables particulièrement volumineuses et en particulier si vous exécutez des copies simultanées dans une table partitionnée, ceci peut demander un très grand nombre de verrous.

Pour éviter une pénurie de verrous, procédez comme suit :

- Affectez au paramètre de configuration number of locks une valeur suffisamment élevée ou
- utilisez l'argument -b *batchsize* de bcp pour copier des batchs de petite taille. Si vous n'utilisez pas l'argument -b, toutes les opérations de copie sont traitées comme un batch unique.

Pour plus d'informations sur bcp, reportez-vous au manuel *Utilitaires*.

Informations sur les partitions

sp_helpartition affiche des informations sur les tables partitionnées. Il affiche ainsi le nombre de pages de données dans la partition et le récapitulatif des informations sur la distribution de données. Emettez sp_helpartition en donnant le nom de table. Cet exemple affiche la distribution des données immédiatement après la création d'un index clusterisé :

```
                sp_helpartition sales
partitionid firstpage   controlpage ptn_data_pages
     ----------- ----------- ----------- --------------
               1        6601        6600           2782
               2       13673       13672           2588
               3       21465       21464           2754
               4       29153       29152           2746
```

```
            5       36737       36736           2705
            6       44425       44424           2732
            7       52097       52096           2708
            8       59865       59864           2755
            9       67721       67720           2851

   (9 rows affected)
   Partitions Average Pages Maximum Pages Minimum Pages Ratio (Max/Avg)
    ---------- ------------- ------------- ------------- -----------------
               9          2735          2851          2588          1.042413
```

sp_helpartition indique l'uniformité de répartition des données entre les partitions. La dernière colonne de la dernière ligne affiche le ratio entre la taille moyenne et la taille maximale de la colonne. Ce ratio permet de déterminer si une requête peut être exécutée en parallèle. Si la taille maximale est deux fois supérieure à la taille moyenne, l'optimiseur ne choisit pas de plan d'exécution parallèle.

On parle de **mauvaise répartition (partition skew)** lorsque les données sont inégalement réparties sur les partitions.

Dans le cas où la table n'est pas partitionnée, sp_helpartition affiche le message "Object is not partitioned". Utilisée sans nom de table, la procédure sp_helpartition affiche les noms de toutes les tables utilisateur de la base de données et le nombre de partitions de chaque table. sp_help appelle sp_helpartition lorsqu'un nom de table est mentionné.

Utilisation de *bcp* pour équilibrer les partitions

Si vous devez charger des données dans une table partitionnée sans index clusterisé et si sp_helpartition indique que certaines partitions contiennent plus de pages que d'autres, vous pouvez utiliser une session de bulkcopy pour équilibrer le nombre de lignes de chaque partition.

L'exemple suivant montre que la table n'a que 487 pages dans une partition et 917 dans une autre :

```
partitionid firstpage   controlpage ptn_data_pages
----------- ----------- ----------- --------------
          1      189825      189824            812
          2      204601      204600            487
          3      189689      189688            917

(3 rows affected)
Partitions Average Pages Maximum Pages Minimum Pages Ratio (Max/Avg)
---------- ------------- ------------- ------------- ---------------
         3           738           917           487        1.242547
```

Il est possible de calculer le nombre de lignes à ajouter à chaque partition de la façon suivante :

- en déterminant le nombre moyen de lignes de chaque partition si celles-ci ont été réparties de manière homogène, c'est-à-dire la somme des lignes actuelles et des lignes à ajouter, divisée par le nombre de partitions ;
- en estimant le nombre actuel de lignes dans chaque partition et en soustrayant ce nombre de la moyenne cible.

La formule peut se résumer comme suit :

```
Lignes à ajouter = (total_lignes_anciennes + total_lignes_nouvelles)
   /#_de_partitions- lignes_dans_cette_partition
```

Dans l'exemple suivant, la procédure utilise des valeurs stockées dans systabstats et syspartitions pour effectuer les calculs :

```
create procedure help_skew @object_name varchar(30), @newrows int
   as
   declare @rows int, @pages int, @rowsperpage int,
           @num_parts int
   select @rows = rowcnt, @pages = pagecnt
       from systabstats
       where id = object_id(@object_name) and indid in (0,1)
   select @rowsperpage = floor(@rows/@pages)
   select @num_parts = count(*) from syspartitions
      where id = object_id("certaine_table")

   select partitionid, (@rows + @newrows)/@num_parts -
       ptn_data_pgs(id, partitionid)*@rowsperpage as rows_to_add
       from syspartitions
       where id = object_id (@object_name)
```

Pour déterminer le nombre de lignes à ajouter à chaque partition de la table customer, dans laquelle 18 000 doivent être copiées, utilisez cette procédure. Les résultats sont indiqués sous la syntaxe.

```
help_skew customer, 18000
partitionid rows_to_add------------------
          1        5255
          2        9155
          3        3995
```

Remarque Si l'inégalité des partitions (partition skew) est importante et si le nombre de lignes à ajouter est faible, cette procédure renvoie des nombres négatifs pour les lignes qui contiennent plus que le nombre moyen de lignes finales.

Par ailleurs, les résultats de requête sont plus précis si les commandes update statistics et update partition statistics ont été exécutées pour mettre à jour les statistiques de la table et des partitions.

Avec les résultats fournis par help_skew, vous pouvez ensuite diviser le fichier contenant les données à charger en plusieurs fichiers de la longueur adéquate ou utiliser les arguments -F (première ligne) et -L (dernière ligne) avec bcp.

Pour plus d'informations, reportez-vous à la section "Utilisation de bcp pour équilibrer les partitions", page 108.

Contrôle de la distribution des données sur les devices avec *sp_helpsegment*

Parfois l'équilibre du nombre de pages de données dans une partition entraîne le déséquilibre de celui sur les devices dans un segment.

Vérification de l'espace libre sur les devices à l'aide de sp_helpsegment Cette partie du rapport de sp_helpsegment pour la même table affichée dans l'exemple de sp_helpartition ci-dessus indique que la distribution de pages sur les devices est encore équilibrée :

```
device                 size            free_pages
---------------------- --------------- -----------
pubtune_detail01       15.0MB          4480
pubtune_detail02       15.0MB          4872
pubtune_detail03       15.0MB          4760
pubtune_detail04       15.0MB          4864
pubtune_detail05       15.0MB          4696
pubtune_detail06       15.0MB          4752
pubtune_detail07       15.0MB          4752
pubtune_detail08       15.0MB          4816
pubtune_detail09       15.0MB          4928
```

Effets d'un déséquilibre des données entre les segments et partitions

Un déséquilibre dans le nombre de pages des différentes partitions indique généralement que les partitions sont saturées sur le device et que des extents ont été affectés sur un autre device physique. On parle alors d'**extent stealing**.

Cela survient lorsque des données sont insérées dans la table à l'aide de la commande insert ou d'une copie "bulk copy" lors de la création d'index clusterisés.

Tout déséquilibre dans le nombre de pages des différentes partitions d'une table peut avoir les conséquences ci-après :

- Les statistiques de la partition utilisées par l'optimiseur s'appuient sur les statistiques affichées par sp_helpartition.

  Aussi longtemps que le partage des données entre les partitions est équilibré, l'optimisation des requêtes parallèles n'est pas perturbée. L'optimiseur sélectionne un balayage de partitions dans la mesure où le nombre de pages de la plus grande partition reste inférieur au double du nombre moyen de pages par partition.

- La parallélisation des E/S peut être réduite en ajoutant des E/S supplémentaires pour certains des devices physiques sur lesquels l'opération d'extent stealing a placé des données.

- La régénération d'un index clusterisé ne permet pas toujours d'obtenir le ré-équilibrage souhaité entre les partitions dans le cas où certaines partitions se trouvent partiellement ou entièrement saturées.

  Pour plus d'informations, reportez-vous à la section "Problèmes relatifs aux devices saturés des tables partitionnées", page 127.

Evaluation du nombre de pages d'une partition

Vous pouvez utiliser la fonction ptn_data_pgs ou les commandes dbcc checktable et dbcc checkdb pour déterminer le nombre de pages de données d'une partition de la table.

Pour plus d'informations sur la commande dbcc, reportez-vous au *Guide d'administration système*.

La fonction ptn_data_pgs retourne le nombre de pages de données d'une partition. Sa syntaxe est la suivante :

ptn_data_pgs(*objet_id*, *partition_id*)

Cet exemple affiche le nombre de pages de chacune des partitions de la table sales :

```
select partitionid,
ptn_data_pgs(object_id("sales"), partitionid) Pages
from syspartitions
where id = object_id("sales")
```

Pour une description complète de ptn_data_pgs, reportez-vous au document *Manuel de référence d'Adaptive Server*.

Dans certains cas, la valeur retournée par ptn_data_pgs peut être inexacte. Si vous soupçonnez une erreur, commencez par exécuter update partition statistics, dbcc checktable, dbcc checkdb ou dbcc checkalloc avant d'utiliser ptn_data_pgs.

Mise à jour des statistiques de partition

Adaptive Server conserve les statistiques sur la répartition des pages dans une table partitionnée et les utilise pour décider de l'utilisation ou non d'un balayage parallèle pour le traitement des requêtes. Lorsque vous partitionnez une table, Adaptive Server stocke des informations sur les pages de données de chaque partition dans la page de contrôle.

Les statistiques se rapportant à une table partitionnée peuvent devenir inexactes dans les cas suivants :

- Le partitionnement de la table a été supprimé puis immédiatement recréé.
- Un grand nombre de lignes a été supprimé.
- Un grand nombre de lignes a été mis à jour et les mises à jour ne s'effectuent pas "sur place".
- Un grand nombre de lignes a été copié par "bulk copy" parallèle dans certaines des partitions.
- Les insertions sont fréquemment annulées.

Si vous estimez que les plans d'exécution des requêtes risquent d'être peu performants en raison de statistiques erronées, exécutez la commande update partition statistics pour mettre à jour les informations de la page de contrôle.

La commande update partition statistics met à jour les informations relatives au nombre de pages de chacune des partitions de la table partitionnée.

La commande update all statistics met également à jour les statistiques relatives aux partitions.

La régénération de l'index clusterisé ou l'exécution de reorg rebuild redistribue automatiquement les données entre les partitions et met à jour les statistiques concernant ces dernières. dbcc checktable, dbcc checkdb et dbcc checkalloc mettent également à jour les statistiques relatives aux partitions à mesure des vérifications.

Syntaxe de *update partition statistics*

La syntaxe à utiliser est :

```
update partition statistics nom_table
    [numéro_partition]
```

Utilisez sp_helpartition pour afficher les numéros de partitions d'une table.

Pour une description complète de update partition statistics, reportez-vous au *Manuel de référence d'Adaptive Server*.

Etapes de partitionnement des tables

Vous devez prévoir le nombre de devices pour le segment de la table afin d'équilibrer les performances d'E/S. Utilisez des disques physiques dédiés au lieu de sections de disques tels que des devices de bases de données pour obtenir les meilleures performances et assurez-vous qu'aucun autre objet ne partage les devices avec la table partitionnée.

Des recommandations pour la création des segments sont fournies dans le *Guide d'administration système.*

Les étapes à suivre pour le partitionnement d'une table dépendent de l'emplacement de la table au début de l'opération. Cette section fournit des exemples pour les situations suivantes :

- La table n'a pas encore été créée et remplie.
- La table existe mais elle ne se trouve pas dans le segment de base de données dans lequel elle doit résider.

- La table se trouve déjà dans le segment sur lequel elle doit résider mais vous souhaitez redistribuer les données afin d'améliorer les performances ou d'ajouter des devices au segment.

Remarque Les sections suivantes décrivent les procédures applicables dans un certain nombre de cas, notamment lorsque d'importantes contraintes d'espace dans la base de données rendent le partitionnement et la création d'index clusterisé très difficiles. Ces procédures complexes ne sont nécessaires que dans ces cas spéciaux. En revanche, si vous disposez de beaucoup de place sur les devices de base de données, le partitionnement et le maintien des performances des tables partitionnées sont simples à réaliser.

Sauvegarde de la base de données après le partitionnement des tables

L'utilisation de la copie "bulk copy" rapide et la création d'index en parallèle permettent d'apporter des modifications à faible journalisation dans la base de données et nécessitent une sauvegarde complète de la base.

Si vous modifiez l'affectation des segments au cours de votre travail sur les tables partitionnées, vous devez également sauvegarder la base de données master puisque les informations d'affectation des segments sont stockées dans sysusages.

La table n'existe pas

Création d'une nouvelle table partitionnée et chargement des données avec bcp :

1 Créez la table sur le segment, à l'aide de la clause on *nom_segment*. Pour plus d'informations sur la création de segments, reportez-vous à la section "Création d'objets sur des segments", page 90.

2 Partitionnez la table, à raison d'une partition pour chacun des devices physiques du segment.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

Remarque Si le fichier de données d'entrée ne suit pas l'ordre de clé clusterisé, que la table va occuper plus de 40 % de l'espace du segment et que vous avez besoin d'un index clusterisé.

Pour plus d'informations, reportez-vous à la section "Procédures spéciales pour les situations difficiles", page 122.

3 Copiez les données dans la table à l'aide d'une copie "bulk copy" parallèle.

Pour consulter des exemples d'utilisation de bcp, reportez-vous à la section "Utilisation de bcp parallèle pour copier des données dans les partitions", page 106.

4 Si vous n'avez pas besoin d'index clusterisé, utilisez sp_helpartition pour vérifier que les données sont réparties de façon équitable entre les partitions.

Pour plus d'informations, reportez-vous à la section "Informations sur les partitions", page 107.

Si vous avez besoin d'un index clusterisé, l'étape suivante peut varier selon que les données ont déjà été triées et qu'elles sont réparties de façon équilibrée entre les partitions.

Si le fichier des données d'entrée se trouvait dans l'ordre de clé d'index et que la répartition des données entre les partitions est équilibrée, vous pouvez utiliser l'option sorted_data et le nom du segment lors de la création de l'index. Cette association d'options s'exécute en série, vérifie l'ordre des clés et construit simultanément l'arbre d'index. Il n'est pas nécessaire de copier des données dans l'ordre des clés ; il n'y a donc pas d'équilibrage de la charge. Si vous n'avez pas besoin de contraintes d'intégrité référentielle, vous pouvez utiliser create index.

Pour plus d'informations, reportez-vous à la section "Utilisation de drop index et decreate clustered index", page 104.

Pour créer un index clusterisé avec des contraintes d'intégrité référentielle, utilisez alter table...add constraint.

Pour plus d'informations, reportez-vous à la section "Utilisation de contraintes et de la commande alter table", page 105.

Si vos données n'étaient pas placées dans l'ordre des clés d'index lors de leur copie, vérifiez que l'espace est suffisant pour créer l'index clusterisé lors de la copie des données.

Utilisez sp_spaceused pour afficher la taille de la table et sp_helpsegment pour vérifier la taille du segment. La création d'un index clusterisé demande environ 120 % de l'espace occupé par la table.

Si l'espace est insuffisant, procédez comme indiqué à la section "Si l'espace est insuffisant pour régénérer l'index clusterisé", page 119.

5 Créez les éventuels index non clusterisés.

6 Sauvegardez la base de données.

La table existe déjà dans un autre emplacement de la base de données

Si la table existe sur le segment par défaut ou sur un autre segment de la base de données, suivez les étapes ci-après pour déplacer les données vers la partition et les distribuer uniformément :

1 Si la table est déjà partitionnée mais que le nombre de ses partitions diffère du nombre de devices sur le segment cible, annulez la partition de la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

2 Partitionnez la table, en créant autant de partitions qu'il existe de devices sur le segment cible.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

3 S'il existe un index clusterisé, supprimez-le. En fonction de la méthode de création de l'index, vous pouvez utiliser soit drop index soit alter table...drop constraint.

Pour plus d'informations, reportez-vous aux sections "Utilisation de drop index et decreate clustered index", page 104 ou alter table...drop constraint et "Utilisation de contraintes et de la commande alter table", page 105.

4 Créez ou re-créez l'index en utilisant la clause on *nom_segment*. Lorsque le nom du segment diffère de celui du segment courant dans lequel la table est stockée, la création de l'index clusterisé entraîne un tri parallèle et répartit les données de façon uniforme entre les partitions à mesure qu'elle copie les lignes en fonction de l'ordre de l'index. Cette étape régénère les index non clusterisés de la table.

   Pour plus d'informations, reportez-vous à la section "Répartition uniforme des données sur les partitions", page 103.

5 Si vous n'avez pas besoin de l'index clusterisé, supprimez-le.

6 Sauvegardez la base de données.

La table existe dans le segment

Si la table existe dans le segment, il peut être nécessaire de :

- Redistribuer les données en régénérant un index clusterisé ou en utilisant une copie "bulk copy" ou
- accroître le nombre de devices du segment.

Redistribution des données

S'il est nécessaire de redistribuer les données entre les partitions, la méthode choisie dépendra de l'espace occupé par les données sur la partition. Si l'espace occupé par la table est inférieur à 40 ou 45 % de l'espace du segment, vous pouvez créer un index clusterisé pour redistribuer les données.

Si la table occupe plus de 40 à 45 % de l'espace du segment, vous devez copier les données à l'extérieur de la table au moyen d'un "bulk copy", tronquer la table puis y réintégrer les données. Le choix des étapes est différent si vous avez besoin d'un index clusterisé ou non et si les données sont déjà ou non dans un ordre de clé clusterisé.

Utilisez les procédures sp_helpsegment et sp_spaceused pour vérifier que l'espace est suffisant pour créer un index clusterisé sur le segment.

Si l'espace est suffisant pour créer ou régénérer l'index clusterisé

Si l'espace est suffisant, suivez la procédure indiquée à la section "Répartition uniforme des données sur les partitions", page 103. Si vous n'avez pas besoin de l'index clusterisé, vous pouvez l'annuler sans modifier la répartition des données.

Sauvegardez la base de données après avoir créé l'index clusterisé.

Si l'espace est insuffisant dans le segment mais disponible sur un autre emplacement du serveur

Si l'espace est suffisant pour copier la table, copiez cette dernière dans un autre emplacement puis régénérez l'index clusterisé pour réintégrer les données dans le segment cible.

La procédure utilisée pourra varier, selon que l'espace de stockage temporaire se trouve :

- sur le segment default (par défaut) de la base de données ou dans tempdb,
- sur d'autres segments de la base de données.

Utilisation du segment par défaut ou de tempdb

1 Utilisez select into pour copier la table dans le segment default ou dans tempdb.

```
select * into temp_sales from sales
```

ou

```
select * into tempdb..temp_sales from sales
```

2 Supprimez la table d'origine.

3 Partitionnez la copie de la table.

4 Créez l'index clusterisé dans le segment sur lequel la table doit résider.

5 Utilisez sp_rename pour restituer à la table son nom d'origine.

6 Sauvegardez la base de données.

Utilisation de l'espace d'un autre segment

Si l'espace disponible se trouve sur un autre segment :

1 Créez un index clusterisé, en indiquant le segment sur lequel se trouve cet espace ; la table est transférée dans cet emplacement.

2 Supprimez l'index.

3 Recréez l'index clusterisé, en indiquant le segment dans lequel les données doivent résider.

4 Sauvegardez la base de données.

Si l'espace est insuffisant pour régénérer l'index clusterisé

Si l'espace est insuffisant et que vous avez besoin de recréer un index clusterisé sur les tables :

1 Copiez les données hors de la table au moyen de "bulk copy".

2 Annulez le partitionnement de la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

3 Tronquez la table à l'aide de truncate table.

4 Supprimez l'index clusterisé à l'aide de drop index ou dealter table...drop constraint.

Supprimez également les index non clusterisés pour éviter les interblocages au cours des sessions de copie "bulk copy" parallèle.

Pour plus d'informations, reportez-vous à la section "Répartition uniforme des données sur les partitions", page 103.

5 Repartitionnez la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

6 Copiez les données dans la table à l'aide d'une copie "bulk copy" parallèle. Veillez à copier les données vers chacun des segments en suivant l'ordre des clés d'index et indiquez le nombre de lignes de chaque partition pour obtenir une répartition uniforme.

Pour plus d'informations, reportez-vous à la section "Utilisation de bcp parallèle pour copier des données dans les partitions", page 106.

7 Régénérez l'index à l'aide des clauses with sorted_data et on *nom_segment*. Cette commande effectue un balayage série de la table et génère l'arbre d'index sans copier les données.

N'utilisez pas de clauses nécessitant une copie des données (fillfactor, ignore_dup_row et max_rows_per_page).

8 Régénérez les éventuels index non clusterisés.

9 Sauvegardez la base de données.

Si l'espace est insuffisant et que l'index clusterisé n'est pas nécessaire

S'il n'existe pas d'index clusterisé et qu'il n'est pas nécessaire d'en créer un :

1 Copiez les données à l'extérieur de la table au moyen d'une copie "bulk copy".

2 Annulez le partitionnement de la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

3 Tronquez la table à l'aide de truncate table.

4 Supprimez les index non clusterisés pour éviter d'éventuels interblocages au cours des sessions de copie "bulk copy" parallèle.

5 Repartitionnez la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

6 Copiez les données au moyen de la copie "bulk copy" parallèle.

Pour plus d'informations, reportez-vous à la section "Utilisation de bcp parallèle pour copier des données dans les partitions", page 106.

7 Régénérez les éventuels index non clusterisés.

8 Sauvegardez la base de données.

S'il n'y a pas d'index clusterisé, que l'espace est insuffisant et qu'un index clusterisé est nécessaire

Pour modifier les clés d'index sur l'index clusterisé d'une table partitionnée ou pour créer un index sur une table qui n'a pas été stockée comme une table sans index, un tri au niveau du système d'exploitation permet d'accélérer le processus.

La création d'un index clusterisé implique de disposer d'un espace équivalent à 120 % de l'espace utilisé par la table pour créer une copie des données et générer l'arbre d'index.

Si vous avez accès à un utilitaire de tri au niveau du système d'exploitation :

1 Copiez les données à l'extérieur de la table au moyen d'une copie "bulk copy".

2 Annulez le partitionnement de la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

3 Tronquez la table à l'aide de truncate table.

4 Supprimez les index non clusterisés pour éviter d'éventuels interblocages au cours des sessions de copie "bulk copy" parallèle.

5 Repartitionnez la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

6 Procédez à un tri sur ce fichier à partir du système d'exploitation.

7 Copiez les données au moyen de la copie "bulk copy" parallèle.

Pour plus d'informations, reportez-vous à la section "Utilisation de bcp parallèle pour copier des données dans les partitions", page 106.

8 Régénérez l'index à l'aide des clauses sorted_data et on *nom_segment*. Cette commande effectue un balayage série de la table et génère l'arbre d'index sans copier les données.

N'utilisez pas de clauses nécessitant une copie des données (fillfactor, ignore_dup_row et max_rows_per_page).

9 Régénérez les éventuels index non clusterisés.

10 Sauvegardez la base de données.

Ajout de devices dans un segment

Si vous avez besoin d'ajouter un device dans un segment, procédez comme indiqué ci-après :

1 Vérifiez l'espace disponible sur les devices liés au segment au moyen de la commande sp_helpsegment.

Si l'espace sur l'un des devices est extrêmement réduit, reportez-vous à la section "Problèmes relatifs aux devices saturés des tables partitionnées", page 127.

Il peut être nécessaire de copier les données hors de la table puis de les y réintégrer pour obtenir une répartition équitable des données.

2 Initialisez chaque device au moyen de disk init puis associez-le à la base de données à l'aide de la commande alter database.

3 Utilisez sp_extendsegment *nom_segment*, *nom_device* pour étendre le segment à chacun des devices. Supprimez les segments default et system de chaque device.

4 Annulez le partitionnement de la table.

   Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

5 Repartitionnez la table, en indiquant le nouveau nombre de devices du segment.

   Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

6 S'il existe un index clusterisé, supprimez-le puis recréez-le. Utilisation de l'option sorted_data.

   Pour plus d'informations, reportez-vous à la section "Répartition uniforme des données sur les partitions", page 103.

7 Sauvegardez la base de données.

Procédures spéciales pour les situations difficiles

Ces techniques sont plus complexes que celles présentées au début de ce chapitre.

Index clusterisés sur des tables de grande taille

Si vous avez besoin de créer un index clusterisé sur une table qui occupera plus de 40 à 45 % de l'espace du segment et que le fichier des données d'entrée n'a pas été trié par clés de l'index clusterisé, la procédure ci-après vous permet d'obtenir une répartition équitable des données à condition que les données que vous copiez au cours de l'étape 6 contiennent un échantillon représentatif des données.

1 Copiez les données hors de la table.

2 Annulez le partitionnement de la table.

   Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

3 Tronquez la table.

4 Repartitionnez la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

5 Supprimez l'index clusterisé et tous les éventuels index non clusterisés. En fonction de la méthode de création de l'index, vous pouvez utiliser l'une ou l'autre des commandes drop index.

Pour plus d'informations, reportez-vous à la section "Utilisation de drop index et decreate clustered index", page 104 ou alter table...drop constraint et "Utilisation de contraintes et de la commande alter table", page 105.

6 Utilisez la copie "bulk copy" parallèle pour copier un volume de données occupant environ 40 % du segment. Il doit s'agir d'un échantillon représentatif des valeurs figurant dans la ou les colonnes de clés de l'index clusterisé.

La copie de 40 % des données donne de meilleures chances d'obtenir de bons résultats que des volumes de données plus réduits et cette partie de la copie "bulk copy" peut être effectuée en parallèle ; la deuxième opération de copie "bulk copy" ne doit pas être effectuée en parallèle.

Pour plus d'informations, reportez-vous à la section "Utilisation de bcp parallèle pour copier des données dans les partitions", page 106.

7 Créez l'index clusterisé sur le segment, n'utilisez pas la clause sorted_data.

8 Utilisez la copie bcp non parallèle dans une session unique pour copier le reste des données. L'index clusterisé dirige les lignes vers les partitions adéquates.

9 Utilisez la procédure sp_helppartition pour contrôler la distribution des pages de données sur les partitions et sp_helpsegment pour contrôler la distribution des pages sur le segment.

10 Créez les éventuels index non clusterisés.

11 Sauvegardez la base de données.

L'un des inconvénients de cette méthode est que, une fois que l'index clusterisé existe, la seconde copie "bulk copy" risque de provoquer un fractionnement des pages de données, entraînant l'occupation d'un espace légèrement supérieur dans la base de données. Toutefois, une fois que l'index clusterisé existe et que toutes les données ont été chargées, les activités de maintenance suivantes peuvent s'effectuer au moyen de méthodes plus simples et plus rapides.

Alternative pour les index clusterisés

Cette procédure peut être utile dans les cas suivants :

- Les données de la table occupent plus de 40 à 45 % du segment.
- Les données de la table ne sont pas triées dans l'ordre de clés clusterisé et vous devez créer un index clusterisé.
- Vous n'obtenez pas les résultats escomptés lorsque vous tentez de charger un échantillon de données représentatif, comme indiqué à la section "Index clusterisés sur des tables de grande taille", page 122.

Les étapes décrites ci-après permettent dans la plupart des cas une redistribution homogène des données mais elles nécessitent quelques précautions :

1 Trouvez la valeur minimale de la colonne de clés pour l'index clusterisé :

```
select min(order_id) from orders
```

2 Si l'index clusterisé existe, supprimez-le. Supprimez également les éventuels index non clusterisés.

Reportez-vous à la section "Utilisation de drop index et decreate clustered index", page 104 ou à "Utilisation de contraintes et de la commande alter table", page 105.

3 Exécutez la commande :

```
set sort_resources on
```

Cette commande désactive les commandes create index. Ainsi, les prochaines commandes create index entraîneront l'affichage d'informations sur le mode de tri, sans créer l'index.

4 Lancez la commande permettant la création de l'index clusterisé et notez le nombre de partitions et les valeurs obtenues en résultat. L'exemple ci-après indique les valeurs à utiliser pour une table utilisant quatre partitions :

```
create clustered index order_cix
    on orders(order_id)
The Create Index is done using Parallel Sort
Sort buffer size: 1500
Parallel degree: 25
Number of output devices: 3
Number of producer threads: 4
Number of consumer threads: 4
The distribution map contains 3 element(s) for 4
partitions.
Partition Element: 1

450977
Partition Element: 2

903269
Partition Element: 3

1356032
Number of sampled records: 2449
```

Ces valeurs, associées à la valeur minimale obtenue à l'étape n°1, sont les valeurs des clés utilisées comme délimiteurs par l'opération de tri lors de l'affectation de lignes à chaque partition.

5 Effectuez une copie "bulk copy" des données hors de la table, en utilisant le mode caractères.

6 Annulez le partitionnement de la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...unpartition", page 102.

7 Tronquez la table.

8 Repartitionnez la table.

Pour plus d'informations, reportez-vous à la section "Syntaxe alter table...partition", page 102.

9 Dans le fichier de données obtenu en résultat, trouvez la valeur de clé minimale et chacune des valeurs clés identifiées à l'étape n° 4. Copiez ces valeurs dans un autre fichier et supprimez-les du fichier de sortie.

10 Copiez ces lignes dans la table, en utilisant la copie "bulk copy" parallèle pour les placer dans le segment qui convient. Dans le cas où les valeurs indiquées plus haut sont utilisées, le fichier peut contenir les informations ci-après :

```
1             Jones         ...
450977        Smith         ...
903269        Harris        ...
1356032       Wilder        ...
```

Les commandes bcp se présentent alors comme suit :

```
bcp testdb..orders:1 in keyrows -F1 -L1
bcp testdb..orders:2 in keyrows -F2 -L2
bcp testdb..orders:3 in keyrows -F3 -L3
bcp testdb..orders:4 in keyrows -F4 -L4
```

A l'issue de cette opération, vous obtenez une ligne sur la première page de chaque partition (ligne identique à celle que la création de l'index aurait affecté à cet emplacement).

11 Désactivez set sort_resources et créez l'index clusterisé dans le segment, à l'aide de l'option with sorted_data.

N'utilisez aucune clause forçant la création de l'index pour copier les lignes de données.

12 Procédez à une copie "bulk copy" pour transférer les données dans la table.

Utilisez une session unique, non parallèle. Si la table dispose d'un index clusterisé, vous ne devez pas indiquer de partition pour la copie "bulk copy". Par ailleurs, l'exécution de plusieurs sessions entraîne des risques d'interblocage.

L'index clusterisé dirige les pages vers la partition adéquate.

13 Utilisez sp_helpartition pour vérifier l'équilibre des pages de données entre les partitions et sp_helpsegment pour vérifier celui sur les segments.

14 Créez les éventuels index non clusterisés.

15 Sauvegardez la base de données.

Même si cette méthode permet d'utiliser pratiquement toutes les pages d'une partition, elle présente des inconvénients.

- La table entière doit être copiée au moyen d'une opération de "bulk copy" lente.
- L'existence de l'index clusterisé est susceptible d'entraîner un fractionnement des pages de données si elles utilisent le verrouillage APL, de sorte que davantage d'espace peut être requis.

Problèmes relatifs aux devices saturés des tables partitionnées

En cas de saturation des partitions, le simple ajout de disques et la régénération des index ne permettent pas toujours de résoudre les problèmes de répartition de la charge. Dans le cas où un device physique servant de support à une partition devient entièrement saturé, l'étape de copie des données lors de la regénération d'index ne permet pas de copier les données vers ce device physique.

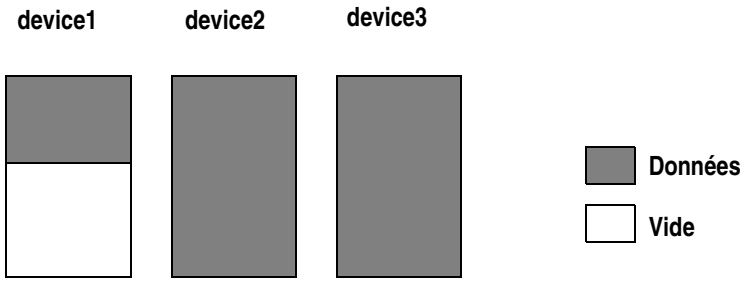
Si un device physique est presque saturé, la régénération de l'index clusterisé ne permet pas toujours d'obtenir une bonne répartition de la charge.

Ajout de disques en cas de saturation des devices

En cas de saturation d'un device physique, la génération d'un index clusterisé entraîne la création de deux partitions sur l'un des autres devices physiques. La figure 5-2 et la figure 5-3 représentent cette situation.

Les devices 2 et 3 sont entièrement saturés, comme indiqué à la figure 5-2.

Figure 5-2 : Table comportant 3 partitions sur 3 devices

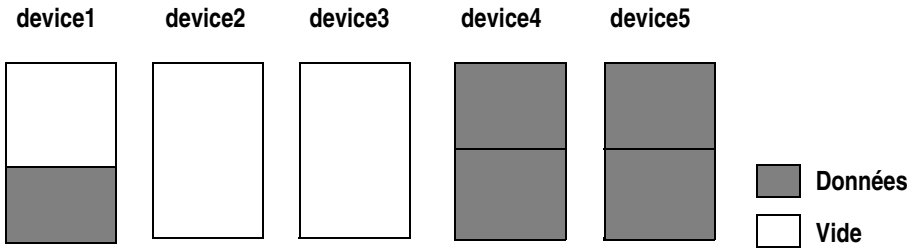


L'ajout de deux devices, le repartitionnement de la table afin d'utiliser cinq partitions et la suppression et régénération de l'index clusterisé produisent les résultats ci-après :

Device 1	Une partition, occupée à 40 % environ.
Devices 2 et 3	Vides. Ces devices ne disposaient pas d'espace disponible lors du lancement de la commande create index, de sorte qu'il n'a pas été possible de créer une partition sur le device pour la copie de l'index.
Devices 4 et 5	Chaque device possède deux partitions qui sont toutes deux occupées à 100 %.

La figure 5-3 répertorie les résultats.

Figure 5-3 : Devices et partitions après la commande create index

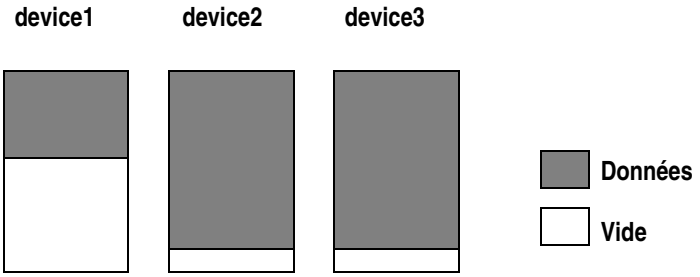


Dès lors qu'un device est entièrement saturé, la seule solution consiste à effectuer une copie "bulk copy" des données hors de la table, à tronquer cette table puis à réintégrer les données dans la table.

Ajout de disques en cas de devices presque saturés

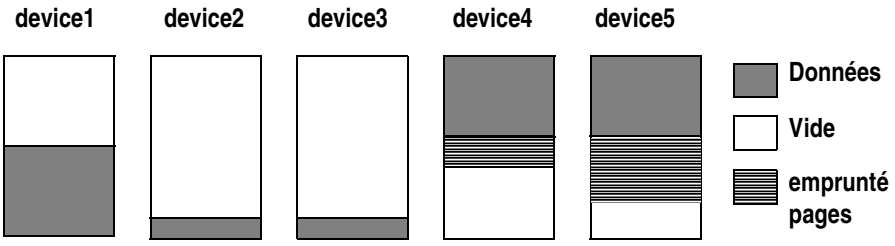
Lorsqu'un device est proche de la saturation, la régénération de l'index clusterisé ne permet pas de répartir équitablement les données entre les devices. Le device qui est proche de la saturation est utilisé pour stocker une petite partie de la partition tandis que les autres affectations d'espace de la partition utilisent des extents sur d'autres devices (extent stealing). La figure 5-4 représente une table dont les devices de données sont pratiquement saturés.

Figure 5-4 : Les partitions occupent presque entièrement les devices



Après l'ajout de devices et la régénération de l'index clusterisé, le résultat obtenu peut être comparable à celui représenté à la figure 5-5.

Figure 5-5 : Extent stealing et répartition inégale des données



Une fois que les partitions s'appuyant sur device 2 et device 3 ont utilisé l'espace qui restait disponible, elles ont commencé à emprunter des extents à device 4 et à device 5.

Dans ce cas, une seconde régénération de l'index peut entraîner une répartition plus équitable des données. Cependant, si l'un des devices est presque entièrement occupé en raison du phénomène d'extent stealing, une nouvelle création d'index ne permet pas de résoudre le problème.

Procédez à une copie "bulk copy" pour transférer les données hors de la table puis les y réintégrer. Il s'agit de la seule solution permettant de remédier à cette forme de déséquilibre.

Pour éviter des situations de ce genre, contrôlez l'utilisation de l'espace sur les devices et ajoutez-en suffisamment tôt si nécessaire.

Problèmes de maintenance et tables partitionnées

Les activités de maintenance des tables partitionnées peuvent varier en fonction de la fréquence et du type des mises à jour effectuées dans ces tables.

Certaines tables partitionnées nécessitent peu de maintenance :

- Les tables en lecture seule ou faisant l'objet d'un nombre très réduit de mises à jour. Dans le second cas, seules les vérifications périodiques de la répartition sont nécessaires.
- Les tables dans lesquelles les insertions sont bien réparties entre les partitions. Les insertions aléatoires dans des tables partitionnées sans index et les insertions réparties de façon homogène en raison d'une clé d'index clusterisé qui place les lignes sur des partitions différentes n'entraînent pas de mauvaise répartition des pages.

  Si les modifications des données entraînent une fragmentation de l'espace et des pages de données partiellement remplies, la régénération de l'index clusterisé peut être périodiquement nécessaire.
- Les tables sans index, dans lesquelles les insertions s'effectuent par copie "bulk copy". Vous pouvez faire appel à la copie "bulk copy" parallèle pour orienter les nouvelles données dans des partitions spécifiques, afin de conserver une bonne répartition de la charge.

Les tables partitionnées nécessitant de fréquentes opérations de suivi et de maintenance sont les tables à index clusterisé qui ont tendance à diriger les nouvelles lignes vers un sous-ensemble des partitions. Un index de clés ascendant est susceptible de demander des opérations de maintenance plus fréquentes.

Contrôles de maintenance périodiques portant sur les tables partitionnées

Le suivi standard des tables partitionnées doit comprendre les vérifications ci-après, en plus des vérifications habituelles de cohérence de la base de données :

- Vérifiez la répartition des données entre les partitions, à l'aide de la procédure sp_helpartition.

  Si certaines partitions sont beaucoup plus importantes ou beaucoup moins importantes que la moyenne, régénérez l'index clusterisé pour redistribuer les données.

- Vérifiez la répartition de l'espace disponible sur les disques sous-jacents, à l'aide de la procédure sp_helpsegment.

- Si vous régénérez l'index clusterisé afin de redistribuer les données pour optimiser les performances des requêtes parallèles, recherchez les devices dont l'occupation est proche de 50 %.

  L'ajout de disques suffisamment tôt, avant que les devices ne soient entièrement saturés, permet d'éviter les procédures complexes décrites plus haut dans ce chapitre.

- Vérifiez les pages libres sur chaque device à l'aide de la procédure sp_helpsegment ou vérifiez les kilo-octets libres à l'aide de sp_helpdb.

En outre, si les tables partitionnées font l'objet des activités décrites à la section "Mise à jour des statistiques de partition", page 112, exécutez update partition statistics.

Il peut être nécessaire de régénérer l'index clusterisé des tables partitionnées pour les raisons ci-après :

- La clé d'index a tendance à affecter des insertions à un sous-ensemble des partitions.
- La suppression a tendance à faire disparaître les données d'un sous-ensemble de partitions, ce qui entraîne un déséquilibre des E/S et des balayages basés sur les partitions.
- La table comporte de nombreuses insertions, mises à jour et suppressions, ce qui signifie que de nombreuses pages de données sont partiellement remplies. Cette situation entraîne un gaspillage d'espace, tant sur le disque que dans le cache et multiplie les opérations d'E/S puisqu'il est nécessaire d'accéder à un plus grand nombre de pages pour de nombreuses requêtes.

CHAPITRE 6

Conception de bases de données

Cette section fournit des informations de base sur la conception des bases de données que les administrateurs et les concepteurs trouveront utiles. Elle traite également les formes normales pour la normalisation et la dénormalisation des bases de données.

Il faut affronter certains des concepts les plus importants et fournir des conseils pour vous aider à passer d'une conception logique à une conception physique d'une base de données sur Adaptive Server.

Sujet	Page
Conception de base	133
Normalisation	135
Dénormalisation pour l'optimisation des performances	140

Conception de base

La conception d'une base de données consiste à transposer les spécifications et les modèles réels en un modèle de base de données qui respecte ces spécifications.

La normalisation d'une base de données est une approche pour la structuration des informations afin d'éviter les redondances et les incohérences et en promouvoir une maintenance, un stockage et une mise à jour efficaces. Plusieurs "règles" ou niveaux de normalisation sont acceptables, chacun étant un perfectionnement du précédent.

Trois formes sont couramment utilisées : première forme normale, seconde forme normale et troisième forme normale. Les premières formes normales, les moins structurées, sont des groupes d'enregistrement parmi lesquels chaque champ (colonne) contient des informations uniques et non répétées. La seconde et la troisième forme normales détaillent les premières formes normales, en les séparant en des tables différentes pour définir ensuite des relations plus fines entre les champs.

Pour les bases de données relationnelles telles qu'Adaptive Server, la conception standard crée des tables en troisième forme normale.

Lorsque vous convertissez un modèle entité-relation de troisième forme normale (3NF) en modèle relationnel :

- Les relations sont converties en tables.
- Les attributs sont convertis en colonnes.
- Les relations sont converties en références de données (références de clé étrangère et primaire).

Conception physique des bases de données pour Adaptive Server

Selon les spécifications et les contraintes relatives aux accès, mettez en œuvre la conception physique des bases de données comme suit :

- dénormalisez, si nécessaire ;
- partitionnez les tables, si nécessaire ;
- regroupez les tables dans des bases de données, si nécessaire ;
- définissez l'utilisation des segments ;
- définissez l'utilisation des devices ;
- mettez en œuvre l'intégrité référentielle des contraintes.

Tailles des pages logiques

Dans Adaptive Server la taille des pages est variable. Vous devez être attentif lors de la définition des tailles de page.

Il est dangereux d'utiliser les devices les plus grands sur une plate-forme limitée à 2 Go. Si vous tentez de configurer un device logique plus grand que 2 Go, si Adaptive Server ne prend pas en charge les grands devices, vous pouvez rencontrer les problèmes suivants :

- Altération des données contenues dans les bases de données (certaines versions n'émettent pas de message d'erreur).
- Impossibilité de décharger ou charger des données dans la base de données.

Normalisation

Lorsqu'une table est normalisée, ses colonnes non-clé dépendent de la clé utilisée.

Dans le modèle relationnel, les tables sont en troisième forme normale. La conception physique normalisée permet une maintenance aisée et ne pose aucun problème de compréhension aux développeurs.

Cependant, une conception totalement normalisée ne donne pas toujours des performances optimales. Sybase conseille de mettre au point une conception adaptée à la troisième forme normale et de dénormaliser lorsque des problèmes de performances apparaissent.

Niveaux de normalisation

Chaque niveau de normalisation dépend du niveau précédent. Par exemple, pour être conforme à la seconde forme normale, les entités doivent être en première forme normale.

Il peut être nécessaire d'étudier à fond les tables d'une base de données pour vérifier si la base de données est normalisée. Il peut être nécessaire de modifier le mode de normalisation en dénormalisant certaines données avant d'appliquer une définition différente de normalisation.

Utilisez les informations suivantes pour vérifier si une base de données a été normalisée ou non puis pour définir les formes normales que vous souhaitez utiliser.

Avantages de la normalisation

La normalisation produit des tables de plus petite taille avec des lignes moins longues :

- plus de lignes par page (moins d'E/S logiques)
- plus de lignes par E/S (performances accrues)
- plus de lignes contenues en mémoire cache (moins d'E/S physiques)

La normalisation présente les avantages suivants :

- La recherche, le tri et la création d'index sont plus rapides, car les tables sont plus petites et une page de données peut contenir plus de lignes.
- Vous disposez généralement de plus de tables.

  Vous pouvez avoir plus d'index clusterisés (un par table), ce qui vous donne plus de souplesse pour optimiser les requêtes.
- La recherche dans un index est souvent plus rapide, car les index sont plus petits.
- Un plus grand nombre de tables permet une meilleure utilisation des segments pour contrôler l'emplacement physique des données.
- Le nombre des index par table est généralement moindre et les commandes de modification de données sont donc plus rapides.
- Le nombre de valeurs NULL et de données redondantes est réduit, ce qui rend votre base de données plus compacte.
- En l'absence de données redondantes, les triggers s'exécutent plus rapidement.
- Les anomalies de modification de données sont réduites.
- La normalisation rend la maintenance plus facile et il est possible de la modifier selon vos besoins.

Bien que les bases de données complètement normalisées aient besoin de plus de jointures, celles-ci sont généralement très rapides si des index sont disponibles dans les colonnes correspondantes.

Adaptive Server est optimisé de façon à conserver les niveaux supérieurs de l'index dans le cache, de sorte que chaque jointure n'effectue qu'une ou deux E/S physiques pour chaque ligne concernée.

Le coût de la recherche de lignes déjà présentes dans le cache de données est extrêmement faible.

Première forme normale

Les règles à respecter dans la première forme normale sont les suivantes :

- Chaque colonne doit être atomique. Elle ne peut pas être subdivisée en plusieurs sous-colonnes.

- Vous ne pouvez pas disposer de colonnes contenant plusieurs valeurs ou groupes répétitifs.
- Chaque position de ligne et de colonne ne peut contenir qu'une valeur.

La table de la figure 6-1 ne respecte pas la première forme normale, car la colonne dept_no contient un groupe répétitif :

Figure 6-1 : Table ne respectant pas la première forme normale

Employee (emp_num, emp_lname, dept__no)

Employee

emp_num	emp_lname	dept_
10052	Jones	A10 C66
10101	Sims	D60

Grpe répétitif

La normalisation crée deux tables et place dept_no dans la deuxième :

Figure 6-2 : Création de deux tables suite au non-respect de la première forme normale

Employee (emp_num, emp_lname) **Emp_dept (emp_num, dept_no)**

Employee

emp_num	emp_lname
10052	Jones
10101	Sims

Emp_dept

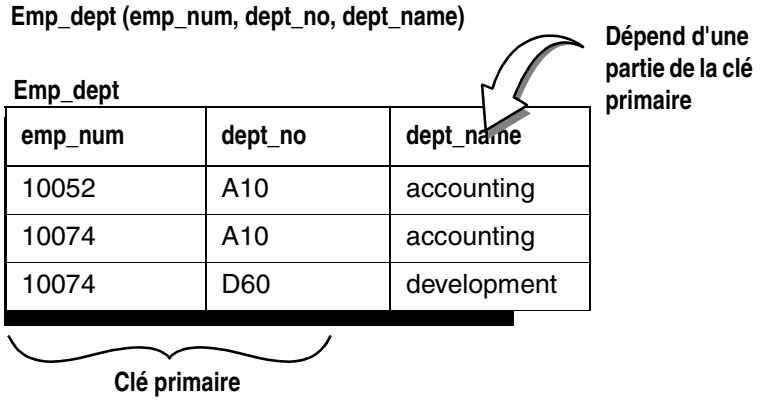
emp_num	dept_no
10052	A10
10052	C66
10101	D60

Deuxième forme normale

Pour qu'une table soit conforme à la deuxième forme normale, chaque champ non-clé doit dépendre de la clé primaire tout entière et non d'une partie de clé primaire composée. Lorsqu'une base de données ne contient que des clés primaires à champ unique, elle est automatiquement en deuxième forme normale.

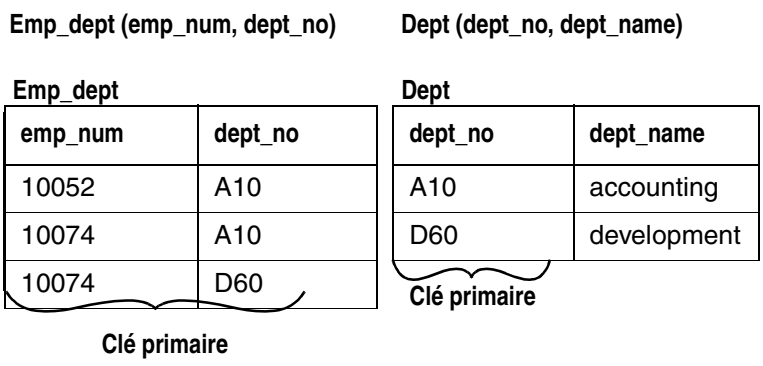
Dans la table de la figure 6-3, la clé primaire est une clé composée sur emp_num et dept_no. Cependant, la valeur de dept_name dépend uniquement de dept_no, et non de la clé primaire tout entière.

Figure 6-3 : Table ne respectant pas la deuxième forme normale



Pour normaliser cette table, placez dept_name dans une deuxième table comme dans la figure 6-4.

Figure 6-4 : Création de deux tables suite au non-respect de la seconde forme normale



Troisième forme normale

Pour qu'une table soit conforme à la troisième forme normale, un champ non-clé ne doit pas dépendre d'un autre champ non-clé.

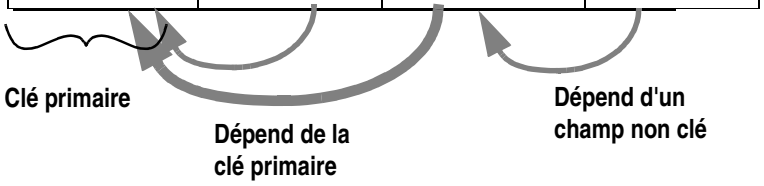
La table de la figure 6-5 ne respecte pas la troisième forme normale car le champ mgr_lname dépend du champ mgr_emp_num, qui n'est pas un champ clé.

Figure 6-5 : Table ne respectant pas la troisième forme normale

Dept (dept_no, dept_name, mgr_emp_num, mgr_lname)

Dept

dept_no	dept_name	mgr_emp_num	mgr_lname
A10	accounting	10073	Johnson
D60	development	10089	White
M80	marketing	10035	Dumont



La solution consiste à diviser la table Dept en deux tables, comme illustré à la figure 6-6. Dans ce cas, la table Employees stocke déjà ces informations ; par conséquent, si vous supprimez le champ mgr_lname de Dept, la table est conforme à la troisième forme normale.

Figure 6-6 : Création de deux tables suite au non-respect de la troisième forme normale

Dept (dept_no, dept_name, mgr_emp_num)

Dept

dept_no	dept_name	mgr_emp_num
A10	accounting	10073
D60	development	10089
M80	marketing	10035

Clé primaire

Employee (emp_num, emp_lname)

Employee

emp_num	emp_lname
10073	Johnson
10089	White
10035	Dumont

Clé primaire

Dénormalisation pour l'optimisation des performances

Une fois que vous avez normalisé votre base de données, vous pouvez exécuter des tests pour vérifier les performances. Il peut être nécessaire de dénormaliser pour des requêtes et/ou applications spécifiques.

Dénormalisation :

- peut être effectuée avec des tables ou des colonnes ;
- suppose une normalisation préalable ;
- implique de bien connaître la façon dont les données sont utilisées.

Vous pouvez souhaiter dénormaliser si :

- La totalité, ou presque, des requêtes les plus fréquentes nécessitent un accès à l'ensemble des données jointes.
- La plupart des applications effectuent des balayages de table lors des jointures de tables.
- La complexité de calcul des colonnes dérivées requiert des tables temporaires ou des requêtes très complexes.

Risques

Pour dénormaliser vous devez avoir une connaissance complète de l'application. En outre, vous ne devez dénormaliser qu'en cas de nécessité, à cause de problèmes de performances.

Par exemple, la colonne ytd_sales de la table titles de la base de données pubs2 est une colonne dénormalisée gérée par un trigger sur la table salesdetail. Les mêmes valeurs peuvent être obtenues en utilisant la requête suivante :

```
select title_id, sum(qty)
    from salesdetail
    group by title_id
```

Pour obtenir les valeurs agrégées et le titre du document, vous devez disposer d'une jointure avec la table titles. Pour ce faire, lancez la requête suivante :

```
select title, sum(qty)
    from titles t, salesdetail sd
    where t.title_id = sd.title_id
    group by title
```

Si vous effectuez fréquemment cette requête, il est avantageux de dénormaliser cette table. Toutefois ce n'est pas sans conséquence : vous devez créer un trigger d'insertion/mise à jour/suppression sur la table salesdetail pour conserver les valeurs d'agrégat dans la table titles.

L'exécution de ce trigger et les modifications apportées à titles entraînent un coût supplémentaire de traitement pour chaque modification de la colonne qty dans salesdetail.

Cette situation reflète l'antagonisme existant entre les applications décisionnelles (DSS), qui nécessitent souvent des synthèses de nombreuses données, et les applications transactionnelles (OLTP), qui effectuent des modifications de données discontinues.

La dénormalisation favorise souvent un type de traitement au détriment des autres.

Quelle que soit la forme de dénormalisation choisie, elle peut générer des problèmes de cohérence des données qui doivent être soigneusement répertoriés et pris en considération lors de la conception d'applications.

Inconvénients

La dénormalisation présente les inconvénients suivants :

- Généralement, elle accélère les recherches mais ralentit les modifications de données.
- Elle est toujours spécifique d'une application et doit être réévaluée si l'application évolue.
- Elle peut entraîner un accroissement de la taille des tables.
- Dans certains cas, elle simplifie la programmation, alors que dans d'autres elle la rend plus complexe.

Avantages sur les performances

La dénormalisation peut améliorer les performances en :

- réduisant le nombre de jointures requises ;
- limitant le nombre de clés étrangères sur les tables ;
- diminuant le nombre d'index, ce qui économise l'espace de stockage et réduit le temps de modification des données ;
- calculant au préalable les valeurs d'agrégat, c'est-à-dire au moment des modifications de données plutôt qu'au moment de la sélection ;
- réduisant, dans certains cas, le nombre de tables.

Données pour la dénormalisation

Lorsque vous décidez de dénormaliser, vous devez analyser, pour les applications de votre environnement, les spécifications d'accès aux données ainsi que les caractéristiques de performances réelles.

Une indexation correcte et quelques autres solutions résolvent souvent bien des problèmes de performances mieux que la dénormalisation.

En particulier, posez-vous les questions suivantes lorsque vous envisagez de dénormaliser :

- Quelles sont les transactions critiques et quel est le temps de réponse voulu ?
- Quelle est la fréquence d'utilisation des transactions ?
- Quelles sont les tables ou les colonnes qu'utilisent les transactions stratégiques ? A combien de lignes accèdent-elles chaque fois ?
- Quelle est la combinaison des types de transaction : select, insert, update et delete ?
- Quel est l'ordre de tri habituel ?
- Quelle concurrence d'accès voulez-vous mettre en œuvre ?
- De quelle taille sont les tables le plus fréquemment utilisées ?
- Certains processus génèrent-ils des données agrégées ?
- Quel est l'emplacement physique des données ?

Techniques

Les techniques de dénormalisation les plus fréquentes sont les suivantes :

- Duplication de colonnes
- Ajout de colonnes dérivées
- Déstructuration de tables

Pour améliorer les performances, vous pouvez en outre dupliquer ou diviser les tables. Ces méthodes ne constituent pas des techniques de dénormalisation, mais elles jouent le même rôle et nécessitent les mêmes précautions.

Duplication de colonnes

Vous pouvez dupliquer des colonnes pour éliminer les jointures fréquentes.

Par exemple, si des jointures fréquentes sont effectuées entre les tables titleauthor et authors pour rechercher le nom de l'auteur, vous pouvez ajouter la colonne au_lname dans titleauthor.

La duplication de colonnes élimine les jointures pour un grand nombre de requêtes. Les problèmes soulevés par cette solution sont les suivants :

- Elle nécessite la maintenance de nouvelles colonnes. Vous devez introduire des modifications dans deux tables et éventuellement dans un grand nombre de lignes d'une des tables.
- Elle requiert davantage d'espace disque car au_lname est dupliquée.

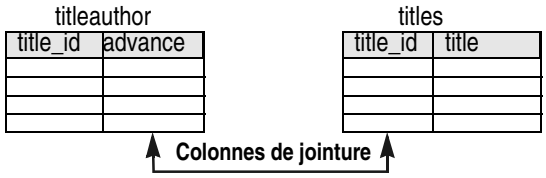
Ajout de colonnes dérivées

L'ajout de colonnes dérivées permet d'éliminer les jointures et de réduire le temps nécessaire pour générer des valeurs d'agrégat. La colonne total_sales dans la table titles de la base de données pubs2 fournit un exemple de colonne dérivée utilisée pour réduire le temps de traitement des valeurs d'agrégat.

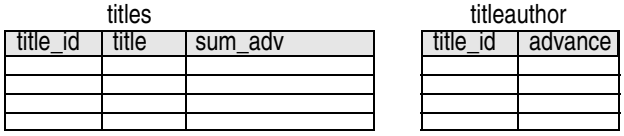
L'exemple de la figure 6-7 met en évidence ces deux avantages. Des jointures fréquentes sont nécessaires entre les tables titleauthor et titles pour fournir l'avance totale pour un titre d'ouvrage particulier sum_adv.

Figure 6-7 : Dénormalisation par ajout de colonnes dérivées

```
select title, sum(advance)
from titleauthor ta, titles t
where ta.title_id = t.title_id
group by title_id
```




```
select title, sum_adv from titles
```



Vous pouvez créer et mettre à jour une colonne de données dérivées dans la table titles, en éliminant à la fois la jointure et l'agrégat lors de l'exécution. Cela augmente l'espace de stockage requis et nécessite une mise à jour de la colonne dérivée lorsque des changements sont effectués dans la table titles.

Déstructuration de tables

Si la plupart des utilisateurs ont besoin de visualiser l'ensemble des données jointes de deux tables, la déstructuration des deux tables permettant d'en créer une seule peut améliorer les performances en éliminant la jointure.

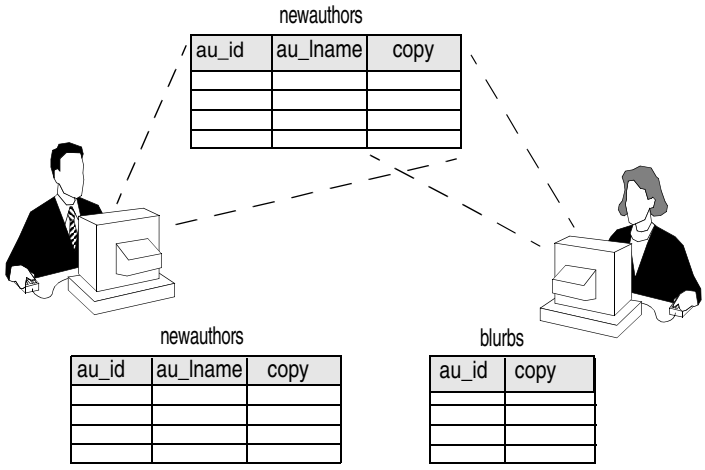
Par exemple, les utilisateurs ont souvent besoin de visualiser simultanément le nom de l'auteur, son identificateur et les données de la colonne copy de la table blurbs. La solution est de déstructurer les deux tables pour en créer une seule. Pour qu'une déstructuration de deux tables soit possible, une relation de un-à-un doit exister entre leurs données.

La déstructuration de tables élimine la jointure, mais la séparation conceptuelle des données est perdue. Si certains utilisateurs nécessitent toujours un accès uniquement aux deux colonnes de chaque table, cet accès peut être restauré par des requêtes sélectionnant seulement les colonnes nécessaires ou par l'utilisation de vues.

Duplication de tables

Si un groupe d'utilisateurs n'a normalement besoin que d'un sous-ensemble des données, vous pouvez dupliquer les tables critiques pour ce groupe.

Figure 6-8 : Dénormalisation par duplication de tables



La duplication montrée à la figure 6-8 réduit les conflits mais impose une gestion des données dupliquées. Par ailleurs, des temps de latence peuvent s'imposer au groupe d'utilisateurs qui n'accèdent qu'aux données copiées.

Fractionnement de tables

Parfois, le fractionnement de tables normalisées peut améliorer les performances. Il existe deux façons de fractionner les tables :

- Horizontalement, en plaçant les lignes dans deux tables distinctes, en fonction des valeurs des données dans une ou plusieurs colonnes.

- Verticalement, en plaçant la clé primaire et certaines colonnes dans une table, et les autres colonnes ainsi que la clé primaire dans une autre table.

Tenez compte du fait que le fractionnement des tables, horizontal ou vertical, rend vos applications plus complexes.

Fractionnement horizontal

Utilisez le fractionnement horizontal si :

- La table est volumineuse et le partitionnement permet de réduire le nombre de pages d'index lues dans une requête.

  Cependant, les index de l'arbre à accès séquentiel comportent généralement peu de niveaux et vous pouvez ajouter un grand nombre de lignes dans une table comportant des clés d'index de petite taille avant que l'arbre à accès séquentiel ne requière plus de niveaux. B-tree requires more levels.

  Un trop grand nombre de niveaux d'index peut poser un problème pour les tables ayant des clés de très grande taille.

- Le point de rupture de la table correspond à une séparation naturelle des lignes, par exemple des présentations de sites géographiques différents ou une opposition entre données historiques/données actuelles.

  Vous pouvez adopter le fractionnement horizontal si vous avez une table contenant d'une part un grand nombre de données historiques peu consultées et d'autre part des données actuelles dont l'accès exige des performances élevées de la part de vos applications.

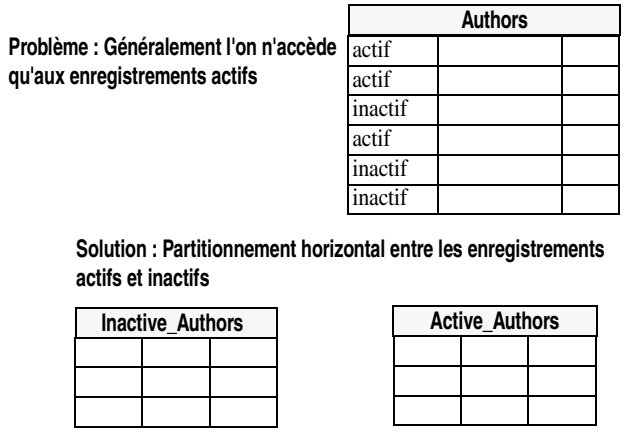
- Le fractionnement de tables répartit les données sur un support de distribution physique ; il existe toutefois d'autres moyens pour parvenir à ce résultat.

Généralement, il est nécessaire d'utiliser des noms de table différents dans les requêtes, selon les valeurs des tables. Pour la plupart des applications de base de données, les avantages du fractionnement des tables ne parviennent pas à compenser cette complexité.

Tant que les clés d'index sont de petite taille et que les requêtes de la table utilisent les index, le fait de doubler ou de tripler le nombre de lignes de la table n'augmente que d'un niveau d'index le nombre de lectures disque requises pour une requête. Lorsqu'un grand nombre de requêtes a recours au balayage de table, le fractionnement horizontal peut apporter un gain de performances assez intéressant pour compenser les opérations de maintenance supplémentaires.

La figure 6-9 montre comment il est possible de diviser la table authors pour séparer les enregistrements actifs des enregistrements inactifs :

Figure 6-9 : Partitionnement horizontal entre les enregistrements actifs et inactifs



Fractionnement vertical

Utilisez le fractionnement vertical si :

- Certaines colonnes sont utilisées plus fréquemment que d'autres.
- La table comporte des lignes de grande dimension et son fractionnement permet de réduire le nombre de pages devant être lues.

Le fractionnement vertical de tables est vraiment utile lorsque les deux conditions ci-dessus sont remplies. Vous augmenterez notablement la vitesse d'extraction des colonnes fréquemment utilisées en plaçant les colonnes très longues rarement utilisées dans une autre table. Si les lignes sont plus courtes, la page de données pourra contenir plus de lignes ; les requêtes accèdent donc à un nombre réduit de pages.

Gestion des données dénormalisées

Quelle que soit la technique de dénormalisation choisie, vous devez garantir la cohérence des données. Pour ce faire, utilisez :

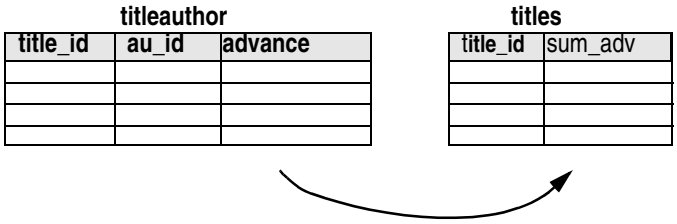
- Les triggers, qui peuvent mettre à jour les données dérivées ou dupliquées à chaque modification des données de la base.
- La logique applicative avec, dans chaque application, des transactions qui mettent à jour les données dénormalisées pour s'assurer que les changements sont atomiques.
- La mise à jour par batch, exécutée à intervalles appropriés pour homogénéiser les données dénormalisées.

Les triggers constituent la méthode la plus efficace pour assurer la cohérence des données, mais ils risquent de ralentir les performances.

Utilisation de triggers

Dans la figure 6-10, la colonne sum_adv de la table titles stocke les données dénormalisées. Un trigger met à jour la colonne sum_adv dès que des modifications sont apportées à la colonne advance de la table titleauthor.

Figure 6-10 : Utilisation de triggers pour conserver les données normalisées



Utilisation de la logique de l'application

Si votre application doit assurer la cohérence des données, elle doit vérifier que les insertions, suppressions ou mises à jour des deux tables s'effectuent dans une transaction unique.

Si vous utilisez la logique d'application, vérifiez que les spécifications de cohérence des données sont répertoriées et connues des développeurs d'applications et des utilisateurs en charge de la maintenance des applications.

Remarque Il est risqué d'utiliser la logique d'application pour gérer les données dénormalisées. La même logique doit être utilisée et mise à jour dans toutes les applications modifiant les données.

Mise à jour par batch

Si une cohérence de 100 % n'est pas nécessaire en permanence, vous pouvez exécuter un batch ou une procédure stockée pendant les heures non ouvrées afin de mettre à jour les données dupliquées ou dérivées.

CHAPITRE 7

Stockage de données

Ce chapitre explique comment Adaptive Server stocke les lignes de données sur des pages et comment les instructions de sélection et de modification de données sont appliquées à ces pages quand il n'existe pas d'index.

Il aide à comprendre comment la création d'index, l'optimisation de vos requêtes et un stockage adéquat des objets peuvent améliorer les performances d'Adaptive Server.

Sujet	Page
Gains de performances grâce à l'optimisation des requêtes	151
Adaptive Server pages	154
Pages gérant l'allocation de l'espace	158
Surcoût lié à l'espace disque	162
Données en vrac : tables sans index clusterisé	168
Opérations d'E/S d'Adaptive Server sur les tables sans index	175
Caches et liaisons d'objets	176
Prélecture asynchrone et E/S de tables sans index	181
Tables sans index : avantages et inconvénients	183
Gestion des tables sans index	183
Journal de transactions : table spéciale sans index	185

Gains de performances grâce à l'optimisation des requêtes

L'optimiseur d'Adaptive Server essaye de trouver le chemin d'accès optimal pour chaque table de la requête en évaluant le coût en E/S physiques nécessaires pour accéder aux données et le nombre de lectures nécessaires pour chaque page stockée dans le cache de données.

Dans la plupart des applications de base de données, la base contient de nombreuses tables, qui comportent toutes un ou plusieurs index. Selon le nombre et le type d'index créés, l'optimiseur choisit les méthodes d'accès suivantes :

- Balayage de table : lecture de toutes les pages de données de la table, représentant parfois des centaines ou des milliers de pages.
- Accès par index : utilisation de l'index afin d'extraire uniquement les pages de données nécessaires, soit parfois trois ou quatre pages seulement.
- Accès par index restreint : utilisation d'un index non clusterisé pour renvoyer les données, sans lire les lignes de données réelles, en n'effectuant qu'une partie des lectures de pages requises pour un balayage de table.

Si vous disposez d'index appropriés sur les tables, la plupart de vos requêtes n'effectueront qu'un nombre réduit de lectures de pages pour accéder aux données.

Traitement des requêtes et lectures de pages

Le temps d'exécution d'une requête est employé en grande partie à lire les pages de données sur le disque. C'est pourquoi les principaux gains de performances (plus de 80 % selon de nombreux experts) sont obtenus lorsqu'on réduit le nombre des opérations de lecture sur le disque pour chaque requête.

Lorsqu'une requête effectue un balayage de table, Adaptive Server lit chaque page de la table car il n'existe aucun index exploitable pour l'aider à extraire les données. Comme les lectures de disque sont longues, le temps de réponse de la requête n'est guère satisfaisant. Par ailleurs, les requêtes qui utilisent les balayages de table ralentissent les performances des autres requêtes s'effectuant sur le serveur.

Les balayages risquent de prolonger l'attente des autres utilisateurs car ils utilisent des ressources système telles que le temps CPU, les E/S disque et le trafic réseau.

Les balayages de table augmentent le nombre d'opérations de lecture sur le disque (E/S) pour la requête. Lorsque vous serez complètement familiarisés avec les méthodes d'accès, les outils d'optimisation, la taille et la structure des tables et les requêtes dans vos applications, vous serez à même d'estimer le nombre d'opérations d'E/S que couvrira une jointure ou une sélection, compte tenu des index disponibles.

Si vous connaissez les colonnes indexées sur les tables ainsi que la taille des index et des tables, vous pouvez, en considérant une requête, prévoir son comportement. Pour des requêtes différentes sur une même table, vous devez pouvoir tirer ces conclusions :

- Cette requête ponctuelle ne renvoie qu'une seule ligne ou un petit nombre de lignes répondant au critère de la clause where.

  La condition de la clause where est indexée ; elle doit effectuer entre deux et quatre E/S sur l'index, plus une pour lire la page de données correcte.

- Toutes les colonnes de la liste de sélection ainsi que la clause where de cette requête sont incluses dans un index non clusterisé. Cette requête balayera probablement le niveau feuille de l'index, soit environ 600 pages.

  Si vous ajoutez une colonne non indexée dans la liste de sélection, le balayage s'exécute sur la table, ce qui nécessite 5000 lectures sur disque.

- Il n'existe pas d'index exploitable pour cette requête. Elle aboutira donc à un balayage de table nécessitant au moins 5000 lectures sur disque.

Ce chapitre décrit le mode de stockage des tables et le mode d'accès aux lignes de données lorsque les index ne sont pas utilisés.

Le chapitre 9, "Fonctionnement des index", décrit les méthodes d'accès relatives aux index. Dans d'autres chapitres, vous apprendrez comment déterminer la méthode d'accès utilisée pour une requête, la taille des tables et des index ainsi que le volume d'E/S d'une requête. Ces chapitres permettent de bien comprendre de quelle façon l'optimiseur détermine le coût d'accès aux données pour vos requêtes.

Adaptive Server pages

L'unité de stockage élémentaire pour Adaptive Server est la **page**. Les tailles des pages sont 2 ko, 4 ko, 8 ko et 16 ko. La taille de la page du serveur est définie lors de la première création de la source de données. Une fois le serveur construit, cette valeur ne peut plus être modifiée. Ces pages contiennent des objets de la base de données, notamment :

- Des pages de données, qui stockent les lignes de données d'une table.
- Des pages d'index, qui stockent les lignes d'index pour tous les niveaux d'un index.
- Des pages d'objets volumineux (LOB), qui contiennent les données des colonnes de type text ou image et des colonnes Java hors ligne.

Adaptive Server peut avoir à traiter des volumes importants de données pour une seule requête, opération DML ou commande. Par exemple, si vous utilisez une table verrouillée au niveau des pages de données seulement (DOL) et une colonne char(2000), Adaptive Server doit allouer de la mémoire pour effectuer la copie des colonnes tout en balayant la table. Des besoins accrus de mémoire pendant une requête signifient une réduction potentielle du débit.

La taille des pages logiques d'Adaptive Server (2 ko, 4 ko, 8 ko ou 16 ko) détermine l'allocation d'espace sur le serveur. Chaque page d'allocation, page d'OAM, page de données, page d'index, page de texte, etc., est construite sur une page logique. Par exemple, si la taille de la page logique d'Adaptive Server est de 8 ko, chaque type de page a une taille de 8 ko. Toutes ces pages utilisent la totalité de l'espace spécifié par la taille de la page logique. Les pages d'OAM ont un plus grand nombre d'entrées OAM pour les pages logiques de grande taille (par exemple 8 ko) que pour celles de petite taille (2 ko).

En-têtes de page et tailles de page

Toutes les pages comportent un en-tête dans lequel sont stockées des informations telles que l'ID d'objet auquel appartient la page et d'autres données utilisées pour gérer l'espace sur la page. Le tableau 7-1 montre le nombre d'octets d'overhead et l'espace utilisable dans des pages de données et d'index.

Tableau 7-1 : Overhead et espace de données utilisateur dans des pages de données et d'index

Plan de verrouillage	Overhead	Octets pour des données utilisateur
Toutes les pages (APL)	32	2016
Pages de données seulement (DOL)	46	2002

Le reste de la page est disponible pour le stockage des lignes de données et d'index.

Pour plus d'informations sur le stockage des colonnes Java ou de type text et image, reportez-vous à la section "Pages d'objets volumineux (LOB)", page 156.

Tailles variables des pages logiques

La commande dataserver vous permet de créer des devices master et des bases de données comportant des pages logiques de 2 ko, 4 ko, 8 ko ou 16 ko. Les pages logiques de grande taille permettent de créer des lignes plus longues, ce qui augmente les performances car Adaptive Server peut accéder à davantage de données chaque fois qu'il lit une page. Par exemple, une page de 16 ko peut contenir 8 fois plus de données qu'une page de 2 ko, une page de 8 ko 4 fois plus qu'une page de 2 ko et ainsi de suite, pour toutes les tailles de pages logiques.

La taille des pages logiques est un paramètre défini à l'échelle du serveur ; il est impossible d'avoir des pages logiques de tailles différentes au sein du même serveur. Toutes les tables ont une taille ne permettant pas que la taille des lignes excède celles des pages. Cela signifie que les lignes ne peuvent pas s'étendre sur plusieurs pages.

Pour tout renseignement spécifique sur l'utilisation de la commande dataserver pour construire votre device master, reportez-vous au guide *Utilitaires*.

Pages de données et d'index

Les pages de données et d'index dans des tables verrouillées au niveau des pages de données seulement (tables DOL) comprennent une table d'offset de ligne dans laquelle figurent des pointeurs désignant l'octet de début de chaque ligne de la page. Chaque pointeur occupe 2 octets.

Les lignes de données et d'index sont insérées dans une page immédiatement après l'en-tête et occupent le reste de la page de manière continue. Pour la totalité des tables et des index sur des tables DOL, la table d'offset de ligne commence au niveau du dernier octet de la page et elle se déploie vers le haut.

Pour chaque ligne, les informations stockées sont les données de la colonne et des informations telles que le numéro de ligne et le nombre de colonnes de longueur variable et de colonnes NULL sur la ligne. Les pages d'index des tables verrouillées au niveau de toutes les pages (tables APL) ne possèdent pas de table d'offset de ligne.

Les lignes ne peuvent dépasser les limites de la page, sauf dans le cas de colonnes de type *text*, *image* et les colonnes Java hors ligne. Chaque ligne de données compte au moins 4 octets d'overhead. Les lignes contenant des données de longueur variable comportent davantage d'overhead.

Pour plus d'informations sur l'overhead et la taille des lignes d'index et de données, reportez-vous au chapitre 15, "Détermination de la taille des tables et des index".

La table d'offset de ligne stocke les pointeurs dirigés vers l'origine de chaque ligne de données de la page.

Pages d'objets volumineux (LOB)

Les colonnes de type *text* et *image* et les colonnes Java hors ligne (LOB) d'une table sont stockées dans une structure distincte composée d'un ensemble de pages. Chaque table contenant une colonne text ou image possède l'une de ces structures. Même une table comportant plusieurs colonnes de type LOB possède une (et une seule) de ces structures de données.

La table elle-même stocke, pour la ligne correspondante, un pointeur de 16 octets renvoyant à la première page de la valeur. Les pages supplémentaires contenant cette valeur sont reliées par des pointeurs renvoyant sur les pages suivantes et précédentes. Chaque valeur est stockée dans sa propre chaîne de pages. La première page contient le nombre d'octets de la valeur text. La dernière page de la chaîne où figure la valeur se termine par un pointeur de page suivante NULL.

La lecture ou l'écriture d'une valeur LOB nécessite la lecture ou l'écriture d'au moins deux pages :

- une pour le pointeur,
- une pour l'emplacement du texte dans l'objet text.

Chaque page LOB peut stocker jusqu'à 1 800 octets. Toute valeur non NULL occupe au moins une page entière.

Les structures LOB sont répertoriées séparément dans sysindexes. La colonne d'ID d'index, indid, est toujours égale à 255 et name est le nom de la table préfixé de la lettre "t".

Extents

Dans Adaptive Server, les pages sont toujours allouées à une table, un index ou un objet LOB. Un bloc de 8 pages s'appelle un **extent**. La taille d'un extent dépend de la taille de page utilisée par le serveur. Elle est de 16 ko sur un serveur de 2 ko, de 64 ko sur un serveur de 8 ko, etc. L'espace minimal que peut occuper une table ou un index est de 1 extent, soit 8 pages. Les extents ne sont désalloués que lorsque toutes les pages de l'extent sont vides.

L'utilisation d'extents dans Adaptive Server est transparente pour l'utilisateur ; celui-ci ne les voit que lorsqu'il examine des rapports sur l'utilisation de l'espace.

Par exemple, les rapports générés par sp_spaceused indiquent l'espace alloué (colonne reserved) et l'espace utilisé par les données et les index. La colonne unused donne l'espace des extents alloué à un objet mais non encore utilisé pour stocker des données.

```
sp_spaceused titles
name   rowtotal reserved data    index_size unused
------ -------- -------- ------- ---------- ------
titles 5000     1392 KB  1250 KB 94 KB     48 KB
```

Dans le rapport ci-dessus, la table titles et ses index disposent d'un espace réservé de 1 392 ko sur divers extents, dont 48 ko (24 pages de données) qui ne sont pas alloués.

Pages gérant l'allocation de l'espace

Outre les pages de données, d'index et d'objets LOB utilisées pour le stockage, Adaptive Server a recours à d'autres types de pages pour gérer le stockage, contrôler l'allocation de l'espace et organiser les objets de la base de données. La table sysindexes contient également des pointeurs utilisés lors de l'accès aux données.

Les pages qui gèrent l'allocation de l'espace et les pointeurs sysindexes sont utilisées pour :

- accélérer la recherche d'objets dans la base,
- accélérer le processus d'allocation et de désallocation de l'espace pour les objets,
- permettre à Adaptive Server d'allouer de l'espace supplémentaire à un objet voisin de l'espace qu'il occupe déjà. Cette proximité contribue également à améliorer les performances en réduisant les temps de déplacement des têtes de lecture.

Les pages suivantes analysent l'utilisation de l'espace disque par les objets de base de données :

- Les pages de la table d'allocation globale (GAM) contiennent des bitmaps pour la base de données tout entière.
- Les pages d'allocation analysent l'utilisation de l'espace et les objets dans des groupes de 256 pages (blocs de 1/2 Mo).
- Les pages de la table d'allocation d'objets (OAM) contiennent des informations sur les extents utilisés pour chaque objet. L'OAM contient au moins une page pour chaque table ou index ; elle sert à repérer l'emplacement des pages des objets dans la base de données.
- Les pages de contrôle gèrent l'allocation de l'espace pour les tables partitionnées. A chaque partition correspond une page de contrôle.

Pages de la table d'allocation globale

Chaque base de données possède une page de table d'allocation globale (GAM). Elle contient un bitmap de toutes les unités d'allocation d'une base de données, chaque unité étant représentée par un bit. Lorsqu'il n'existe plus d'extents disponibles pour le stockage d'objets dans une unité d'allocation, le bit correspondant dans la GAM prend la valeur 1.

Ce système permet d'accélérer l'allocation d'espace aux objets. La page GAM n'est pas visible par les utilisateurs : elle apparaît dans les catalogues système sous le nom de table sysgams.

Pages d'allocation

Lorsque vous créez une base de données ou que vous ajoutez de l'espace dans une base, l'espace est divisé en unités d'allocation de 256 pages de données. La première page de chaque **unité d'allocation** est appelée page d'allocation. La page 0 et toutes les pages multiples de 256 sont des pages d'allocation.

La page d'allocation garde en mémoire l'espace de chaque extent de l'unité d'allocation en enregistrant l'ID d'objet et l'ID d'index pour l'objet stocké dans l'extent, ainsi que le nombre de pages libres et occupées. La page d'allocation contient l'ID de page pour la table ou la page d'OAM de l'index.

Pages de la table d'allocation d'objets

Chaque table, index ou chaîne de texte possède une ou plusieurs pages d'OAM (Object Allocation Map ou table d'allocation d'objets) qui sont stockées sur des pages allouées à la table ou à l'index. Si une table possède plusieurs pages d'OAM, elles sont liées de façon à former une chaîne. Ces pages d'OAM stockent des pointeurs renvoyant aux unités d'allocation qui contiennent les pages de l'objet recherché.

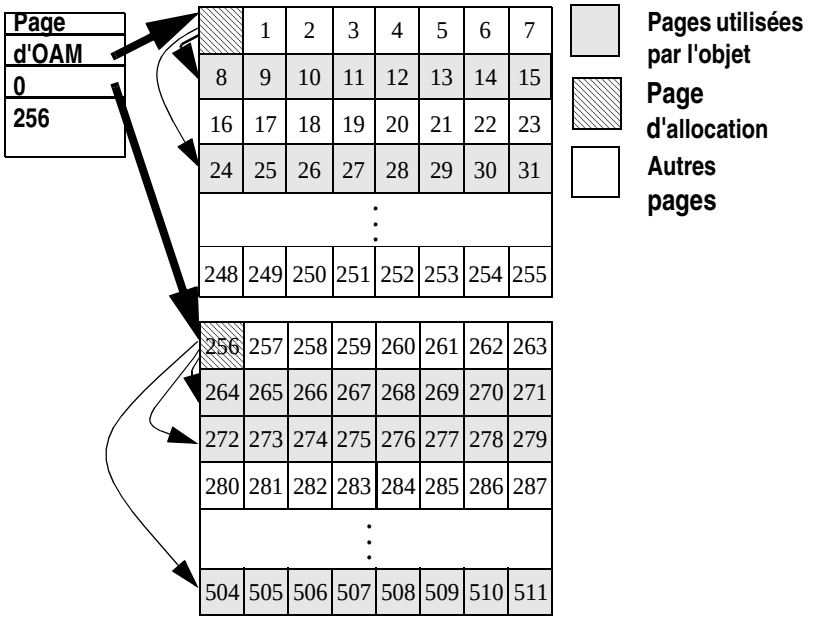
La première page de la chaîne contient des conseils d'allocation, qui indiquent par exemple quelle page d'OAM de la chaîne comporte des informations sur les unités d'allocation disposant d'espace. Cette méthode permet d'allouer rapidement à un objet un supplément d'espace à proximité des pages qu'il occupe déjà.

Gestion du stockage des objets par les pages d'OAM et les pages d'allocation

La figure 7-1 illustre de quelle façon les unités d'allocation, les extents et les objets sont gérés par les pages d'OAM et les pages d'allocation.

- Deux unités d'allocation sont représentées : l'une commence à la page 0 et l'autre à la page 256. La première page de chacune d'elles est la page d'allocation.
- Une table est stockée sur quatre extents commençant au niveau des pages 1 et 24 sur la première unité d'allocation et au niveau des pages 272 et 504 sur la deuxième unité d'allocation.
- La première page de la table est sa page d'OAM. Elle renvoie à la page d'allocation de chaque unité d'allocation sur laquelle l'objet utilise des pages, c'est-à-dire aux pages 0 et 256.
- Les pages d'allocation 0 et 256 stockent les ID d'objets de la table et des informations sur les extents et les pages utilisées sur ces extents. Ainsi, la page d'allocation 0 pointe sur les pages 1 et 24 de la table et la page d'allocation 256 pointe sur les pages 272 et 504.

Figure 7-1 : Pointeurs des pages d'allocation et de la page d'OAM



L'allocation de pages permet de ne pas séparer les pages d'un objet

Adaptive Server s'efforce de garder ensemble les pages allouées à un objet. En règle générale :

- S'il existe une page non allouée dans l'extent courant, elle est attribuée à cet objet.
- S'il n'existe pas de page disponible dans l'extent courant mais qu'il existe une page non allouée dans un autre extent de l'objet, Adaptive Server utilise cet extent.
- Si tous les extents de l'objet sont occupés mais qu'il en existe de disponibles sur l'unité d'allocation, le nouvel extent est alloué à l'une des unités déjà utilisées.

Table *sysindexes* et accès aux données

La table sysindexes stocke des informations sur les tables indexées et non indexées. Elle contient une ligne où :

- Pour chaque table APL, la colonne indid est à 0 si la table ne contient pas d'index clusterisé ou à 1 dans le cas contraire.
- Pour chaque table DOL, la colonne indid est toujours à 0 pour la table.
- Pour chaque index, non clusterisé et clusterisé, d'une table DOL, les ID d'index sont compris entre 2 et 250.
- Pour chaque table comportant une ou plusieurs colonnes LOB, l'ID d'index est toujours égal à 255 pour l'objet LOB.

Chaque ligne de sysindexes stocke des pointeurs renvoyant à une table ou à un index afin d'accélérer l'accès aux objets. Le tableau 7-2 montre de quelle façon sont utilisés ces pointeurs lors de l'accès aux données.

Tableau 7-2 : Utilisation des pointeurs de sysindexes lors de l'accès aux données

Colonne	Utilisation pour l'accès aux tables	Utilisation pour l'accès aux index
root	Si indid est égal à 0 et si la table est partitionnée et verrouillée au niveau de toutes les pages (APL), root pointe vers la dernière page de cette table.	Sert à retrouver la page racine de l'arbre d'index.
first	Renvoie à la première page de données dans la chaîne des pages pour les tables APL.	Renvoie à la première page de niveau feuille dans un index non clusterisé ou dans un index clusterisé d'une table DOL.
doampg	Renvoie à la première page d'OAM de la table.	
ioampg		Renvoie à la première page d'OAM d'un index.

Surcoût lié à l'espace disque

Quelle que soit la taille de la page logique pour laquelle il est configuré, Adaptive Server alloue de l'espace aux objets (tables, index, chaîne de pages de texte) extents, équivalant à huit pages logiques. Autrement dit, si un serveur est configuré pour des pages logiques de 2 ko, il alloue un extent de 16 ko pour chacun de ces objets. S'il est configuré pour des pages logiques de 16 ko, il alloue un extent de 128 ko pour chacun de ces objets.

Il en va de même pour les catalogues système. Quand le serveur comporte beaucoup de petites tables, l'espace occupé peut être important si le serveur utilise des pages logiques de grande taille.

Par exemple, s'il s'agit d'un serveur configuré pour des pages logiques de 2 ko, systypes (avec environ 31 lignes brèves, un index clusterisé et un index non clusterisé) se voit réserver 3 extents, soit 48 ko de mémoire. Si vous modifiez le serveur de façon à utiliser des pages de 8 ko, l'espace réservé à systypes est toujours de 3 extents, soit 192 ko de mémoire.

Pour un serveur configuré pour 16 ko, systypes a besoin de 384 ko d'espace disque. Pour les petites tables, l'espace inutilisé dans le dernier extent peut devenir important sur les serveurs qui utilisent des pages logiques de grande taille.

Les bases de données sont également concernées. Chacune d'elle comprend les catalogues système et leurs index. Si vous passez à une page logique de taille supérieure, vous devez tenir compte de la quantité d'espace disque requise par chaque base de données.

Nombre de colonnes et taille

Le nombre maximal de colonnes que vous pouvez créer dans une table est :

- 1024 pour les colonnes de longueur fixe pour les tables verrouillées au niveau de toutes les pages (APL) et celles verrouillées uniquement au niveau des pages de données (DOL)
- 254 pour les colonnes de longueur variable d'une table APL
- 1024 pour les colonnes de longueur variable d'une table DOL

La taille maximale d'une colonne dépend des éléments suivants :

- La longueur (fixe ou variable) des colonnes de la table.

- La taille de page logique de la base de données. Par exemple, dans une base de données avec des pages logiques de 2 ko, la taille maximale d'une colonne d'une table APL peut correspondre à une ligne, soit environ 1962 octets, moins les informations de contrôle. De même, pour une page de 4 ko, la taille maximale d'une colonne de table APL peut correspondre à 4010 octets, moins les informations de contrôle. Pour plus d'informations, reportez-vous au tableau 7-3.
- Si vous tentez de créer une table avec une colonne de longueur fixe supérieure aux limites de la taille de page logique, create table renvoie un message d'erreur.

Tableau 7-3 : Longueur maximale de ligne et de colonne - APL et DOL

Plan de verrouillage	Taille de la page	Longueur maximale de ligne	Longueur maximale de colonne
Tables APL	2 ko (2048 octets)	1962	1960 octets
	4 ko (4096 octets)	4010	4008 octets
	8 ko (8192 octets)	8106	8104 octets
	16 ko (16384 octets)	16298	16296 octets
Tables DOL	2 ko (2048 octets)	1964	1958 octets
	4 ko (4096 octets)	4012	4006 octets
	8 ko (8192 octets)	8108	8102 octets
	16 ko (16384 octets)	16300	16294 octets si la table ne comprend pas de colonne de longueur variable
	16 ko (16384 octets)	16300 (soumise à un offset *maximale de début* de varlen = 8191)	8191–6–2 = 8183 octets si la table comprend au moins une colonne de longueur variable.*
	* Cette taille comprend six octets pour l'overhead de ligne et deux octets pour le champ de longueur de ligne		

La taille maximale d'une colonne de longueur fixe d'une table DOL avec une taille de page logique de 16 ko diffère selon que la table contient ou non des colonnes de longueur variable. L'offset initial maximal d'une colonne de longueur variable est de 8191. Si la table a des colonnes de longueur variable, la somme de la partie de longueur fixe de la ligne et des overheads ne peut pas dépasser 8191 octets et la taille maximale de toutes les colonnes de longueur fixe est limitée à 8183 octets, lorsque la table contient des colonnes de longueur variable.

Colonnes de longueur variable des tables APL

Les tables APL qui contiennent une colonne de longueur variable (par exemple, varchar, varbinary etc.) ont un overhead minimal pour chaque ligne égal à :

- Deux octets pour l'overhead initial de ligne.
- Deux octets pour la longueur de la ligne.
- Deux octets pour la table d'offset de colonne à la fin de la ligne. Cela vaut toujours *n+1* octets, *n* étant le nombre de colonnes de longueur variable de la ligne.

Une table à une seule colonne a un overhead d'au moins six octets, plus l'overhead supplémentaire pour la table d'ajustement. La taille maximale de la colonne, en tenant compte de l'overhead total, est inférieure ou égale à la longueur de la colonne augmentée du nombre d'octets pour la table d'ajustement, plus six octets d'overhead.

Tableau 7-4 : Taille maximale des colonnes de longueur variable d'une table APL

Taille de la page	Longueur maximale de ligne	Longueur maximale de colonne
2 ko (2048 octets)	1962	1948
4 ko (4096 octets)	4010	3988
8 ko (8192 octets)	8096	8058
16 ko (16384 octets)	16298	16228

Colonnes de longueur variable dépassant la taille de la page logique

Si votre table utilise des pages logiques de 2 ko, vous pouvez créer quelques colonnes de longueur variable dont la longueur totale de la ligne dépasse la longueur maximale de ligne pour une page de 2 ko. Cela vous permet de créer des tables dans lesquelles quelques colonnes de longueur variable contiennent la taille maximale possible. Toutefois, lorsque vous exécutez create table, vous recevez un message d'avertissement indiquant que la taille de la ligne résultante risque de dépasser la taille maximale de la ligne et de faire échouer par la suite une commande insert ou update.

Par exemple, si vous créez une table qui utilise une taille de page de 2 ko et contient une colonne de longueur variable de 1975 octets, Adaptive Server crée la table mais émet un message d'avertissement. Toutefois, si vous tentez d'insérer des données dépassant la longueur maximale de la ligne (1962 octets), l'opération échoue.

Colonnes de longueur variable des tables DOL

Pour une seule colonne de longueur variable dans une table DOL, l'overhead minimal pour chaque ligne est :

- Six octets pour l'overhead initial de ligne.
- Deux octets pour la longueur de la ligne.
- Deux octets pour la table d'offset de colonne à la fin de la ligne. Chaque entrée d'offset de colonne est de deux octets. Il existe *n* entrées, n étant le nombre de colonnes de longueur variable de la ligne.

L'overhead total est de 10 octets. Il n'y a pas de table d'ajustement pour les lignes DOL. La taille réelle d'une colonne de longueur variable est de :

```
longueur de colonne + 10 octets d'overhead
```

Tableau 7-5 : Taille maximale pour les colonnes de longueur variable d'une table DOL

Taille de la page	Longueur maximale de ligne	Longueur maximale de colonne
2 ko (2048 octets)	1964	1954
4 ko (4096 octets)	4012	4002
8 ko (8192 octets)	8108	7998
16 ko (16384 octets)	16300	162290

Les tables DOL avec des colonnes de longueur variable doivent avoir un offset inférieur à 8191 octets pour que toutes les insertions puissent s'effectuer. Par exemple, l'insertion suivante échoue parce que l'offset total dépasse 8191 octets :

```
create table t1(
    c1 int not null,
    c2 varchar(5000) not null
    c3 varchar(4000) not null
    c4 varchar(10) not null
    plus de colonnes de longueur fixe)
cvarlen varchar(nnn)) lock datarows
```

L'offset des colonnes c2, c3 et c4 est de 9010, par conséquent l'insertion échoue dans son ensemble.

Restrictions pour la conversion de plans de verrouillage ou l'utilisation de select into

Que vous utilisiez alter table pour modifier un plan de verrouillage ou select into pour copier des données dans une nouvelle table, les restrictions suivantes s'appliquent.

Pour les serveurs qui utilisent des tailles de page différentes de 16 ko, la longueur maximale d'une colonne de longueur variable dans une table APL est inférieure à celle d'une table DOL. Vous pouvez donc convertir en mode DOL le plan de verrouillage d'une table APL avec une colonne de longueur variable de taille maximale. La conversion d'une table DOL contenant au moins une colonne de longueur variable de taille maximale en mode APL est restreinte. Adaptive Server renvoie un message d'erreur et l'opération est abandonnée.

Sur les serveurs qui utilisent des pages de 16 ko, les tables APL peuvent contenir des colonnes de longueur variable de taille substantiellement supérieure à celles des tables DOL.Vous pouvez convertir des tables de DOL à APL, mais la conversion du plan de verrouillage de APL à DOL est restreinte. Adaptive Server émet un message d'erreur et l'opération est abandonnée.

Notez que ces restrictions des conversions de plan de verrouillage ne se produisent que si la table source contient des données qui dépassent les limites de la table cible. Dans ce cas, Adaptive Server émet un message d'erreur en remplaçant le format de ligne d'un plan de verrouillage par celui d'un autre. Si la table est vide, cette transformation de données n'est pas nécessaire et l'opération de changement de plan de verrouillage réussit. Toutefois, lors d'une opération d'insertion ou de mise à jour sur la table, les utilisateurs peuvent rencontrer des erreurs dues aux limitations sur la taille de la colonne ou de la ligne pour le schéma cible de la table modifiée.

Organisation des colonnes dans les tables DOL par taille des colonnes de longueur variable

Pour les tables DOL qui utilisent des colonnes de longueur variable, placez les colonnes les plus longues vers la fin de la table. Cela permet de créer des tables avec des lignes de très grande taille. Par exemple, sur un serveur avec pages de 16 ko, la définition de table suivante est acceptable :

```
create table t1 (
    c1 int not null,
    c2 varchar(1000) null,
    c3 varchar(4000) null,
    c4 varchar(9000) null) lock datarows
```

En revanche, la définition de table suivante n'autorise pas de nouvelles insertions. L'offset potentiel initial de la colonne c2 est supérieur à la limite de 8192 octets en raison de la colonne c4 de 9000 octets :

```
create table t2 (
    c1 int not null,
    c4 varchar(9000) null,
    c3 varchar(4000) null,
    c2 varchar(1000) null) lock datarows
```

La table est créée mais les futures insertions peuvent échouer.

Nombre de lignes par page de données

Le nombre de lignes permises pour une page de données DOL est déterminé par :

- La taille de la page
- Un overhead de 10 octets pour l'ID de ligne, qui spécifie une adresse de redirection de ligne.

Le tableau 7-6 affiche le nombre maximal de lignes de données que peut contenir une page de données DOL :

Tableau 7-6 : Nombre maximal de lignes de données pour une page de données DOL

Taille de la page	Nombre maximal de lignes
2 ko	166
4 ko	337
8 ko	678
16 ko	1361

Les pages de données APL peuvent contenir un maximum de 256 lignes. Chaque page nécessitant doit avoir un indicateur de ligne d'un octet, les grandes pages avec lignes courtes comportent de l'espace non utilisé.

Par exemple, si Adaptive Server est configuré avec des pages logiques de 8 ko et des lignes de 25 octets, la page comporte 1275 octets d'espace non utilisé, une fois la table d'offset de ligne et l'en-tête de page pris en compte.

Nombres maximaux

Arguments des procédures stockées

Le nombre maximal d'arguments des procédures stockées est 2048. Reportez-vous au *Guide de l'utilisateur Transact-SQL* pour de plus amples informations.

Extraction de données avec des limites optimisées

Adaptive Server version 12.5 et ultérieure peut stocker des données ayant des limites différentes de celles des versions précédentes. Les clients doivent également pouvoir traiter les nouvelles limites des données. Si vous utilisez des versions précédentes d'Open Client et Open Server, elles ne peuvent pas traiter les données dans les cas suivants :

- Mise à niveau vers Adaptive Server version 12.5.
- Suppression et recréation de tables à larges colonnes.
- Insertion de données étendues.

Pour plus d'informations, reportez-vous à la section Open Client.

Données en vrac : tables sans index clusterisé

Si vous créez une table sans index clusterisé sur Adaptive Server, elle est stockée sous forme de *table sans index*. Les lignes de données ne sont pas stockées dans un ordre particulier. La présente section explique comment Adaptive Server effectue les sélections, insertions, suppressions et mises à jour sur ces tables en l'absence d'index exploitable facilitant l'extraction des données.

L'expression "absence d'index exploitable" est importante pour comprendre la décision de l'optimiseur d'effectuer un balayage de table. Parfois, il existe un index sur les colonnes nommées dans la clause where, mais l'optimiseur calcule qu'il est plus coûteux de l'utiliser que de balayer la table.

Vous trouverez dans d'autres chapitres de ce manuel des explications sur la manière dont l'optimiseur évalue le coût des requêtes au moyen des index ; vous y apprendrez également pourquoi l'optimiseur effectue tel ou tel choix.

Les balayages de table ont toujours lieu lorsque vous sélectionnez toutes les lignes d'une table. Seules font exception les requêtes comportant uniquement des colonnes qui constituent des clés dans un index non clusterisé.

Pour plus d'informations, reportez-vous à la section "Couverture par index", page 229.

Les sections suivantes décrivent la manière dont Adaptive Server repère les lignes dans une table sans index exploitable.

Plans de verrouillage et différences entre les tables sans index

Dans une table APL, les pages de données sont associées à une liste de pages doublement liée par des pointeurs sur chacune des pages. Les pages de ce type de table ne sont pas liées dans une chaîne de pages.

Dans une table APL, chaque page contient un pointeur qui renvoie à la page suivante et à la page précédente de la chaîne. Lorsqu'il est nécessaire d'insérer une nouvelle page, les pointeurs des deux pages adjacentes sont modifiés de façon à renvoyer à cette nouvelle page. Lorsqu'Adaptive Server balaie une table APL, il lit les pages dans l'ordre défini par les pointeurs.

Les pages sont doublement liées à chaque niveau d'index dans les tables APL et au niveau feuille des index dans les tables DOL. Si une table APL est partitionnée, il existe une chaîne de pages pour chaque partition.

Une autre différence entre les tables APL et les tables DOL est que ces dernières utilisent des ID de ligne fixes. Plus précisément, dans une table DOL, les ID de ligne (combinaison du numéro de page et du numéro de ligne dans la page) ne changent pas lors du traitement normal des requêtes.

Ils ne changent que dans le cas d'une opération nécessitant la copie de lignes de données, par exemple durant un processus reorg rebuild ou lors de la création d'un index clusterisé.

Pour plus d'informations sur les ID de ligne fixes et leur effet sur les opérations pour une table sans index, reportez-vous aux sections "Suppression dans une table DOL sans index", page 173 et "Tables DOL sans index", page 174.

Opérations de sélection sur les tables sans index

Lorsque vous lancez une opération de sélection (select) sur une table sans index non clusterisé exploitable, Adaptive Server doit balayer chaque page de données de la table pour rechercher toutes les lignes correspondant aux critères de recherche. Il peut trouver une ou plusieurs lignes ou n'en trouver aucune.

Dans des tables APL sans index

Pour les tables APL, Adaptive Server lit la colonne first dans sysindexes, lit la première page dans le cache, puis suit les pointeurs de pages suivantes jusqu'à la dernière page de la table.

Tables DOL sans index

Etant donné que les pages des tables DOL ne sont pas liées dans une chaîne, une requête select sur une table sans index utilise les pages d'OAM et d'allocation pour repérer toutes les lignes de cette table. La page d'OAM renvoie aux pages d'allocation, qui pointent vers les extents et les pages de la table.

Insertion de données dans une table APL sans index

Lorsque vous insérez des données dans une table APL sans index, la ligne de données est toujours ajoutée dans la dernière page de la table. Si une table ne comprend aucun index clusterisé et si elle n'est pas partitionnée, l'entrée sysindexes.root de cette table comporte un pointeur renvoyant à sa dernière page ; ce pointeur permet de localiser la page où les données doivent être insérées.

Si la dernière page est pleine, une nouvelle page est allouée dans l'extent courant et liée à la chaîne. Si l'extent est plein, Adaptive Server recherche des pages vides dans d'autres extents utilisés par la table. En l'absence de pages disponibles, un nouvel extent est alloué à la table.

Conflits lors des insertions dans des tables sans index

Dans des tables APL sans index, les performances sont considérablement altérées du fait que la page doit être verrouillée au moment de l'ajout de la ligne et que le verrou est maintenu jusqu'à la fin de la transaction. Lorsque plusieurs utilisateurs tentent d'insérer simultanément des données dans une table APL sans index, chacun doit attendre la fin de la transaction précédente pour réaliser l'insertion.

Ce problème de conflit au niveau de la dernière page dans les tables sans index se retrouve dans les cas suivants :

- insertion d'une seule ligne à l'aide d'insert.
- insertion de plusieurs lignes à l'aide de select into ou insert...select ou de plusieurs instructions insert regroupées dans un batch,
- bulkcopy dans la table.

Il existe des solutions pour pallier les conflits de dernière page dans les tables sans index, notamment :

- passer à un verrouillage des pages de données ou des lignes de données
- créer un index clusterisé qui oriente les insertions vers des pages différentes
- partitionner la table, ce qui crée plusieurs points d'insertion dans celle-ci et donc plusieurs "dernières pages" dans les tables APL.

D'autres principes sont également applicables aux transactions susceptibles de générer un conflit de verrouillage, en particulier :

- Réduction du temps d'exécution des transactions
- éviter toute activité du réseau et toute interaction des utilisateurs lorsqu'une transaction pose des verrous

Insertion de données dans une table DOL sans index

Lorsque les utilisateurs insèrent des données dans une table DOL sans index, Adaptive Server consigne les numéros de pages dans lesquelles ont eu lieu les dernières insertions ; ce numéro servira ensuite d'indication pour les prochaines tâches nécessitant de l'espace. Les insertions suivantes seront alors dirigées vers l'une de ces pages. Si la page est pleine, Adaptive Server alloue une nouvelle page et remplace l'ancienne indication par le nouveau numéro de page.

Les risques de blocage sont moindres lors d'insertions simultanées dans des tables DOL sans index. Toutefois, si un blocage survient, Adaptive Server alloue un petit nombre de pages vides et oriente les nouvelles insertions vers ces pages, en se reportant pour ce faire aux numéros des dernières pages allouées.

En ce qui concerne les tables verrouillées au niveau des lignes de données, il se produit un blocage uniquement au moment où les modifications apportées à la page de données sont écrites ; bien que les verrous de ligne soient posés pour la durée de la transaction, il reste possible d'insérer d'autres lignes dans la page. Les verrouillages au niveau de la ligne permettent à plusieurs transactions de poser des verrous sur la page.

Sur des tables DOL, Adaptive Server admet un léger temps de blocage immédiatement après l'allocation de nombreuses pages, de sorte que les nouvelles pages allouées sont remplies avant que d'autres ne soient allouées.

Conflit durant des insertions dans une table sans index

Les risques de conflit lors d'insertions dans des tables sans index sont considérablement réduits sur les tables DOL, mais ils existent malgré tout. Si ces conflits ralentissent les insertions, vous disposez de certains moyens pour les résoudre ; vous pouvez notamment :

- opter pour un verrouillage des lignes de données si la table utilise un verrouillage des pages de données ;
- utiliser un index clusterisé pour répartir les insertions ;
- partitionner la table, ce qui fournit des indications supplémentaires et permet l'allocation de nouvelles pages sur chaque partition en cas de blocage.

Suppression de données d'une table sans index

Lorsque vous supprimez des lignes dans une table sans index exploitable, Adaptive Server balaie la totalité des données de la table à la recherche des lignes à supprimer. En effet, rien ne lui permet de déterminer combien de lignes correspondent aux critères de la requête sans examiner chaque ligne.

Suppression dans une table APL sans index

Lorsqu'une ligne de données est supprimée d'une page dans une table APL, les lignes qui suivent remontent vers le haut afin que les données restent groupées.

Suppression dans une table DOL sans index

Lorsque vous supprimez des données d'une table DOL sans index, un balayage de table s'impose s'il n'existe pas d'index exploitable. Les pages d'OAM et d'allocation sont utilisées pour localiser les pages.

L'espace dans la page n'est pas immédiatement récupéré. Dans les tables DOL, les lignes doivent gérer des ID de ligne fixes et être réinsérées à la même place en cas d'annulation de la transaction.

Après une suppression, les lignes sont déplacées sur la page afin de libérer un espace non fragmenté, ce qui suppose le recours à l'un des processus suivants :

- Processus housekeeper
- Insertion qui doit trouver de l'espace sur la page
- Commande reorg reclaim_space

Suppression de la dernière ligne d'une page

Si vous supprimez la dernière ligne d'une page, la page est libérée. Si d'autres pages de l'extent sont toujours utilisées par la table, la page libérée peut à nouveau être utilisée par la table en cas de besoin.

Si toutes les autres pages de l'extent sont vides, l'extent tout entier est libéré. Il peut alors être alloué à d'autres objets de la base de données. En revanche, la première page de données d'une table ou d'un index n'est jamais libérée.

Mise à jour de données dans une table sans index

Comme les autres opérations effectuées sur les tables sans index, update donne lieu à un balayage de la table pour rechercher les lignes à modifier, lorsqu'il n'existe pas d'index exploitable sur les colonnes désignées dans la clause where.

Dans des tables APL sans index

Les mises à jour de tables APL sans index peuvent être exécutées de différentes manières :

- Si la longueur des lignes ne change pas, la ligne mise à jour remplace la ligne existante et les données ne sont pas déplacées sur la page.
- Si la longueur de la ligne change et la page compte suffisamment d'espace libre, la ligne reste à sa place et les autres lignes remontent ou descendent de façon à maintenir la contiguïté des données sur la page.

  Les pointeurs d'offset de ligne figurant en fin de page sont ajustés en fonction du nouvel emplacement des lignes.
- Si l'espace disponible sur la page est insuffisant pour accueillir la ligne, celle-ci est supprimée de la page courante et insérée sous forme d'une "nouvelle" ligne sur la dernière page de la table.

  Ce type de mise à jour peut provoquer un conflit sur la dernière page de la table, tout comme les insertions. En l'absence d'index non clusterisé, toutes les références d'index à la ligne doivent être mises à jour.

Tables DOL sans index

Un des impératifs spécifiques des tables DOL est que l'ID de ligne ne doit jamais changer (sauf lors de reconstructions intentionnelles de la table). Par conséquent, les mises à jour effectuées dans ces tables peuvent être réalisées à l'aide des deux premières méthodes précédemment décrites, dès l'instant que la ligne tient dans la page.

Mais lorsque, toujours dans ce type de table, une ligne mise à jour ne tient plus sur la page, un processus appelé **redirection de lignes** exécute les opérations suivantes :

- la ligne est insérée dans une page différente et
- un pointeur vers l'ID de ligne sur la nouvelle page est stocké à l'emplacement d'origine de la ligne.

En cas de redirection de lignes vers une autre page, il n'est pas nécessaire que les index soient modifiés. Tous les index continuent de pointer sur l'ID de ligne initial.

Si la ligne doit être redirigée une nouvelle fois vers une autre page, l'emplacement d'origine est mis à jour de façon à désigner la nouvelle page–la ligne n'est jamais redirigée au-delà d'un saut de page de son emplacement d'origine.

La redirection des lignes améliore la concurrence d'accès durant les opérations de mise à jour car les index n'ont pas besoin de mise à jour. Cependant, elle peut ralentir l'extraction des données, car une tâche doit lire la page au niveau de l'emplacement d'origine puis lire la page dans laquelle les données ont été stockées après la redirection.

Pour effacer des lignes redirigées dans une table, utilisez la commande reorg.

Pour plus d'informations sur les mises à jour, reportez-vous à la section "Mise à jour des opérations", page 501.

Opérations d'E/S d'Adaptive Server sur les tables sans index

Lorsqu'une requête demande une page de données, Adaptive Server commence par vérifier si cette dernière est accessible dans le cache de données. Si la page n'est pas accessible, elle doit être lue sur disque. Après son installation, Adaptive Server dispose d'un seul cache de données configuré pour des E/S de 2 ko. Un administrateur système peut :

- configurer plusieurs caches,
- lier des tables, des index ou des chaînes de texte aux caches.
- configurer les caches de données de façon à exécuter des E/S dont la taille correspond à un multiple de la taille de page (jusqu'à 8 pages de données, soit un extent).

Pour utiliser ces caches de la manière la plus efficace possible et réduire les opérations d'E/S, l'optimiseur d'Adaptive Server peut :

- opter pour la prélecture simultanée de plusieurs pages de données (jusqu'à huit pages)
- choisir parmi plusieurs stratégies de mise en mémoire cache.

Prélecture séquentielle (E/S étendues)

Adaptive Server permet à un administrateur système de configurer les caches de données pour des E/S étendues. Lorsqu'un cache est configuré pour des E/S étendues, Adaptive Server peut effectuer une prélecture des pages de données.

Les caches ont des zones de buffers qui dépendent de la taille des pages logiques, ce qui permet à Adaptive Server de lire jusqu'à un extent complet (huit pages de données) en une seule opération d'E/S.

La plus grande partie du temps requis par des opérations d'E/S étant consacrée à la recherche et au positionnement, la lecture de huit pages pour une E/S de 16 ko se déroule presque huit fois plus vite que la lecture d'une page (E/S de 2 ko), de sorte que les requêtes balayées par la table doivent être plus performantes en cas d'E/S étendues.

Lorsque plusieurs pages lues au cours d'une même opération d'E/S, elles sont traitées comme une seule unité : elles sont conservées ensemble dans le cache et, si l'une d'elles est modifiée, elles sont écrites ensemble sur le disque.

Pour plus d'informations sur la configuration des caches de mémoire pour les E/S étendues, reportez-vous au chapitre 14, "Utilisation et performances de la mémoire".

Caches et liaisons d'objets

Il est possible de lier une table à un cache. Lorsqu'une table n'est pas liée à un cache spécifique mais que sa base de données l'est, toutes les E/S passent par le cache lié à la base.

Sinon, toutes les E/S se font dans le cache par défaut. Le cache de données par défaut peut être configuré pour des E/S étendues. Si vos applications contiennent des tables sans index, elles fonctionneront sans doute mieux si elles utilisent un cache prévu pour des E/S de 16 ko.

Données sans index, E/S et stratégie sur les caches

Chaque cache de données d'Adaptive Server est géré comme une chaîne de buffers MRU/LRU ("most recently used/less recently used" : utilisation la plus récente/utilisation la plus ancienne). Au fur et à mesure de leur durée de séjour dans le cache, les buffers se déplacent de l'extrémité MRU vers l'extrémité LRU.

Lorsque les pages en mémoire cache qui ont été modifiées atteignent un point de la chaîne MRU/LRU appelé **marqueur de vidage**, Adaptive Server lance une écriture asynchrone sur ces pages. Cela garantit la mise à jour et la pertinence des pages se trouvant à l'extrémité LRU du cache.

Présentation des stratégies de caches

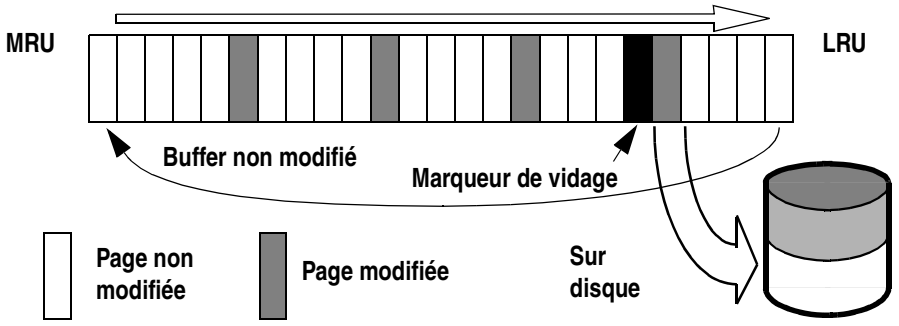
Adaptive Server dispose de deux stratégies principales pour assurer une utilisation efficace des caches de données :

- la stratégie de substitution LRU, généralement appliquée pour les pages auxquelles une requête doit accéder plusieurs fois ou pour les pages à mettre à jour
- la stratégie MRU, dite de *lecture-élimination* et utilisée pour les pages auxquelles la requête ne doit accéder qu'une seule fois.

Stratégie de remplacement LRU

La stratégie de remplacement LRU lit les pages de données en séquence dans le cache et remplace le buffer non utilisé depuis le plus longtemps. Le buffer est placé à l'extrémité MRU de la chaîne de buffers de données. Au fur et à mesure que d'autres pages sont lues dans le cache, il se déplace vers l'extrémité LRU.

Figure 7-2 : La stratégie LRU récupère une page non modifiée à l'extrémité LRU du cache



Utilisation de la stratégie LRU

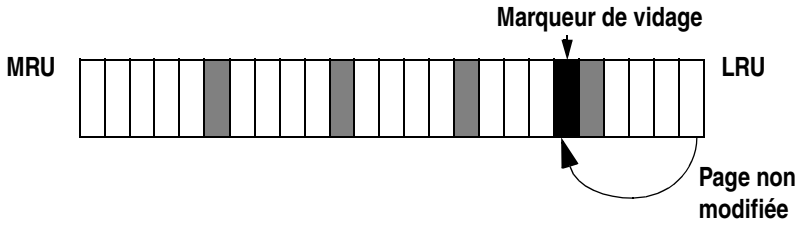
Adaptive Server utilise la stratégie LRU pour :

- les instructions qui modifient les données contenues dans les pages
- les pages requises plusieurs fois par une même requête
- les pages d'OAM
- la plupart des pages d'index
- les requêtes dans lesquelles la stratégie LRU est spécifiée

Stratégie de remplacement MRU

La stratégie MRU (lecture-élimination) est souvent employée pour le balayage de tables sans index. Cette stratégie place des pages dans le cache juste avant le marqueur de vidage (voir la figure 7-3).

Figure 7-3 : La stratégie MRU place les pages juste avant le marqueur de vidage



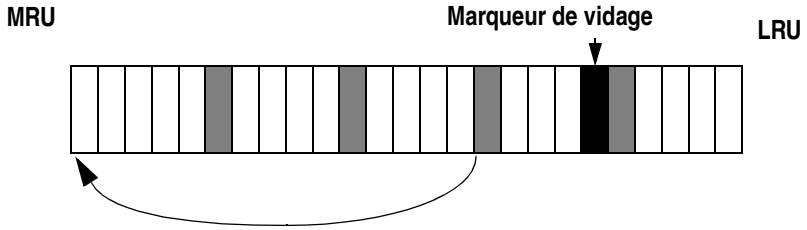
La lecture-élimination est le plus souvent utilisée pour les requêtes qui n'ont besoin d'une page qu'une seule fois. Celles-ci comprennent notamment :

- la plupart des balayages de table dans des requêtes qui n'utilisent pas de jointure
- une ou plusieurs tables dans une jointure.

Les pages requises n'étant placées qu'une seule fois au niveau du marqueur de vidage, elles ne font pas sortir les autres pages du cache.

La stratégie de lecture-élimination est employée uniquement pour les pages effectivement lues sur le disque pour la requête. Si une page se trouve déjà dans le cache, elle est placée à l'extrémité MRU du cache.

Figure 7-4 : Recherche de la page requise dans le cache



Opérations de sélection et mise en mémoire cache

Dans la plupart des cas, les opérations de sélection portant sur une table seule sans index utilisent :

- la taille d'E/S la plus grande possible et
- la stratégie de remplacement MRU (lecture-élimination)

Pour les données sans index, les opérations de sélection qui effectuent des E/S étendues peuvent être très efficaces. Adaptive Server peut lire séquentiellement tous les extents dans une table.

A l'exception des cas où la table sans index balayée constitue la table interne d'une jointure, la stratégie de substitution MRU lit les pages dans le cache et les élimine aussitôt, puisqu'elles ne sont requises qu'une seule fois par la requête.

Remarque Les E/S étendues sont efficaces pour traiter les tables APL sans index, à condition que les chaînes de pages ne soient pas fragmentées.

Pour plus d'informations sur la maintenance des données sans index, reportez-vous à la section "Gestion des tables sans index", page 183.

Modification de données et mémoire cache

Adaptive Server s'efforce de réduire au minimum les écritures sur disque en conservant les pages modifiées dans le cache. Plusieurs utilisateurs peuvent modifier une page de données pendant qu'elle se trouve dans le cache. Ces modifications sont consignées dans le journal de transactions mais les pages d'index et de données modifiées ne sont pas immédiatement écrites sur disque.

Mise en mémoire cache et insertions dans des tables sans index

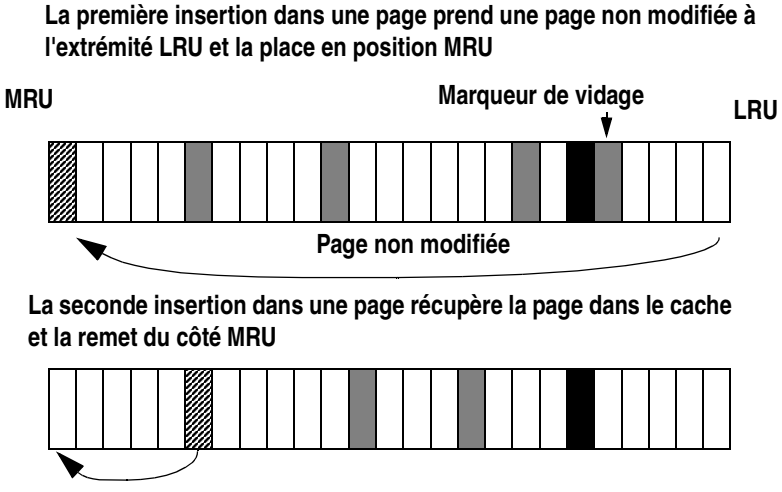
Pour les insertions dans les tables sans index, l'insertion a lieu :

- sur la dernière page d'une table verrouillée au niveau de toutes les pages (APL)
- sur une page qui a été récemment utilisée avec succès pour une insertion, dans une table DOL

Si une insertion constitue la première ligne d'une nouvelle page, Adaptive Server alloue un buffer de données non modifié pour stocker la page de données, comme indiqué dans la figure 7-5. A mesure que d'autres processus lisent des pages en mémoire, cette page se déplace dans le cache de données vers l'extrémité LRU.

Si la page fait l'objet d'une deuxième insertion alors qu'elle se trouve encore dans le cache, elle est localisée et déplacée vers l'extrémité MRU.

Figure 7-5 : Insertions dans une page de table sans index se trouvant dans le cache de données



La page de données modifiée reste dans le cache jusqu'à ce qu'elle atteigne l'extrémité LRU de la chaîne des pages. La page peut être modifiée ou référencée plusieurs fois pendant qu'elle est en cache, mais elle n'est écrite sur le disque que dans les conditions suivantes :

- la page dépasse le marqueur de vidage,
- un point de reprise ou la tâche housekeeper l'écrit sur le disque.

"Cache de données", page 325 fournit des détails sur ces processus.

Mise en mémoire cache et opérations de mise à jour et de suppression dans des tables sans index

Lorsque vous mettez à jour ou supprimez une ligne dans une table sans index, les effets au niveau du cache de données sont les mêmes que pour une insertion. Si la page requise se trouve déjà dans le cache, le buffer tout entier (soit, selon la taille d'E/S, une ou plusieurs pages) est placé à l'extrémité MRU de la chaîne et la ligne est modifiée.

Si la page n'est pas dans le cache, Adaptive Server la transfère du disque dans le cache et l'analyse pour déterminer si les lignes qu'elle contient répondent aux clauses de la requête. Son positionnement dans la chaîne MRU/LRU est différent selon qu'il faut ou non modifier des données :

- Si des données de la page doivent être modifiées, le buffer est placé à l'extrémité MRU. La page reste dans le cache jusqu'au moment où elle est vidée sur le disque ; elle peut y être mise à jour autant de fois que nécessaire ou lue par d'autres utilisateurs.
- Si aucune modification n'est requise, le buffer est placé dans le cache avant le marqueur de vidage.

Prélecture asynchrone et E/S de tables sans index

La prélecture asynchrone améliore les performances des requêtes qui effectuent des balayages de table. Les tâches qui doivent effectuer une opération physique d'E/S libèrent le moteur du serveur (CPU) pendant que l'E/S s'exécute.

Si un balayage de table nécessite la lecture de 1000 pages dont aucune n'est en cache, une E/S de 2 ko sans prélecture asynchrone exige que la tâche effectue 1000 boucles, avec une alternance des exécutions et des phases de veille. Dans ce cas, une E/S de 16 ko serait préférable, puisqu'elle ne nécessiterait que 125 boucles.

La prélecture asynchrone peut demander toutes les pages d'une unité d'allocation qui appartiennent à une table, lorsque la tâche extrait la première page de l'unité. Si une table de 1000 pages occupe simplement 4 unités d'allocation, la tâche requiert moins de cycles dans les boucles d'exécution et de veille.

Type de E/S	Boucles	Etapes de chaque boucle
E/S de 2 ko aucune prélecture	1000	Demande de page. Veille jusqu'à la lecture de la page sur le disque. Attente d'un cycle d'exécution sur le moteur (CPU) d'Adaptive Server. Lecture des lignes de la page.
E/S de 16 ko aucune prélecture	125	Demande d'extent. Veille jusqu'à la lecture de l'extent sur le disque. Attente d'un cycle d'exécution sur le moteur (CPU) d'Adaptive Server. Lecture des lignes dans les 8 pages.
prélecture	4	Demande de toutes les pages d'une unité d'allocation. Veille jusqu'à la lecture de la première page sur le disque. Attente d'un cycle d'exécution sur le moteur (CPU) d'Adaptive Server. Lecture de toutes les lignes dans toutes les pages du cache.

Les performances réelles dépendent de la taille du cache et des autres activités qui se déroulent dans le cache de données.

Pour plus d'informations sur la prélecture asynchrone, reportez-vous au chapitre 25, "Optimisation de la prélecture asynchrone".

Tables sans index : avantages et inconvénients

L'accès séquentiel au disque est performant, notamment pour des E/S étendues et des prélectures asynchrones. Cependant, la table entière doit être balayée pour permettre l'extraction d'une valeur, ce qui peut avoir des conséquences au niveau du cache de données et des autres requêtes.

Les insertions en batch peuvent se révéler efficaces avec les E/S séquentielles. Toutefois, un goulet d'étranglement peut se produire sur la dernière page lorsque plusieurs processus tentent d'insérer des données en même temps.

Les tables sans index fonctionnent bien à condition qu'elles ne soient pas trop grandes et qu'elles ne soient pas trop souvent modifiées. En revanche, cette structure ne convient pas aux grandes tables auxquelles s'appliquent des requêtes visant à renvoyer un sous-ensemble de lignes.

Cette solution est intéressante pour les tables qui :

- sont relativement petites et n'utilisent qu'un petit nombre de pages ;
- ne nécessitent pas d'accès direct à une ligne aléatoire unique ;
- ne nécessitent pas le tri des jeux de résultats.

Les tables sans index partitionnées constituent une solution utile pour les tables qui font fréquemment l'objet d'insertions en batch de gros volumes, puisque le surcoût lié à la suppression et à la création d'index clusterisés est alors excessif. Cette exception mise à part, les tables sans index n'ont pas vraiment de raison d'être. La plupart des applications s'exécutent en effet mieux lorsque leurs tables sont dotées d'index clusterisés.

Gestion des tables sans index

Au bout d'un certain temps, les E/S sur des tables sans index risquent de perdre en efficacité à mesure que le stockage se fragmente. Les suppressions et les mises à jour peuvent donner lieu à :

- Un grand nombre de pages partiellement remplies.
- Des E/S étendues inefficaces du fait que les extents risquent de contenir de nombreuses pages vides.
- Lignes redirigées dans les tables verrouillées sur les données seulement.

Méthodes

Une fois que les suppressions et les mises à jour ont libéré de l'espace sur les pages ou des pages dans les extents, procédez de l'une des façons suivantes pour récupérer l'espace dans les tables sans index :

- Utilisez la commande reorg rebuild (uniquement sur les tables DOL).
- Créez puis supprimez un index clusterisé.
- Utilisez bcp (l'utilitaire bulkcopy) et truncate table.

Utilisation de la commande *reorg rebuild* pour récupérer de l'espace

reorg rebuild copie toutes les lignes de données dans de nouvelles pages et reconstruit les index non clusterisés sur la table sans index. reorg rebuild n'est utilisable que dans les tables DOL.

Récupération d'espace par la création d'un index clusterisé

Vous pouvez créer et supprimer un index clusterisé sur une table sans index pour récupérer l'espace vide, si cette table comporte de nombreuses pages partiellement remplies suite à des opérations de mise à jour et de suppression. Pour permettre la création d'un index clusterisé, la base de données doit comporter un espace disponible correspondant à au moins 120 pour cent de la taille de la table.

Pour plus d'informations, reportez-vous à la section "Détermination de l'espace disponible pour des activités de maintenance", page 404.

Récupération d'espace à l'aide de *bcp*

Pour récupérer de l'espace avec bcp :

1 Copiez la table dans un fichier à l'aide de bcp.

2 Tronquez la table au moyen de la commande truncate table.

3 Copiez à nouveau la table à l'aide de bcp.

Pour connaître des procédures permettant de travailler avec des tables partitionnées, reportez-vous à la section "Etapes de partitionnement des tables", page 113.

Pour plus d'informations sur bcp, reportez-vous au document *Utilitaires* correspondant à votre plate-forme.

Journal de transactions : table spéciale sans index

Dans Adaptive Server, le journal de transactions est une table sans index spéciale qui stocke des informations sur les modifications apportées aux données de la base. Le journal de transactions est toujours une table sans index ; en effet, chaque nouvel enregistrement de transaction est ajouté à la fin du journal. Le journal de transactions ne contient aucun index.

Les chapitres suivants de ce manuel expliquent comment améliorer les performances du journal de transactions. La meilleure technique consiste à utiliser la clause log on avec create database, pour placer le journal de transactions sur un device autre que celui contenant les données.

Pour plus d'informations sur la création de base de données, reportez-vous au *Guide d'administration système*.

Les écritures sont très fréquentes dans le journal de transactions. Evitez qu'elles n'entrent en conflit avec d'autres E/S de la base de données, généralement effectuées de façon dispersée sur les pages de données.

Placez les journaux sur des devices physiques autres que ceux contenant les données et les pages d'index. Etant donné que le journal est séquentiel, la tête de lecture n'a que rarement à rechercher un emplacement, ce qui assure un débit d'E/S élevé sur le journal.

En dehors des opérations de reprise, les opérations qui nécessitent des lectures dans le journal de transactions sont les suivantes :

- Toute modification de données effectuée en mode différé.
- Triggers contenant des références aux tables insérées ou supprimées. Ces tables sont créées à partir des enregistrements du journal de transactions pour les besoins de la requête.
- Annulations de transaction.

Dans la plupart des cas, les pages du journal de transactions pour ces types de requête sont toujours disponibles dans le cache de données au moment où Adaptive Server a besoin de les lire, donc aucune E/S disque n'est requise.

CHAPITRE 8

Indexation pour l'optimisation des performances

Ce chapitre présente les principaux outils d'analyse des requêtes permettant de choisir les index et répertorie les critères de sélection des index pour les requêtes ponctuelles, les requêtes à intervalle et les jointures.

Sujet	Page
Incidence des index sur les performances	187
Symptômes d'une mauvaise indexation	189
Limites des index et ressources requises	192
Choix des index	193
Techniques de choix d'index	202
Index et gestion des statistiques	205
Conseils supplémentaires d'indexation	207

Incidence des index sur les performances

L'ajout d'index soigneusement choisis dans une base de données bien conçue permet d'obtenir d'excellentes performances avec Adaptive Server. Cependant, l'ajout d'index sans une analyse appropriée au préalable peut réduire les performances globales de votre système. De plus, les insertions, les mises à jour et les suppressions peuvent prendre plus de temps si les index à mettre à jour sont nombreux.

Analysez la charge de travail de l'application et générez des index en fonction du besoin d'optimisation des performances des principaux processus.

L'optimiseur de requêtes d'Adaptive Server utilise un modèle d'estimation basé sur le calcul des probabilités. Il analyse les ressources nécessaires aux différents plans d'exécution de requêtes et choisit celui qui en consomme le moins. Comme les ressources les plus consommatrices de temps dans les requêtes sont les E/S disque, la création d'un ensemble d'index adapté à vos applications permet à l'optimiseur d'utiliser les index pour :

- éviter les balayages de table lors de l'accès aux données
- cibler des pages de données contenant des valeurs particulières dans requête ponctuelle
- définir les limites supérieure et inférieure de lecture des données d'une requête à intervalle
- éviter complètement l'accès aux pages de données lorsque l'index couvre la requête
- utiliser des données ordonnées pour éviter les tris ou contribuer à des jointures par fusion sur des jointures à boucle imbriquée.

Vous pouvez également créer des index pour respecter l'unicité des données et répartir de façon aléatoire les emplacements de stockage des insertions.

Détection des problèmes d'indexation

Les principaux symptômes d'une indexation insuffisante ou incorrecte sont les suivants :

- L'exécution des instructions select dure trop longtemps.
- Une jointure entre deux ou plusieurs tables prend énormément de temps.
- Les traitements de modification de données ont du mal à s'exécuter, alors que les instructions select se déroulent bien.
- Les requêtes ponctuelles (par exemple "where colvalue = 3") s'exécutent bien, contrairement aux requêtes à intervalle (par exemple, "where colvalue > 3 and colvalue < 30").

Ces problèmes sont décrits dans les sections ci-après.

Symptômes d'une mauvaise indexation

L'un des buts principaux de l'amélioration des performances grâce aux index est d'éviter les balayages de table. Dans un balayage de table, chaque page doit être lue sur le disque.

Lorsqu'une requête cherche une valeur unique dans une table comptant 600 pages de données, elle effectue 600 lectures, physiques et logiques. En revanche, si un index pointe vers la valeur d'une donnée, 2 ou 3 lectures peuvent suffire pour cette même requête, ce qui correspond à une amélioration des performances de l'ordre de 200 à 300 pour cent.

Dans un système fonctionnant avec un disque de 12 ms, un index peut faire passer le temps de lecture de plusieurs secondes à moins d'une seconde. En outre, des E/S disque fréquentes ralentissent la capacité globale de traitement.

Balayages de table provoqués par un manque d'index

Si les instructions select et les jointures prennent trop de temps, il est probable qu'il n'existe pas d'index adapté ou, s'il en existe un, qu'il n'est pas utilisé par l'optimiseur.

showplan indique si l'accès à la table s'effectue via un balayage de table ou à l'aide d'un index. Si vous pensez que l'utilisation d'un index est nécessaire alors que showplan fait état d'un balayage de table, reportez-vous aux résultats de dbcc traceon (302) pour comprendre la raison de ce choix. Cet utilitaire affiche les estimations de coût concernant toutes les clauses susceptibles d'optimiser une requête.

Si une clause ne figure pas dans les résultats de dbcc traceon(302), cela peut être dû à un problème d'écriture de la clause. A l'inverse, si une clause dont vous pensez qu'elle risque de limiter le balayage est incluse dans les résultats de dbcc traceon(302), examinez soigneusement les données concernant son évaluation et celles du plan sélectionné par dbcc traceon(310).

Index insuffisamment sélectif

Un index est sélectif lorsqu'il aide l'optimiseur à trouver une ligne ou un groupe de lignes particulier. Une indexation sur un identificateur unique, tel qu'un numéro de sécurité sociale, est extrêmement sélective puisqu'elle permet à l'optimiseur de localiser une seule ligne. Une indexation sur une colonne non unique, telle que le sexe (M, F), n'est pas très sélective et l'optimiseur n'utilise ce type d'index que dans certains cas.

Index ne gérant pas les requêtes à intervalle

En général, les index clusterisés et les couvertures par index sont performants dans les requêtes à intervalle et dans les arguments de recherche (SARG) qui correspondent à de nombreuses lignes. Les requêtes à intervalle qui référencent les clés d'index non couvrants utilisent les index pour des intervalles renvoyant un nombre limité de lignes.

A mesure que le nombre de lignes renvoyées par la requête augmente, l'utilisation d'un index non clusterisé ou d'un index clusterisé dans une table verrouillée au niveau des pages de données seulement (table DOL) peut s'avérer plus coûteuse qu'un balayage de table.

Un trop grand nombre d'index ralentit la modification des données

Si vous constatez que les performances de modification de données sont médiocres, vérifiez que le nombre d'index n'est pas trop élevé. Les index favorisent les instructions select, mais ils ralentissent les modifications de données.

Chaque opération d'insertion ou de suppression se répercute au niveau feuille (et parfois à des niveaux supérieurs) d'un index clusterisé d'une table DOL, ainsi qu'au niveau des index non clusterisés, quel que soit le plan de verrouillage en vigueur.

Dans des tables verrouillées au niveau de toutes les pages (tables APL), les mises à jour appliquées aux clés d'index clusterisés peuvent entraîner des déplacements de lignes vers d'autres pages, ce qui nécessite la mise à jour de chaque index non clusterisé. Analysez les besoins de chaque index et essayez de supprimer ceux qui sont inutiles ou peu utilisés.

Entrées d'index trop volumineuses

Essayez de limiter la taille des entrées d'index. Vous pouvez créer des index avec des clés de 600 octets maximum mais ces index ne pourront stocker qu'un très petit nombre de lignes par page, ce qui augmente le nombre d'E/S disque nécessaires aux requêtes. Un index compte plusieurs niveaux de plusieurs pages chacun.

L'exemple suivant montre, à partir des informations renvoyées par sp_estspace comment le nombre de pages d'index et de niveaux de feuille requis augmente avec la taille de la clé. Des index non clusterisés sont générés avec des clés de 10, 20 ou 40 caractères.

```
create table demotable (c10 char(10),
                        c20 char(20),
                        c40 char(40))
create index t10 on demotable(c10)
create index t20 on demotable(c20)
create index t40 on demotable(c40)
sp_estspace demotable, 500000
```

Le tableau 8-1 répertorie les résultats.

Tableau 8-1 : Influence de la taille des clés sur les index et les niveaux

Index, taille de la clé	Pages de niveau feuille	Niveaux d'index
t10, 10 octets	4311	3
t20, 20 octets	6946	3
t40, 40 octets	12501	4

Le résultat indique que l'index sur la colonne de 10 caractères compte trois niveaux alors que les deux index sur les colonnes de 20 et 40 caractères comptent chacun quatre niveaux.

Le nombre de pages requis augmente de plus de 50 pour cent à chaque niveau.

Exception concernant les lignes de données et les index étendus

Cette procédure peut être utile dans les cas suivants :

- La table contient de très longues lignes, de sorte que leur nombre par page est faible.
- L'ensemble des requêtes exécutées sur la table propose des choix logiques pour une couverture par index.
- Les requêtes renvoient un nombre suffisamment élevé de lignes.

Par exemple, si une table contient de longues lignes et une seule ligne par page, une requête devant renvoyer 100 lignes devra accéder à 100 pages de données. Un index couvrant cette requête, même si les lignes sont longues, peut améliorer les performances.

Ainsi, si les lignes sont longues de 240 octets, l'index stocke 8 lignes par page et la requête n'a besoin d'accéder qu'à 12 pages d'index.

Limites des index et ressources requises

Sur Adaptive Server, les index présentent les limites suivantes :

- Vous ne pouvez créer qu'un index clusterisé par table car les données de ce type d'index sont triées en fonction de la clé de l'index.
- Vous ne pouvez pas créer plus de 249 index non clusterisés par table.
- Une clé peut contenir jusqu'à un maximum de 31 colonnes. Le nombre maximum d'octets par clé d'index est de 600.
- Lorsque vous créez un index clusterisé, Adaptive Server a besoin d'espace pour copier les lignes de la table et affecter de l'espace aux pages d'index clusterisé. Il a également besoin d'espace pour régénérer les index non clusterisés de la table.

  Les besoins en espace varient selon le degré de remplissage des pages de la table lorsque vous commencez, et selon les propriétés de gestion qui sont appliquées à la table et aux pages d'index.

  Pour plus d'informations, reportez-vous à la section "Détermination de l'espace disponible pour des activités de maintenance", page 404.
- Les contraintes d'intégrité référentielle unique et primary key génèrent des index uniques permettant d'appliquer les restrictions imposées au niveau des clés. Par défaut, les contraintes de type unique génèrent des index non clusterisés et celles de type primary key, des index clusterisés.

Choix des index

Lorsque vous effectuez une sélection d'index, vous devez vous poser les questions suivantes :

- Quels index sont généralement associés à une table donnée ?
- Quels sont les principaux processus qui utilisent la table ?
- Quel est le rapport entre les opérations de sélection et les modifications de données réalisées sur la table ?
- Un index clusterisé a-t-il été attribué à la table ?
- L'index clusterisé peut-il être remplacé par un index non clusterisé ?
- L'un des index couvre-t-il une ou plusieurs requêtes principales ?
- Un index composé est-il nécessaire pour que l'unicité d'une clé primaire composée soit respectée ?
- Quels index peuvent être définis comme uniques ?
- Quelles sont les principales spécifications concernant les tris ?
- Existe-t-il des requêtes qui utilisent un ordre décroissant des jeux de résultats ?
- Les index gèrent-ils des jointures et les contrôles d'intégrité référentielle ?
- L'indexation a-t-elle une incidence sur le type de modification (immédiate ou différée) ?
- Quels index sont nécessaires au positionnement du curseur ?
- Si des lectures de données modifiées sont requises, existe-t-il des index uniques pour prendre en charge le balayage ?
- Des colonnes IDENTITY doivent-elles être ajoutées aux tables et aux index afin de générer des index uniques ? Les index uniques sont nécessaires pour les curseurs modifiables et les lectures de données modifiées.

Lorsque vous choisissez le nombre d'index à utiliser, tenez compte des éléments suivants :

- Contraintes d'espace
- Chemins d'accès à la table

- Pourcentage des modifications de données par rapport aux opérations de sélection
- Performances requises en matière de génération d'états comparées à celles des traitements transactionnels (OLTP)
- Incidence des performances sur les modifications d'index
- Nombre de fois où vous pouvez lancer update statistics

Clés d'index et clés logiques

Vous devez distinguer les clés d'index des clés logiques. Les clés logiques font partie de la conception de la base de données et définissent les relations entre les tables. clés primaires, clés étrangères et clés communes.

Lorsque vous optimisez les requêtes en créant des index, il est possible d'utiliser ces clés logiques comme clés physiques servant à la création d'index. Vous pouvez générer des index sur des colonnes ne correspondant pas à des clés logiques ou disposer de clés logiques qui ne sont pas utilisées comme clés d'index.

Pour améliorer les performances, choisissez des clés d'index. Créez des index sur les colonnes qui supportent des jointures, des arguments de recherche et des conditions de tri dans les requêtes.

Une erreur courante consiste à créer l'index clusterisé pour une table sur la clé primaire, même si elle n'est jamais utilisée dans les requêtes à intervalle ou dans le classement des jeux de résultats.

Recommandations relatives aux index clusterisés

Vous trouverez ci-après quelques recommandations générales concernant les index clusterisés.

- La plupart des tables APL doivent avoir des index clusterisés ou utiliser des partitions pour réduire les conflits sur la dernière page des tables sans index.

  Si le nombre de transactions est élevé, le verrouillage de la dernière page limite considérablement le débit.

- Si votre environnement requiert de nombreuses insertions, la clé de l'index clusterisé ne doit pas être placée sur une valeur qui augmente de façon uniforme, comme une colonne IDENTITY.

  Pour réduire les conflits de verrous tout en disposant d'une clé exploitable par de nombreuses requêtes, choisissez une clé qui répartit les insertions sur les pages de façon "aléatoire". Souvent, la clé primaire n'est pas conforme à cette recommandation.

  Le problème est moins grave sur des tables DOL, mais il est à l'origine de bien des conflits de verrous sur les tables APL.

- Les index clusterisés sont très performants lorsque la clé correspond à l'argument de recherche des requêtes à intervalle, comme dans le cas suivant :

  ```
  where colvalue >= 5 and colvalue < 10
  ```

  Dans des tables APL, les lignes sont gérées dans l'ordre des clés et les pages sont reliées séquentiellement, ce qui améliore les performances des requêtes utilisant un index clusterisé.

  Dans les tables DOL, les lignes sont gérées dans l'ordre des clés une fois l'index créé, mais la clusterisation diminue avec le temps.

- Les colonnes utilisées dans les clauses order by et dans les jointures constituent également des clés d'index clusterisés performantes.

- Si possible, évitez d'inclure des colonnes fréquemment modifiées comme clés d'index clusterisés dans des tables APL.

  Lorsque les clés sont modifiées, les lignes doivent être déplacées vers une nouvelle page. De plus, si l'index clusterisé n'est pas unique, les modifications sont effectuées en différé.

Choix des index clusterisés

Choisissez les index conformes aux types de clauses where ou de jointures que vous exécutez. Les index clusterisés peuvent être constitués des éléments suivants :

- La clé primaire, lorsqu'elle est utilisée pour les clauses where et qu'elle répartit les inserts de façon aléatoire
- Les colonnes auxquelles l'accès s'effectue par le biais d'un intervalle comme :

  ```
  col1 between 100 and 200
  col12 > 62 and < 70
  ```

- Les colonnes utilisées par order by
- Les colonnes qui ne sont pas souvent modifiées
- Les colonnes utilisées dans les jointures

Si vous avez plusieurs possibilités, choisissez en priorité l'ordre physique le plus souvent utilisé.

En second choix, optez pour les requêtes à intervalle. Lors des tests de performances, vérifiez s'il existe des "goulots d'étranglement" résultant de conflits de verrouillage.

Candidats en tant qu'index non clusterisés

Pour choisir les colonnes à utiliser dans un index non clusterisé, prenez en considération tous les cas pour lesquels le choix effectué ne convient pas. Envisagez également d'utiliser des colonnes qui peuvent améliorer les performances grâce à un index couvrant la requête.

Dans les tables DOL, les index clusterisés peuvent procéder par couverture puisqu'ils comportent un niveau feuille au-dessus du niveau de données.

Dans les pages APL, les requêtes à intervalle non couvertes par un index fonctionnent bien avec des index clusterisés, mais elles ne sont pas forcément améliorées par des index non clusterisés, selon la taille de l'intervalle.

Pour couvrir les requêtes principales tout en traitant les requêtes les moins fréquentes, utilisez des index composés :

- Les requêtes les plus importantes doivent pouvoir exécuter des requêtes ponctuelles et des lectures avec correspondance.
- Les autres requêtes doivent pouvoir effectuer des lectures sans correspondance, en utilisant l'index et en évitant les balayages de table.

Autres recommandations concernant les index

Vous trouverez ci-après d'autres considérations à prendre en compte dans le choix d'un index :

- Si une clé d'index est unique, définissez-la comme telle afin que l'optimiseur détecte immédiatement qu'il n'existe qu'une seule ligne correspondant à un argument de recherche ou à une jointure.

- Si la conception de votre base de données prévoit d'utiliser l'intégrité référentielle (mots-clés references ou foreign key...references dans l'instruction create table), les colonnes référencées *doivent* posséder un index unique, sinon la tentative de créer une contrainte d'intégrité référentielle échoue.

  Cependant, Adaptive Server ne génère pas automatiquement un index sur la colonne de référence. Si votre application modifie des clés primaires ou supprime des lignes dans les tables correspondantes, vous pouvez créer un index sur la colonne de référence afin d'éviter un balayage de table.

- Si vos applications utilisent des curseurs, reportez-vous à la section "Utilisation d'index et conditions requises pour les curseurs", page 696.

- Si vous créez un index sur une table dans laquelle de nombreuses insertions sont prévues, utilisez fillfactor pour limiter temporairement les fractionnements de pages, améliorer la concurrence d'accès et réduire les risques d'interblocage.

- Si vous créez un index sur une table accessible en lecture seule, utilisez un fillfactor (facteur de remplissage) de valeur 100 afin de réduire le plus possible la taille de la table ou de l'index.

- Réduisez le plus possible la taille de la clé. Vos arbres d'index restent plus plats, ce qui accélère les traversées.

- Utilisez des types de données réduits chaque fois que cela est satisfaisant.

  - En interne, les données numériques peuvent être comparées plus rapidement que les chaînes.

  - Les données de type caractères de longueur variable et de type binaire requièrent un surcoût de lignes plus important que les données de longueur fixe. Par conséquent, s'il y a peu de différence entre la longueur moyenne d'une colonne et la longueur définie, utilisez une longueur fixe. Par définition, les données de type caractères et de type binaire qui acceptent les valeurs NULL sont de longueur variable.

  - Utilisez dans la mesure du possible des types de données de longueur fixe, non NULL pour les colonnes courtes qui seront utilisées comme clés d'index.

- Assurez-vous que les types de données des colonnes de jointure dans les différentes tables soient compatibles. Si Adaptive Server doit convertir un type de données sur un côté d'une jointure, il est possible qu'il n'utilise pas d'index pour cette table.

  Pour plus d'informations, reportez-vous à la section "Types de données incompatibles et optimisation des requêtes", page 434.

Choix des index non clusterisés

Lorsque vous envisagez de créer des index non clusterisés, vous devez évaluer si l'amélioration du temps d'extraction des données compense l'augmentation du temps de modification de ces données. Vous devez en outre prendre en considération les points suivants :

- Quel espace les index vont-ils occuper ?
- Quel est le degré de volatilité de la colonne candidate ?
- Quel est le degré de sélectivité des clés d'index ? Un balayage serait-il plus approprié ?
- Les valeurs en double sont-elles nombreuses ?

Pour des questions d'overhead lors de la modification des données, ajoutez des index non clusterisés uniquement si les tests indiquent qu'ils sont utiles.

Coût en performances des modifications de données

Pour les tables APL, tous les index non clusterisés doivent être mis à jour :

- Pour chaque insertion dans la table
- Pour chaque suppression de la table

Une mise à jour de la table qui modifie une partie d'une clé d'index nécessite de mettre à jour uniquement cet index.

Pour les tables APL, tous les index doivent être mis à jour :

- Lors de chaque mise à jour qui entraîne le déplacement d'une ligne vers une autre page suite à la modification d'une clé d'index clusterisé
- Chaque fois qu'une ligne est concernée par le fractionnement d'une page de données

Dans les tables APL, des verrous exclusifs sont posés sur les pages d'index concernées, et ce pour toute la durée de la transaction, ce qui augmente les conflits de verrous ainsi que l'overhead de traitement.

Les performances de certaines applications sont pénalisées de façon inacceptable pour seulement trois ou quatre index sur des tables qui subissent d'importantes modifications des données. D'autres fonctionnent très bien alors que les tables sont plus nombreuses.

Choix des index composés

Si l'analyse des besoins révèle plusieurs colonnes candidates pour servir de clé d'index clusterisé, vous pouvez envisager un accès de type clusterisé avec un index composé qui couvre une requête particulière ou un ensemble de requêtes, à savoir : Ces tables :

- Requêtes sur un intervalle.
- Agrégats vectoriels (groupés) lorsqu'ils comprennent à la fois les colonnes groupées et les colonnes de regroupement. Les arguments de recherche doivent également être inclus dans l'index.
- Les requêtes qui renvoient un grand nombre de valeurs dupliquées.
- Les requêtes incluant une clause order by.
- Les requêtes effectuant des balayages de table, mais qui utilisent un sous-ensemble réduit des colonnes de la table.

Les tables en lecture seule ou en lecture majoritaire peuvent être largement indexées tant que l'espace disponible dans la base de données est suffisant. Si le nombre de modifications est faible et celui des sélections élevé, vous devez prévoir des index pour toutes les requêtes fréquentes. Evaluez attentivement les améliorations des performances qui peuvent résulter de l'utilisation d'un index couvrant la requête.

Ordre des clés et performances dans les index composés

Les requêtes par couverture peuvent faire état d'excellents temps de réponse lorsque les colonnes principales sont utilisées.

La requête suivante s'exécute très rapidement avec l'index non clusterisé composé établi sur au_lname, au_fname, au_id :

```
select au_id
    from authors
where au_fname = "Eliot" and au_lname = "Wilk"
```

Cette requête ponctuelle doit lire uniquement les niveaux supérieurs de l'index et une seule page au niveau de la ligne feuille dans l'index non clusterisé d'une table de 5 000 lignes.

La requête suivante, qui semble identique (elle utilise le même index), n'est pas aussi performante. En effet, bien que procédant toujours par couverture, elle effectue une recherche sur au_id :

```
select au_fname, au_lname
    from authors
where au_id = "A1714224678"
```

Du fait qu'elle n'inclut pas la principale colonne de l'index, elle doit balayer le niveau feuille complet, ce qui correspond à près de 95 lectures.

L'ajout d'une colonne dans la liste de sélection de cette requête aggrave encore la situation (bien que le changement paraisse minime) :

```
select au_fname, au_lname, phone
    from authors
where au_id = "A1714224678"
```

Ici, la requête effectue un balayage de table, soit la lecture de 222 pages. Dans ce cas, les performances sont nettement moins bonnes. Avec un argument de recherche qui ne correspond pas à la colonne principale, Adaptive Server ne dispose que de deux méthodes d'accès possibles : un balayage de table, ou un balayage avec couverture d'index.

Il ne balaie pas le niveau feuille de l'index si l'argument n'est pas une colonne principale. Un index composé n'est performant que s'il couvre la requête ou si la première colonne apparaît dans la clause where.

Pour une requête contenant la colonne principale de l'index composé, l'ajout d'une colonne non incluse dans l'index n'entraîne la lecture que d'une page de données. La requête ci-dessous doit lire la page de données pour trouver le numéro de téléphone :

```
select au_id, phone
    from authors
where au_fname = "Eliot" and au_lname = "Wilk"
```

Le tableau 8-2 indique les caractéristiques de performances des différentes clauses where avec un index non clusterisé sur au_lname, au_fname, au_id et aucun autre index sur la table.

Tableau 8-2 : Ordre des index non clusterisés composés et incidence sur les performances

Colonnes indiquées dans la clause where	Performances avec les colonnes indexées dans la liste de sélection	Performances avec les autres colonnes dans la liste de sélection
au_lname ou au_lname, au_fname ou au_lname, au_fname, au_id	Performances satisfaisantes ; l'index est utilisé pour descendre l'arborescence ; pas d'accès au niveau des données	Performances satisfaisantes ; l'index est utilisé pour descendre l'arborescence ; accès aux données (une lecture de page supplémentaire par ligne)
au_fname ou au_id ou au_fname, au_id	Performances moyennes ; lecture de l'index avant de renvoyer les valeurs	Performances médiocres, pas d'utilisation de l'index, balayage de table

Choisissez l'ordre de l'index composé de telle sorte que la majorité des requêtes constitue un sous-ensemble ordonné.

Avantages et inconvénients des index composés

Les index composés présentent les avantages suivants :

- Un index composé peut couvrir de nombreuses requêtes.
- Si une requête comporte des arguments de recherche sur chaque clé, l'index composé a besoin de moins d'E/S que la même requête utilisant un index sur un simple attribut.
- Les index composés constituent un bon moyen de garantir l'unicité des attributs multiples.

Les éléments suivants sont particulièrement adaptés à l'emploi d'index composés :

- les tables de consultation,
- les colonnes souvent sollicitées simultanément,
- les colonnes utilisées pour les agrégats vectoriels,
- les colonnes qui génèrent un sous-ensemble fréquemment utilisé à partir d'une table comportant de très longues lignes.

Les index composés présentent les inconvénients suivants :

- Les index composés tendent à avoir des entrées volumineuses. Par conséquent un plus petit nombre d'entrées d'index par page et plus de pages d'index à lire.
- La mise à jour de l'un des attributs d'un index composé entraîne une modification de l'index. Ne choisissez pas les colonnes souvent soumises à des modifications.

Il est préférable d'éviter :

- les index presque aussi volumineux que la table,
- les index composés pour lesquels seule une clé mineure est utilisée dans la clause where.

Techniques de choix d'index

Cette section analyse deux requêtes qui doivent accéder à une table unique et les choix d'index possibles correspondants. Ces deux requêtes sont les suivantes :

- une requête à intervalle qui renvoie un grand nombre de lignes,
- une requête ponctuelle qui ne renvoie qu'une ou deux lignes.

Choix d'un index pour une requête à intervalle

Supposons que vous deviez améliorer les performances de la requête suivante :

```
select title
    from titles
    where price between $20,00 and $30,00
```

Il existe un certain nombre de statistiques élémentaires concernant la table :

- Elle contient 1 000 000 lignes et utilise le verrouillage de toutes les pages (APL).
- Les pages contiennent 10 lignes, elles sont pleines à 75 pour cent et la table comporte donc environ 135 000 pages.
- 190 000 (19 %) des titres sont vendus entre 20 et 30 dollars.

Sans index, la requête balaye les 135 000 pages.

Avec un index clusterisé sur price, la requête identifie le premier ouvrage à 20 dollars et commence à lire les lignes suivantes les unes après les autres jusqu'au dernier ouvrage à 30 dollars. Avec des pages pleines à 75 pour cent, le nombre moyen de lignes par page est de 7,5. Pour lire 190 000 lignes, la requête lirait environ 25 300 pages, plus 3 ou 4 pages d'index.

Avec un index non clusterisé sur price et une répartition aléatoire des valeurs price, l'utilisation de l'index pour trouver les lignes correspondant à cette requête implique la lecture de 19 % des pages de niveau feuille de l'index, soit environ 1 500 pages.

Si les prix sont répartis de façon aléatoire, le nombre de pages de données à lire sera sans doute plus élevé, peut-être même égal à celui des lignes répondant au critère, soit 190 000. Puisqu'un balayage de la table ne requiert que 135 000 pages, l'index non clusterisé ne sera probablement pas utilisé.

Une autre option consisterait à créer un index non clusterisé sur price, title. La requête peut réaliser une lecture d'index avec correspondance, détecter, à l'aide de l'index, la première page contenant un ouvrage à 20 dollars, puis balayer les niveaux feuille suivants jusqu'à ce qu'un prix supérieur à 30 dollars soit trouvé. Cet index implique environ 35 700 pages de niveau feuille, soit à peu près 19 % des pages de l'index ou 6 800 lectures pour balayer les pages de niveau feuille.

Pour cette requête, la meilleure méthode est donc de créer un index non clusterisé sur price, title.

Ajout d'une requête ponctuelle avec des critères d'indexation différents

Le choix de l'index pour la requête à intervalle portant sur price n'a pas posé de problème, compte tenu des index utiles envisagés. A présent, supposons que la requête nécessite également une recherche sur titles :

```
select price
from titles
where title = "Looking at Leeks"
```

Vous savez qu'il existe très peu de titres en double et que la requête ne renvoie donc qu'une ou deux lignes.

Si l'on considère cette requête et la précédente, le tableau 8-3 montre quatre stratégies d'indexation possibles, avec une estimation du coût pour chaque index. Les évaluations relatives au nombre de pages d'index et de données ont été réalisées avec sp_estspace à partir d'un taux de remplissage de 75 % :

```
sp_estspace titles, 1000000, 75
```

Les valeurs ont été arrondies afin de faciliter la comparaison.

Tableau 8-3 : Comparaison des stratégies d'indexation des deux requêtes

Choix d'indexation possible		Pages d'index	Requête à intervalle sur price	Requête ponctuelle sur title
1	Index non clusterisé sur title Index clusterisé sur price	36 800 650	Index clusterisé d'environ 26 600 pages (135 000 * 0,19) Avec E/S de 16 ko : 3 125 E/S	Index non clusterisé, 6 E/S
2	Index clusterisé sur title non clusterisé sur price	3 770 6 076	Balayage de table, 135 000 pages Avec E/S de 16 ko : 17 500 E/S	Index clusterisé, 6 E/S
3	Index non clusterisé sur title, price	36 835	Lecture d'index sans correspondance, près de 35 700 pages Avec E/S de 16 ko : 4 500 E/S	Index non clusterisé, 5 E/S
4	Index non clusterisé sur price, title	36,835	Lecture d'index avec correspondance, environ 6 800 pages (35 700 * 0,19) Avec E/S de 16 ko : 850 E/S	Lecture d'index sans correspondance, près de 35 700 pages Avec E/S de 16 ko : 4 500 E/S

Les chiffres du tableau 8-3 indiquent les résultats suivants :

- Pour la requête à intervalle sur price, la solution 4 est la plus appropriée, les choix 1 et 3 restant satisfaisants avec des E/S de 16 ko.
- Pour la requête ponctuelle sur titles, les choix d'indexation 1, 2 et 3 sont excellents.

La meilleure stratégie d'indexation pour la combinaison de ces deux requêtes consiste à utiliser deux index :

- Le choix 4, pour les requêtes à intervalle sur price.
- Le choix 2, pour des requêtes ponctuelles sur title, puisque l'index clusterisé occupe très peu de place.

Il se peut que vous ayez besoin d'informations supplémentaires pour déterminer la stratégie d'indexation à utiliser pour supporter plusieurs requêtes. Les éléments suivants sont généralement à prendre en considération :

- Quelle est la fréquence de chaque requête ? Combien de fois est-elle exécutée par heure ou par jour ?
- Quels sont les temps de réponse requis ? Le critère temps est-il spécialement important pour l'une de ces requêtes ?
- Quels sont les temps de réponse requis pour les mises à jour ? La création de plusieurs index ralentit-elle les mises à jour ?
- L'intervalle des valeurs est-il caractéristique ? Des intervalles de prix plus ou moins étendus, par exemple de 20 à 50 dollars, sont-ils souvent utilisés ? Quelle incidence les différents intervalles ont-ils sur le choix de l'index ?
- Le cache de données est-il étendu ? Ces requêtes sont-elles suffisamment importantes pour fournir un cache de 35 000 pages pour les index composés non clusterisés des solutions 3 ou 4 ? Les performances seraient excellentes si cet index était lié à son propre cache.
- Quels sont les autres arguments de recherche et les autres requêtes utilisés ? La table est-elle fréquemment jointe à d'autres tables ?

Index et gestion des statistiques

Pour vous assurer que les index évoluent avec votre système :

- Surveillez les requêtes pour déterminer si les index sont toujours appropriés pour vos applications.

  Régulièrement, vérifiez les plans d'exécution des requêtes comme indiqué au chapitre 35, "Utilisation de set showplan", et les statistiques d'E/S pour les requêtes fréquemment utilisées. Accordez une attention particulière aux index non couvrants qui supportent des requêtes à intervalle. Si la répartition des données change, il est probable que les index soient remplacés par des balayages de table.
- Supprimez et recréez les index qui limitent les performances.

- Maintenez les statistiques des index à jour.
- Utilisez les propriétés de gestion de l'espace pour réduire le nombre des fractionnements de pages ou la fréquence des opérations de maintenance.

Suppression des index qui limitent les performances

Supprimez les index qui limitent les performances. Si, au cours de la journée, une application modifie des données et génère des rapports durant la nuit, vous pouvez supprimer certains des index le matin pour les recréer le soir.

Souvent, les concepteurs créent de nombreux index qui sont rarement, voire jamais, utilisés par l'optimiseur de requêtes. Assurez-vous que les index soient basés sur les transactions et les processus en cours d'exécution et non sur la structure de la base de données d'origine.

Pour savoir si vos index sont utilisés, vérifiez les plans d'exécution de requêtes.

Pour plus d'informations sur la gestion des index, reportez-vous aux sections "Gestion des index et des statistiques sur les colonnes", page 393 et "Reconstruction des index", page 394.

Choix des propriétés de gestion de l'espace des index

Les propriétés de gestion de l'espace aident à réduire la fréquence de maintenance des index. En particulier, fillfactor réduit le nombre de fractionnements sur les pages feuille des index non clusterisés et sur les pages de données des tables APL avec index clusterisés.

Pour plus d'informations sur le choix des valeurs de fillfactor pour les index, reportez-vous au chapitre 13, "Définition des propriétés de gestion de l'espace",.

Conseils supplémentaires d'indexation

Les suggestions suivantes permettent d'améliorer les performances lorsque vous créez et utilisez les index :

- Modifiez la conception logique afin d'utiliser une colonne artificielle et une table de consultation pour les tables requérant un grand nombre d'entrées d'index.
- Réduisez la taille des entrées pour les index fréquemment utilisés.
- Supprimez les index dans les périodes de fréquentes mises à jour et recréez-les avant des périodes de sélections fréquentes.
- Si vous effectuez souvent des opérations de mise à jour d'index, configurez votre serveur afin d'accélérer le tri.

  Pour plus d'informations sur les paramètres de configuration qui permettent un tri plus rapide, reportez-vous à la section "Configuration d'Adaptive Server pour accélérer le tri", page 391.

Création de colonnes artificielles

Lorsque les index deviennent trop volumineux, surtout les index composés, il est préférable de créer une colonne artificielle affectée à une ligne, ainsi qu'une table de consultation secondaire utilisée pour la conversion entre les ID internes et les colonnes d'origine.

Cette opération peut augmenter le temps de réponse de certaines requêtes, mais grâce à un index plus compact et des lignes de données plus courtes, elle permet une amélioration globale des performances, qui vaut généralement la peine d'être tentée.

Entrées d'index courtes pour réduire l'overhead

Evitez de stocker des ID entièrement numériques comme données de type caractère. Utilisez des ID entiers ou numériques chaque fois que possible pour :

- économiser l'espace de stockage sur les pages de données,
- réduire les entrées d'index,
- améliorer les performances, car les comparaisons internes sont plus rapides.

Les entrées d'index sur les colonnes varchar requièrent un overhead plus important que sur des colonnes char. Pour les clés d'index courtes, notamment celles dont la longueur varie peu dans les données de colonne, utilisez char afin d'obtenir des entrées d'index plus compactes.

Suppression et reconstruction des index

Avant de procéder à une importante série d'insertions, vous pouvez supprimer des index non clusterisés que vous reconstruirez ultérieurement. Vous accélérez ainsi les insertions et les opérations de bulk copy, car les index non clusterisés n'ont pas besoin d'être mis à jour à chaque insertion.

Pour plus d'informations, reportez-vous à la section "Reconstruction des index", page 394.

CHAPITRE 9

Fonctionnement des index

Ce chapitre décrit comment Adaptive Server stocke les index et comment il les utilise pour améliorer l'extraction de données lors d'opérations de sélection (select), de mise à jour (update), de suppression (delete) et d'insertion (insert).

Sujet	Page
Types d'index	210
Index clusterisés dans des tables APL	213
Index non clusterisés	222
Couverture par index	229
Index et mise en mémoire cache	232

Les index sont l'élément de conception physique le plus important pour obtenir des performances élevées dans les tâches de bases de données :

- Ils évitent les balayages de table. La lecture de centaines de pages de données est inutile car quelques pages d'index et pages de données satisfont à de nombreuses requêtes.
- Pour certaines requêtes, des données peuvent être extraites à partir d'un index non clusterisé sans nécessiter l'accès aux lignes de données.
- Les index clusterisés permettent des insertions de données aléatoires, évitant ainsi l'apparition de goulets d'étranglement sur la dernière page d'une table.
- Si l'ordre de l'index correspond à l'ordre des colonnes dans une clause order by, la présence d'index peut éviter les phases de tri.

Outre les gains de performances, les index garantissent l'unicité des données.

Les index sont des objets de base de données créés pour une table et permettant d'accéder directement à une ou plusieurs lignes de données. Ils stockent les valeurs d'une ou des clés nommées lors de la création de l'index ainsi que des pointeurs logiques vers les pages de données ou vers d'autres pages d'index.

Bien que les index accélèrent la recherche de données, ils ralentissent les modifications de données qui, dans la plupart des cas, requièrent également la mise à jour des index. Pour une indexation optimale, il faut :

- bien comprendre le comportement des requêtes qui accèdent à des tables sans index, à des tables avec index clusterisés et à des tables avec index non clusterisés ;
- bien comprendre les combinaisons de requêtes qui s'exécutent sur votre serveur ;
- connaître les principes de fonctionnement de l'optimiseur d'Adaptive Server.

Types d'index

Adaptive Server propose deux types d'index :

- Les index clusterisés, où les données de la table sont physiquement stockées dans l'ordre des clés de l'index :
  - dans les tables verrouillées au niveau de toutes les pages (tables APL), les lignes sont stockées dans l'ordre des clés des pages et les pages sont liées dans l'ordre des clés ;
  - pour les tables verrouillées au niveau des pages de données seulement (tables DOL), les index permettent d'orienter le stockage des données dans des lignes et des pages mais l'ordre strict des clés n'est pas maintenu.
- Les index non clusterisés, où l'ordre de stockage des données de la table n'est pas lié aux clés d'index.

Vous ne pouvez créer qu'un seul index clusterisé par table car il existe un seul ordre physique possible des lignes de données. Vous pouvez créer au maximum 249 index non clusterisés par table.

Une table qui ne contient pas d'index clusterisé est appelée table sans index. Les lignes d'une table sans index ne se présentent pas dans un ordre particulier et toutes les nouvelles lignes sont ajoutées en fin de table. Le chapitre 7, "Stockage de données", traite des tables sans index et des opérations SQL sur les tables sans index.

Pages d'index

Les entrées d'index sont stockées sous forme de lignes sur des pages d'index, dans un format similaire à celui des lignes de données sur les pages de données. Les entrées d'index stockent les valeurs de clé ainsi que des pointeurs sur les niveaux inférieurs de l'index, sur les pages de données ou sur des lignes de données.

Adaptive Server utilise l'indexation en arbre à accès séquentiel afin que chaque nœud de la structure d'index puisse avoir plusieurs enfants.

Une entrée d'index est habituellement bien plus petite qu'une ligne de données et une page d'index plus dense qu'une page de données. Si une ligne de données comporte 200 octets (y compris le surcoût de ligne), on obtient 10 lignes par page.

Un index sur un champ de 15 octets aurait environ 100 lignes par page d'index (les pointeurs nécessitent de 4 à 9 octets par ligne), selon le type et le niveau d'index).

Les index comportent plusieurs niveaux :

- Niveau racine
- Niveau feuille
- Niveaux intermédiaires

Niveau racine

Le niveau racine est le niveau supérieur de l'index. Il n'existe qu'une seule page racine. Pour que l'index d'une table APL très petite tienne sur une seule page, aucun niveau intermédiaire ou feuille n'est créé et la page racine stocke des pointeurs vers les pages de données.

Les tables DOL ont toujours un niveau feuille entre la page racine et les pages de données.

Pour les tables plus importantes, la page racine stocke des pointeurs vers les pages d'index de niveau intermédiaire ou de niveau feuille.

Niveau feuille

Le niveau inférieur de l'index est le niveau feuille. A ce niveau, l'index contient une valeur clé pour chaque ligne de la table et les lignes sont stockées dans l'ordre de tri par clé d'index.

- Pour les index clusterisés des tables APL, le niveau feuille correspond aux données. C'est le seul niveau comportant une ligne d'index par ligne de données.
- Dans les index non clusterisés et les index clusterisés des tables DOL, le niveau feuille contient la valeur de clé d'index pour chaque ligne, un pointeur vers la page où se trouve la ligne et un pointeur vers les lignes de la page de données.

  Le niveau feuille est situé immédiatement au-dessus des données ; il contient une ligne d'index par ligne de données. Les lignes d'index sont stockées dans les pages dans l'ordre des valeurs de clé.

Niveaux intermédiaires

Tous les niveaux entre la racine et la feuille sont des niveaux intermédiaires. Un index sur une table importante ou un index utilisant des clés longues peut avoir de nombreux niveaux intermédiaires. A l'inverse, une très petite table APL peut ne pas en avoir, auquel cas les pages racine pointent directement vers le niveau feuille.

Taille des index

Le tableau 9-1 indique les nouvelles limites concernant la taille des index des tables APL et DOL :

Tableau 9-1 : Longueur de ligne d'index

Taille de la page	Longueur de ligne d'index visible par l'utilisateur	Longueur de ligne d'index interne
2 ko (2048 octets)	600	650
4 k (4096 octets)	1250	1310
8 ko (8192 octets)	2600	2670
16 ko (16384 octets)	5300	5390

Etant donné que vous pouvez créer des tables avec des colonnes plus larges par rapport à la limite de la clé d'index, ces colonnes ne sont plus indexables. Par exemple, si vous effectuez les opérations suivantes sur une page du serveur de 2 k et que vous essayez de créer un index sur c3, la commande échoue et Adaptive Server affiche un message d'erreur car la colonne c3 est plus large par rapport à la limite imposée pour la taille des lignes (600 octets).

```
create table t1 (
c1 int
c2 int
c3 char(700))
```

"Non-indexable" ne vous empêche pas d'utiliser ces colonnes dans les clauses de recherche. Bien qu'une colonne soit non indexable (voir c3 ci-dessus), vous pouvez néanmoins créer des statistiques la concernant. De même, si vous incluez la colonne dans une clause where, celle-ci sera évaluée pendant l'optimisation.

Index clusterisés dans des tables APL

Dans les index clusterisés des tables APL, les pages de niveau feuille sont également les pages de données et toutes les lignes sont stockées dans l'ordre physique des clés.

L'ordre physique signifie que :

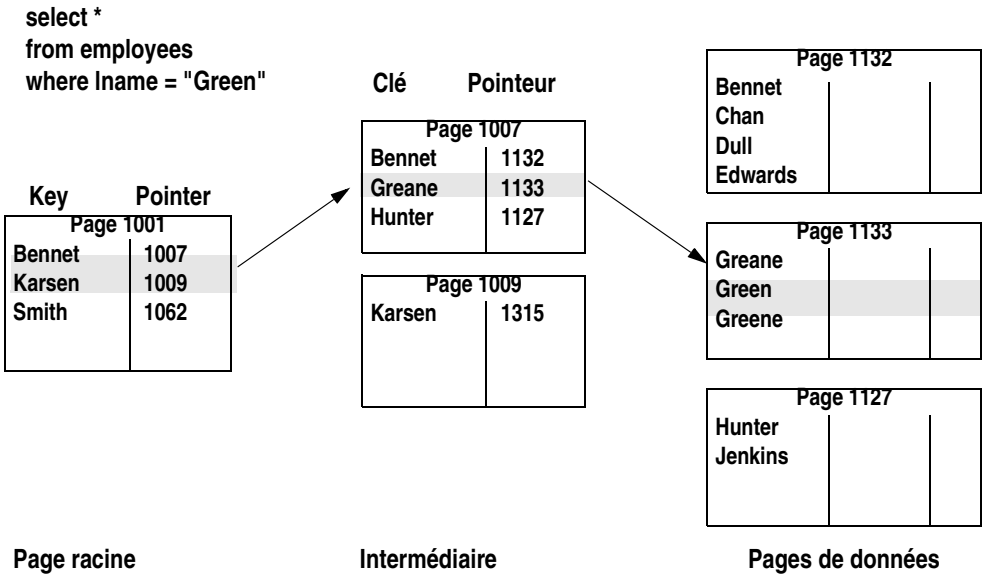
- Toutes les entrées d'une page de données sont dans l'ordre de la clé d'index.
- En suivant les pointeurs renvoyant à la page suivante des données, Adaptive Server lit la totalité de la table dans l'ordre des clés d'index.

Sur les pages de niveau racine et intermédiaire, chaque entrée pointe vers une page de niveau suivant.

Index clusterisés et sélections

Pour sélectionner un nom (last name) particulier à l'aide d'un index clustérisé, Adaptive Server utilise d'abord sysindexes pour rechercher la page racine. Il examine les valeurs de la page racine, puis suit les pointeurs page, en exécutant une recherche binaire sur chaque page accédée lors du parcours de l'index. Voir la figure 9-1 ci-dessous.

Figure 9-1 : Sélection d'une ligne utilisant un index clusterisé sur une table APL



Sur la page du niveau racine, "Green" se trouve après "Bennet" mais avant "Karsen", donc le pointeur de "Bennet" vers la page 1007 est utilisé. A la page 1007, "Green" se trouve après "Greane,"mais avant "Hunter," donc le pointeur vers la page 1133 est utilisé vers la page de données contenant la ligne qui est renvoyée à l'utilisateur.

Cette recherche via l'index clustérisé requiert :

- une lecture pour le niveau racine de l'index,
- une lecture pour le niveau intermédiaire,
- une lecture pour la page de données.

Ces lectures sont effectuées soit sur le cache (il s'agit de **lecture logique**), soit sur le disque (il s'agit de **lecture physique**). Sur les tables fréquemment utilisées, les niveaux supérieurs des index sont souvent repérés dans le cache, les niveaux inférieurs et les pages de données étant lus sur le disque.

Index clusterisés et insertions

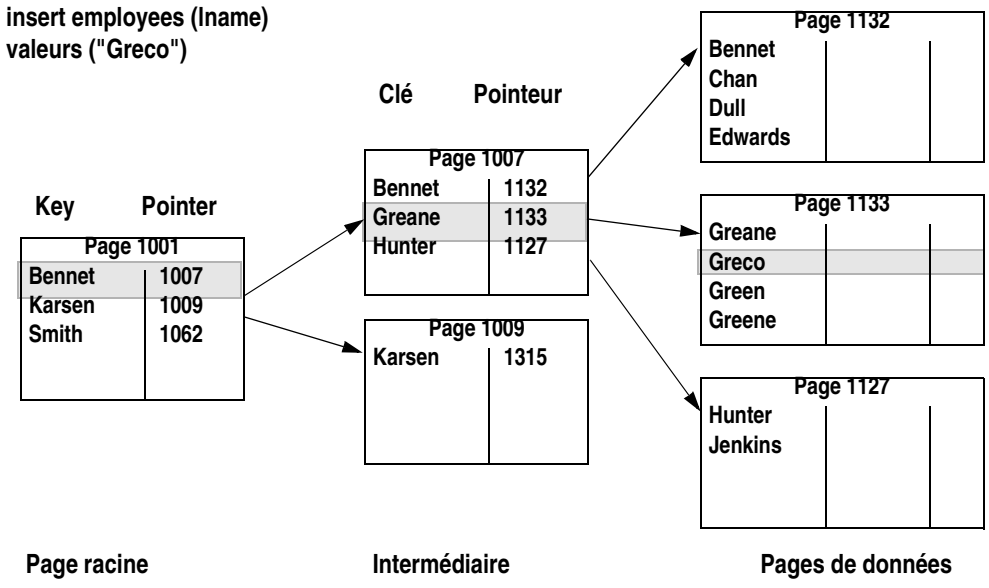
Quand vous insérez une ligne dans une table verrouillée APL comportant un index clusterisé, celle-là doit être positionnée physiquement en fonction de la valeur de clé de la table.

D'autres lignes de la page de données se déplacent vers le bas, comme requis, pour faire de la place à la nouvelle valeur. Tant que la place pour la nouvelle ligne existe sur la page, l'insertion n'a aucun effet sur les autres pages de la base de données.

L'index clustérisé est utilisé pour rechercher l'emplacement de la nouvelle ligne.

La figure 9-2 est un exemple simple d'une page de données existante disposant de l'espace nécessaire pour la nouvelle ligne. Dans cet exemple, les valeurs de clé de l'index ne doivent pas forcément être modifiées.

Figure 9-2 : Insertion d'une ligne dans une table APL comportant un index clustérisé



Fractionnement des pages pleines

Si la page de données ne dispose pas de l'espace nécessaire pour la nouvelle ligne, un fractionnement de page est nécessaire.

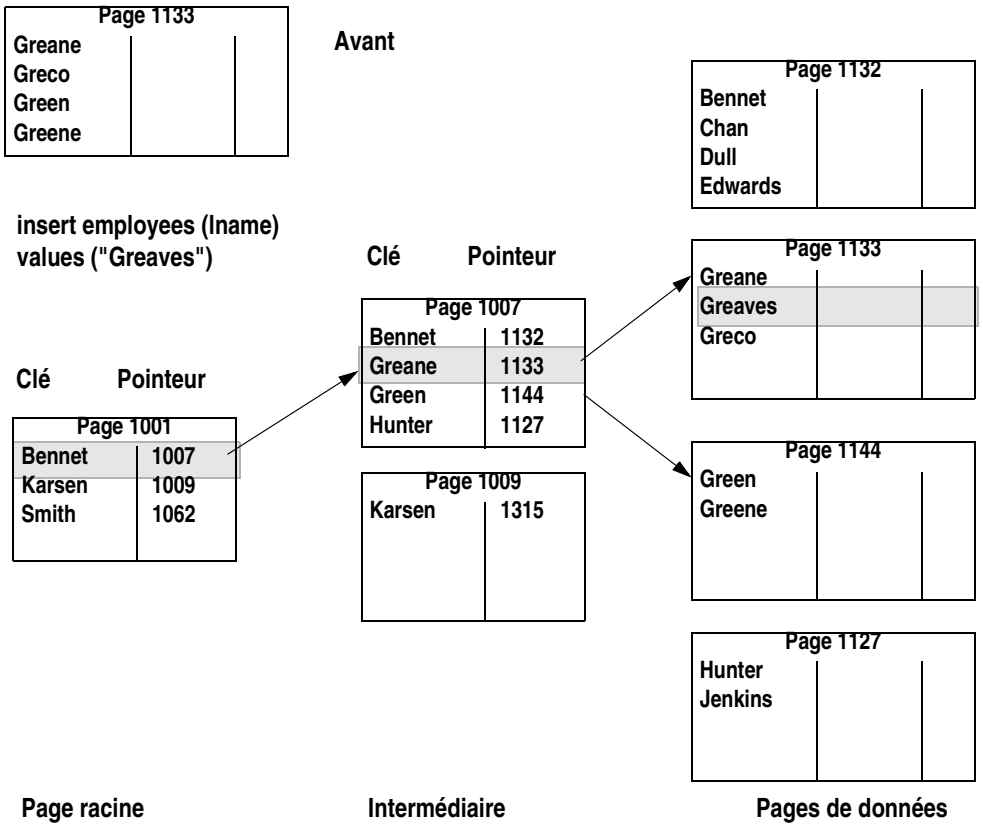
- Une nouvelle page de données est allouée sur un extent déjà utilisé par la table. En l'absence de pages disponibles, un nouvel extent est alloué.
- Les pointeurs page suivante et page précédente des pages adjacentes sont modifiés afin d'insérer la nouvelle page dans la chaîne de pages. Cela requiert la lecture de ces pages dans la mémoire et leur verrouillage.
- Plus ou moins la moitié des lignes sont déplacées dans la nouvelle page, la nouvelle ligne étant insérée dans l'ordre.
- Les niveaux supérieurs de l'index clustérisé sont modifiés afin de pointer vers la nouvelle page.
- Si la table contient également des index non clustérisés, tous les pointeurs vers les lignes de données concernées doivent être modifiées afin de pointer ver la nouvelle page et la nouvelle ligne.

Dans certains cas, le fractionnement de page est géré de façon légèrement différente.

Pour plus d'informations, reportez-vous à la section "Exceptions au fractionnement de page", page 217.

Dans la figure 9-3, le fractionnement de page nécessite l'ajout d'une nouvelle ligne à une page d'index existante, page 1007.

Figure 9-3 : Fractionnement de page dans une table APL comportant un index clustérisé



Exceptions au fractionnement de page

Il existe des exceptions aux fractionnements 50-50 de page :

- Si vous insérez une ligne volumineuse ne tenant pas dans la page précédant ou suivant la page qui nécessite le fractionnement, deux nouvelles pages sont allouées, une pour la ligne volumineuse et une pour les lignes suivantes.
- Si possible, Adaptive Server garde les valeurs en double ensemble lors du fractionnement de page.

- Si Adaptive Server détecte que toutes les insertions prennent place en fin de page, à cause d'une valeur de clé croissante, la page n'est pas fractionnée au moment d'insérer une nouvelle ligne ne tenant pas au bas de la page. Dans ce cas, une nouvelle page est allouée et la ligne prend place dans la nouvelle page.
- Si Adaptive Server détecte que les insertions prennent place dans l'ordre à d'autres endroits de la page, la page est fractionnée au point d'insertion.

Fractionnement des pages d'index

Si une nouvelle ligne doit être ajoutée à une page d'index pleine, le processus de fractionnement de la page d'index est similaire à celui de la page de données.

Une nouvelle page est allouée et la moitié des lignes d'index est déplacée dans la nouvelle page.

Une nouvelle ligne est insérée au niveau supérieur suivant de l'index afin de pointer vers la nouvelle page d'index.

Incidence des performances sur le fractionnement de page

Les fractionnements de page sont des opérations coûteuses. Outre le travail courant consistant à déplacer les lignes, à allouer les pages et à journaliser les opérations, le coût est accru par :

- la mise à jour de l'index clustérisé lui-même,
- la mise à jour des pointeurs page des pages adjacentes afin de conserver les liens de page,
- la mise à jour de toutes les entrées des index non clustérisés pointant vers les lignes concernées par le fractionnement.

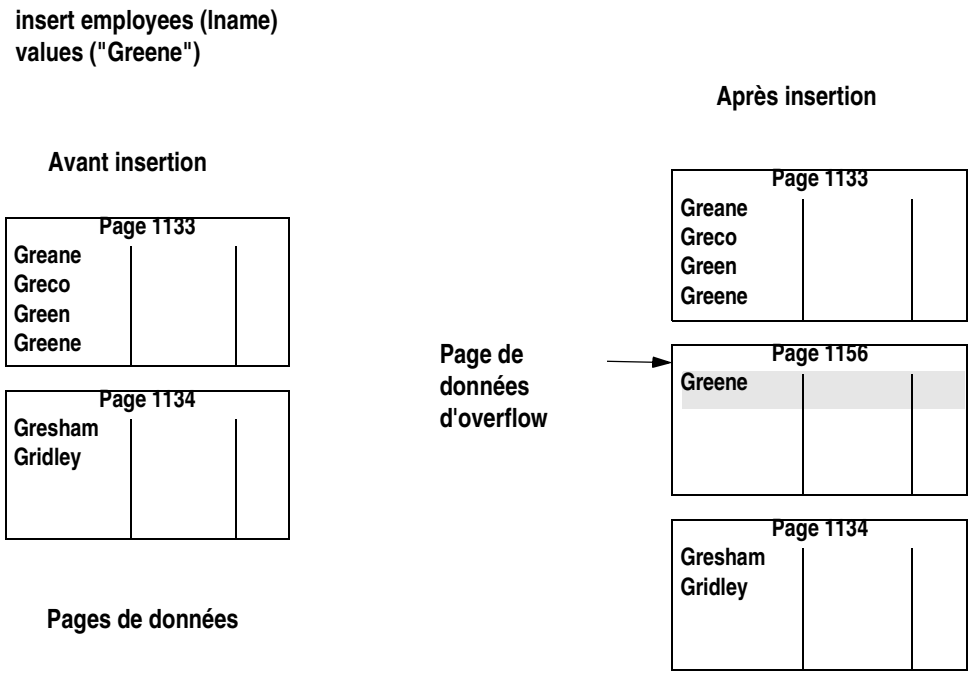
Quand vous créez un index clustérisé pour une table qui deviendra volumineuse avec le temps, vous pouvez utiliser fillfactor pour faire de la place sur les pages de données et d'index. Cela limite le nombre de fractionnements de page pour un certain temps.

Pour plus d'informations, reportez-vous à la section "Choix des propriétés de gestion de l'espace des index", page 206.

Pages d'overflow

Des pages d'overflow spéciales sont crées pour les index clustérisés non uniques des tables APL lorsqu'une ligne nouvellement insérée possède la même clé que la dernière ligne d'une page de données pleine. Une nouvelle page de données est allouée et liée à une chaîne de pages et la ligne nouvellement insérée prend place dans la nouvelle page (voir la figure 9-4).

Figure 9-4 : Ajout d'une page d'overflow à un index clusterisé, table APL



Les seules lignes qui prendront place sur cette page d'overflow sont des lignes supplémentaires possédant la même valeur de clé. Dans un index clustérisé non unique possédant plusieurs valeurs de clé en double, il peut y avoir plusieurs pages d'overflow pour la même valeur.

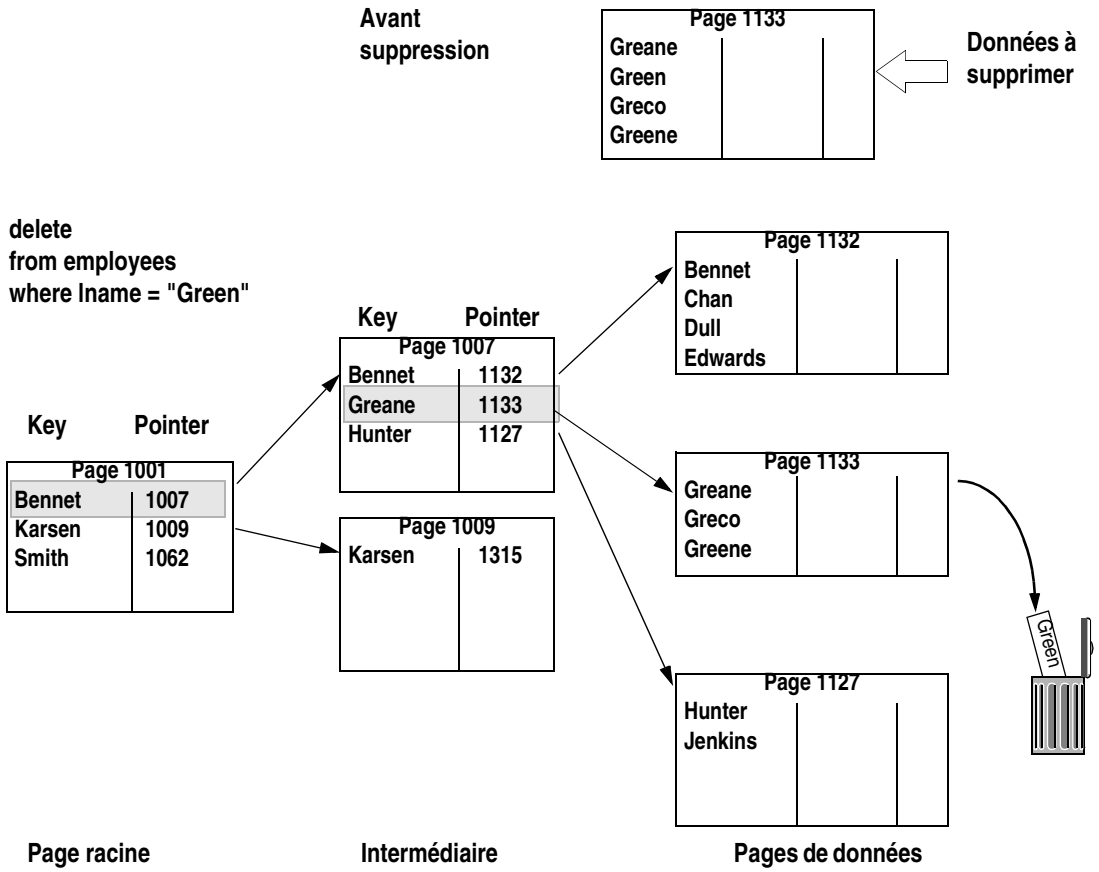
L'index clustérisé ne contient pas de pointeurs directement vers les pages d'overflow. Dans ce cas, les pointeurs de la page suivante sont utilisés pour suivre la chaîne des pages d'overflow jusqu'à ce qu'une valeur ne correspondant pas à la valeur de recherche ne soit rencontrée.

Index clusterisés et suppressions

Lorsqu'une ligne est supprimée d'une table APL comportant un index clustérisé, les autres lignes de la page remontent vers le haut pour occuper l'espace vide afin que les données restent groupées sur la page.

La figure 9-5 montre une page de 4 lignes avant la suppression de la seconde. Les deux lignes suivant la ligne supprimée remontent vers le haut.

Figure 9-5 : Suppression d'une ligne dans une table comportant un index clustérisé



Suppression de la dernière ligne d'une page

Si la dernière ligne est supprimée de la page de données, celle-ci est libérée et les pointeurs page suivante et précédente de la page adjacente sont modifiés.

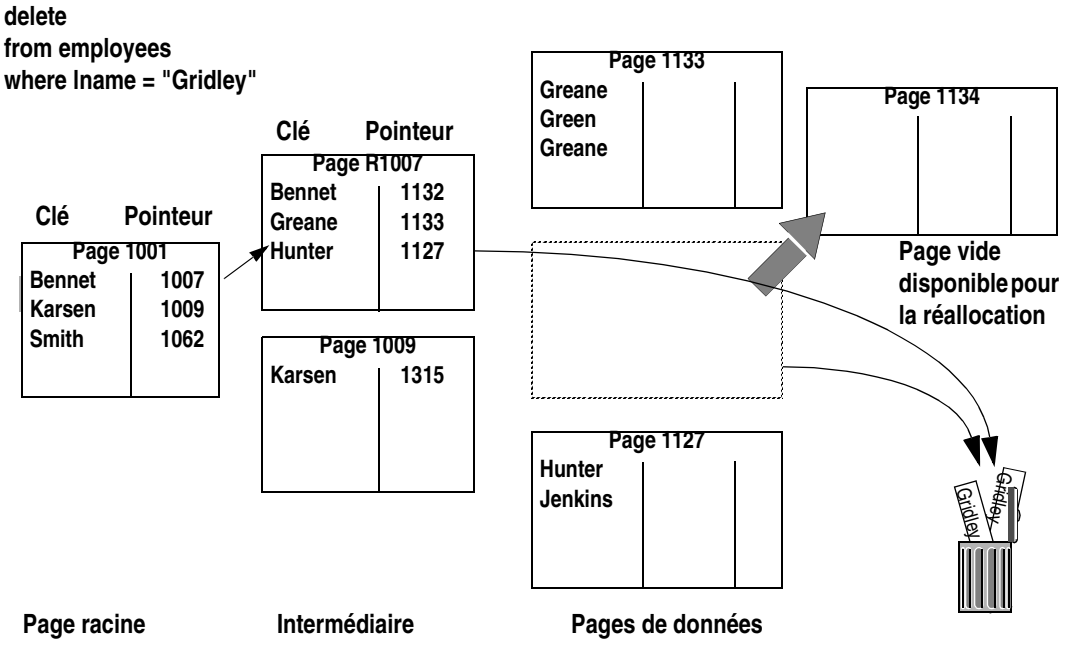
Les lignes pointant vers cette page aux niveaux feuille et intermédiaire de l'index sont supprimées.

Si la page de données libérée se trouve sur le même extent que les autres pages de la table, elle peut être réutilisée lorsque cette table nécessite une page supplémentaire.

Si la page de données libérée correspond à la dernière page de l'extent de la table, l'extent est également libéré et peut servir pour le développement d'autres objets de la base de données.

Dans la figure 9-6, qui montre la table après la suppression, le pointeur vers la page supprimée a été supprimé de la page d'index 1007 et les lignes d'index suivantes de la page ont été déplacées vers le haut pour maintenir la contiguïté de l'espace utilisé.

Figure 9-6 : Suppression de la dernière ligne d'une page (après la suppression)



Fusions de pages d'index

Si vous supprimez un pointeur d'une page d'index, en ne laissant qu'une ligne sur cette page, la ligne est déplacée sur une page adjacente et la page vide est désaffectée. Les pointeurs sur la page parent sont mis à jour pour refléter les modifications.

Index non clusterisés

L'arbre à accès séquentiel pour les index non clustérisés fonctionne à peu près de la même façon que pour les index clustérisés mais avec quelques différences. Dans les index non clusterisés :

- Les pages feuille diffèrent des pages de données.
- Le niveau feuille stocke une paire de pointeurs clé pour *chaque ligne* de la table.
- Les pages du niveau feuille stockent les clés d'index et les pointeurs de page outre un pointeur dirigé vers la table d'offset de ligne de la page de données. Cette combinaison de pointeur de page et de nombre d'offset de ligne s'appelle **ID de ligne**.
- Les niveaux racine et intermédiaire stockent les clés d'index et les pointeurs de page dirigés vers d'autres pages d'index. Ils stockent également l'ID de ligne de la ligne de données de la clé.

Avec des clés de même taille, les index non clustérisés requièrent plus d'espace que les index clustérisés.

Pages feuille revisitées

La page feuille d'un index est le niveau inférieur de l'index où toutes les clés de l'index apparaissent dans l'ordre de tri.

Dans les index clusterisés de tables APL, les lignes de données sont stockées par clé d'index, ainsi, par définition, le niveau données est le niveau feuille. C'est le seul niveau de l'index clustérisé comportant une ligne d'index par ligne de données. Les index clusterisés des tables APL sont des index non denses.

Le niveau au-dessus des données comporte un pointeur pour chaque *page* de données et non pour chaque *ligne* de données.

Les index non clusterisés et les index clusterisés des tables de données DOL ont un niveau feuille juste au-dessus du niveau de données : ce niveau contient une paire de pointeurs clé pour chaque ligne de données. Ces index sont denses. Au niveau au-dessus des données, ils comportent une ligne d'index par ligne de données.

Structure des index non clusterisés

La table de la figure 9-7 montre un index non clustérisé sur lname. Les lignes de données tout à droite affichent les pages dans l'ordre croissant et classées par employee_id (10, 11, 12 et ainsi de suite) à cause de la présence d'un index clustérisé sur cette colonne.

Les pages racine et intermédiaire stockent :

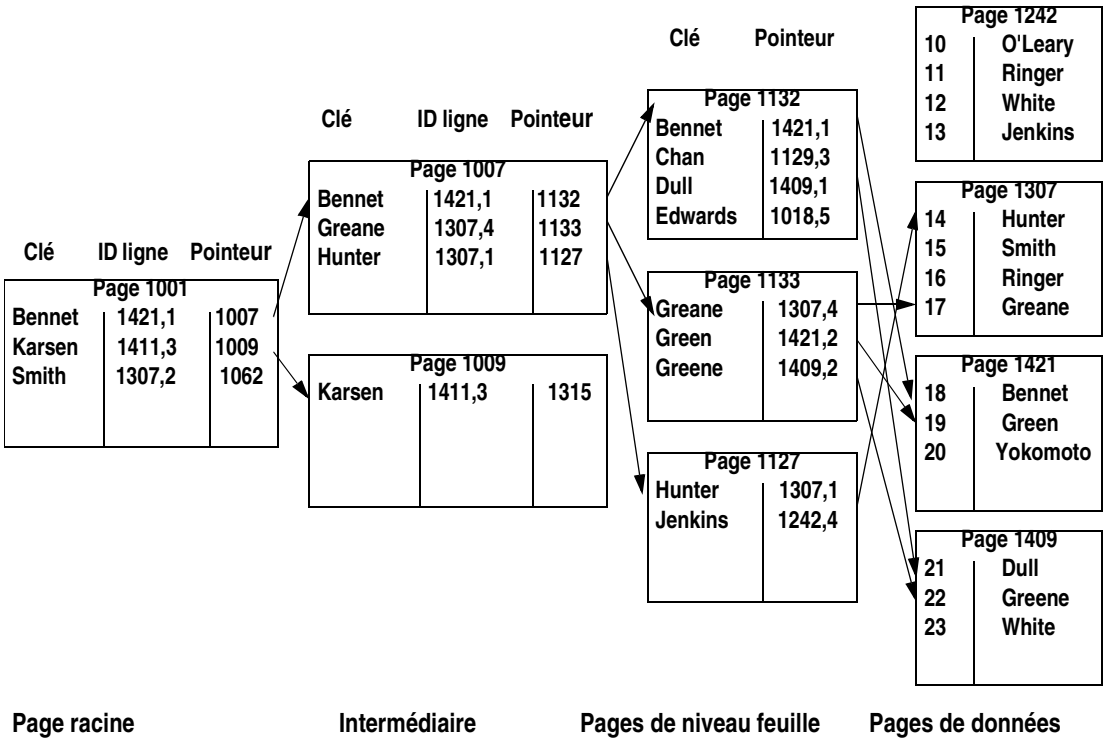
- la valeur clé
- l'ID de ligne
- le pointeur au niveau suivant de l'index

Le niveau feuille stocke :

- la valeur clé
- l'ID de ligne

L'ID de ligne des niveaux supérieurs de l'index est utilisé pour les index qui permettent les clés en double. Si une modification de données affecte la clé d'index ou supprime une ligne, l'ID de ligne identifie clairement toutes les occurrences de la clé à tous les niveaux d'index.

Figure 9-7 : Structure des index non clusterisés



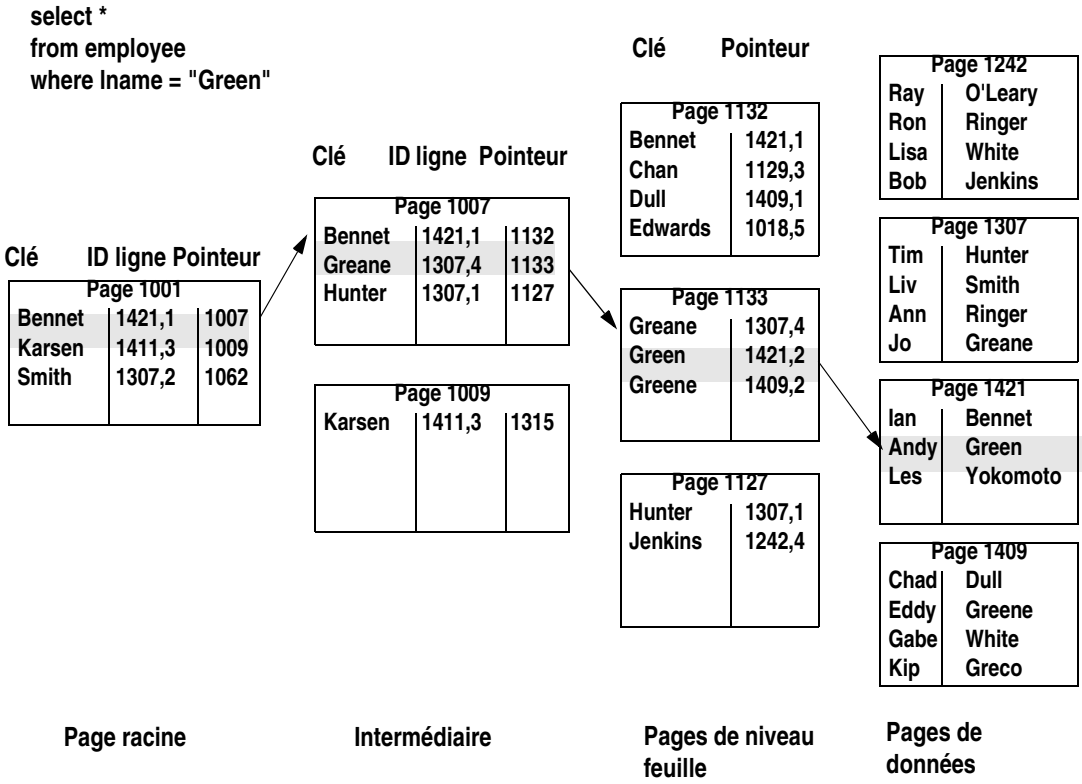
Index non clusterisés et sélections

Lorsque vous sélectionnez une ligne à l'aide d'un index non clusterisé, la recherche commence au niveau racine. sysindexes.root stocke le numéro de la page racine de l'index non clusterisé.

Dans la figure 9-8, "Green" se trouve après "Bennet" mais avant "Karsen", donc le pointeur dirigé vers la page 1007 est utilisé.

"Green" se trouve après "Greane" mais avant "Hunter", donc le pointeur vers la page 1133 est suivi. La page 1133 est la page feuille et elle montre que la ligne de "Green" se trouve en deuxième position sur la page 1421. Cette dernière est extraite, l'octet "2" de la table d'offset est contrôlé et la ligne est renvoyée à partir de la page de données.

Figure 9-8 : Sélection de lignes à l'aide d'un index non clusterisé



Performances des index non clusterisés

La requête de la figure 9-8 nécessite l'E/S suivante :

- une lecture pour la page de niveau racine,
- une lecture pour la page de niveau intermédiaire,
- une lecture pour la page de niveau feuille,
- une lecture pour la page de données.

Si vos applications utilisent fréquemment un index non clusterisé particulier, les pages de niveau racine et de niveau intermédiaire se trouvent probablement en mémoire cache et il suffit d'une ou deux E/S disque.

Index non clusterisés et insertions

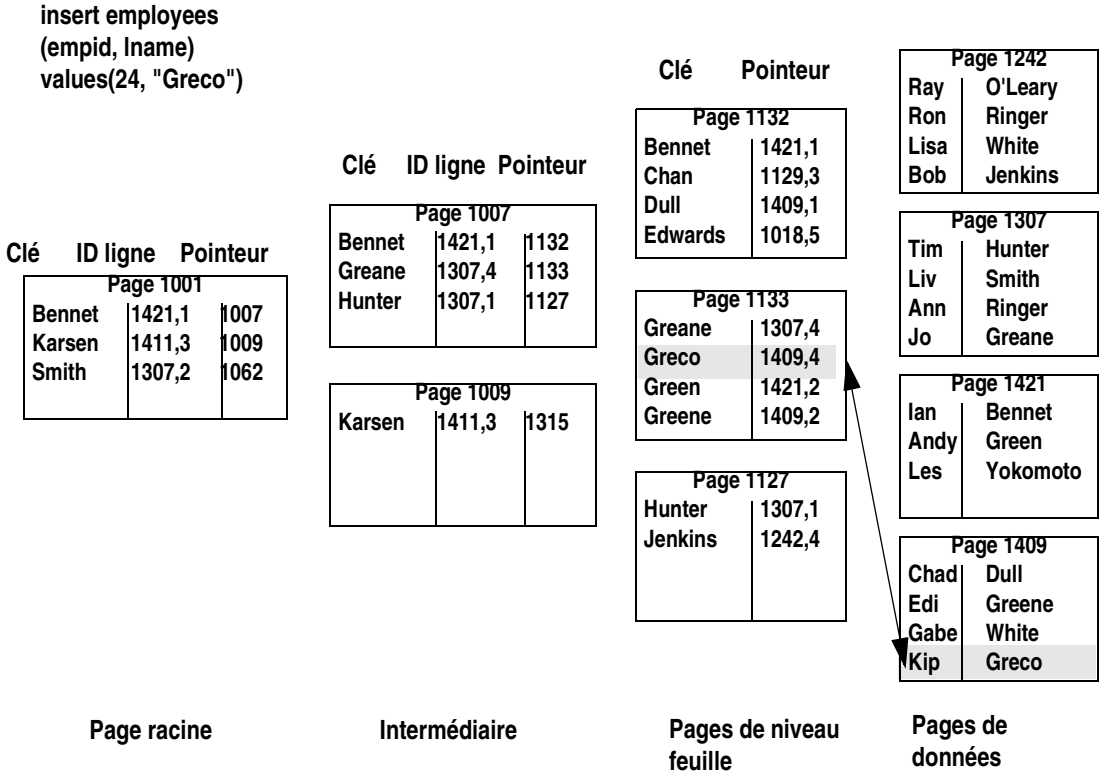
Lorsque vous insérez des lignes dans une table qui possède un index non clusterisé, l'insertion est effectuée à la dernière page de la table.

Si la table est partitionnée, l'insertion est effectuée à la dernière page, dans l'une des partitions. Ensuite, l'index non clusterisé est mis à jour pour inclure la nouvelle ligne.

Si la table possède un index clusterisé, ce dernier est utilisé pour trouver l'emplacement de la ligne. L'index clusterisé est mis à jour, si nécessaire, ainsi que chaque index non clusterisé, pour inclure la nouvelle ligne.

La figure 9-9 présente une insertion dans une table comportant un index non clusterisé. La ligne prend place en fin de table. Une ligne, contenant la nouvelle valeur de la clé et l'ID de ligne, est également insérée au niveau feuille de l'index non clusterisé.

Figure 9-9 : Insertion dans une table comportant un index non clusterisé

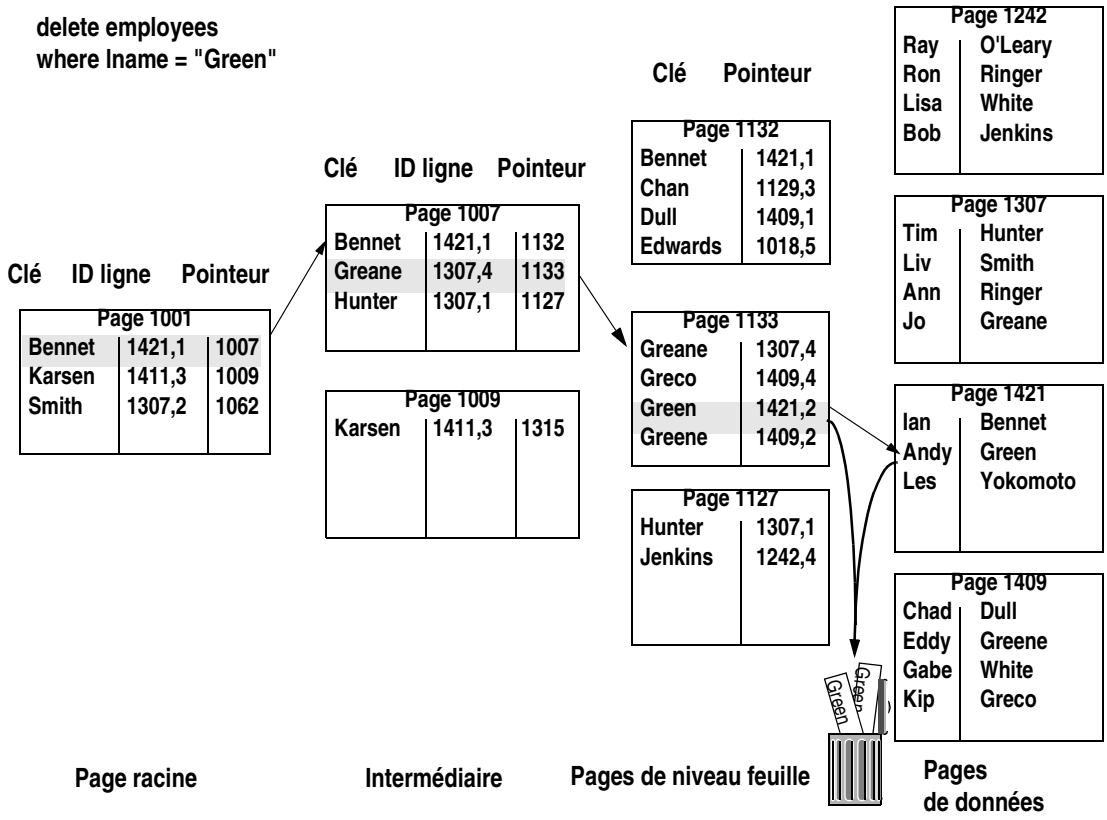


Index non clusterisés et suppressions

Lorsque vous supprimez une ligne d'une table, la requête peut utiliser un index non clusterisé sur les colonnes de la clause where pour localiser la ligne de données à supprimer (voir la figure 9-10).

La ligne de niveau feuille de l'index non clusterisé qui pointe sur la ligne de données est également supprimée. S'il existe d'autres index non clusterisés sur la table, les lignes de niveau feuille de ces index sont également supprimées.

Figure 9-10 : Suppression d'une ligne dans une table comportant un index non clustérisé



Si la dernière ligne est supprimée de la page de données, celle-ci est libérée et les pointeurs de pages adjacents sont mis à jour dans les tables APL. Les références à la page sont également supprimées dans les niveaux supérieurs de l'index.

Si la suppression laisse une seule ligne sur une page d'index intermédiaire, les pages d'index peuvent être fusionnées, comme avec les index clusterisés.

Pour plus d'informations, reportez-vous à la section "Fusions de pages d'index", page 222.

Il n'existe pas de fusion automatique sur les pages de données ; par conséquent, si vos applications effectuent de nombreuses suppressions aléatoires, vous risquez de vous retrouver avec des pages comportant très peu de lignes, voire une seule.

Index clusterisés sur les tables DOL

Dans les tables DOL, les index clusterisés sont structurés comme les index non clusterisés. Ils comprennent un niveau feuille au-dessus des pages de données. Le niveau feuille contient les valeurs de clé et l'ID de ligne de chaque ligne de la table.

Contrairement aux index clusterisés des tables APL, les lignes de données dans les tables DOL ne sont pas obligatoirement maintenues dans l'ordre exact des clés. Les index orientent en fait les lignes vers des pages qui possèdent des clés adjacentes ou proches.

Lorsqu'une ligne doit être insérée dans une table DOL avec un index clusterisé, la clé d'index qui précède immédiatement la valeur à insérer est utilisée. Les pointeurs d'index servent à retrouver cette page et la ligne est insérée dans la page, à condition qu'il y ait suffisamment de place. Si l'espace manque, la ligne est insérée dans une page de la même unité d'allocation ou dans une autre unité d'allocation déjà utilisée par la table.

Pour que les données restent relativement groupées lors des insertions et des mises à jour d'une table DOL, vous pouvez définir des propriétés de gestion de l'espace afin de conserver un espace suffisant sur les pages (à l'aide des commandes fillfactor et exp_row_size) ou dans les unités d'allocation (à l'aide de la commande reservepagegap).

Voir le chapitre 13, "Définition des propriétés de gestion de l'espace".

Couverture par index

La **couverture par index** peut considérablement améliorer les performances lorsque toutes les colonnes nécessaires à une requête sont incluses dans l'index.

Vous pouvez créer des index sur plusieurs clés. On parle alors d'*index composés*. Les index composés peuvent comporter jusqu'à 31 colonnes qui s'ajoutent pour totaliser 600 octets au maximum.

Si vous créez un index non clusterisé composé sur chaque colonne référencée dans la liste de sélection et dans les clauses where, having, group by et order by, l'accès à ce seul index permettra de répondre à la requête.

Puisque le niveau feuille d'un index non clusterisé ou d'un index clusterisé de table DOL contient les valeurs de clé pour chaque ligne de la table, les requêtes qui n'accèdent qu'à ces valeurs de clé peuvent extraire les informations en utilisant le niveau feuille de l'index non clusterisé comme s'il s'agissait des données réelles de la table. C'est ce qu'on appelle une couverture par index.

Deux types de lecture d'index sont possibles dans une couverture par index :

- la lecture d'index avec correspondance,
- la lecture d'index sans correspondance.

Pour chacun de ces types, les clés d'index doivent contenir toutes les colonnes nommées dans la requête. Les lectures avec correspondance présentent des contraintes supplémentaires.

La section "Choix des index composés", page 199 décrit les types de requête qui tirent profit de la couverture par index.

Couverture par index avec correspondance

Ce type de couverture par index permet d'éviter la dernière lecture (qui extrait la page de données) pour chaque ligne renvoyée par la requête.

Pour les requêtes ponctuelles, qui renvoient seulement une ligne, le gain de performances est faible (une seule page).

Pour les requêtes séquentielles, le gain de performances est plus élevé, puisque la couverture par index économise une lecture pour chaque ligne renvoyée par la requête.

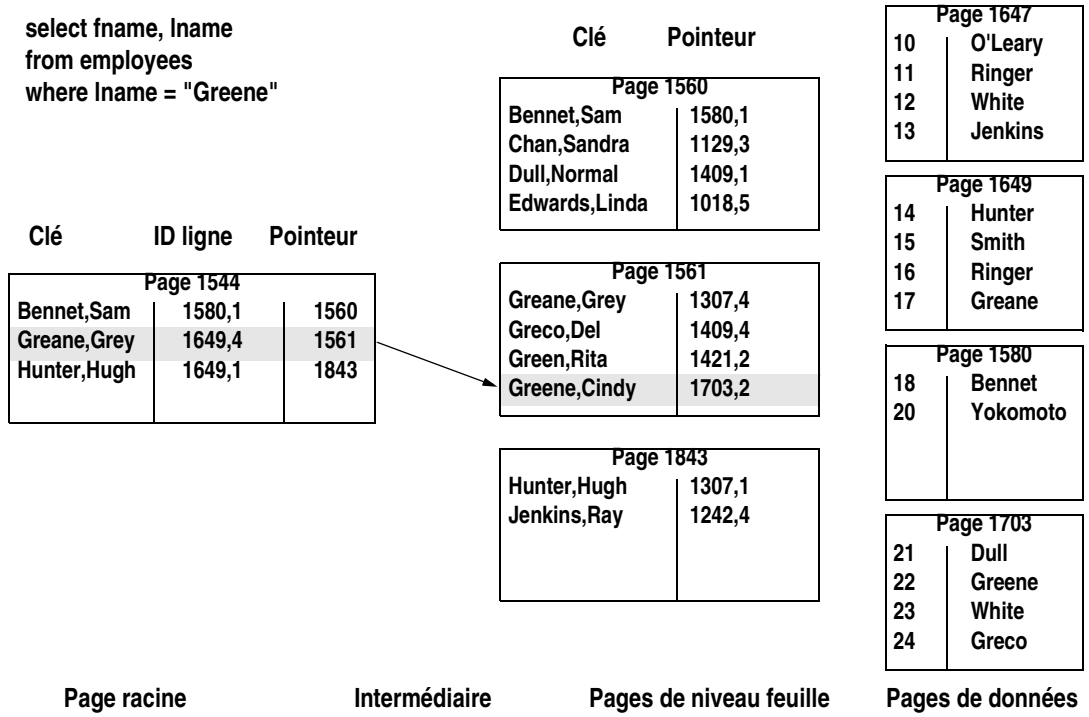
Pour qu'une couverture par index avec correspondance soit possible, l'index doit contenir toutes les colonnes nommées dans la requête. De plus, les colonnes figurant dans les clauses where de la requête doivent inclure la colonne principale de l'index.

Par exemple, pour un index sur les colonnes A, B, C et D, les ensembles suivants peuvent effectuer les lectures avec correspondance : A, AB, ABC, AC, ACD, ABD, AD et ABCD. Les colonnes B, BC, BCD, BD, C, CD ou D, qui ne comprennent pas la colonne principale, ne sont pas utilisables dans des lectures avec correspondance.

Lorsque vous effectuez une lecture d'index avec correspondance, Adaptive Server utilise les méthodes d'accès standard pour se déplacer depuis la racine de l'index jusqu'à la page de niveau feuille non clusterisée contenant la première ligne.

Dans la figure 9-11, l'index non clusterisé sur lname, fname couvre la recherche. La clause where inclut la colonne principale et toutes les colonnes de la liste de sélection sont incluses dans l'index, de sorte qu'il n'est pas nécessaire d'accéder à la page de données.

Figure 9-11 : L'accès à l'index avec correspondance évite la lecture de la ligne



Couverture par index sans correspondance

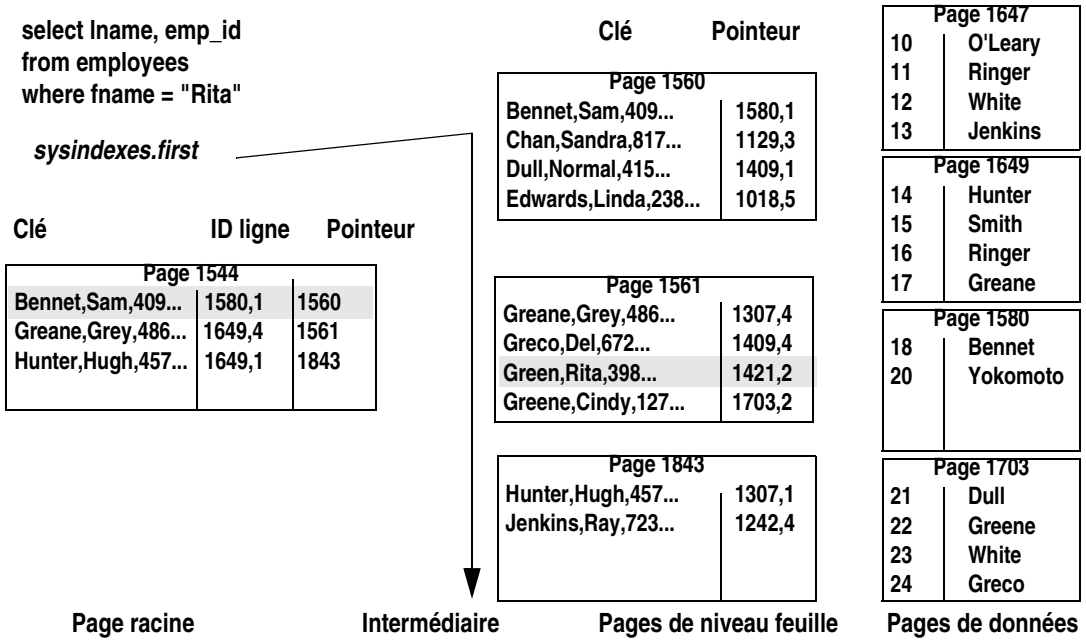
Lorsque les colonnes spécifiées dans la clause where n'incluent pas le nom de la colonne principale de l'index mais que toutes les colonnes spécifiées dans la liste de sélection et les autres clauses de la requête (telles que group by ou having) sont incluses dans l'index, Adaptive Server économise des E/S en balayant le niveau feuille de l'index non clusterisé, au lieu de balayer la table.

Toutefois, il ne peut pas effectuer de lecture avec correspondance car la colonne principale de l'index n'est pas spécifiée.

La requête de la figure 9-12 montre une lecture d'index sans correspondance. Cette requête n'utilise pas les colonnes principales de l'index mais toutes les colonnes nécessaires à la requête se trouvent dans l'index non clusterisé sur lname, fname, emp_id.

La lecture sans correspondance doit examiner toutes les lignes du niveau feuille. Il balaye toutes les pages d'index de niveau feuille, en commençant par la première page. Il n'a aucun moyen de savoir combien de lignes correspondent aux critères de la requête et doit donc consulter chaque ligne de l'index. Comme il doit commencer à la première page du niveau feuille, il peut utiliser le pointeur de sysindexes.first au lieu de parcourir l'index.

Figure 9-12 : Une lecture d'index sans correspondance



Index et mise en mémoire cache

La section "Opérations d'E/S d'Adaptive Server sur les tables sans index", page 175 présente les concepts fondamentaux relatifs au cache de données d'Adaptive Server et explique comment les caches sont utilisés lors de la lecture des tables sans index.

Les pages d'index mises en mémoire cache sont gérées de façon spécifique :

- Les pages d'index racine et intermédiaires utilisent toujours la stratégie LRU.
- Les pages d'index ne peuvent utiliser qu'un seul cache alors que les pages de données en utilisent un autre, si l'index est lié à un cache différent.
- Les couvertures avec lecture d'index peuvent utiliser la stratégie delecture-élimination.
- Les pages d'index peuvent passer dans le cache plusieurs fois, si le paramètre number of index trips est configuré.

Quand une requête utilisant un index est exécutée, les pages racine, intermédiaires, feuille et ligne de données sont lues dans cet ordre. Si ces pages ne sont pas en mémoire cache, elles sont lues dans l'extrémité MRU du cache, puis déplacées vers l'extrémité LRU au fur et à mesure que d'autres pages sont lues.

Chaque fois qu'une page est trouvée en mémoire cache, elle est déplacée à l'extrémité MRU de la chaîne de pages, ainsi la page racine et les niveaux supérieurs de l'index restent en mémoire cache.

Utilisation de caches distincts pour les pages de données et d'index

Les index et les tables qu'ils indexent utilisent des caches distincts. L'administrateur système ou le propriétaire de la table peut lier un index clusterisé ou non clusterisé à un cache et sa table à un autre.

Nombre de cycles effectués par l'index dans le cache

Les pages d'index sont conservées en mémoire cache selon une méthode spécifique. Les pages de données, quant à elles, n'effectuent qu'un seul cycle : elles sont lues à l'extrémité MRU du cache ou placées juste avant le marqueur de vidage, selon la stratégie de cache choisie pour la requête.

Une fois que les pages atteignent l'extrémité LRU du cache, le buffer de la page correspondante est réutilisé lorsqu'une autre page doit être lue dans le cache.

Pour les pages d'index, un compteur contrôle le nombre de cycles MRU/LRU qu'elles peuvent effectuer.

Si le compteur a une valeur supérieure à 0, il est diminué de 1 lorsque la page d'index atteint l'extrémité LRU de la chaîne de pages, puis la page se repositionne à l'extrémité MRU.

Par défaut, le nombre de cycles qu'une page d'index effectue dans le cache est défini sur 0. Pour modifier cette valeur, l'administrateur système (SA) peut définir le paramètre de configuration number of index trips.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

CHAPITRE 10

Configuration et optimisation des verrouillages

Ce chapitre traite des types de verrous utilisés dans Adaptive Server et des commandes ayant une incidence sur le verrouillage ; vous pouvez trouver une introduction aux Concepts de verrouillage dans le document Guide d'administration système Adaptive Server.

Sujet	Page
Verrouillage et performances	235
Configuration des verrous et des seuils de conversion des verrous	241
Choix du plan de verrouillage d'une table	251

Verrouillage et performances

Le verrouillage réduit les performances d'Adaptive Server car il limite la concurrence d'accès. Une augmentation du nombre d'utilisateurs accédant simultanément au serveur peut accroître les conflits de verrous, ce qui engendre une baisse des performances. Les verrous ont un impact sur les performances lorsque :

- Les processus attendent la libération de verrous.

  Chaque fois qu'un processus attend qu'un autre processus termine sa transaction et libère ses verrous, le temps de réponse global et le débit en pâtissent.

- Les transactions aboutissent souvent à des interblocages.

  Dès qu'un interblocage se produit, une transaction est annulée et doit être relancée par l'application. Lorsque les interblocages sont fréquents, le débit des applications est sérieusement ralenti.

  Pour réduire ces interblocages, choisissez de préférence un verrouillage au niveau des pages de données ou des lignes de données, ou redéfinissez le mode d'accès aux données par les transactions.

- La création d'index verrouille les tables.

  Pendant la création d'un index clusterisé, l'accès à la table est interdit à tous les utilisateurs.

  Pendant la création d'un index non clusterisé, toutes les mises à jour sont interdites.

  Dans les deux cas, il est préférable de créer des index au cours des périodes de faible activité du serveur.

- La désactivation du mécanisme de détection d'interblocage provoque un conflit de verrou d'attente.

  Si vous définissez le paramètre deadlock checking period à 0, le serveur effectue plus souvent des recherches d'interblocages. Le processus de détection d'interblocages maintient des verrous d'attente sur les structures de verrouillage en mémoire pendant qu'il effectue ses recherches.

  Dans un environnement de production hautement transactionnel, ne choisissez pas la valeur 0.

Utilisation de *sp_sysmon* et *sp_object_stats*

Les sections ci-dessous indiquent comment limiter les conflits de verrous en modifiant la valeurs de certains paramètres de configuration.

Les procédures sp_object_stats et sp_sysmon permettent de déterminer si des conflits de verrous nuisent à votre système et de connaître les incidences des réglages d'optimisation visant à limiter ces conflits.

Pour plus d'informations sur l'utilisation de sp_object_stats, reportez-vous à la section "Identification des tables pour lesquelles la concurrence d'accès est problématique", page 287.

Pour plus d'informations sur l'utilisation de sp_sysmon, reportez-vous à la section "Gestion des verrous", page 1036.

Les conflits de verrous peuvent nuire aux performances d'Adaptive Server et aux temps de réponse.

Réduction des conflits de verrouillage

Le conflit de verrouillage peut avoir des conséquences sur le débit et le temps de réponse d'Adaptive Server. Vous devez tenir compte du verrouillage lors de la conception de la base de données, et contrôler son fonctionnement lors de la conception de l'application.

Ainsi, il est possible de changer le plan de verrouillage des tables ou de modifier la conception des applications ou des tables qui rencontrent fréquemment des conflits. Exemple :

- Ajoutez des index afin de limiter les conflits, notamment pour les opérations de suppression et de mise à jour.
- Utilisez des transactions courtes afin que les verrous ne restent pas posés trop longtemps.
- Vérifiez les goulots d'étranglement, notamment pour les insertions dans des tables ALP sans index.

Ajout d'index pour réduire les conflits

Une instruction update ou delete qui n'a aucun index exploitable sur ses arguments de recherche doit effectuer un balayage de table et donc maintenir un verrouillage de table exclusif jusqu'à la fin de l'opération. Si la tâche qui modifie les données met également à jour d'autres tables :

- vous pouvez la bloquer en sélectionnant des requêtes ou autres mises à jour.
- Elle peut rester bloquée et devoir attendre, tout en posant un grand nombre de verrous.
- Elle peut être en interblocage avec d'autres tâches.

La création d'un index adéquat pour la requête permet à l'instruction de modification de données d'utiliser des verrous de page ou de ligne, améliorant ainsi la concurrence d'accès à la table. Si vous ne pouvez pas créer d'index pour une transaction de mise à jour ou de suppression longue, utilisez un curseur pour cette opération avec de fréquentes instructions commit transaction afin de limiter le nombre de verrous de page.

Réduction du temps d'exécution des transactions

Une transaction qui pose des verrous doit s'exécuter le plus rapidement possible. En particulier, il est déconseillé d'utiliser des transactions qui nécessitent une interaction avec l'utilisateur alors qu'elles effectuent des verrouillages.

Tableau 10-1 : Exemples

	Avec verrouillage de page	Avec verrouillage de ligne
`begin tran`		

`select balance`
`from account holdlock`
`where acct_number = 25`
	Verrou d'intention partagé sur la table
	Verrou de page partagé
		Verrou d'intention partagé sur la table
		Verrou de ligne partagé

	Si l'utilisateur s'absente maintenant, aucun autre utilisateur ne peut mettre à jour les lignes de cette page.	Si l'utilisateur s'absente maintenant, aucun autre utilisateur ne peut mettre à jour cette ligne.

`update account`
`set balance = balance + 50`
`where acct_number = 25`
	Verrou d'intention exclusif de table
	Verrou de mise à jour sur la page de données
	verrou exclusif sur la page de données
		Verrou d'intention exclusif de table
		Verrou de mise à jour de ligne sur la page de données
		verrou exclusif de ligne sur la page de données

	Personne ne peut lire les lignes de la page qui contient cette ligne.	Personne ne peut lire cette ligne.
`commit tran`		

Evitez si possible tout trafic de réseau dans les transactions. Le réseau est en effet plus lent qu'Adaptive Server. L'exemple suivant présente une transaction exécutée à partir d'isql en deux fois.

`begin tran`
`update account`
`set balance = balance + 50`
`where acct_number = 25`
`go`
	isql batch sent to Adaptive Server
	Maintien des verrous en attendant la validation (commit)

`update account`
`set balance = balance - 50`
`where acct_number = 45`
`commit tran`
`go`
	isql batch sent to Adaptive Server
	Verrous libérés

La réduction du temps d'exécution des transactions est primordiale pour les modifications de données qui se répercutent sur des clés d'index non clusterisé dans des tables verrouillées au niveau de toutes les pages (ALP).

En effet, les index non clusterisés sont denses : le niveau immédiatement supérieur à celui des données comporte une ligne pour chaque ligne de la table. Toutes les opérations d'insertion et de suppression effectuées sur la table, ainsi que la mise à jour des valeurs de clé, touchent au moins une page d'index non clusterisé (et les pages contiguës dans la chaîne de pages en cas de rupture de page ou de libération d'espace).

Si le verrouillage d'une page de données risque de ralentir l'accès à un petit nombre de lignes, les verrous posés sur des pages d'index fréquemment utilisées peut bloquer l'accès à un ensemble de lignes bien plus important.

Comment éviter les points de contention

Les points de contention se produisent lorsque toutes les mises à jour sont effectuées sur une page, comme dans les tables ALP sans index, où toutes les données sont insérées sur la dernière page de la chaîne de pages.

Par exemple, une table historique non indexée qui est mise à jour par tous les utilisateurs subira toujours des conflits de verrou sur la dernière page. L'exemple suivant montre un résultat de sp_sysmon où 11,9 % des insertions sur une table sans index sont en attente de verrou.

```
Derniers verrous de page sur les tables sans index
Granted                 3,0    0,4    185     88,1 %
Waited                  0,4    0,0     25     11,9 %
```

Les solutions possibles sont :

- Changer de plan en adoptant un verrouillage des pages de données ou des lignes de données.

  Du fait que ces plans de verrouillage n'ont pas de chaînage de pages, ils peuvent allouer des pages supplémentaires en cas de blocage lors des insertions.

- Partitionner la table. La partition d'une table sans index crée un grand nombre de chaînes de pages dans la table, augmentant ainsi le nombre des dernières pages dans lesquelles les utilisateurs peuvent réaliser des insertions.

  Les insertions concurrentes provoqueront moins de blocages, puisque plusieurs dernières pages sont accessibles. La partition permet d'améliorer la concurrence d'accès pour les tables sans index sans qu'il soit nécessaire de créer des tables distinctes pour les différents groupes d'utilisateurs.

Pour plus d'informations sur la partition de tables, reportez-vous à la section "Amélioration des performances d'insertion avec les partitions", page 96.

- Créer un index clusterisé pour répartir les mises à jour sur toutes les pages de données de la table.

  Comme la partition, cette solution crée plusieurs points d'insertion dans la table. Cependant, elle génère un overhead car il est nécessaire de conserver l'ordre physique des lignes de la table.

Autres instructions relatives au verrouillage

Ces instructions peuvent vous aider à réduire les conflits de verrous et à améliorer les performances :

- Utilisez le niveau de verrouillage le plus bas requis par chaque application. N'utilisez le niveau d'isolement 2 ou 3 que lorsque cela est nécessaire.

  Les mises à jour effectuées par d'autres transactions peuvent rester en attente jusqu'à ce qu'une transaction exécutée au niveau d'isolement 3 libère l'un de ses verrous partagés.

  N'utilisez le niveau d'isolement 3 que lorsque des lectures uniques ou des lignes fantômes risquent d'interférer avec les résultats souhaités.

  Si le niveau d'isolement 3 n'est requis que pour quelques requêtes, utilisez dans ces dernières le mot-clé holdlock ou la clause at isolation serializing au lieu d'appliquer set transaction isolation level 3 à toute la transaction.

  En revanche, si ce niveau est requis pour la plupart des requêtes, spécifiez set transaction isolation level 3, mais utilisez noholdlock ou at isolation read committed dans les requêtes exécutables au niveau d'isolement 1.

- Si vous devez effectuer un grand nombre d'insertions, de mises à jour et de suppressions sur des tables actives, vous limiterez les risques de blocage en effectuant l'opération dans le cadre d'une procédure enregistrée utilisant un curseur, avec des validations fréquentes.

- Si votre application doit renvoyer une ligne, interagissez avec l'utilisateur, puis mettez la ligne à jour, en pensant à utiliser l'horodatage et la fonction tsequal plutôt que holdlock.

- Si vous utilisez un logiciel tiers, étudiez soigneusement le modèle de verrouillage des applications pour détecter d'éventuels problèmes de concurrence d'accès.

D'autres réglages permettent de limiter les conflits de verrouillage. Par exemple, si un processus verrouille une page et doit effectuer une E/S physique pour lire une autre page, le verrou sera maintenu plus longtemps si la deuxième page ne se trouve pas en mémoire cache.

Dans ce cas, une meilleure utilisation du cache ou le recours à des E/S étendues peuvent limiter les conflits. Vous avez également tout intérêt à optimiser l'indexation et la répartition des E/S physiques sur les différents disques.

Configuration des verrous et des seuils de conversion des verrous

L'administrateur système peut configurer :

- le nombre total de verrous utilisables par les processus sur Adaptive Server
- la taille de la table de hachage de verrou et le nombre de verrous d'attente qui protègent la table de hachage de verrou au niveau page ou ligne, la table de hachage de verrou de table et la table de hachage de verrou d'adresse
- la limite de temporisation des verrous au niveau serveur ainsi que pour les transactions distribuées
- le seuil de conversion des verrous, au niveau serveur, pour une base de données ou des tables spécifiques
- le nombre de verrous par moteur et le nombre de verrous transférés entre la liste globale des verrous disponibles et les moteurs

  Pour plus d'informations sur ces paramètres, reportez-vous aux sections au document *Guide d'administration système d'Adaptive Server*.

Configuration du nombre maximal de verrous sur Adaptive Server

Par défaut, Adaptive Server est configuré avec 5000 verrous. L'administrateur système peut modifier cette limite à l'aide de la procédure sp_configure. Exemple :

```
sp_configure "number of locks", 25000
```

Vous devrez probablement modifier également la valeur du paramètre total memory de sp_configure, car chaque verrou utilise de la mémoire.

Le nombre de verrous requis par une requête peut varier considérablement en fonction du plan de verrouillage, du nombre de traitements parallèles simultanés et du type d'action exécutée par les transactions. L'expérience et l'habitude sont les meilleurs atouts pour configurer correctement les valeurs de votre système.

Vous pouvez commencer par définir arbitrairement 20 verrous pour chaque connexion concurrente active, plus 20 verrous par processus de travail. Envisagez d'augmenter ces valeurs dans les cas suivants :

- si vous modifiez les tables pour utiliser le verrouillage au niveau des lignes de données
- si des requêtes s'exécutent au niveau d'isolement 2 ou 3, ou qu'elles utilisent l'option serializable ou holdlock
- si vous activez le traitement parallèle des requêtes, notamment au niveau d'isolement 2 ou 3
- si les mises à jour de plusieurs lignes sont fréquentes
- si vous avez augmenté la valeur des seuils de conversion des verrous

Estimation du *nombre de verrous* pour des tables verrouillées au niveau des pages de données seulement (DOL)

L'adoption d'un verrouillage des pages de données seulement (DOL) peut exiger un plus grand nombre de verrous ou, au contraire, réduire le nombre de verrous nécessaires :

- Les tables verrouillées au niveau des pages de données seules ont besoin de moins de verrous que les tables ALP, puisque dans les premières, les requêtes ne posent pas de verrous spécifiques sur les pages d'index.

- Les tables qui utilisent un verrouillage des lignes de données peuvent nécessiter un grand nombre de verrous. Bien que dans ces tables, aucun verrou ne soit posé sur les pages d'index, les commandes de modification des lignes de données risquent de poser davantage de verrous.

  Les requêtes qui s'exécutent au niveau d'isolement 2 ou 3 peuvent poser de nombreux verrous de ligne.

Commandes d'insertion et verrous

Une insertion dans une table APL a besoin de *N*+1 verrous, *N* étant le nombre d'index. La même insertion dans une table DOL verrouille uniquement la page ou la ligne de données.

Requêtes *select* et verrous

Les balayages au niveau d'isolement 1, avec read committed with lock définie avec l'option hold locks à 1, posent des verrous qui se chevauchent sur les lignes ou les pages, de sorte qu'ils maintiennent, au plus, deux verrous de page en même temps.

Cependant, les balayages au niveau d'isolement 2 et 3, notamment en cas de verrouillage des lignes de données, peuvent poser de très nombreux verrous, surtout si ces balayages se déroulent en parallèle. Avec un verrouillage au niveau des lignes de données et sans blocage lors de la conversion des verrous, le nombre maximum de verrous dont un balayage de table peut avoir besoin est :

```
row lock promotion HWM * parallel_degree
```

Si un conflit dû à des verrous exclusifs empêche le passage à un verrouillage de page, les balayages risquent de poser un très grand nombre de verrous.

Au lieu d'adapter le nombre de verrous aux exigences des requêtes s'exécutant au niveau d'isolement 2 ou 3, configurez plutôt les applications qui portent sur de nombreuses lignes de telle sorte qu'elles utilisent la commande lock table. Cette commande pose un verrou de table sans tenter d'obtenir des verrous de page individuels.

Pour plus d'informations sur l'utilisation de lock table, reportez-vous à la section "Commande lock table", page 272.

Commandes de modification des données et verrous

Pour les tables qui utilisent un plan de verrouillage des lignes de données, les commandes de modification de données peuvent exiger plus de verrous que sur des tables verrouillées au niveau de toutes les pages (ALP) ou des pages de données seulement (DOL).

Par exemple, une transaction qui procède à de nombreuses insertions dans une table sans index peut poser quelques verrous de page seulement sur une table ALP, mais en cas de verrouillage des lignes, elle aura besoin d'un verrou pour chaque ligne insérée. De même, les transactions qui mettent à jour ou suppriment de nombreuses lignes de données peuvent poser de nombreux verrous sur les tables verrouillées au niveau des lignes.

Configuration de la table de hachage de verrou

Le paramètre de configuration lock hashtable size configure le nombre de compartiments contenus dans la table de hachage de verrou. La taille par défaut de cette table est adaptée à la plupart des installations.

Cependant, si les utilisateurs sont nombreux et si vous avez dû augmenter le paramètre number of locks pour faire face aux demandes, vérifiez la longueur moyenne de la chaîne de hachage à l'aide de la procédure sp_sysmon, et ce dans les périodes de forte activité. Si la longueur moyenne des chaînes de hachage dépasse 4 ou 5, il convient d'augmenter lock hashtable size en élevant la valeur courante à la puissance 2 suivante.

La longueur de la chaîne de hachage peut être élevée pour des batchs d'insertion importants, tels que des opérations de bulkcopy. Ce phénomène est normal et il n'est pas nécessaire de redéfinir la taille de la table de hachage de verrou.

Définition des seuils de conversion des verrous

Les seuils de conversion des verrous définissent le nombre de verrous de page acceptés par une tâche ou un processus de travail avant qu'Adaptive Server n'essaie de poser un verrou de table sur l'objet. Vous pouvez définir ces seuils au niveau du serveur, de la base de données ou de la table.

Les valeurs par défaut conviennent parfaitement à plusieurs tailles de table. Si vous définissez des valeurs plus élevées, les requêtes auront plus de mal à obtenir des verrous de table, en particulier sur des tables très volumineuses qui comportent souvent des centaines de pages verrouillées par différentes requêtes.

Remarque La conversion des verrous implique toujours deux niveaux : conversion des verrous de page en verrous de table, ou des verrous de lignes en verrous de table. Les verrous de ligne ne sont jamais convertis en verrous de page.

Conversion des verrous et sessions de balayage

La conversion des verrous se déroule pour chaque session de balayage.

Une *session de balayage* désigne le processus par lequel Adaptive Server contrôle les balayages de tables au sein d'une transaction. Une transaction peut comporter plusieurs sessions de balayage pour les raisons suivantes :

- Une table peut être balayée plusieurs fois dans le cadre d'une transaction lorsqu'il existe des jointures, des sous-requêtes, des clauses exists, etc.

  Chaque balayage de la table correspond à une session de balayage.

- Une requête exécutée en parallèle balaye une table à l'aide de plusieurs processus de travail.

  Chaque processus est associé à une session de balayage distincte.

Lorsqu'une table entière risque d'être nécessaire, un verrou de table est plus efficace que plusieurs verrous de page ou le ligne. Tout d'abord, une tâche pose des verrous de page ou de ligne, puis elle tente de passer à un verrou de table lorsqu'une session de balayage pose plus de verrous de page ou de ligne que n'en spécifie le seuil de conversion des verrous.

Etant donné que la conversion des verrous s'effectue au niveau de la session de balayage, aucune session de balayage ne doit dépasser le seuil de conversion, mais le nombre total de verrous de page ou de ligne pour l'ensemble de la transaction peut être supérieur à ce seuil. Certains verrous peuvent être maintenus tout au long de la transaction ; si celle-ci comporte plusieurs sessions de balayage, elle risque d'accumuler un nombre considérable de verrous.

La conversion de verrous ne peut pas se dérouler si une autre tâche pose des verrous en conflit avec le type de verrou dont elle a besoin. Par exemple, si un processus maintient un verrou de page exclusif, aucun autre processus ne peut obtenir un verrou de table tant que les verrous exclusifs de page ne sont pas libérés.

Lorsque la conversion de verrous est impossible en raison de conflits, un processus peut cumuler des verrous de page ou de ligne au point de dépasser le seuil de conversion et même d'utiliser tous les verrous disponibles dans Adaptive Server.

Les paramètres de conversion des verrous sont les suivants :

- Pour les tables ALP et DOL, page lock promotion HWM, page lock promotion LWM et page lock promotion PCT.
- Pour les tables verrouillées au niveau des lignes de données, row lock promotion HWM, row lock promotion LWM et row lock promotion PCT.

Les abréviations contenues dans ces paramètres sont :

- HWM, high water mark (seuil maximal)
- LWM, low water mark (seuil minimal)
- PCT, percent (pour cent)

Seuil maximal de conversion de verrous

page lock promotion HWM et row lock promotion HWM fixent un nombre maximum de verrous de page ou de ligne autorisés sur une table avant qu'Adaptive Server ne tente d'imposer un verrou de table. La valeur par défaut est 200.

Lorsque le nombre de verrous posés au cours d'une session de balayage dépasse ce seuil, Adaptive Server essaie de poser un verrou de table.

Si vous définissez un seuil maximal supérieur à 200, les tâches ou les processus de travail risquent moins de poser un verrou de table. Par exemple, si un processus met à jour plus de 200 lignes dans une table très volumineuse au cours d'une transaction, un seuil maximal de conversion supérieur à 200 empêche ce processus de poser un verrou de table.

Si vous définissez un seuil maximal à moins de 200, les probabilités qu'une tâche ou un processus de travail a de poser un verrou de table augmentent.

Seuil minimal de conversion de verrous

page lock promotion LWM et row lock promotion LWM fixent un nombre minimum de verrous autorisés sur une table avant qu'Adaptive Server ne tente de poser un verrou de table. La valeur par défaut est 200. Adaptive Server ne cherche jamais à poser un verrou de table tant que le seuil minimal n'est pas atteint.

Le seuil minimal doit être inférieur ou égal au seuil maximal correspondant.

Si vous définissez un seuil minimal très élevé, les probabilités pour une tâche ou un processus de travail de poser un verrou de table diminuent, mais en contrepartie, ils utilisent davantage de verrous et risquent donc d'épuiser tous les verrous disponibles sur Adaptive Server. Ce risque est particulièrement important pour les requêtes qui mettent à jour de nombreuses lignes dans une table verrouillée au niveau des lignes, ou qui sélectionnent de nombreuses lignes dans une table verrouillée au niveau des lignes avec un niveau d'isolement 2 ou 3.

Si des conflits de verrous empêchent la conversion, augmentez la valeur du paramètre de configuration number of locks.

Pourcentage de conversion de verrous

page lock promotion PCT et row lock promotion PCT définissent le pourcentage de pages ou de lignes verrouillées (en fonction de la taille de la table), au-delà duquel Adaptive Server tente de poser un verrou de table lorsque le nombre de verrous est compris entre le seuil maximal (lock promotion HWM) et le seuil minimal (lock promotion LWM).

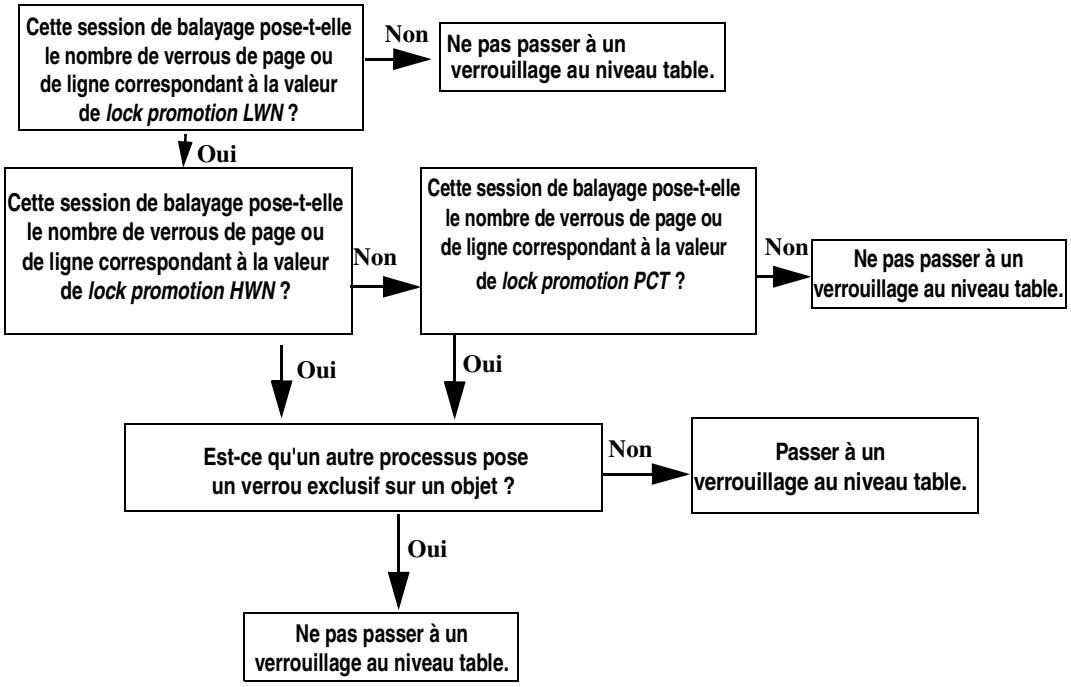
La valeur par défaut est 100.

Adaptive Server tente de convertir les verrous de page ou de lignes en un verrou de table, lorsque le nombre total de verrous posés sur la table dépasse :

```
(PCT * nombre de pages ou de lignes de la table) / 100
```

Si vous attribuez à lock promotion PCT une valeur très faible, les probabilités qu'une transaction utilisateur pose un verrou de table augmentent. La figure 10-1 montre comment Adaptive Server détermine la nécessité de convertir le type de verrous sur une table.

Figure 10-1 : Logique de conversion de verrous



Définition des seuils de conversion des verrous sur l'ensemble du serveur

Au niveau du serveur, la commande suivante définit le paramètre page lock promotion LWM à 100, page lock promotion HWM à 2000 et page lock promotion PCT à 50 pour toutes les tables DOL et ALP :

```
sp_setpglockpromote "server", NULL, 100, 2000, 50
```

Dans cet exemple, la tâche ne tente pas d'effectuer la conversion à un verrou de table à moins que le nombre de verrous sur la table ne soit compris entre 100 et 2000.

Si une commande nécessite plus de 100 mais moins de 2000 verrous, Adaptive Server compare le nombre des verrous au pourcentage de verrous dans la table.

Si ce nombre est supérieur au nombre de pages calculé avec le pourcentage, Adaptive Server tente de poser un verrou de table.

sp_setrowlockpromote définit les paramètres de configuration pour toutes les tables verrouillées au niveau des lignes de données :

```
sp_setrowlockpromote "server", null, 300, 500, 50
```

La valeur par défaut des paramètres de conversion de verrous convient à la plupart des applications.

Définition des seuils de conversion des verrous pour une table ou une base de données

Pour configurer la conversion des verrous pour une table ou une base de données, initialisez les trois seuils de conversion des verrous. Exemple :

```
sp_setpglockpromote "table", titles, 100, 2000, 50
sp_setrowlockpromote "table", authors, 300, 500, 50
```

Une fois les valeurs initialisées, vous pouvez modifier n'importe quelle valeur individuellement. Par exemple, pour modifier uniquement lock promotion PCT, utilisez la commande suivante :

```
sp_setpglockpromote "table", titles, null, null, 70
sp_setrowlockpromote "table", authors, null, null, 50
```

Pour configurer une source de données, utilisez :

```
sp_setpglockpromote "database", pubs3, 1000, 1100, 45
sp_setrowlockpromote "database", pubs3, 1000, 1100, 45
```

Priorité des valeurs

Vous pouvez modifier les seuils de conversion des verrous pour une base de données utilisateur ou uniquement pour une table. Les valeurs définies pour la table ont priorité sur les paramètres définis au niveau de la base de données ou du serveur ; les valeurs définies pour une base de données ont priorité sur celles définies à l'échelle du serveur.

Les valeurs établies à l'échelle du serveur s'appliquent à toutes les tables utilisateur sur le serveur, sauf si la base de données ou la table ont des valeurs de conversion des verrous qui leur sont propres.

Suppression des valeurs définies pour une base de données et une table

Pour supprimer les seuils de conversion des verrous définis pour une table ou une base de données, utilisez la procédure système sp_dropglockpromote ou sp_droprowlockpromote. Lorsque vous supprimez les seuils de conversion des verrous d'une base de données, les tables pour lesquelles des seuils de conversion des verrous n'ont pas été configurés utiliseront les valeurs définies pour le serveur.

Lorsque la suppression s'applique à une table, Adaptive Server utilise ensuite les seuils de conversion définis pour la base de données ou, si ceux-ci n'existent pas, les valeurs définies pour le serveur. Vous ne pouvez pas supprimer les seuils de conversion des verrous définis au niveau du serveur.

Utilisation de *sp_sysmon* pour l'optimisation des seuils de conversion de verrous

Utilisez sp_sysmon pour savoir le nombre et le type de conversions de verrous.

Pour plus d'informations, reportez-vous à la section "Conversions de verrou", page 1044.

En cas de problème, cherchez les signes de conflit de verrouillage dans les données "Granted" et "Waited" de la section "Lock Detail" des résultats de sp_sysmon.

Pour plus d'informations, reportez-vous à la section "Détail des verrous", page 1040.

Si les conflits de verrouillage sont nombreux et la conversion des verrous fréquente, envisagez de modifier les seuils de conversion de verrous pour les tables concernées.

Utilisez Adaptive Server Monitor pour voir comment la modification du seuil de conversion de verrous agit au niveau objet.

Choix du plan de verrouillage d'une table

En général, le choix du plan de verrouillage d'une nouvelle table est fonction des risques potentiels de conflits de verrous que les applications peuvent rencontrer. Pour changer le plan de verrouillage d'une table existante, il convient d'évaluer les conflits éventuels mais il faut également prendre en compte les performances de l'application.

Les paragraphes suivants décrivent quelques situations caractéristiques et des principes généraux qui vous aideront à choisir un plan de verrouillage adapté :

- Les applications nécessitent un accès clustérisé aux lignes de données en raison de requêtes de type intervalle ou de clauses order by.

  Le verrouillage de toutes les pages (allpages) offre un accès clusterisé plus performant que le verrouillage des données seules.

- Un grand nombre d'applications accède à environ 10 à 20 % des lignes de données, avec de nombreuses mises à jour et sélections des mêmes données.

  Utilisez un verrouillage au niveau des lignes de données ou des pages de données seules afin de réduire les conflits, notamment sur les tables à haut conflit d'accès.

- La table n'a pas d'index et le taux des insertions est élevé.

  Utilisez un verrouillage au niveau des lignes de données pour éviter les conflits. Si le nombre de lignes insérées par batch est élevé, le verrouillage des pages de données est également acceptable. Le verrouillage de toutes les pages comporte plus de risques de conflit sur la dernière page des tables sans index.

- Les applications doivent gérer un taux très élevé de transactions ; les conflits seront probablement très peu nombreux.

  Utilisez un verrouillage de toutes les pages, car une réduction du surcoût au niveau des verrous et des loquets améliore les performances.

Analyse des applications existantes

Si vos applications sont souvent confrontées à des problèmes de blocage et d'interblocage, procédez comme suit afin d'analyser le problème :

1 Vérifiez les interblocages et les conflits de verrouillage :

- Utilisez sp_object_stats pour déterminer les tables qui posent un problème de blocage.
- Identifiez les tables concernées par l'interblocage, soit à l'aide de la procédure sp_object_stats, soit en activant le paramètre de configuration print deadlock information.

2 Si la table utilise un verrouillage de toutes les pages, vérifiez que la structure modifiée de l'index clusterisé sur les tables DOL ne nuit pas aux performances.

Pour plus d'informations, reportez-vous à la section "Tables dont les performances des index clusterisés doivent rester élevées", page 255.

3 Si la table utilise un verrouillage de toutes les pages, convertissez-le en verrouillage des pages de données et voyez si le problème de concurrence d'accès est ainsi résolu.

4 Réexécutez les tests de concurrence. Si la concurrence d'accès reste un problème, adoptez un verrouillage au niveau des lignes de données.

Choix d'un plan de verrouillage sur la base des statistiques relatives aux conflits

Si la table utilise un plan de verrouillage de toutes les pages, les statistiques fournies par sp_object_stats font état des conflits de verrous de page et d'index.

Si les conflits de verrous représentent 15 % ou plus de l'ensemble des verrous (partagés, de mise à jour et exclusifs), sp_object_stats recommande de passer à un verrouillage des pages de données. Effectuez la modification et exécutez de nouveau sp_object_stats.

Si, même avec un verrouillage des pages de données, les conflits sont supérieurs à 15 %, sp_object_stats vous recommande d'adopter un verrouillage des lignes de données. Cette approche en deux temps s'appuie sur les éléments suivants :

- Le passage d'un verrouillage ALP à un des plans de verrouillage des données seules (DOL) est long et coûteux en termes d'E/S, alors que le passage de l'un des types de verrouillage des données seules à l'autre est rapide et ne nécessite pas la copie de la table.
- Le verrouillage au niveau des lignes nécessite plus de verrous et entraîne un surcoût plus important.

  Si, une fois que vous avez appliqué un verrouillage des pages de données sur les tables à risque, vos applications rencontrent peu de conflits, il est inutile de passer à un verrouillage des lignes de données, vous éviterez ainsi le surcoût qui lui est associé.

  Remarque Le nombre de verrous disponibles pour tous les processus sur le serveur est limité par le paramètre de configuration number of locks.

  Le passage à un verrouillage DOL réduit le nombre de verrous requis puisque les pages d'index ne sont plus verrouillées.

  Le passage à un verrouillage des lignes de données peut en revanche augmenter le nombre de verrous requis, puisqu'un verrou s'applique à chaque ligne.

  Pour plus d'informations, reportez-vous à la section "Estimation du nombre de verrous pour des tables verrouillées au niveau des pages de données seulement (DOL)", page 242.

Lorsque vous analysez le résultat fourni par sp_object_stats, examinez les tables qui sont utilisées ensemble dans les transactions de vos applications. Le verrouillage des tables utilisées ensemble dans des requêtes et des transactions peut en effet avoir une incidence sur les conflits de verrous des autres tables.

La réduction des conflits sur une table peut également entraîner des conflits sur d'autres tables ou aggraver un conflit sur une autre table, qui n'apparaissait pas en raison du blocage de la première table. Exemple :

- Les conflits de verrouillage sont élevés pour deux tables qui sont mises à jour dans des transactions impliquant plusieurs tables. Les applications verrouillent d'abord la TableA, puis tentent de poser des verrous sur la TableB et se bloquent, en posant des verrous sur la TableA.

Des tâches supplémentaires exécutant la même application se bloquent en essayant de poser des verrous sur la TableA. Pour les deux tables, les conflits sont importants, ainsi que les temps d'attente.

Le passage sur la TableB à un verrouillage des données seules peut réduire les conflits sur les deux tables.

- Pour la TableT, les conflits sont importants, de sorte que le plan de verrouillage existant est remplacé par un verrouillage des données seules.

  La réexécution de la procédure sp_object_stats montre à présent des conflits sur la TableX, qui jusque là n'en avait que très peu. Les conflits sur la TableX étaient masqués par le problème de blocage sur la TableT.

Si votre application utilise de nombreuses tables, vous pouvez adopter progressivement un verrouillage des données seules, en l'appliquant aux tables comportant le plus de conflits de verrous. Testez ensuite les résultats des modifications apportées en réexécutant la procédure sp_object_stats.

Vous devez exécuter vos tests standard de contrôle des performances avant et après les modifications.

Contrôle et gestion des tables après la conversion

Après être passé, sur une ou plusieurs tables, à un plan de verrouillage des données seules :

- Vérifiez les plans d'exécution de requête et les statistiques d'E/S, notamment pour les requêtes qui utilisent des index clusterisés.
- Examinez les tables pour déterminer l'incidence du nouveau plan de verrouillage :
  - sur les taux de clusterisation, en particulier pour les tables avec des index clusterisés
  - sur le nombre de lignes redirigées vers d'autres pages de la table

Applications peu susceptibles de bénéficier d'un verrouillage des données seules

Cette section décrit les tables et les types d'application qui tirent peu de profit du passage au verrouillage des données seules, ou qui risquent même d'exiger un surcroît de gestion après la conversion.

Tables dont les performances des index clusterisés doivent rester élevées

Si des requêtes nécessitant des performances élevées utilisent des index clusterisés pour renvoyer de nombreuses lignes dans l'ordre d'indexation, vous risquez de constater une dégradation des performances si vous adoptez sur les tables concernées un verrouillage des données seules. Dans les tables DOL, les index clusterisés sont structurés comme les index non clusterisés.

Des algorithmes de positionnement servent à maintenir les lignes nouvellement insérées à proximité des lignes existantes ayant des valeurs adjacentes, dès l'instant que l'espace disponible sur les pages voisines le permet.

Pour une table DOL avec un index clusterisé, les performances doivent être proches de ce qu'elles seraient sur cette même table ALP immédiatement après une commande create clustered index ou reorg rebuild, mais les performances, notamment en présence d'E/S étendues, déclinent si les taux de clusterisation diminuent du fait des insertions et de la redirection des lignes vers d'autres pages.

Sur les tables dans lesquelles peu d'insertions sont effectuées, les performances restent élevées. En revanche, sur les tables où les insertions sont nombreuses, l'administrateur système peut supprimer puis recréer l'index clusterisé, ou exécuter plus fréquemment la commande reorg rebuild.

L'utilisation des propriétés de gestion de l'espace telles que fillfactor, exp_row_size et reservepagegap peut contribuer à réduire la fréquence des opérations de maintenance. Dans certains cas, l'utilisation d'un plan de verrouillage de toutes les pages d'une table peut, même s'il existe certains conflits, améliorer les performances des requêtes qui effectuent des lectures d'index clusterisés, par opposition au verrouillage des données seules.

Tables avec des lignes de longueur maximale

Aussi, dans les premières, la longueur maximale des lignes est légèrement plus courte que dans les secondes.

Dans des tables utilisant un verrouillage des données seules et contenant uniquement des colonnes de longueur fixe, la taille de ligne maximale est de 1958 octets de données utilisateur. Dans les tables ALP, le maximum autorisé est de 1960 octets.

Pour des tables contenant des colonnes à longueur variable, il convient de soustraire deux octets par colonne (y compris les colonnes qui admettent les valeurs NULL). Par exemple, la taille de ligne maximale pour des données utilisateur dans une table DOL contenant quatre colonnes de longueur variable est de 1950 octets.

Si, pour une table ALP et contenant des colonnes de longueur fixe avec plus de 1958 octets, vous tentez de changer le plan de verrouillage, la commande échoue dès que la lecture du schéma de la table commence.

Si, toujours pour une table APL et contenant des colonnes de longueur variable avec certaines lignes qui dépassent la taille maximale autorisée par le verrouillage des données seules, vous tentez de changer le plan de verrouillage, la commande alter table échoue dès qu'elle rencontre la première ligne trop longue à convertir.

CHAPITRE 11

Utilisation des commandes de verrouillage

Ce chapitre traite des types de verrous utilisés dans Adaptive Server et des commandes ayant une incidence sur le verrouillage.

Sujet	Page
Spécification du plan de verrouillage d'une table	257
Contrôle des niveaux d'isolement	264
Verrouillage readpast	269
Curseurs et verrouillage	269
Commandes supplémentaires de verrouillage	272

Spécification du plan de verrouillage d'une table

Les plans de verrouillage suivants d'Adaptive Server vous permettent de choisir le type de verrouillage le mieux adapté à chaque table dans le cadre de votre application et d'ajuster le plan de verrouillage adopté pour une table si des conflits ou des problèmes de performances l'exigent. Les outils permettant de spécifier les plans de verrouillage sont les suivants :

- sp_configure, pour spécifier un plan de verrouillage par défaut à l'échelle du serveur,
- create table, pour spécifier le plan de verrouillage des tables créées,
- alter table, pour changer le plan de verrouillage d'une table,
- select into, pour spécifier le plan de verrouillage d'une table créée suite à la sélection de données dans d'autres tables.

Définition d'un schéma de verrouillage sur l'ensemble du serveur

Le paramètre de configuration lock scheme définit le plan de verrouillage à utiliser pour les nouvelles tables, si la commande create table n'en précise pas.

Pour savoir quel est le plan de verrouillage en vigueur, utilisez :

```
sp_configure "lock scheme"
```

```
Nom paramètre    Défaut     Mémoire        Valeur        Valeur
en exécution                 utilisée       configurée
---------------- ---------- -----------    ------------  -----------
lock scheme      allpages             0    datarows      datarows
```

La syntaxe pour changer de plan de verrouillage est :

```
sp_configure "lock scheme", 0,
{allpages | datapages | datarows }
```

La commande suivante définit le mode de verrouillage au niveau des pages de données (datapages) comme plan par défaut pour le serveur :

```
sp_configure "lock scheme", 0, datapages
```

Lorsque vous installez Adaptive Server pour la première fois, lock scheme est défini à allpages.

Spécification d'un plan de verrouillage avec *create table*

Vous pouvez spécifier le plan de verrouillage d'une nouvelle table à l'aide de la commande create table. Respectez la syntaxe suivante :

```
create table table_name (column_name_list)
    [lock {datarows | datapages | allpages }]
```

Si vous ne spécifiez pas de plan de verrouillage pour une table, la valeur par défaut définie par le paramètre de configuration lock scheme à l'échelle du serveur est appliquée.

La commande ci-après spécifie un verrouillage au niveau des lignes de données (datarows) pour la table new_publishers :

```
create table new_publishers
(pub_id     char(4)      not null,
 pub_name   varchar(40)  null,
 city       varchar(20)  null,
 state      char(2)      null)
lock datarows
```

Le plan de verrouillage spécifié avec create table remplace le paramètre par défaut défini pour le serveur.

Pour plus d'informations, reportez-vous à la section "Définition d'un schéma de verrouillage sur l'ensemble du serveur", page 258.

Changement de plan de verrouillage à l'aide de *alter table*

Pour changer le plan de verrouillage d'une table, utilisez la commande alter table. Respectez la syntaxe suivante :

```
alter table nom_table
    lock {allpages | datapages | datarows}
```

La commande ci-après remplace le plan de verrouillage existant de la table titles par un plan de verrouillage au niveau des lignes de données (datarows) :

```
alter table titles lock datarows
```

alter table gère le passage d'un plan de verrouillage à un autre, quel qu'il soit. Le passage d'un verrouillage de toutes les pages (allpages) à un verrouillage des données seules nécessite la copie des lignes de données sur de nouvelles pages et la recréation des index sur la table.

Cette opération implique plusieurs étapes et exige un espace suffisant pour la copie de la table et des index. Le temps requis dépend de la taille de la table et du nombre d'index.

Le passage d'un verrouillage des pages de données (datapages) à un verrouillage des lignes de données (datarows) ou l'inverse n'exige pas la copie de pages de données ni, par conséquent, la reconstruction des index. Le passage d'un des plans de verrouillage des données seules à l'autre ne met à jour que les tables système et se déroule en quelques secondes.

Remarque Vous ne pouvez pas utiliser un verrouillage des données seules sur des tables contenant des lignes d'une taille maximale, ou presque, c'est-à-dire d'une longueur de 1962 (y compris les deux octets de la table d'offset de ligne).

Pour les tables verrouillées au niveau des pages de données seulement (DOL) qui ne contiennent que des colonnes de longueur fixe, la taille maximale des lignes de données utilisateur est de 1960 octets (y compris les 2 octets de la table d'offset de ligne).

Les tables contenant des colonnes de longueur variable exigent 2 octets supplémentaires par colonne (y compris les colonnes qui admettent les valeurs NULL).

Pour plus d'informations sur les lignes et l'overhead (surcoût) de ligne, reportez-vous au chapitre 15, "Détermination de la taille des tables et des index".

Changement de plan de verrouillage – avant et après

Avant de passer d'un plan de verrouillage de toutes les pages (allpages) à un plan de verrouillage des données seules, ou l'inverse, il est préférable d'exécuter les opérations suivantes :

- Si la table est partitionnée et si update statistics n'a pas été exécutée depuis les dernières modifications importantes, exécutez update statistics sur la table dont vous allez changer le plan de verrouillage. La commande alter table...lock est plus performante si les statistiques des tables partitionnées sont précises.

  Le changement du plan de verrouillage n'a aucune incidence sur la répartition des données dans les partitions ; les lignes de la partition 1 sont copiées dans la partition 1 de la copie de la table.

- Exécutez une sauvegarde de la base de données.
- Définissez les propriétés de gestion de l'espace qui doivent s'appliquer à la copie de la table ou aux index reconstruits.

  Pour plus d'informations, reportez-vous à la section chapitre 13, "Définition des propriétés de gestion de l'espace".

- Déterminez si l'espace libre est suffisant.

Pour plus d'informations, reportez-vous à la section "Détermination de l'espace disponible pour des activités de maintenance", page 404.

- Si, dans la base de données, certaines tables sont partitionnées et nécessitent un tri parallèle :
  - Utilisez sp_dboption pour définir l'option de base de données select into/bulkcopy/pllsort à vrai et exécutez checkpoint dans la base de données.
  - Définissez votre configuration pour des performances optimales de tri parallèle.

Après exécution de *alter table*

- Exécutez dbcc checktable sur la table et dbcc checkalloc sur la base de données pour garantir la cohérence de la base.
- Exécutez une sauvegarde de la base de données.

Remarque Après être passé d'un verrouillage de toutes les pages à un verrouillage des données seules, ou l'inverse, il est impossible d'utiliser la commande dump transaction pour sauvegarder le journal de transactions.

Vous devez au préalable effectuer une sauvegarde complète de la base.

Coût relatif au passage du verrouillage de toutes les pages au verrouillage des données seules ou inversement

Le passage du verrouillage de toutes les pages en verrouillage des données seulement ou l'inverse est une opération coûteuse, en termes d'E/S. Le temps nécessaire dépend de la taille de la table et du nombre d'index à recréer. L'importance du coût tient principalement aux E/S requises pour copier les tables et recréer les index. Certaines écritures dans le journal des transactions sont également indispensables.

Lors du remplacement du verrouillage de toutes les pages par un verrouillage des données seules ou inversement, la commande alter table...lock effectue les actions suivantes :

- Toutes les lignes de la table sont copiées sur de nouvelles pages et remises en forme conformément au nouveau format. Si vous passez à un verrouillage des données seules, les lignes de moins de 10 octets sont remplies jusqu'à atteindre 10 octets. Dans le sens inverse (d'un verrouillage de toutes les pages à un verrouillage des données seules), le remplissage est supprimé pour les lignes de moins de 10 octets.
- Tous les index de la table sont supprimés puis recréés.
- Les anciennes pages de la table sont supprimées.
- Les tables système sont mises à jour en fonction du nouveau plan de verrouillage.
- Le compteur associé à la table est mis à jour afin que les plans d'exécution de requête soient recompilés.

S'il existe sur la table un index clusterisé, les lignes sont copiées dans l'ordre des clés de cet index sur les nouvelles pages de données. Sinon, en cas de conversion du verrouillage de toutes les pages en verrouillage des données seules, elles sont copiées dans l'ordre du chaînage de pages.

La commande entière alter table...lock est exécutée comme une transaction unique afin de permettre la reprise. Un verrou de table exclusif est posé sur la table pour la durée de la transaction.

Le passage d'un verrouillage des pages de données (datapages) à un verrouillage des lignes de données (datarows) ou l'inverse n'exige pas la copie de pages de données ni la reconstruction des index. Il met simplement à jour les tables système. La définition de sp_dboption "select into/bulkcopy/pllsort" n'est pas obligatoire.

Performances du tri durant l'exécution de *alter table*

Si la table modifiée est partitionnée, le tri parallèle peut être utilisé pour la reconstruction des index. Les performances de alter table sont considérablement améliorées si le cache de données et le serveur sont configurés pour optimiser le tri parallèle.

Durant l'exécution de alter table, les index sont recréés un à un. Si votre système dispose d'un nombre suffisant de moteurs et si le débit des E/S permet la gestion simultanée des opérations create index, vous réduirez le temps global requis pour changer de plan de verrouillage en procédant comme suit :

- suppression des index non clusterisés,
- modification du plan de verrouillage,
- configuration du système pour des performances de tri parallèle optimales,
- recréation immédiate de deux index non clustérisés ou plus.

Spécification d'un plan de verrouillage à l'aide de *select into*

Vous pouvez spécifier un plan de verrouillage au moment de la création d'une nouvelle table, avec la commande select into. Respectez la syntaxe suivante :

```
select [all | distinct] liste_select
        into [[base_de_données.]propriétaire.]nom_table
         [lock {datarows | datapages | allpages }]
from ...
```

Si vous ne spécifiez pas de plan de verrouillage à l'aide de select into, la nouvelle table utilise le plan défini par défaut à l'échelle du serveur par le paramètre lock scheme.

La commande suivante spécifie un verrouillage des lignes de données pour la table créée :

```
select title_id, title, price
into bus_titles
lock datarows
from titles
where type = "business"
```

Les tables temporaires créées avec le format de dénomination #nom_table sont des tables mono-utilisateur, de sorte que les conflits de verrou ne sont pas un problème. Pour les tables temporaires qui sont partagées par plusieurs utilisateurs, c'est-à-dire créées sur le modèle tempdb..nomtable, il est possible d'utiliser n'importe quel plan de verrouillage.

Contrôle des niveaux d'isolement

Vous pouvez définir le niveau d'isolement de transaction utilisé par les commandes select :

- Pour toutes les requêtes de la session, à l'aide de la commande set transaction isolation level,
- Pour une requête seule, avec la clause at isolation,
- Pour des tables spécifiques d'une requête, avec les mots-clés holdlock, noholdlock et shared.

Lorsque vous définissez le niveau de verrouillage de vos applications, choisissez le niveau le plus bas répondant aux exigences de votre modèle de production. La configuration simultanée au niveau de la session et des requêtes permet aux transactions s'exécutant en même temps d'obtenir les résultats escomptés tout en réduisant au maximum les risques de blocage.

Remarque Si vous utilisez le niveau d'isolement 2 (lectures répétables) sur des tables en mode verrouillage de toutes les pages, le niveau d'isolement 3 (lectures sérialisables) est également appliqué.

Pour plus d'informations sur les niveaux d'isolement, reportez-vous au *Guide d'administration système*.

Définition des niveaux d'isolement pour une session

La norme SQL spécifie le niveau 3 comme niveau d'isolement par défaut. Pour mettre en œuvre ce niveau, vous disposez de la commande Transact-SQL set transaction isolation level. Par exemple, pour définir le niveau 3 comme niveau d'isolement par défaut de votre session, utilisez la commande suivante :

```
set transaction isolation level 3
```

Si la session est imposée au niveau 3, vous pouvez choisir le niveau 1 pour une requête à l'aide de l'option noholdlock, comme décrit ci-dessous.

Si vous utilisez le niveau d'isolement par défaut d'Adaptive Server (défini à 1) ou si vous avez défini un niveau 0 ou 2 avec la commande set transaction isolation level, vous pouvez imposer le niveau 3 en utilisant l'option holdlock afin que les verrous partagés soient maintenus jusqu'à la fin de la transaction.

Pour déterminer le niveau d'isolement courant d'une session, vous pouvez utiliser la variable globale @@*isolation*.

Syntaxe des options de verrouillage au niveau requête et au niveau table

Les options holdlock, noholdlock et shared peuvent être définies pour chaque table dans une instruction select, la clause at isolation s'appliquant à toute la requête.

```
select liste_sélection
    from nom_table [holdlock | noholdlock] [shared]
        [, nom_table [[holdlock | noholdlock] [shared]
    {clauses where/group by/order by/compute clauses}
    [at isolation {
        [ read uncommitted | 0] |
        [ read committed | 1 ] |
        [ repeatable read | 2 ]|
         [ serializable | 3 ] ]
```

La commande readtext doit respecter la syntaxe suivante :

```
readtext [[ base_de_données.]propriétaire.]nom_table. nom_colonne
    pointeur_texte offset size
    [holdlock | noholdlock] [readpast]
    [using {bytes | chars | characters}]
    [at isolation {
        [ read uncommitted | 0] |
        [ read committed | 1 ] |
        [ repeatable read | 2 ]|
         [ serializable | 3 ] } ]
```

Utilisation de *holdlock*, *noholdlock* ou *shared*

Pour ignorer le niveau de verrouillage d'une session, utilisez les options holdlock, noholdlock et shared sur les tables mentionnées dans les commandes select ou readtext :

Niveau à utiliser	Mot-clé	Effet
1	noholdlock	Ne garde pas les verrouillages jusqu'à la fin de la transaction ; à utiliser pour mettre en œuvre le niveau 1 lorsque le niveau 3 est défini.
2, 3	holdlock	Maintient les verrouillages partagés jusqu'à la fin de la transaction ; permet de mettre en œuvre le niveau 3 lorsque le niveau 1 est défini.
N/A	partagée	Utilisation des verrous partagés à la place des verrous de mise à jour pour des instructions select dans des curseurs ouverts pour la mise à jour.

Ces mots-clés modifient le mode de verrouillage de la transaction. Avec holdlock, tous les verrous sont maintenus jusqu'à la fin de la transaction.

Si vous spécifiez holdlock dans une requête pendant que le niveau d'isolement 0 est appliqué à la session, Adaptive Server émet un avertissement, ignore la clause holdlock et ne pose pas de verrou pendant l'exécution de la requête.

Si vous spécifiez holdlock et read uncommitted, Adaptive Server émet un message d'erreur et la requête n'est pas exécutée.

Utilisation de la clause *at isolation*

Vous pouvez modifier le niveau d'isolement pour toutes les tables de la requête en utilisant la clause at isolation avec la commande select ou readtext. Les options de la clause at isolation sont les suivantes :

Niveau à utiliser	Option	Effet
0	read uncommitted (lecture des modifications non validées)	Lit les modifications non validées ; permet de mettre en oeuvre le niveau 0 (lecture de données modifiées, ou dirty read) lorsque le niveau 1, 2 ou 3 est défini.
1	read committed (lecture validée)	Ne lit que les modifications validées ; attend la libération des verrous ; permet de lire uniquement les modifications validées (niveau 0), sans autoriser le maintien des verrous.
2	repeatable read	Maintient les verrous partagés jusqu'à la fin de la transaction ; permet de mettre en œuvre le niveau 2 lorsque le niveau 0 ou 1 est défini.
3	serializable	Maintient les verrous partagés jusqu'à la fin de la transaction ; permet de mettre en œuvre le niveau 3 lorsque le niveau 1 ou 2 est défini.

Par exemple, l'instruction suivante interroge la table titles en utilisant le niveau d'isolement 0 :

```
select *
from titles
at isolation read uncommitted
```

Pour plus d'informations sur l'option transaction isolation level et la clause at isolation, reportez-vous au *Guide de l'utilisateur Transact-SQL.*

Définition de verrous plus contraignants

Si le niveau d'isolement 1 est en général suffisant mais que quelques requêtes exigent un niveau supérieur, vous pouvez imposer ce dernier de façon sélective en utilisant des clauses dans l'instruction select :

- Utilisez repeatable read pour imposer le niveau 2.
- Utilisez holdlock ou at isolation serializable pour imposer le niveau 3.

Le mot-clé holdlock restreint le verrou de page ou de table partagé. Il s'applique :

- aux verrous partagés,
- à la table ou à la vue pour laquelle il est spécifié,
- pendant l'exécution de l'instruction ou de la transaction comportant l'instruction.

La clause at isolation s'applique à toutes les tables de la clause from, uniquement pendant la transaction. Les verrous sont libérés à la fin de la transaction.

Dans une transaction, holdlock indique à Adaptive Server de maintenir les verrous partagés jusqu'à la fin de la transaction au lieu de les libérer dès que la table, la vue ou la page de données requise n'est plus utilisée. Adaptive Server maintient toujours les verrous exclusifs jusqu'à la fin de la transaction.

Dans l'exemple ci-dessous, l'utilisation de holdlock garantit des résultats cohérents pour les deux requêtes :

```
begin transaction
select branch, sum(balance)
    from account holdlock
    group by branch
select sum(balance) from account
commit transaction
```

La première requête pose un verrou partagé sur la table account afin qu'aucune transaction ne puisse mettre à jour les données avant l'exécution de la seconde requête. Ce verrou n'est libéré qu'à la fin de la transaction contenant la commande holdlock.

Utilisation de *read committed*

Si le niveau d'isolement de votre session est 0 et si vous devez lire uniquement des modifications validées, utilisez la clause at isolation level read committed.

Définition de verrous moins contraignants

Contrairement à holdlock, le mot-clé noholdlock empêche Adaptive Server de conserver des verrous acquis lors de l'exécution de cette instruction, quel que soit le niveau d'isolation actuel de la transaction.

noholdlock est utile lorsque des transactions exigent le niveau d'isolement 2 ou 3. Si ces transactions contiennent des requêtes ne nécessitant pas le maintien des verrous partagés jusqu'à la fin de la transaction, vous pouvez spécifier noholdlock pour les requêtes concernées afin d'améliorer la concurrence d'accès du serveur.

Par exemple, si le niveau d'isolement des transactions défini est 3 (dans ce cas, la requête select maintient des verrous partagés jusqu'à la fin de la transaction), la commande suivante libère les verrous à la fin du balayage de la page ou de la ligne :

```
select balance from account noholdlock
    where acct_number < 100
```

Utilisation de *read uncommitted*

Si le niveau d'isolement de la session est 1, 2 ou 3 et si vous voulez lire des données modifiées, vous pouvez utiliser la clause at isolation level read uncommitted.

Utilisation de *shared*

Le mot-clé shared indique à Adaptive Server d'utiliser un verrou partagé (à la place d'un verrou de mise à jour) sur une table ou une vue spécifiée dans un curseur.

Pour plus d'informations, reportez-vous à la section "Utilisation du mot-clé shared", page 271.

Verrouillage readpast

Le verrouillage readpast permet de sélectionner et de lire le texte de requêtes afin d'ignorer silencieusement toutes les lignes ou pages verrouillées avec des verrous incompatibles. Les requêtes ne se bloquent pas, ne se terminent pas et n'envoient pas de messages d'erreur ou d'avertissement à l'utilisateur. Ce type de verrouillage est prévu pour des applications de traitement des files d'attente.

En général, dans ces applications, les requêtes renvoient la première ligne verrouillée qui correspond à la requête. C'est par exemple le cas d'une application de suivi des appels concernant les services de support : la requête doit trouver la ligne associée au dateur le plus ancien, qui ne soit pas verrouillée par un autre membre du support.

Pour plus d'informations sur le verrouillage readpast, reportez-vous au document *Transact-SQL User's Guide.*

Curseurs et verrouillage

Les techniques de verrouillage des curseurs sont semblables aux autres modes de verrouillage d'Adaptive Server. Pour les curseurs déclarés en mode read only ou sans la clause for update, Adaptive Server pose un verrou partagé sur la page de données contenant la position courante du curseur.

Lorsque d'autres lignes sont extraites pour le curseur, Adaptive Server pose un verrou sur la page suivante, le curseur passe sur cette page et le verrou de la page précédente est libéré (sauf si vous utilisez le niveau d'isolement 3).

Dans le cas de curseurs déclarés avec for update, Adaptive Server utilise par défaut des verrous de mise à jour lors du balayage des tables ou des vues référencées avec la clause for update du curseur.

Si la liste for update est vide, toutes les tables et vues référencées dans la clause from de l'instruction select reçoivent des verrous de mise à jour. Un verrou de mise à jour est un verrou spécial de lecture, qui indique que le processus de lecture peut modifier les données d'un moment à l'autre. Un verrou de mise à jour admet d'autres verrous partagés sur la page, mais interdit d'autres verrous de mise à jour et les verrous exclusifs.

Si une ligne est mise à jour ou supprimée via un curseur, la transaction de modification de données pose un verrou exclusif. Tous les verrous exclusifs générés par des mises à jour de curseur dans une transaction sont maintenus jusqu'à la fin de la transaction, même si le curseur est fermé avant.

Cela s'applique également aux verrous partagés ou de mise à jour des curseurs utilisant le mot-clé holdlock ou le niveau d'isolement 3.

Les paragraphes suivants décrivent le fonctionnement du verrouillage pour les curseurs à chaque niveau d'isolement :

- Au niveau 0, Adaptive Server n'utilise pas de verrous sur les pages des tables de la base qui contiennent une ligne représentant la position courante d'un curseur. Les curseurs ne posent pas de verrou de lecture lors de leurs balayages, afin de permettre à d'autres applications d'accéder aux données.

  Cependant, les curseurs utilisant ce niveau d'isolement ne sont pas modifiables et requièrent un index unique sur la table de la base pour garantir l'exactitude.

- Au niveau 1, Adaptive Server utilise des verrous partagés ou de mise à jour sur les pages des tables de la base ou sur les pages d'index de niveau feuille qui contiennent une ligne représentant la position courante d'un curseur.

  La page reste verrouillée jusqu'à ce que le curseur sorte de la page à la suite d'une instruction fetch.

- Au niveau 2 ou 3, Adaptive Server utilise des verrous partagés ou de mise à jour sur les pages des tables de la base ou sur les pages d'index de niveau feuille qui ont été lues au cours d'une transaction à l'aide du curseur.

  Adaptive Server maintient les verrous jusqu'à la fin de la transaction ; il ne libère pas les verrous quand la page de données n'est plus nécessaire ou lorsque l'on ferme le curseur.

Si vous ne définissez pas les options close on endtran ou chained, le curseur reste ouvert après la fin de la transaction et le verrou de page correspondant n'est pas libéré. Il peut également continuer de poser des verrous au fur et à mesure qu'il lit des lignes supplémentaires.

Utilisation du mot-clé *shared*

Lorsque vous déclarez un curseur modifiable à l'aide de la clause for update, vous pouvez indiquer à Adaptive Server d'utiliser des verrous partagés de page (au lieu de verrous de mise à jour de page) dans l'instruction declare cursor :

```
declare nom_curseur cursor
    for select liste_sélection
    from {nom_table | nom_vue} shared
     for update [of liste_nom_colonne]
```

Cela permet à d'autres utilisateurs d'obtenir un verrou de mise à jour sur cette table ou sur une table sous-jacente de la vue.

Vous pouvez utiliser le mot-clé holdlock conjointement au mot-clé shared après chaque nom de table ou de vue, mais holdlock doit précéder shared dans l'instruction select. Exemple :

```
declare authors_crsr cursor
for select au_id, au_lname, au_fname
    from authors holdlock shared
    where state != 'CA'
    for update of au_lname, au_fname
```

Les options holdlock ou shared lors de la définition d'un curseur modifiable ont les conséquences suivantes :

- Si vous omettez les deux options, le curseur maintient un verrou de mise à jour sur la ligne ou la page contenant la ligne courante.

  Les autres utilisateurs ne peuvent pas mettre à jour (que ce soit par le biais d'un curseur ou de tout autre moyen) la ligne désignée par la position du curseur (pour les tables verrouillées au niveau des lignes de données) ou les lignes de cette page (pour les tables ALP et DOL).

  Ils peuvent en revanche déclarer un curseur sur les mêmes tables que celles de votre curseur, et lire les données, mais la pose de verrous exclusif ou de mise à jour sur la ligne ou la page en cours d'utilisation leur est interdite.

- Si vous spécifiez l'option shared, le curseur maintient un verrou partagé sur la ligne ou la page contenant la ligne en cours d'extraction.

  Les autres utilisateurs ne peuvent pas mettre à jour la ligne en cours d'utilisation, ou les lignes de la page, que ce soit par l'intermédiaire d'un curseur ou d'un autre moyen. Ils peuvent cependant lire les lignes de la page.

- Si vous spécifiez l'option holdlock, des verrous de mise à jour sont maintenus sur toutes les lignes ou pages extraites (si vous n'utilisez pas de transaction), ou uniquement sur les pages extraites depuis la dernière validation ou annulation (si vous utilisez une transaction).

  Les autres utilisateurs ne peuvent pas mettre à jour les lignes ou les pages extraites, que ce soit par l'intermédiaire d'un curseur ou d'un autre moyen.

  Ils peuvent en revanche déclarer un curseur sur les mêmes tables que celles de votre curseur, mais ne peuvent pas obtenir de verrou de mise à jour sur les lignes ou les pages extraites.

- Si vous spécifiez les deux options, le curseur maintient des verrous partagés sur toutes les lignes ou les pages extraites (si vous n'utilisez pas de transactions), ou uniquement sur les lignes ou les pages extraites depuis la dernière validation ou annulation.

  Les autres utilisateurs ne peuvent pas mettre à jour les lignes ou les pages extraites, que ce soit par l'intermédiaire d'un curseur ou d'un autre moyen.

Commandes supplémentaires de verrouillage

Commande *lock table*

Dans des transactions, vous pouvez explicitement verrouiller une table avec la commande lock table.

- Pour verrouiller immédiatement la table complète, au lieu d'attendre que la conversion des verrous ait lieu.
- Lorsque la requête ou les transactions utilisent plusieurs balayages dont aucun ne verrouille un nombre suffisant de pages ou de lignes pour déclencher la conversion des verrous, le nombre total de verrous étant toutefois très élevé.

- Lorsque des tables volumineuses, notamment celles qui utilisent un verrouillage des lignes de données, font l'objet d'accès au niveau d'isolement 2 ou 3 et que la conversion des verrous risque d'être bloquée par d'autres tâches. L'utilisation de la commande lock table peut dans ce cas éviter que les verrous viennent à manquer.

Les verrous de table sont libérés à la fin de la transaction.

lock table vous permet de spécifier un délai d'attente. Si le verrou de table ne peut pas être accordé dans le délai fixé, un message d'erreur s'affiche mais la transaction n'est pas annulée.

Pour obtenir un exemple de procédure enregistrée qui utilise des temporisations de verrou et vérifie la présence d'un message d'erreur, reportez-vous à la section lock table dans le *Manuel de référence Adaptive Server*. Si la procédure a été lancée par l'administrateur système, elle continue de s'exécuter et renvoie un message d'erreur aux autres utilisateurs.

Temporisations de verrous

Vous pouvez spécifier le temps qu'une tâche doit attendre un verrou :

- au niveau serveur, à l'aide du paramètre de configuration lock wait period,
- pour une session ou dans une procédure enregistrée, à l'aide de la commande set lock wait,
- Pour une commande lock table.

Pour plus d'informations sur ces commandes, reportez-vous au *Guide de l'utilisateur Transact-SQL*.

Sauf pour la commande lock table, une tâche qui tente de poser un verrou et échoue dans le délai imparti renvoie un message d'erreur, puis la transaction est annulée.

L'utilisation de temporisations est utile lorsque vous supprimez des tâches qui posent des verrous puis qui attendent, bloquant ainsi les autres utilisateurs. Cependant, du fait que les transactions sont annulées et que les utilisateurs peuvent simplement soumettre de nouveau leurs requêtes, la temporisation d'une transaction signifie que la tâche doit se répéter.

Vous pouvez utiliser la procédure sp_sysmon afin de contrôler le nombre de tâches qui dépassent le délai fixé pour l'attente d'un verrou.

Pour plus d'informations, reportez-vous à la section "Gestion des caches par cache", page 1055.

CHAPITRE 12

Informations sur les verrous

Ce chapitre présente les outils qui permettent de générer des rapports sur les verrous et leur fonctionnement.

Sujet	Page
Outils de verrouillage	275
Interblocages et concurrence d'accès	280
Identification des tables pour lesquelles la concurrence d'accès est problématique	287
Informations sur la gestion des verrous	289

Outils de verrouillage

Les procédures système sp_who, sp_lock et sp_familylock fournissent des informations sur les verrous posés par les utilisateurs et signalent les processus bloqués par d'autres transactions.

Informations relatives aux processus bloqués

sp_who fournit des informations sur les processus. Si une commande lancée par un utilisateur est bloquée par des verrous posés par une autre tâche ou un processus de travail, la mention "lock sleep" apparaît dans la colonne status pour signaler que la tâche ou le processus en question attend la libération d'un verrou.

La colonne blk_spid ou block_xloid affiche l'ID de processus de la tâche ou de la transaction qui maintient actuellement les verrous.

Pour obtenir avec sp_who des informations relatives à un utilisateur spécifique d'Adaptive Server, ajoutez le nom de celui-ci. Si vous n'indiquez pas de nom d'utilisateur, le résultat de sp_who concerne tous les processus d'Adaptive Server.

Remarque Dans ce chapitre, le résultat généré par sp_lock ne fait pas état de la colonne class, ceci afin de faciliter la lisibilité des informations. La colonne class indique les noms des curseurs qui maintiennent des verrous ou la mention "Non Cursor Lock".

Affichage des verrous

Pour afficher la liste des verrous courants posés sur Adaptive Server, utilisez la procédure système sp_lock :

```
                        sp_lock
fid spid loid locktype          table_id   page  row dbname   context
--- ---- ---- ---------------- ---------- ----- --- -------- ----------------
  0  15   30  Ex_intent         208003772     0   0 sales    Fam dur
  0  15   30  Ex_page           208003772  2400   0 sales    Fam dur, Ind pg
  0  15   30  Ex_page           208003772  2404   0 sales    Fam dur, Ind pg
  0  15   30  Ex_page-blk       208003772   946   0 sales    Fam dur
  0  30   60  Ex_intent         208003772     0   0 sales    Fam dur
  0  30   60  Ex_page           208003772   997   0 sales    Fam dur
  0  30   60  Ex_page           208003772  2405   0 sales    Fam dur, Ind pg
  0  30   60  Ex_page           208003772  2406   0 sales    Fam dur, Ind pg
  0  35   70  Sh_intent          16003088     0   0 sales    Fam dur
  0  35   70  Sh_page            16003088  1096   0 sales    Fam dur, Inf key
  0  35   70  Sh_page            16003088  3102   0 sales    Fam dur, Range
  0  35   70  Sh_page            16003088  3113   0 sales    Fam dur, Range
  0  35   70  Sh_page            16003088  3365   0 sales    Fam dur, Range
  0  35   70  Sh_page            16003088  3604   0 sales    Fam dur, Range
  0  49   98  Sh_intent         464004684     0   0 master   Fam dur
  0  50  100  Ex_intent         176003658     0   0 stock    Fam dur
  0  50  100  Ex_row            176003658 36773   8 stock    Fam dur
  0  50  100  Ex_intent         208003772     0   0 stock    Fam dur
  0  50  100  Ex_row            208003772 70483   1 stock    Fam dur
  0  50  100  Ex_row            208003772 70483   2 stock    Fam dur
  0  50  100  Ex_row            208003772 70483   3 stock    Fam dur
  0  50  100  Ex_row            208003772 70483   5 stock    Fam dur
  0  50  100  Ex_row            208003772 70483   8 stock    Fam dur
  0  50  100  Ex_row            208003772 70483   9 stock    Fam dur
 32  13   64  Sh_page           240003886 17264   0 stock
 32  16   64  Sh_page           240003886 4376    0 stock
 32  17   64  Sh_page           240003886 7207    0 stock
 32  18   64  Sh_page           240003886 12766   0 stock
 32  18   64  Sh_page           240003886 12767   0 stock
 32  18   64  Sh_page           240003886 12808   0 stock
```

```
32  19   64  Sh_page          240003886 22367   0 stock
32  32   64  Sh_intent         16003088     0   0 stock    Fam dur
32  32   64  Sh_intent         48003202     0   0 stock    Fam dur
32  32   64  Sh_intent         80003316     0   0 stock    Fam dur
32  32   64  Sh_intent        112003430     0   0 stock    Fam dur
32  32   64  Sh_intent        176003658     0   0 stock    Fam dur
32  32   64  Sh_intent        208003772     0   0 stock    Fam dur
32  32   64  Sh_intent        240003886     0   0 stock    Fam dur
```

Cet exemple montre l'état de verrouillage des processus en série et de deux processus en parallèle :

- spid 15 pose un verrou d'intention exclusif sur une table, un verrou de page de données et deux verrous de page d'index. Le suffixe "blk" indique que ce processus en bloque un autre qui a besoin de poser un verrou ; spid 15 bloque un autre processus. Dès que le processus bloquant prend fin, les autres processus remontent dans la liste.
- spid 30 pose un verrou d'intention exclusif sur une table, un verrou sur une page de données et deux verrous sur des pages d'index.
- spid 35 exécute une requête à intervalle au niveau d'isolement 3. Il pose des verrous séquentiels sur plusieurs pages et un verrou de clé infini.
- spid 49 désigne la tâche qui a exécuté sp_lock ; elle pose un verrou d'intention partagé sur la table spt_values dans master pendant l'exécution.
- spid 50 pose des verrous d'intention sur deux tables et plusieurs verrous de ligne.
- fid 32 fait état de plusieurs spid qui posent des verrous : le processus parent (spid 32) pose des verrous d'intention partagés sur 7 tables, tandis que les processus de travail posent des verrous de page partagés sur une des tables.

La colonne locktype indique non seulement si le verrou est partagé (préfixe "Sh"), exclusif (préfixe "Ex"), ou de mise à jour, mais également s'il est posé sur une table ("table" ou "intention") ou sur une "page" ou une "ligne".

Le suffixe "demand" indique que le processus obtiendra un verrou exclusif après la libération de tous les verrous partagés.

Pour plus d'informations sur la demande de verrous, reportez-vous au *Guide d'administration système*.

La colonne context comporte un ou plusieurs des éléments ci-dessous :

- "Fam dur" indique que la tâche maintiendra le verrou jusqu'à la fin de la requête, c'est-à-dire pour la durée d'activité de la famille des processus de travail. Les verrous d'intention partagés sont un exemple de verrous "Fam dur".

  Pour une requête en parallèle, le processus de coordination pose toujours un verrou d'intention partagé sur la table, qui n'est libéré qu'à la fin de la requête parallèle. Si cette dernière fait partie d'une transaction et que des instructions antérieures ont modifié des données, le processus de coordination maintient des verrous pour la durée de la famille sur toutes les pages de données modifiées.

  Les processus de travail peuvent maintenir ce type de verrous lorsque la requête est exécutée au niveau d'isolement 3.

- "Ind pg" indique des verrous sur des pages d'index (tables verrouillées au niveau de toutes les pages uniquement).
- "Inf key" indique un verrou de clé infini, utilisé sur des tables verrouillées au niveau des pages de données seulement pour certaines requêtes à intervalle exécutées au niveau d'isolement 3.
- "Range" indique un verrou séquentiel, utilisé pour certaines requêtes au niveau d'isolement 3.

Pour obtenir des informations de verrouillage sur un login particulier, indiquez le spid du processus :

```
sp_lock 30
```

```
fid spid loid locktype         table_id   page  row dbname   context
--- ---- ---- ---------------- ---------- ----- --- -------- ----------------
  0  30   60  Ex_intent         208003772     0   0 sales    Fam dur
  0  30   60  Ex_page           208003772   997   0 sales    Fam dur
  0  30   60  Ex_page           208003772  2405   0 sales    Fam dur, Ind pg
  0  30   60  Ex_page           208003772  2406   0 sales    Fam dur, Ind pg
```

Si le spid que vous spécifiez est aussi le fid d'une famille de processus, sp_who imprime des informations pour tous les processus.

Vous pouvez également demander des informations relatives aux verrous sur les deux spids :

```
                    sp_lock 30, 15
fid spid loid locktype         table_id   page  row dbname   context
--- ---- ---- ---------------- ---------- ----- --- -------- ----------------
  0  15   30  Ex_intent         208003772     0   0 sales    Fam dur
  0  15   30  Ex_page           208003772  2400   0 sales    Fam dur, Ind pg
  0  15   30  Ex_page           208003772  2404   0 sales    Fam dur, Ind pg
  0  15   30  Ex_page-blk       208003772   946   0 sales    Fam dur
  0  30   60  Ex_intent         208003772     0   0 sales    Fam dur
  0  30   60  Ex_page           208003772   997   0 sales    Fam dur
  0  30   60  Ex_page           208003772  2405   0 sales    Fam dur, Ind pg
  0  30   60  Ex_page           208003772  2406   0 sales    Fam dur, Ind pg
```

Affichage des verrous

sp_familylock affiche la liste des verrous posés par une famille. L'exemple suivant montre que le processus de coordination (fid 51, spid 51) maintient un verrou d'intention partagé sur la table et que chaque processus de travail maintient un verrou de page partagé :

```
                    sp_familylock 51
fid spid loid locktype         table_id    page  row dbname   context
--- ---- ---- ---------------- ----------  ----- --- -------- ----------------
 51  23  102  Sh_page           208003772    945   0 sales
 51  51  102  Sh_intent          16003088      0   0 sales    Fam dur
 51  51  102  Sh_intent          48003202      0   0 sales    Fam dur
 51  51  102  Sh_intent         176003658      0   0 sales    Fam dur
 51  51  102  Sh_intent         208003772      0   0 sales    Fam dur
```

Vous pouvez aussi spécifier deux ID pour sp_familylock.

Blocage dans une famille lors des fusions de buffers réseau

Lorsque de nombreux processus de travail renvoient des résultats de requête, il peut se produire un blocage entre les processus. L'exemple suivant montre cinq processus de travail bloqués par un sixième :

```
                    sp_who 11
fid spid status     loginame origname hostname blk dbname cmd
--- ---- ---------- -------- -------- -------- --- ------ --------------
 11   11 sleeping   diana    diana    olympus  0   sales  SELECT
 11   16 lock sleep diana    diana    olympus  18  sales  PROCESSUS DE TRAVAIL
 11   17 lock sleep diana    diana    olympus  18  sales  PROCESSUS DE TRAVAIL
 11   18 send sleep diana    diana    olympus  0   sales  PROCESSUS DE TRAVAIL
 11   19 lock sleep diana    diana    olympus  18  sales  PROCESSUS DE TRAVAIL
 11   20 lock sleep diana    diana    olympus  18  sales  PROCESSUS DE TRAVAIL
 11   21 lock sleep diana    diana    olympus  18  sales  PROCESSUS DE TRAVAIL
```

Chaque processus de travail pose un verrou exclusif d'adresse sur le buffer réseau pour y consigner des résultats. Lorsque le buffer est plein, il est envoyé au client et le verrou est maintenu jusqu'à ce que l'écriture réseau prenne fin.

Interblocages et concurrence d'accès

Lorsque deux pages de données, deux pages d'index ou deux tables sont verrouillées par deux processus utilisateur, il y a **interblocage** si chaque processus tente de poser un verrou sur l'objet verrouillé par l'autre. Dans ce cas, le premier processus attend que le second libère le verrou, mais le second processus ne le libèrera pas tant que le premier processus n'aura pas libéré son verrou.

Interblocages au niveau serveur et au niveau application

Lorsqu'un interblocage de tâches se produit dans Adaptive Server, un mécanisme permet de le détecter, annule une des transactions et envoie des messages à l'utilisateur ainsi que dans le journal des erreurs Adaptive Server. Il peut se produire des situations d'interblocage au niveau application, dans lesquelles un client ouvre plusieurs connexions qui doivent ensuite attendre la libération de verrous posés par une autre connexion de la même application.

Il ne s'agit pas là de véritables interblocages de niveau serveur et les mécanismes de détection d'Adaptive Server ne les localisent donc pas.

Exemple d'interblocage au niveau application

Certains développeurs simulent les curseurs en utilisant deux connexions ou plus depuis DB-Library™. Une connexion effectue une opération select et l'autre effectue des mises à jour ou des suppressions sur les mêmes tables. Il en résulte des interblocages au niveau application. Exemple :

- La connexion A maintient un verrou partagé sur une page. Tant qu'il existe des lignes en attente d'Adaptive Server, un verrou partagé est maintenu sur la page courante.
- La connexion B requiert un verrou exclusif sur les mêmes pages, puis attend.
- L'application attend que la connexion B aboutisse avant d'appeler le programme nécessaire pour supprimer le verrou partagé. Or, cette connexion ne se produit jamais.

Du fait que la connexion A ne demande jamais le verrou posé par la connexion B, il s'agit bien d'un interblocage au niveau applicatif et non pas associé au serveur.

Interblocages de tâches serveur

Un exemple d'interblocage entre deux processus est illustré ci-dessous.

T19	Déroulement des opérations	T20
`begin transaction`	Lancement de T19 et T20.	`begin transaction`

`update savings`
`set balance = balance - 250`
`where acct_number = 25`
	T19 pose un verrou exclusif sur "savings" tandis que T20 pose un verrou exclusif sur "checking".	`update checking`
		`set balance = balance - 75`
		`where acct_number = 45`

`update checking`
`set balance = balance + 250`
`where acct_number = 45`
	T19 attend que T 20 libère son verrou pendant que T20 attend que T19 libère le sien ; il se produit un interblocage.	`update savings`
		`set balance = balance + 75`
		`where acct_number = 25`

`commit transaction`		`commit transaction`

Si les transactions T19 et T20 s'exécutent simultanément et qu'elles posent des verrous exclusifs avec leurs instructions update, elles sont prises en interblocage, en attendant en vain que l'autre libère son verrou.

Adaptive Server détecte les interblocages et détermine l'utilisateur dont la transaction a consommé le moins de temps CPU.

Adaptive Server annule cette transaction, en avertit l'application en affichant le message 1205 et autorise l'autre processus à continuer.

L'exemple ci-dessus montre deux instructions de modification de données qui créent un interblocage. Les interblocages peuvent également se produire entre un processus qui pose des verrous partagés et un autre qui pose des verrous exclusifs.

En environnement multi-utilisateur, chaque application doit vérifier si les transactions de modification de données n'entraînent pas le message 1205 afin de détecter les éventuels interblocages. Elle indique que la transaction utilisateur risquant un interblocage a été détectée et, par conséquent, annulée. Le programme d'application doit donc la relancer.

Interblocages et requêtes parallèles

Les processus de travail, qui peuvent poser uniquement des verrous partagés, ne sont pas pour autant à l'abri d'interblocages provoqués par des processus ayant posé des verrous exclusifs. Les verrous doivent remplir au moins l'une des conditions suivantes :

- Un processus de coordination maintient un verrou de table exigé par une requête parallèle.

  Il peut par ailleurs maintenir sur d'autres tables des verrous exclusifs demandés par une requête antérieure d'une transaction.

- Une requête parallèle s'exécute au niveau d'isolement des transactions 3, ou avec holdlock et maintient des verrous.

- Une requête parallèle procède à la jointure d'au moins deux tables tandis qu'un autre processus effectue une série de mises à jour sur ces tables dans le cadre d'une transaction.

Un processus de travail unique peut être pris en interblocage, comme c'est le cas entre deux processus en série. Par exemple, un processus de travail effectuant une jointure entre deux tables peut être pris en interblocage avec un processus en série mettant à jour ces tables.

Dans certains cas, les interblocages entre des processus en série et des familles impliquent un niveau d'indirection.

Par exemple, si une tâche maintient un verrou exclusif sur la tableA et a besoin d'un verrou sur la tableB qui est verrouillée par un verrou de type family-duration par un processus de travail, la tâche doit attendre la fin de la transaction dont fait partie le processus.

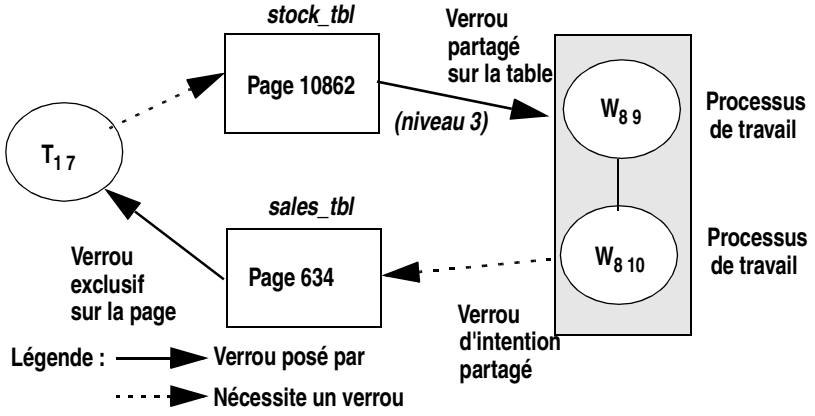
Si un autre processus de travail de la même famille a besoin d'un verrou sur la tableA, un interblocage se produit. figure 12-1 en donne un exemple :

- La famille identifiée par fid 8 exécute une requête parallèle comprenant la jointure entre les tables stock_tbl et sales_tbl, au niveau de transaction 3.
- La tâche en série identifiée par spid 17 (T17) insère des données dans stock_tbl et sales_tbl au sein d'une transaction.

Les opérations aboutissant à l'interblocage sont les suivantes :

- W8 9, processus de travail de fid 8 et de spid 9, maintient un verrou partagé sur la page 10 862 de stock_tbl.
- T17 maintient un verrou exclusif sur la page 634 de sales_tbl. T17 a besoin d'un verrou exclusif sur la page 10 862, qui ne peut être accordé tant que W8 9 n'a pas libéré son verrou partagé.
- Le processus de travail W8 10 requiert un verrou partagé sur la page 634, qui ne peut être accordé tant que T17 n'a pas libéré son verrou exclusif.

Figure 12-1 : Interblocage impliquant une famille de processus de travail



Consignation des informations d'interblocage dans le journal d'erreurs

Les interblocages au niveau serveur sont détectés et signalés à l'application par Adaptive Server et consignés dans le journal d'erreurs du serveur. Le message d'erreur envoyé à l'application correspond à l'erreur 1205.

Le message consigné dans le journal d'erreurs, par défaut, indique uniquement qu'un interblocage s'est produit. La numérotation incluse dans le message indique le nombre d'interblocages depuis le dernier démarrage du serveur.

```
03:00000:00029:1999/03/15 13:16:38.19 server  Deadlock Id 11 detected
```

Dans ce résultat, le processus de fid 0 et de spid 29 a commencé la recherche d'interblocages, c'est pourquoi les valeurs de ses fid et spid figurent comme deuxième et troisième valeurs dans le message d'interblocage. (La première valeur, 03, est le numéro de moteur.)

Pour obtenir plus d'informations sur les tâches en interblocage, définissez le paramètre de configuration print deadlock information à 1. Cette valeur provoquera l'envoi au journal d'erreurs et à la session du terminal sur lequel le serveur a démarré de messages plus détaillés concernant les interblocages.

Toutefois, en attribuant la valeur 1 à print deadlock information, vous risquez de provoquer une détérioration des performances d'Adaptive Server. N'utilisez cette configuration que lorsque vous recherchez les causes d'interblocage.

Les messages d'interblocage contiennent des informations détaillées, indiquant notamment :

- l'ID de famille et l'ID de processus serveur des tâches concernées
- les commandes et les tables impliquées dans les interblocages et, si une procédure stockée est concernée, le nom de celle-ci
- le type de verrou posé par chaque tâche et le type de verrou que chaque tâche a essayé d'acquérir
- les ID de login du serveur (suid).

Dans le rapport suivant, le spid 29 est en interblocage avec une tâche parallèle de fid 94, spid 38. L'interblocage implique des demandes de verrous exclusifs et partagés sur la table authors. Le spid 29 est choisi comme non prioritaire, victime de l'interblocage :

```
Deadlock Id 11: detected. 1 deadlock chain(s) involved.

Deadlock Id 11: Process (Familyid 94, 38) (suid 62) was executing a SELECT
command at line 1.
Deadlock Id 11: Process (Familyid 29, 29) (suid 56) was executing a SELECT
command at line 1.
SQL Text: insert authors (au_id, au_fname, au_lname) values ('A999999816',
'Bill', 'Dewart')

Deadlock Id 11: Process (Familyid 0, Spid 29) was waiting for a 'exclusive page'
lock on page 1155 of the 'authors' table in database 8 but process (Familyid
94, Spid 38) already held a 'shared page' lock on it.
Deadlock Id 11: Process (Familyid 94, Spid 38) was waiting for a 'shared page'
lock on page 2336 of the 'authors' table in database 8 but process (Familyid
29, Spid 29) already held a 'exclusive page' lock on it.
Deadlock Id 11: Process (Familyid 0, 29) was chosen as the victim. End of
deadlock information.
```

Comment éviter les interblocages

Il est possible que des interblocages se produisent lorsque de nombreuses transactions longues sont exécutées en même temps dans une base de données. Les interblocages deviennent plus fréquents à mesure que les conflits de verrous augmentent entre les transactions, ce qui réduit la concurrence d'accès.

Les méthodes permettant de réduire les conflits de verrous (par exemple, éviter d'utiliser des verrous de table ou interdire le maintien de verrous partagés) sont décrites au chapitre 10, "Configuration et optimisation des verrouillages"..

Ordre de verrouillage des objets

Des applications bien conçues peuvent réduire les interblocages en posant des verrous toujours dans le même ordre. Les modifications de plusieurs tables doivent toujours être effectuées dans le même ordre.

Prenons l'exemple fourni par la figure 12-1 : l'interblocage peut être évité si la mise à jour de la table savings ou checking a d'abord eu dans les deux transactions. Ainsi, l'une des transactions obtient le verrou exclusif la première et est traitée, pendant que l'autre attend et reçoit son verrou exclusif sur la même table lorsque la première transaction prend fin.

Pour des applications qui utilisent un grand nombre de tables et des transactions qui mettent à jour plusieurs tables, définissez un ordre de verrouillage commun à tous les développeurs.

Retard du contrôle d'interblocage

Adaptive Server contrôle au bout d'une période minimale tous les processus qui attendent la libération d'un verrou (processus mis en veille) afin de détecter d'éventuels interblocages. Cette opération est très coûteuse pour les applications qui sont en attente sans être prises en interblocage.

Si vos applications subissent rarement des interblocages, Adaptive Server peut espacer cette recherche afin de réduire le surcoût qui en découle. Pour définir le temps (en millisecondes) minimal d'attente des processus avant qu'une recherche d'interblocages soit effectuée, utilisez le paramètre de configuration deadlock checking period.

Les valeurs admises sont comprises entre 0 et 2 147 483, la valeur par défaut étant 500. deadlock checking period est un paramètre dynamique, c'est-à-dire que la valeur définie est immédiatement prise en considération.

Si vous définissez la valeur 0, Adaptive Server lance la recherche d'interblocages dès que le processus commence à attendre un verrou. Si vous choisissez la valeur 600, Adaptive Server lance la recherche lorsque le processus a attendu au moins 600 ms. Voici un exemple de syntaxe :

```
sp_configure "deadlock checking period", 600
```

Plus la valeur de deadlock checking period est élevée, plus le processus attendra avant que les éventuels interblocages ne soient détectés. Cependant, comme Adaptive Server accorde la plupart des verrous demandés avant que le temps d'attente imparti ne se soit écoulé, les processus concernés ne subissent pas le surcoût généré par la recherche d'interblocages.

Pour détecter d'éventuels interblocages, Adaptive Server contrôle tous les processus à intervalles réguliers, en fonction de la valeur de deadlock checking period. Si Adaptive Server lance une recherche alors que le délai d'attente a été modifié, le processus doit attendre l'intervalle suivant.

Ainsi, un processus peut attendre pendant un délai égal à deux fois la valeur de deadlock checking period avant que la recherche d'interblocages soit effectuée. La procédure sp_sysmon vous aide à optimiser ce délai.

Pour plus d'informations, reportez-vous à la section "Détection des interblocages", page 1043.

Identification des tables pour lesquelles la concurrence d'accès est problématique

sp_object_stats imprime des informations au niveau table sur les conflits de verrouillage. Vous pouvez l'utiliser pour :

- obtenir des informations sur toutes les tables dont les conflits de verrouillage sont nombreux
- obtenir des informations relatives aux conflits sur les tables dans une base de données
- obtenir des informations relatives aux conflits sur des tables seules.

La syntaxe est la suivante :

sp_object_stats *interval* [, *top_n*
[, *nom_base* [, *nom_obj* [, *option_rpt*]]]]

Pour évaluer les conflits de verrous sur toutes les tables de toutes les bases de données, spécifiez un seul intervalle. L'exemple suivant contrôle les conflits pendant 20 minutes et fournit des statistiques sur les 10 tables qui présentent le plus de conflits de verrous :

```
sp_object_stats "00:20:00"
```

Les arguments possibles de sp_object_stats sont les suivants :

- *top_n* : permet de spécifier le nombre de tables à inclure dans le rapport. La valeur par défaut est 10. Pour inclure les 20 premières tables comportant le plus de conflits, utilisez par exemple :

  ```
  sp_object_stats "00:20:00", 20
  ```

- *nom_base* : imprime des statistiques sur la base de données spécifiée.
- *nom_objet* : mesure les conflits pour la table spécifiée.
- *option_rpt* : spécifie le type de rapport :
  - rpt_locks signale les octrois (grants), les attentes, les interblocages et les délais d'attente pour les tables à haut risque de conflit. rpt_locks est l'option par défaut.
  - rpt_objlist fournit uniquement les noms des objets ayant le plus haut niveau d'activité de verrouillage.

Voici un exemple de résultat pour la table titles, qui utilise un verrouillage au niveau des pages de données :

```
Object Name: pubtune..titles (dbid=7, objid=208003772,lockscheme=Datapages)


  Page Locks     SH_PAGE               UP_PAGE               EX_PAGE
  ----------    ----------            ----------            ----------
  Grants:            94488                  4052                  4828
  Waits:               532                   500                   776
  Deadlocks:             4                     0                    24
  Wait-time:      20603764 ms           14265708 ms           2831556 ms
  Contention:         0.56%                 10.98%                13.79%

 *** Consider altering pubtune..titles to Datarows locking.
```

Le tableau 12-1 explique les valeurs.

Tableau 12-1 : Résultat de sp_object_stats

Ligne de résultat	Valeur
Grants	Nombre de fois que le verrou a été octroyé immédiatement.
Waits	Nombre de fois que la tâche ayant besoin d'un verrou a dû attendre.
Deadlocks	Nombre d'interblocages qui se sont produits.
Wait-times	Temps total, exprimé en millisecondes, que toutes les tâches ont passé à attendre un verrou.
Contention	Pourcentage exprimant le temps passé par une tâche à attendre ou le temps passé en interblocage.

sp_object_stats recommande de changer le plan de verrouillage lorsque le nombre total de conflits sur une table dépasse 15 %, comme suit :

- Si la table utilise un verrouillage de toutes les pages, la procédure vous conseille de passer à un verrouillage des pages de données.
- Si la table utilise un verrouillage des pages de données, elle vous conseille de passer à un verrouillage des lignes de données.

Informations sur la gestion des verrous

Le résultat généré par sp_sysmon fournit des statistiques sur les verrouillages et les interblocages traités dans ce chapitre.

Ces statistiques permettent de déterminer si les conflits de verrouillage ont une incidence sur les performances d'Adaptive Server.

Pour plus d'informations sur la procédure sp_sysmon et les statistiques relatives aux verrous, reportez-vous à la section "Gestion des verrous", page 1036.

Pour déterminer avec plus de précision les problèmes de verrouillage, utilisez Adaptive Server Monitor.

CHAPITRE 13

Définition des propriétés de gestion de l'espace

Définir des propriétés de gestion de l'espace contribue à réduire le travail de maintenance nécessaire pour garantir des performances optimales au niveau des tables et des index.

Sujet	Page
Réduction des opérations de maintenance des index	291
Réduction de la redirection de lignes	298
Réservation d'espace pour les lignes redirigées et les insertions	305
Utilisation de max_rows_per_page dans des tables APL	314

Réduction des opérations de maintenance des index

Par défaut, Adaptive Server crée des index dont les pages sont pleines au niveau feuille mais laissent de l'espace libre pour deux lignes sur les pages intermédiaires.

L'option fillfactor de la commande create index permet de spécifier le degré de remplissage des pages d'index et des pages de données des index clusterisés. Lorsque vous utilisez fillfactor, les lignes de données et d'index occupent davantage d'espace disque que celui qui est défini par défaut (sauf lorsque le facteur de remplissage est de 100 pour cent).

Si vous créez des index pour des tables destinées à s'accroître, vous pouvez réduire l'incidence des divisions de pages sur les tables et les index en utilisant l'option fillfactor de la commande create index.

L'option fillfactor n'est appliquée qu'au moment de la création de l'index ; elle ne reste pas en vigueur au-delà de cette opération.

Lorsque vous exécutez create index, la valeur de fillfactor spécifiée dans la commande est appliquée comme suit :

- Index clustérisé :
  - sur une table verrouillée au niveau de toutes les pages (table APL), fillfactor est appliqué aux pages de données,
  - sur une table verrouillée au niveau des pages de données seulement (table DOL), fillfactor est appliqué aux pages feuille de l'index et les pages de données sont entièrement remplies (à moins qu'une procédure sp_chgattribute n'ait servi à enregistrer un fillfactor pour la table).
- Index non clusterisé – la valeur de fillfactor est appliquée aux pages de niveau feuille de l'index.

Les valeurs de fillfactor spécifiées avec create index sont appliquées au moment où l'index est créé. Elles ne sont pas enregistrées dans les tables sysindexes et, à mesure que le temps passe, le remplissage complet des pages de données ou d'index n'est pas maintenu.

Vous pouvez aussi utiliser sp_chgattribute pour stocker des valeurs de fillfactor qui seront utilisées lors de l'exécution de reorg rebuild sur une table.

Pour plus d'informations, reportez-vous à la section "Définition des valeurs de fillfactor", page 294.

Avantages de *fillfactor*

Vous pouvez temporairement améliorer les performances en attribuant une valeur faible à fillfactor. Toutefois, cet avantage diminue au fur et à mesure que les insertions dans la base de données accroissent la quantité d'espace utilisée sur les pages de données ou d'index.

Une réduction du facteur de remplissage apporte les avantages suivants :

- il réduit les divisions de pages du niveau feuille dans les index, ainsi que des pages de données dans les tables APL ;
- il améliore la clusterisation des lignes de données dans les tables DOL disposant d'index clusterisés et dans lesquelles sont effectuées de nombreuses insertions ;
- il limite les conflits de verrous sur les tables verrouillées au niveau des pages, car il réduit la probabilité que deux processus exigent simultanément les mêmes pages de données ou d'index ;

- il contribue à préserver l'efficacité des E/S étendues pour les pages de données et pour les niveaux feuille des index non clusterisés, car les divisions de page sont moins fréquentes. Il est donc plus probable que les huit pages d'un extent se suivent.

Inconvénients de *fillfactor*

Si vous utilisez fillfactor avec une valeur particulièrement basse, vous risquez de constater les effets suivants sur les requêtes et sur les opérations de maintenance :

- Le nombre de pages à lire pour chaque requête est plus important qu'avec un balayage de table ou un balayage des pages de niveau feuille sur un index non clusterisé.

  Dans certains cas, cela peut aussi entraîner l'ajout d'un niveau à la structure de l'arbre binaire de l'index, puisque le nombre de pages du niveau de données et, parfois, de chaque niveau d'index, augmente.

- Les commandes dbcc doivent vérifier davantage de pages, de sorte que la vérification dure plus longtemps.

- La durée d'exécution de dump database s'accroît, en raison de l'accroissement du nombre de pages à sauvegarder. dump database copie toutes les pages de données, mais ne sauvegarde pas celles qui ne sont pas encore utilisées.

  Les sauvegardes et les chargements durent plus longtemps et utilisent plus de bandes.

- L'efficacité des facteurs de remplissage diminue avec le temps. Si vous utilisez l'option fillfactor pour réduire l'impact des divisions de pages sur les performances, vous devez contrôler votre système et recréer les index lorsque les divisions de pages commencent à influer sur les performances.

Définition des valeurs de *fillfactor*

sp_chgattribute permet d'enregistrer un pourcentage de fillfactor pour chaque index et pour la table. La valeur de fillfactor que vous définissez avec sp_chgattribute est appliquée :

- lorsque vous exécutez reorg rebuild pour rétablir les taux de clusterisation des tables DOL et de leurs index,
- lorsque vous utilisez alter table...lock pour modifier le plan de verrouillage d'une table ou que vous utilisez une commande alter table...add/modify nécessitant la copie de la table,
- lorsque vous exécutez create clustered index et qu'il existe une valeur enregistrée pour la table.

Le fillfactor enregistré n'est pas appliqué lorsque des index non clusterisés sont reconstruits après une commande create clustered index :

- Si une valeur de fillfactor est spécifiée avec create clustered index, elle s'applique à chaque index non clusterisé.
- Si aucune valeur fillfactor n'est spécifiée avec create clustered index, la valeur par défaut définie au niveau serveur par le paramètre de configuration default fill factor percent s'applique à tous les index.

exemples de *fillfactor*

Les exemples suivants illustrent l'application de différentes valeurs de fillfactor.

Absence de valeurs de *fillfactor* enregistrées

Si aucune valeur de fillfactor n'est enregistrée dans sysindexes, le fillfactor spécifié dans les commandes "create index" s'applique, comme l'indique le tableau 13-1.

```
create clustered index title_id_ix
on titles (title_id)
with fillfactor = 80
```

Tableau 13-1 : valeurs de fillfactor appliquées si aucune valeur n'est enregistrée pour la table

Commande	Table APL	Table DOL
create clustered Index	Pages de données : 80	Pages de données : entièrement remplies Pages de niveau feuille : 80
Reconstructions d'index non clusterisés	Pages de niveau feuille : 80	Pages de niveau feuille : 80

Les index non clusterisés utilisent le fillfactor spécifié dans la commande create clustered index.

Si fillfactor n'est pas spécifié dans create clustered index, les index non clusterisés utilisent toujours la valeur par défaut définie au niveau du serveur ; ils n'utilisent jamais une valeur venant de sysindexes.

Valeurs utilisées pour *alter table...lock* et *reorg rebuild*

Lorsqu'aucune valeur de fillfactor n'est enregistrée, les commandes alter table...lock et reorg rebuild appliquent la valeur définie pour le serveur par le paramètre de configuration default fill factor percentage. Le fillfactor par défaut est appliqué comme le montre le tableau 13-2.

Tableau 13-2 : valeurs de fillfactor appliquées lors des reconstructions

Commande	Table APL	Table DOL
Régénération d'index clusterisé	Pages de données : valeur par défaut	Pages de données : entièrement remplies Pages de niveau feuille : valeur par défaut
Reconstructions d'index non clusterisés	Pages de niveau feuille : par défaut	Pages de niveau feuille : par défaut

Valeur de *fillfactor* enregistrée au niveau table ou pour un index clusterisé

Cette commande enregistre une valeur de fillfactor égale à 50 pour la table :

```
sp_chgattribute titles, "fillfactor", 50
```

Avec une valeur de fillfactor égale à 50 enregistrée au niveau table, la commande suivante create clustered index applique les valeurs de fillfactor comme indiqué dans le tableau 13-3.

```
create clustered index title_id_ix
on titles (title_id)
with fillfactor = 80
```

Tableau 13-3 : Utilisation des valeurs enregistrées de fillfactor pour des index clusterisés

Commande	Table APL	Table DOL
create clustered index	Pages de données : 80	Pages de données : 50 Pages de niveau feuille : 80
Reconstructions d'index non clusterisés	Pages de niveau feuille : 80	Pages de niveau feuille : 80

Remarque Lorsque vous exécutez create clustered index, toute valeur de fillfactor enregistrée pour la table dans sysindexes est remise à 0.

Pour avoir un effet sur le remplissage des tables DOL lors d'une commande create clustered index ou reorg, vous devez d'abord exécuter sp_chgattribute.

Effets de *alter table...lock* lorsque des valeurs sont enregistrées

Les valeurs enregistrées de fillfactor sont utilisées lorsqu'une commande alter table...lock copie des tables et reconstruit les index.

Tables avec index clusterisés

En mode verrouillage APL, la table et l'index clusterisé partagent la ligne sysindexes, de sorte qu'une seule valeur de fillfactor peut être enregistrée et utilisée pour la table et l'index clusterisé. Vous pouvez définir la valeur de fillfactor pour les pages de données en spécifiant le nom de la table ou le nom de l'index clusterisé. La commande suivante enregistre la valeur 50 :

```
sp_chgattribute titles, "fillfactor", 50
```

La commande ci-dessous enregistre la valeur 80 et remplace la valeur 50 précédente :

```
sp_chgattribute "titles.clust_ix", "fillfactor", 80
```

Si, sur la table titles, vous passez à un verrouillage DOL après exécution des commandes sp_chgattribute ci-dessus, la valeur enregistrée 80 de fillfactor est utilisée pour les pages de données et les pages feuille de l'index clusterisé.

Dans une table DOL, les informations sur l'index clusterisé sont stockées dans une ligne distincte, dans sysindexes. La valeur fillfactor que vous spécifiez pour la table s'applique aux pages de données et la valeur fillfactor que vous spécifiez pour l'index clusterisé s'applique au niveau feuille de l'index clusterisé.

Lorsqu'une table DOL est modifiée de façon à utiliser un verrouillage de toutes les pages (APL), la valeur de fillfactor enregistrée pour la table est utilisée pour les pages de données. Le fillfactor enregistré pour l'index clusterisé est ignoré.

Le tableau 13-4 affiche les facteurs de remplissage utilisés sur les pages d'index et de données par une commande alter table...lock exécutée après les commandes sp_chgattribute ci-dessus.

Tableau 13-4 : Effets des valeurs enregistrées de fillfactor durant l'exécution de alter table

alter table...lock	Index non clusterisé	Index clustérisé
Passage d'un verrouillage APL à un verrouillage DOL	Pages de données : 80	Pages de données : 80 Pages de niveau feuille : 80
Passage d'un verrouillage DOL à un verrouillage APL	Pages de données : 80	Pages de données : 80

Remarque alter table...lock définit toutes les valeurs enregistrées de fillfactor pour une table à 0.

valeurs enregistrées de *fillfactor* pour des index non clusterisés

Chaque index non clusterisé est représenté par une ligne distincte de sysindexes. Les commandes ci-dessous enregistrent des valeurs différentes pour deux index non clusterisés :

```
sp_chgattribute "titles.ncl_ix", "fillfactor", 90
sp_chgattribute "titles.pubid_ix", "fillfactor", 75
```

Le tableau 13-5 montre les effets d'une commande reorg rebuild sur une table DOL lorsque les commandes sp_chgattribute ci-dessus sont utilisées pour enregistrer des valeurs de fillfactor.

Tableau 13-5 : Effets des valeurs enregistrées de fillfactor durant l'exécution de reorg rebuild

reorg rebuild	Index non clusterisé	Index clustérisé	index non clusterisés
Table DOL	Pages de données : 80	Pages de données : 50 Pages de niveau feuille : 80	Pages de niveau feuille de ncl_ix : 90 Pages de niveau feuille de pubid_ix : 75

Utilisation des options *sorted_data* et *fillfactor*

L'option sorted_data de la commande create index est utilisée lorsque les données à trier sont déjà dans l'ordre de la clé d'index. Ceci permet à la commande create clustered index d'ignorer la phase de copie lors de la création d'un index clusterisé.

Par exemple, si des données copiées par l'utilitaire 'bcp' dans une table sont déjà classées dans l'ordre de la clé d'index clusterisé, la création d'un index avec l'option sorted_data se déroule sans tri. S'il n'est pas nécessaire de copier les données dans de nouvelles pages, l'option fillfactor n'est pas appliquée. D'autres options de create index peuvent toutefois nécessiter la phase de copie.

Pour plus d'informations, reportez-vous à la section "Création d'un index sur des données triées", page 391.

Réduction de la redirection de lignes

Le fait de spécifier une taille de ligne prévue pour une table DOL est utile lorsque l'application admet des lignes contenant des valeurs nulles ou l'insertion de champs courts de type caractère et de longueur variable, et que la taille de ces lignes augmente au fil des mises à jour. La définition d'une taille de ligne prévue a pour objectif de réduire la redirection des lignes vers d'autres pages.

Par exemple, la table titles de la base pubs2 contient de nombreuses colonnes *varchar* et des colonnes autorisant les valeurs NULL. La taille de ligne maximale pour cette table est de 331 octets et la taille moyenne (indiquée par optdiag) est de 184 octets, mais il est possible d'insérer une ligne de moins de 40 octets puisque de nombreuses colonnes admettent les valeurs NULL. Dans une table DOL, l'insertion de lignes courtes puis leur mise à jour peuvent entraîner une redirection des lignes.

Pour plus d'informations, reportez-vous aux "Tables DOL sans index", page 170.

Vous pouvez définir la taille de ligne prévue pour les tables contenant des colonnes de longueur variable en utilisant :

- le paramètre exp_row_size dans une instruction create table ;
- sp_chgattribute, pour une table existante ;
- une valeur par défaut définie au niveau serveur par le paramètre de configuration default exp_row_size percent. Cette valeur est appliquée à toutes les tables contenant des colonnes de longueur variable, à moins que la commande create table ou sp_chgattribute ne soit utilisée pour définir explicitement une taille de ligne ou pour indiquer que les lignes des pages de données doivent être entièrement remplies.

Si vous spécifiez une taille de ligne prévue pour une table APL, la valeur est enregistrée dans sysindexes, mais elle n'est pas appliquée lors des insertions et des mises à jour.

Si, par la suite, la table passe à un verrouillage DOL, la valeur de exp_row_size est appliquée durant la conversion et à toutes les insertions et mises à jour suivantes.

Valeurs par défaut, minimale et maximale de *exp_row_size*

Le tableau 13-6 répertorie les valeurs minimale et maximale de la taille de ligne prévue, ainsi que les valeurs spéciales 0 et 1 et leur signification.

Tableau 13-6 : Valeurs possibles pour la taille de ligne prévue

Valeurs de exp_row_size	Valeurs minimale, maximale et spéciale
Minimale	La plus grande valeur entre : • 2 octets • La somme de toutes les colonnes de longueur fixe
Maximale	Longueur de ligne de données maximale
0	La valeur par défaut définie pour le serveur est utilisée
1	Remplissage complet de toutes les pages ; aucun espace n'est réservé en vue de l'accroissement des lignes

Vous ne pouvez pas spécifier une taille de ligne prévue pour les tables qui ne contiennent que des colonnes de longueur fixe. Les colonnes qui admettent les valeurs NULL sont par définition des colonnes de longueur variable puisque leur longueur est nulle lorsqu'elles contiennent une valeur NULL.

Valeur par défaut

Si vous ne spécifiez pas de taille de ligne prévue ou si vous spécifiez la valeur 0 lorsque vous créez une table DOL contenant des colonnes de longueur variable, Adaptive Server utilise l'espace indiqué par la paramètre de configuration default exp_row_size percent pour toutes les tables contenant des colonnes de longueur variable.

Pour plus d'informations sur ce paramètre et l'allocation d'espace aux pages de données, reportez-vous à la section "Définition d'une taille de ligne prévue par défaut au niveau serveur", page 301. Utilisez sp_help pour connaître la longueur définie des colonnes de la table.

Spécification d'une taille de ligne prévue avec *create table*

L'instruction create table ci-dessous spécifie une taille de ligne prévue de 200 octets :

```
create table new_titles (
      title_id    tid,
      title       varchar(80) not null,
      type        char(12),
      pub_id      char(4) null,
      price       money null,
      advance     money null,
      total_sales int null,
      notes       varchar(200) null,
      pubdate     datetime,
      contract    bit                              )
lock datapages
with exp_row_size = 200
```

Ajout ou modification d'une taille de ligne prévue

Pour ajouter ou modifier la taille de ligne prévue pour une table, utilisez sp_chgattribute. La commande ci-dessous définit la taille de ligne prévue à 190 pour la table new_titles :

```
sp_chgattribute new_titles, "exp_row_size", 190
```

Si vous souhaitez appliquer à une table la valeur default exp_row_size percent à la place d'une valeur explicite en cours, entrez :

```
sp_chgattribute new_titles, "exp_row_size", 0
```

Pour un remplissage complet des pages sans réservation de pages en vue d'un accroissement des lignes, définissez la valeur à 1.

La modification de la taille de ligne prévue avec sp_chgattribute n'a pas de répercussion immédiate sur le stockage des données existantes. La nouvelle valeur s'applique :

- Lorsqu'un index clusterisé sur la table est créée ou que la commande reorg rebuild est exécutée sur la table. La taille de ligne prévue est appliquée lorsque des lignes sont copiées dans de nouvelles pages de données.

  Si vous augmentez exp_row_size et si vous recréez l'index clusterisé ou exécutez la commande reorg rebuild, la nouvelle copie de la table risque de nécessiter davantage de place.

- Lorsque des modifications de données seront appliquées dans une page.

Définition d'une taille de ligne prévue par défaut au niveau serveur

default exp_row_size percent réserve un pourcentage de la taille de page pour des mises à jour ultérieures. La valeur par défaut, 5, fixe à 5 % l'espace disponible par page de données, pour toutes les tables DOL contenant des colonnes de longueur variable. Compte tenu des 2002 octets disponibles sur les pages de données d'une table DOL, la valeur par défaut réserve 100 octets pour un accroissement possible des lignes. La commande ci-dessous définit la valeur par défaut à 10 % :

```
sp_configure "default exp_row_size percent", 10
```

La définition de default exp_row_size percent à 0 indique qu'aucun espace n'est réservé en vue d'une possible augmentation des lignes dans des tables dont la taille de ligne prévue n'est pas explicitement définie avec create table ou sp_chgattribute.

Si une taille de ligne prévue est spécifiée avec create table ou sp_chgattribute, cette valeur a la priorité sur la valeur définie au niveau serveur.

Affichage de la taille de ligne prévue pour une table

Utilisez sp_help pour afficher la taille de ligne prévue pour une table :

```
sp_help titles
```

Si la valeur est égale à 0 et si la table contient des colonnes de longueur variable ou admettant les valeurs NULL, utilisez sp_configure afin de connaître la valeur par défaut définie pour le serveur :

```
sp_configure "default exp_row_size percent"
```

La requête ci-dessous affiche la valeur de la colonne exp_rowsize pour toutes les tables utilisateur d'une base de données :

```
select object_name(id), exp_rowsize
from sysindexes
where id > 100 and (indid = 0 or indid = 1)
```

Choix d'une taille de ligne prévue pour une table

La définition d'une taille de ligne prévue aide à limiter le nombre de lignes redirigées uniquement si les lignes s'accroissent après leur première insertion dans la table. Pour une définition correcte, il faut que :

- votre application ne génère qu'un petit pourcentage de lignes redirigées,
- vous ne perdiez pas trop d'espace sur les pages de données en raison d'une taille de ligne prévue trop grande.

Utilisation de *optdiag* pour vérifier les lignes redirigées

Pour les tables contenant déjà des données, utilisez optdiag afin d'afficher des statistiques les concernant. La ligne "Data row size" indique la taille moyenne de ligne de données, y compris le surcoût de ligne. L'exemple suivant de résultat optdiag pour la table titles indique 12 lignes redirigées et une taille moyenne de ligne de données de 184 octets :

```
Statistiques de la table :           "titles"

     Nombre de pages de données :    655
     Nombre de pages vides :         5
     Nombre de lignes de données :   4959.000000000
     Nombre de lignes redirigées :     12.000000000
     Nombre de lignes supprimées :     84.000000000
     Nombre de CR de pages :            0.000000000
     OAM + nombre d'allocations
     de page :                          6
     Pages en extent d'allocation :     1
     Taille des lignes de données :   184.000000000
```

Vous pouvez aussi utiliser optdiag pour vérifier le nombre de lignes redirigées dans une table et déterminer si la valeur existante de exp_row_size réduit le nombre des lignes redirigées générées par vos applications.

Pour plus d'informations sur optdiag, reportez-vous au chapitre 36, "Affichage des tables de statistiques avec optdiag".

Consultation de *systabstats* pour vérifier le nombre de lignes redirigées

Pour connaître le nombre de lignes redirigées dans une table, vous pouvez vérifier le contenu de la colonne forwrowcnt de la table systabstats. Cette requête vérifie le nombre de lignes redirigées pour toutes les tables utilisateur (dont les ID d'objet sont supérieurs à 100) :

```
select object_name(id) , forwrowcnt
from systabstats
where id > 100 and (indid = 0 or indid = 1)
```

Remarque Le nombre des lignes redirigées est mis à jour en mémoire et le processus housekeeper le copie régulièrement sur disque.

Pour consulter la table systabstats avec SQL, utilisez au préalable la procédure sp_flushstats afin de disposer des statistiques les plus récentes. optdiag écrit les statistiques sur disque avant d'afficher les valeurs.

Conversion de *max_rows_per_page* en *exp_row_size*

Si une valeur max_rows_per_page est définie pour une table APL, elle sert à calculer une taille de ligne prévue durant l'exécution de la commande alter table...lock. La formule est décrite dans le tableau 13-7.

Tableau 13-7 : Conversion de max_rows_per_page en exp_row_size

Valeur de max_rows_per_page	Valeur de exp_row_size
0	Une valeur en pourcentage définie par default exp_row_size percent
255	1 (pages entièrement remplies)
1-254	La valeur la plus petite entre : • Taille de ligne maximale • 2002/valeur de max_rows_per_page

Par exemple, si max_rows_per_page est défini à 10 pour une table verrouillée APL avec une taille de ligne maximale définie à 300 octets, la valeur de exp_row_size sera de 200 (2002/10) après le passage à un verrouillage DOL.

Si max_rows_per_page est défini à 10 alors que la taille de ligne maximale définie n'est que de 150, la taille de ligne prévue sera fixée à 150.

Contrôle et gestion des tables qui utilisent la taille de ligne prévue

Après avoir défini une taille de ligne prévue pour une table, utilisez optdiag ou des requêtes sur systabstats pour vérifier le nombre des lignes redirigées que vos applications continuent de générer. Exécutez reorg forwarded_rows si vous estimez que le nombre de lignes redirigées est trop élevé et qu'il influe sur les performances. reorg forwarded_rows utilise des transactions courtes et interfère très peu avec d'autres activités, de sorte que vous pouvez l'exécuter parallèlement aux applications.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Si l'application génère un grand nombre de lignes redirigées, vous pouvez augmenter la taille de ligne prévue pour la table en utilisant sp_chgattribute.

Toutefois, un certain pourcentage de redirection est tolérable. Si l'exécution de reorg pour supprimer les lignes redirigées ne cause aucun problème au niveau des applications, ou si vous pouvez exécuter reorg à des moments de faible activité, vous pouvez autoriser un petit pourcentage de renvoi sans que cela provoque de graves problèmes de performances.

La définition d'une taille de ligne prévue augmente l'espace de stockage et le nombre des E/S nécessaires pour lire un groupe de lignes. Si l'augmentation du nombre d'E/S liée à l'augmentation de l'espace de stockage est importante, le fait de tolérer la redirection des lignes et, éventuellement, d'exécuter reorg peut avoir une incidence moindre sur les performances.

Réservation d'espace pour les lignes redirigées et les insertions

La définition d'une valeur reservepagegap peut, sur certaines tables, réduire la fréquence des activités de maintenance, telles que l'exécution de reorg rebuild et la recréation d'index, visant à maintenir des performances optimales. Sur les tables verrouillées au niveau des données, l'optimisation des performances nécessite une bonne clusterisation des données sur les pages, les extents et les unités d'allocation utilisées.

La clusterisation des pages de données et d'index sur le support physique est élevée tant que les zones utilisées pour le stockage des lignes redirigées et pour les lignes insérées dans l'ordre de la clé d'index restent contiguës. La propriété de gestion de l'espace reservepagegap permet de réserver des pages vides lorsque l'accroissement de la table exige l'allocation de nouvelles pages.

Immédiatement après la création d'un index clusterisé sur la table ou après l'exécution de la commande reorg rebuild, les taux de clusterisation des pages et des lignes sont généralement de 1.0, ou très proches de 1.0. Cependant, les futures modifications de données peuvent entraîner la redirection de lignes et nécessiter l'allocation de pages de données et d'index supplémentaires pour le stockage des insertions.

La définition d'un intervalle de page réservée permet de réduire la fragmentation du stockage ainsi que la fréquence de reconstruction des index ou d'exécution de la commande reorg rebuild sur la table.

Opérations d'allocation d'extent et *reservepagegap*

Les commandes qui allouent plusieurs pages de données effectuent une **allocation d'extent** qui attribue huit pages à la fois au lieu d'une seule page chaque fois que nécessaire. L'allocation d'extent réduit les opérations de journalisation puisqu'un seul enregistrement est créé au lieu de huit.

Les commandes qui procèdent à une allocation d'extent sont les suivantes : select into, create index, reorg rebuild, bcp, alter table...lock, ainsi que les options de contrainte alter table...unique et primary key (car elles créent des index). De plus, les commandes alter table qui ajoutent, suppriment ou modifient des colonnes exigent parfois une opération de copie de table. Toutes ces commandes allouent un extent et, sauf si un intervalle de page réservée est en vigueur, remplissent les huit pages.

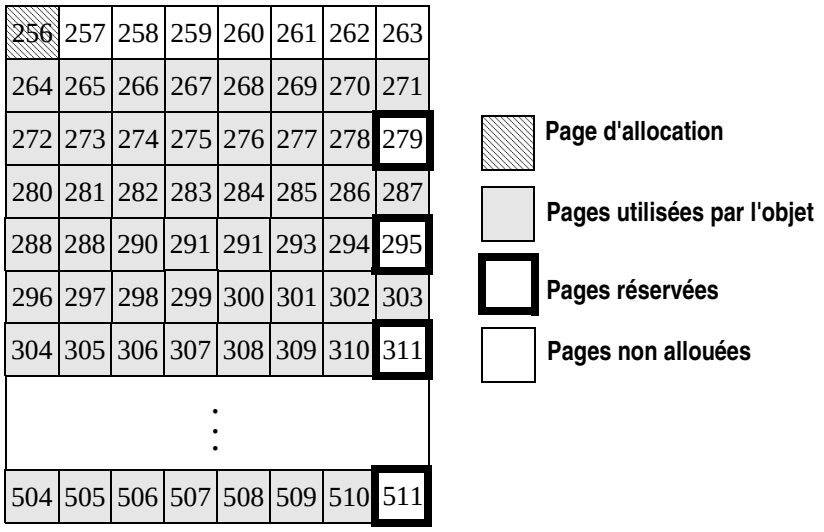
Vous pouvez spécifier la valeur de reservepagegap en nombre de pages, en indiquant un taux de pages vides par rapport aux pages remplies. Par exemple, si vous spécifiez la valeur 8 pour reservepagegap, une opération qui alloue un extent va remplir 7 pages et laisser la huitième vide.

Ces pages vides sont utilisables pour le stockage des lignes redirigées et pour le maintien de la clusterisation des lignes de données dans l'ordre de la clé d'index, ceci sur des tables DOL contenant des index clusterisés.

Etant donné que les opérations d'allocation d'extent doivent allouer des extents entiers, la première page de chaque unité d'allocation n'est pas utilisée car elle contient la page d'allocation. Par exemple, si vous créez un index clusterisé sur une table volumineuse sans spécifier d'intervalle de page réservée, chaque unité d'allocation va contenir 7 pages vides non allouées, 248 pages utilisées et la page d'allocation. Ces 7 pages serviront à stocker les lignes redirigées et les insertions, ce qui permet de garder ces données, avec les index clusterisés, sur la même unité d'allocation. L'utilisation de la propriété reservepagegap permet donc de laisser des pages vides supplémentaires sur chaque unité d'allocation.

La figure 13-1 montre l'aspect d'une unité d'allocation après création d'un index clusterisé avec la propriété reservepagegap définie à 16 sur la table. Les pages qui partagent le premier extent avec l'unité d'allocation ne sont pas utilisées et ne sont pas allouées à la table. Les pages 279, 295 et 311 sont les pages inutilisées sur des extents alloués à la table.

Figure 13-1 : Pages réservées après création d'un index clusterisé



Spécification d'un intervalle de page réservée avec *create table*

La commande suivante create table spécifie la propriété reservepagegap avec une valeur de 16 :

```
create table more_titles (
      title_id    tid,
      title       varchar(80) not null,
      type        char(12),
      pub_id      char(4) null,
      price       money null,
      advance     money null,
      total_sales int null,
      notes       varchar(200) null,
      pubdate     datetime,
      contract    bit
    )
lock datarows
with reservepagegap = 16
```

Toute opération qui effectue une allocation d'extent sur la table more_titles laisse une page vide pour 15 pages remplies.

La valeur par défaut de reservepagegap est 0, indiquant qu'aucun espace n'est réservé. La valeur maximale est 255, indiquant qu'une page reste inutilisée sur chaque unité d'allocation.

Spécification d'un intervalle de page réservée avec *create index*

La commande suivante spécifie la propriété reservepagegap avec la valeur 10 pour les pages d'index non clusterisé :

```
create index type_price_ix
on more_titles(type, price)
with reservepagegap = 10
```

Vous pouvez également spécifier la valeur reservepagegap avec les options alter table...constraint, primary key et unique, qui créent des index. L'exemple suivant crée une contrainte d'unicité :

```
alter table more_titles
add constraint uniq_id unique (title_id)
with reservepagegap = 20
```

Modification de *reservepagegap*

La commande suivante utilise sp_chgattribute pour modifier l'intervalle de page réservée sur la table titles et lui attribuer la valeur 20 :

```
sp_chgattribute more_titles, "reservepagegap", 20
```

La commande ci-dessous définit l'intervalle de page réservée pour l'index title_ix à 10 :

```
sp_chgattribute "titles.title_ix",
    "reservepagegap", 10
```

sp_chgattribute modifie uniquement les valeurs dans les tables système ; après l'exécution de cette procédure, les données ne sont pas transférées vers d'autres pages. La modification de reservepagegap pour une table a les conséquences suivantes sur le stockage futur :

- Lorsque des données sont copiées dans la table par l'utilitaire 'bcp', l'intervalle de page réservée est appliqué à l'espace nouvellement alloué mais les pages existantes ne sont pas concernées.
- Lorsque la commande reorg rebuild est exécutée sur une table, l'intervalle de page réservée est appliqué lorsque la table est copiée dans de nouvelles pages de données.
- Lorsqu'un index clusterisé est créé, l'intervalle de page réservée tel qu'il est défini pour la table est appliqué aux pages de données.

L'intervalle de page réservée est appliqué aux pages d'index :

- durant l'exécution de alter table...lock, pendant la reconstruction des index de la table,
- durant l'exécution des commandes reorg rebuild qui travaillent sur les index,
- durant l'exécution des commandes create clustered index et alter table qui créent un index clusterisé, les index non clusterisés étant pour leur part recréés.

Exemples de *reservepagegap*

Les exemples suivants montrent l'application de reservepagegap durant l'exécution des commandes alter table et reorg rebuild.

reservepagegap spécifié uniquement pour la table

Les commandes suivantes spécifient une valeur de reservepagegap pour la table, mais pas dans les commandes create index :

```
sp_chgattribute titles, "reservepagegap", 16
create clustered index title_ix on titles(title_id)
create index type_price on titles(type, price)
```

Le tableau 13-8 répertorie les valeurs appliquées lors de l'exécution de reorg rebuild ou lors de la suppression puis de la création d'un index clusterisé.

Tableau 13-8 : valeurs de reservepagegap appliquées avec une valeur enregistrée pour la table

Commande	Table APL	Table DOL
create clustered index ou reconstruction d'index clusterisé suite à reorg rebuild	Pages de données et d'index : 16	Pages de données : 16 Pages d'index : 0 (extents remplis)
Reconstruction d'index non clusterisé	Pages d'index : 0 (extents remplis)	Pages d'index : 0 (extents remplis)

La valeur de reservepagegap pour la table s'applique aussi bien aux pages d'index que de données sur une table APL avec un index clusterisé. Pour une table DOL, la valeur de reservepagegap de cette table s'applique aux pages de données mais pas aux pages de l'index clusterisé.

reservepagegap spécifié pour un index clusterisé

Ces commandes spécifient des valeurs différentes de reservepagegap pour la table et l'index clusterisé et une valeur pour l'index non clusterisé type_price :

```
sp_chgattribute titles, "reservepagegap", 16
create clustered index title_ix on titles(title)
    with reservepagegap = 20
create index type_price on titles(type, price)
    with reservepagegap = 24
```

Le tableau 13-9 montre les effets de cette série de commandes.

Tableau 13-9 : valeurs de reservepagegap appliquées aux pages d'index

Commande	Table APL	Table DOL
create clustered index ou reconstruction d'index clusterisé suite à reorg rebuild	Pages de données et d'index : 20	Pages de données : 16 Pages d'index : 20
Reconstructions d'index non clusterisés	Pages d'index : 24	Pages d'index : 24

Pour les tables APL, la valeur de reservepagegap spécifiée avec create clustered index s'applique à la fois aux pages de données et aux pages d'index. Pour les tables DOL, la valeur de reservepagegap spécifiée avec create clustered index s'applique uniquement aux pages d'index. S'il existe une valeur enregistrée de reservepagegap pour la table, elle est appliquée aux pages de données.

Choix d'une valeur pour *reservepagegap*

Le choix d'une valeur pour reservepagegap dépend des éléments suivants :

- existence ou non d'un index clusterisé sur la table,
- taux des insertions dans la table,
- nombre de lignes redirigées dans la table, et
- fréquence de recréation de l'index clusterisé et d'exécution de la commande reorg rebuild.

Lorsque la propriété reservepagegap est correctement configurée, un nombre suffisant de pages est réservé à l'allocation de nouvelles pages pour les tables et les index, afin que les taux de clusterisation de la table, de l'index clusterisé et des pages de niveau feuille des index non clusterisés restent élevés entre deux tâches de maintenance standard.

Contrôle des valeurs de *reservepagegap*

Pour vérifier le taux de clusterisation et le nombre de lignes redirigées dans les tables, utilisez optdiag. La diminution des taux de clusterisation peut également indiquer la nécessité d'exécuter des commandes reorg pour améliorer les performances :

- Si le taux de clusterisation des pages de données d'un index clustérisé est bas, exécutez reorg rebuild ou supprimez l'index puis créez-le de nouveau.
- Si le taux de clusterisation des pages d'index d'un index non clusterisé est bas, supprimez cet index puis créez-le de nouveau.

Pour réduire la fréquence d'exécution des commandes reorg visant à maintenir des taux de clusterisation appropriés, augmentez légèrement la valeur de reservepagegap avant d'exécuter reorg rebuild.

Pour plus d'informations sur optdiag, reportez-vous au chapitre 36, "Affichage des tables de statistiques avec optdiag".

Options *reservepagegap* et *sorted_data* dans *create index*

Lorsque vous créez un index clusterisé sur une table dont les données sont déjà dans l'ordre de la clé d'index, l'utilisation de l'option sorted_data élimine la phase de copie des pages de données dans l'ordre de la clé d'index lorsque les tables ne sont pas partitionnées. L'option reservepagegap peut être spécifiée dans les commandes create clustered index afin de garder des pages vides sur les extents utilisés par la table, en vue d'un futur accroissement. Il existe des règles qui déterminent l'entrée en vigueur des options. Vous ne pouvez pas utiliser sp_chgattribute pour modifier la valeur de reservepagegap et tirer profit des deux options.

Si vous spécifiez les deux options avec create clustered index :

- Sur des tables APL non partitionnées, si la valeur de reservepagegap spécifiée avec create clustered index correspond aux valeurs déjà enregistrées dans sysindexes, l'option sorted_data a la priorité. Les pages de données ne sont pas copiées, de sorte que reservepagegap n'est pas appliqué. Si la valeur de reservepagegap spécifiée avec des commandes create clustered index diffèrent des valeurs enregistrées dans sysindexes, les pages de données sont copiées et la valeur de reservepagegap indiquée dans la commande leur est appliquée.

- Dans des tables DOL, la valeur de reservepagegap spécifiée avec create clustered index n'est appliquée qu'aux pages d'index. Les pages de données ne sont pas copiées.

Utilisation de l'option *sorted_data*

Parallèlement à reservepagegap, d'autres options incluses dans la commande create clustered index peuvent exiger un tri ; dans ce cas, l'option sorted_data est ignorée.

Pour plus d'informations, reportez-vous à la section "Création d'un index sur des données triées", page 391.

En particulier, tenez compte des commentaires suivants concernant l'utilisation de reservepagegap :

- Dans des tables partitionnées, toute commande create clustered index qui exige la copie de pages de données effectue un tri parallèle puis copie les pages dans l'ordre du tri, en appliquant les valeurs de reservepagegap lorsque les pages sont copiées dans de nouveaux extents.
- Chaque fois que l'option sorted_data n'est pas remplacée par d'autres options de create clustered index, un balayage de table est effectué afin de déterminer si les données sont stockées dans l'ordre de la clé d'index. L'index est construit durant le balayage, sans tri.

Le tableau 13-10 indique le mode d'application de ces règles.

Tableau 13-10 : options reservepagegap et sorted_data

	Table partitionnée	Table non partitionnée
Table APL		
create index with sorted_data et valeur reservepagegap correspondante	Les pages de données ne sont pas copiées ; l'index est construit à mesure que les pages sont balayées.	Les pages de données ne sont pas copiées ; l'index est construit à mesure que les pages sont balayées.
create index with sorted_data et valeur reservepagegap différente	Le tri parallèle est effectué, avec application de reservepagegap lorsque les pages sont stockées dans de nouveaux emplacements dans l'ordre du tri.	Les pages de données sont copiées, avec application de reservepagegap et construction de l'index à mesure que les pages sont copiées. Aucun tri n'est effectué.
Table DOL		
create index with sorted_data et toute valeur de reservepagegap	reservepagegap n'est appliqué qu'aux pages d'index ; les pages de données ne sont pas copiées.	reservepagegap n'est appliqué qu'aux pages d'index ; les pages de données ne sont pas copiées.

Options en correspondance avec les objectifs

Si vous souhaitez redistribuer les pages de données d'une table en prévoyant de l'espace pour un futur accroissement :

- Dans les tables APL, supprimez puis recréez l'index clusterisé sans utiliser l'option sorted_data. Spécifiez la valeur de reservepagegap voulue dans la commande create clustered index, si celle enregistrée dans sysindexes ne vous convient pas.
- Dans les tables DOL, utilisez sp_chgattribute pour définir reservepagegap à la valeur souhaitée, puis supprimez et recréez l'index clusterisé sans utiliser l'option sorted_data. La valeur de reservepagegap enregistrée pour la table s'applique aux pages de données. Si la valeur de reservepagegap est spécifiée dans la commande create clustered index, elle ne s'applique qu'aux pages d'index.

Pour créer un index clusterisé sans copier de pages de données :

- Dans les tables APL, utilisez l'option sorted_data sans spécifier reservepagegap avec la commande create clustered index. Vous pouvez aussi spécifier une valeur qui corresponde à celle enregistrée dans sysindexes.
- Dans les tables DOL, utilisez l'option sorted_data. Si une valeur de reservepagegap est spécifiée dans la commande create clustered index, elle s'applique uniquement aux pages d'index et n'entraîne pas la copie des pages de données.

Si vous envisagez d'utiliser l'option sorted_data après une opération de bulkcopy, une commande select into ou toute autre commande qui procède à une allocation d'extent, définissez reservepagegap à la valeur de votre choix pour les pages de données, avant de copier les données, ou spécifiez cette valeur dans la commande select into. Une fois que les pages de données ont été allouées et remplies, la commande suivante applique reservepagegap aux pages d'index uniquement, puisque la copie des pages de données n'est pas nécessaire :

```
create clustered index title_ix
on titles(title_id)
with sorted_data, reservepagegap = 32
```

Utilisation de *max_rows_per_page* dans des tables APL

La définition d'un nombre maximal de lignes par page peut réduire les conflits de verrous pour les tables APL et leurs index. Le plus souvent, il est préférable, sur ces tables, de passer à un verrouillage DOL. Si toutefois, pour une raison quelconque, vous ne pouvez pas changer le plan de verrouillage et si les conflits de verrous constituent un problème, la définition de la valeur max_rows_per_page peut améliorer les performances.

Lorsque les lignes sont peu nombreuses dans les pages de données ou d'index, les risques de conflits de verrous sont limités. Les clés étant réparties sur un plus grand nombre de pages, il est probable que la page recherchée ne soit pas utilisée par quelqu'un d'autre. Pour modifier le nombre de lignes par page, ajustez les valeurs de fillfactor ou max_rows_per_page de vos tables et index.

fillfactor (défini par sp_configure ou create index) détermine le degré de remplissage qu'adopte Adaptive Server lorsqu'il crée un index sur des données existantes. Compte tenu que l'option fillfactor contribue à réduire le nombre de divisions de pages, le nombre des verrous exclusifs est également limité sur l'index, ce qui améliore les performances. Cependant, la valeur de fillfactor n'est pas conservée lors des modifications ultérieures des données. max_rows_per_page (défini par sp_chgattribute, create index, create table ou alter table) est similaire à fillfactor, sauf qu'Adaptive Server maintient la valeur de max_rows_per_page à mesure que les données sont modifiées.

La diminution des valeurs de fillfactor ou de max_rows_per_page entraîne une augmentation des E/S requises pour lire un même nombre de pages de données, ainsi qu'une augmentation de la mémoire utilisée par le cache de données et du nombre de verrous. De plus, une valeur de max_rows_per_page moins élevée pour une table risque de se traduire par un plus grand nombre de divisions de page lors de l'insertion de données dans la table.

Réduction des conflits de verrouillage

La valeur de max_rows_per_page spécifiée dans une commande create table, create index ou alter table limite le nombre de lignes autorisées sur une page de données ou sur une page de niveau feuille d'index (clusterisé ou non). Cela réduit les conflits de verrouillage et améliore la concurrence d'accès pour les tables le plus fréquemment interrogées.

max_rows_per_page s'applique aux pages de données d'une table non indexée ou aux pages de niveau feuille d'un index. A la différence de fillfactor, dont la valeur n'est pas conservée après la création d'une table ou d'un index, Adaptive Server garde la valeur de max_rows_per_page lors de l'insertion ou de la suppression de lignes.

La commande suivante crée la table sales en limitant à quatre le nombre maximal de lignes par page :

```
create table sales
      (stor_id         char(4)      not null,
      ord_num          varchar(20)  not null,
      date             datetime     not null)
      with max_rows_per_page = 4
```

Si vous créez une table en indiquant une valeur pour max_rows_per_page, puis un index clusterisé sur la table sans spécifier max_rows_per_page, l'index utilise la valeur de max_rows_per_page définie dans l'instruction create table. La création d'un index clusterisé avec max_rows_per_page modifie la valeur pour les pages de données de la table.

Index et *max_rows_per_page*

La valeur par défaut de max_rows_per_page (0) crée des index clusterisés avec des pages de données pleines, des index non clusterisés avec des pages de niveau feuille pleines et laisse suffisamment d'espace dans l'arbre à accès séquentiel de l'index, qu'il soit clusterisé ou non.

Pour les tables sans index et les index clusterisés, les valeurs de max_rows_per_page sont comprises entre 0 et 256.

Pour les index non clusterisés, la valeur maximale de max_rows_per_page est égale au nombre de lignes d'index que peut contenir la page de niveau feuille, mais ne peut pas être supérieure à 256. Pour déterminer la valeur maximale, retirez 32 (c'est-à-dire la taille de l'en-tête) à la taille de la page et divisez le résultat par la taille de clé d'index. L'instruction suivante calcule la valeur maximale de max_rows_per_page pour un index non clusterisé :

```
select (@@pagesize - 32)/minlen
    from sysindexes
    where name = "indexname"
```

select into et *max_rows_per_page*

select into ne transmet pas la valeur max_rows_per_page de la table source, mais crée la nouvelle table avec max_rows_per_page égale à 0. Utilisez sp_chgattribute pour définir la valeur de max_rows_per_page pour la table cible.

Application de *max_rows_per_page* aux données existantes

sp_chgattribute configure la valeur de max_rows_per_page d'une table ou d'un index. sp_chgattribute s'applique à toutes les opérations futures, mais pas aux pages existantes. Par exemple, pour attribuer la valeur 1 à max_rows_per_page pour la table authors, entrez :

```
sp_chgattribute authors, "max_rows_per_page", 1
```

Pour appliquer une valeur de max_rows_per_page à des données existantes, deux méthodes sont possibles :

- Si la table a un index clusterisé, supprimez et recréez l'index avec une valeur de max_rows_per_page.
- Utilisez l'utilitaire bcp comme suit :
  a Exportez les données de la table.
  b Tronquez la table.
  c Définissez la valeur de max_rows_per_page à l'aide de sp_chgattribute.
  d Réimportez les données dans la table.

CHAPITRE 14

Utilisation et performances de la mémoire

Ce chapitre décrit la manière dont Adaptive Server utilise les caches de procédures et de données et il traite d'autres domaines concernés par la configuration de la mémoire. En général, plus la quantité de mémoire disponible est importante, plus le temps de réponse d'Adaptive Server est réduit.

Sujet	Page
Incidence de la mémoire sur les performances	317
Quantité de mémoire à configurer	318
Caches sur Adaptive Server	321
Cache des procédures	322
Cache de données	325
Configuration du cache de données pour améliorer les performances	330
Recommandations relatives au cache de données nommé	343
Maintien des performances du cache de données pour les E/S étendues	354
Vitesse de reprise	356
Audit et performances	358

Le *Guide d'administration système* décrit le processus qui permet de définir les valeurs de configuration de la mémoire les mieux adaptées pour Adaptive Server ainsi que les besoins en mémoire pour d'autres options de configuration du serveur.

Incidence de la mémoire sur les performances

Une plus grande quantité de mémoire réduit les E/S disque, ce qui améliore les performances, puisque l'accès à la mémoire est beaucoup plus rapide que l'accès au disque. Lorsqu'un utilisateur émet une requête, les pages de données et d'index doivent se trouver en mémoire ou être lues en mémoire, afin que leurs valeurs soient examinées. Si les pages résident déjà en mémoire, Adaptive Server n'a pas besoin d'effectuer d'E/S disque.

L'ajout de mémoire est peu coûteux et aisé, alors qu'il est onéreux de trouver des solutions alternatives pour résoudre les problèmes de mémoire. Il est donc conseillé d'attribuer le plus de mémoire possible à Adaptive Server.

Les conditions de mémoire ci-dessous peuvent induire de faibles performances :

- La taille du cache de données total est trop petite.
- La taille du cache de procédures est trop petite.
- Seul le cache par défaut est configuré sur un système SMP comportant plusieurs processeurs actifs, ce qui provoque un conflit avec le cache de données.
- La taille des caches de données configurés par l'utilisateur n'est pas appropriée à certaines applications utilisateur.
- La taille des E/S configurées n'est pas appropriée à certaines requêtes.
- La taille de la file d'attente d'audit n'est pas appropriée si la fonction d'audit est installée.

Quantité de mémoire à configurer

La mémoire est l'aspect le plus important à prendre en considération lors de la configuration d'Adaptive Server. La mémoire est occupée par divers paramètres de configuration, caches de procédures et de données. La définition adéquate des divers paramètres de configuration et des caches est essentielle aux bonnes performances du système.

La mémoire totale allouée au démarrage est la somme de la mémoire nécessaire à toutes les opérations de configuration d'Adaptive Server. Cette valeur peut être obtenue à partir du paramètre de configuration en lecture seule 'total logical memory'. Cette valeur est calculée par Adaptive Server. Le paramètre de configuration 'max memory' doit être supérieur ou égal au paramètre 'total logical memory'. 'max memory' indique la quantité de mémoire allouée aux besoins d'Adaptive Server.

La mémoire qu'Adaptive Server alloue par défaut au démarrage est fonction du paramètre 'total logical memory'. Cependant, si le paramètre de configuration 'allocate max shared memory' a été défini, la mémoire allouée est fonction du paramètre 'max memory'. Le paramètre de configuration 'allocate max shared memory' permet à un administrateur système d'allouer la totalité de la mémoire à disposition d'Adaptive Server au démarrage.

Les points clé pour la configuration de la mémoire sont :

- L'administrateur système doit déterminer la taille de la mémoire partagée à disposition d'Adaptive Server et définir le paramètre 'max memory' sur cette valeur.
- Le paramètre de configuration 'allocate max shared memory' peut être activé au démarrage et lors de l'exécution afin d'allouer la totalité de la mémoire partagée jusqu'à atteindre la valeur fixée pour 'max memory' avec une petite quantité de segments de mémoire partagée. Une grande quantité de segments de mémoire partagée comporte une dégradation des performances sur certaines plates-formes. Pour déterminer la quantité optimale de segments de mémoire partagée, reportez-vous à la documentation de votre système d'exploitation. Une fois qu'un segment de mémoire partagée a été alloué, il ne peut pas être libéré avant le prochain redémarrage du serveur.
- Configurez les divers paramètres de configuration, si les paramètres par défauts sont insuffisants.
- A présent, la différence entre 'max memory' et 'total logical memory' constitue une mémoire supplémentaire à disposition du cache de procédures, des caches de données ou d'autres paramètres de configuration.

  La quantité de mémoire qu'Adaptive Server alloue au démarrage est déterminée par le paramètre 'total logical memory' ou 'max memory'. Si cette valeur est trop élevée :

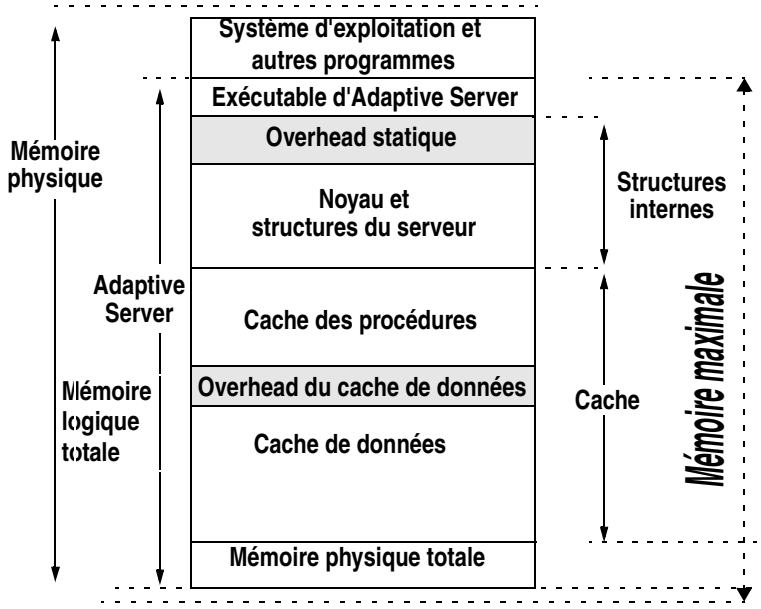
  - Adaptive Server peut ne pas démarrer, si les ressources physiques de votre machine sont insuffisantes.
  - S'il démarre, les taux de pagination du système d'exploitation peuvent augmenter de manière considérable et ce dernier devra probablement être reconfiguré, pour compenser.

Le *Guide d'administration système* traite de manière approfondie des sujets suivants :

- Configuration de la quantité de mémoire totale utilisée par Adaptive Server.
- Paramètres configurables qui utilisent de la mémoire et ont un impact sur la quantité de mémoire restant pour le traitement des requêtes.
- La gestion d'un plus grand nombre de caractères littéraux exige qu'Adaptive Server alloue de la mémoire pour les données utilisateur sous forme de chaîne. De plus, au lieu d'allouer statistiquement des buffers de la plus grande taille, Adaptive Server alloue la mémoire de façon dynamique. Cela signifie que la mémoire est allouée pour les buffers locaux selon les nécessités et d'une taille maximale, même si des buffers de grande taille ne sont pas nécessaires. Ces requêtes de gestion de la mémoire peuvent comporter une dégradation marginale des performances d'Adaptive Server lors de la gestion des données sous forme de caractères étendus.
- Si vous désirez qu'Adaptive Server gère plus de 1 000 colonnes sur une seule table ou qu'il traite plus de 10 000 arguments comme procédures stockées, le serveur doit installer et allouer de la mémoire pour diverses structures de données internes pour ces objets. Une augmentation du nombre de petites tâches exécutées régulièrement peut comporter une dégradation des performances quand il s'agit de requêtes concernant une grande quantité de ces éléments. Cette dégradation augmente au fur et à mesure que la quantité de colonnes et d'arguments traités comme procédures stockées augmente.
- La mémoire allouée de façon dynamique (contrairement à la mémoire allouée par redémarrage d'Adaptive Server) comporte une légère dégradation des performances du serveur.
- Quand Adaptive Server utilise de grandes pages logiques, toutes les E/S disque sont exécutées en fonction des grandes pages logiques. Par exemple, si Adaptive Server utilise une page logique de 8 k, il extrait des données du disque en blocs de 8 k. Cela se traduit en un débit supérieur des E/S, même si celui-ci est limité par la largeur de bande E/S du contrôleur.

La mémoire restante est disponible pour le cache de procédures et le cache de données. La figure 14-1 illustre la répartition de la mémoire.

Figure 14-1 : Utilisation de la mémoire par Adaptive Server



Caches sur Adaptive Server

Une fois que le cache des procédures et le cache des données sont configurés, il n'y a aucune division ou mémoire restante.

- Le **cache des procédures** est utilisé pour les procédures stockées et les triggers et pour les besoins en mémoire à court terme, par exemple des statistiques et des plans d'exécution de requêtes en parallèle.
- Le **cache de données** est utilisé pour les pages de données, d'index et de journal. Le cache de données peut être divisé en caches nommés distincts auxquels sont allouées des bases de données ou des objets de base de données spécifiques.

Définissez la taille du cache des procédures sur une valeur absolue à l'aide de sp_configure. Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Cache des procédures

Adaptive Server conserve en mémoire une chaîne MRU/LRU (most recently used/least recently used, plus récemment utilisé/moins récemment utilisé) des plans d'exécution de requête pour les procédures stockées. Au fur et à mesure que les utilisateurs exécutent les procédures stockées, Adaptive Server recherche dans le cache des procédures un plan d'exécution de requête à utiliser. Si un plan d'exécution de requête est disponible, il est placé à l'extrémité MRU de la chaîne et l'exécution débute.

S'il n'existe pas de plan en mémoire ou si tous les exemplaires sont en cours d'utilisation, l'arbre de requêtes de la procédure est lu dans la table sysprocedures. Il est ensuite optimisé à l'aide des paramètres fournis à la procédure, puis mis à l'extrémité MRU de la chaîne et son exécution débute. Les plans à l'extrémité LRU de la chaîne de pages, qui ne sont pas en cours d'utilisation, sont supprimés du cache.

La mémoire allouée au cache de procédures maintient les plans d'exécution de requête optimisés (et occasionnellement les arbres) pour tous les batchs, y compris les triggers.

Si une procédure ou un trigger est utilisé simultanément par plusieurs utilisateurs, plusieurs de leurs exemplaires se trouveront dans le cache. Si le cache des procédures est trop petit, un utilisateur qui essaie d'exécuter des procédures stockées ou des requêtes qui déclenchent des triggers, reçoit un message d'erreur et doit renouveler la requête. Il y a libération d'espace lorsque des plans inutilisés sortent du cache.

Lors de l'installation initiale d'Adaptive Server, la taille par défaut du cache des procédures est de 3271 pages de mémoire. La valeur optimale du cache des procédures varie d'une application à l'autre et à mesure que les types d'utilisation changent. Le paramètre de configuration définissant la taille, procedure cache size, est présenté dans le *Guide d'administration système*.

Informations sur la taille du cache des procédures

Au démarrage d'Adaptive Server, le journal d'erreurs fait état de la taille du cache des procédures.

Buffers de procédures

Représente le nombre maximal d'objets compilés qui peuvent résider en même temps dans le cache des procédures.

En-têtes des procédures

Représente 1e nombre de pages dédiées au cache des procédures. Chaque objet du cache a besoin d'au moins une page.

Contrôle des performances du cache des procédures

sp_sysmon fournit des indications sur l'exécution des procédures stockées et sur le nombre de lectures sur le disque nécessaire pour ces procédures.

Pour plus d'informations, reportez-vous à la section "Gestion du cache de procédures", page 1063.

Erreurs du cache des procédures

Si la quantité de mémoire est insuffisante pour charger un autre arbre ou plan d'exécution de requête, ou si le nombre maximum d'objets compilés est déjà en cours d'utilisation, Adaptive Server renvoie l'erreur 701.

Dimensionnement du cache des procédures

Sur un serveur de production, vous désirez réduire les lectures de procédure sur le disque. Lorsqu'un utilisateur souhaite exécuter une procédure, Adaptive Server doit pouvoir trouver un arbre ou un plan inutilisé dans le cache des procédures, pour les procédures les plus communes. La fréquence avec laquelle le serveur trouve un plan disponible dans le cache est appelée le **taux de présence dans le cache**. Un taux élevé de présence dans le cache pour les procédures du cache améliore les performances.

Les formules de la figure 14-2 constituent un bon point de départ.

Figure 14-2 : Formules permettant de dimensionner le cache des procédures

Taille du cache des procédures	=	(Nbre max. d'utilis. concurrents) * (Taille du plan le plus grand) * 1,25
Taille minimale du cache des procédures requise	=	(Nbre de procédures principales) * (Taille moyenne du plan)

Si vous avez des procédures stockées imbriquées (la procédure A appelle la procédure B qui appelle la procédure C), elles doivent toutes se trouver en même temps dans le cache. Ajoutez les tailles des procédures imbriquées et utilisez la somme la plus importante au lieu de "Taille du plan le plus grand" dans la formule de la figure 14-2.

La taille minimale du cache des procédures est la plus petite quantité de mémoire permettant à au moins un exemplaire de chaque objet compilé fréquemment utilisé, de résider dans le cache. Cependant, le cache des procédures peut également servir de mémoire supplémentaire lors de l'exécution, par exemple quand une requête ad hoc utilise le mot-clé distinct qui utilise la fonction interne lmlink qui allouera la mémoire de façon dynamique à partir du cache des procédures. Alors le create index utilisera également la mémoire du cache des procédures et peut générer l'erreur 701, bien qu'aucune procédure stockée ne soit concernée.

Pour plus d'informations sur le dimensionnement du cache des procédures, reportez-vous à la section "Utilisation de sp_monitor pour évaluer l'occupation de la CPU", page 42.

Evaluation de la taille d'une procédure stockée

Pour obtenir une estimation globale de la taille d'une procédure stockée, d'une vue ou d'un trigger, utilisez :

```
select(count(*) / 8) +1
    dans sysprocedures
où id = object_id ("nom_procédure")
```

Par exemple, pour trouver la taille de titleid_proc dans pubs2 :

```
select(count(*) / 8) +1
    dans sysprocedures
où id = object_id("titleid_proc")
-----------
          3
```

Cache de données

Le cache de données par défaut et d'autres caches sont configurés comme valeurs absolues. Le cache de données contient les pages des objets le plus récemment interrogés, qui sont en général :

- sysobjects, sysindexes et d'autres tables système pour chaque base de données,
- les pages du journal actif pour chaque base de données,
- les plus hauts niveaux et une partie des niveaux les plus bas des index fréquemment utilisés,
- les dernières pages de données consultées.

Cache par défaut au moment de l'installation

Lors de la première installation d'Adaptive Server, celui-ci comporte un cache de données unique qui est utilisé par tous les processus et les objets d'Adaptive Server pour les pages de données, d'index et de journal. La taille par défaut est 8 Mo.

Les pages suivantes décrivent le mode d'utilisation de ce cache de données unique. La section "Configuration du cache de données pour améliorer les performances", page 330 indique comment améliorer les performances en divisant le cache de données en caches nommés et comment lier des objets spécifiques à ces derniers.

La plupart des concepts sur le vieillissement, sur le vidage du buffer et les stratégies de cache s'appliquent aux caches de données définis par l'utilisateur et au cache de données par défaut.

Vieillissement des pages dans le cache de données

Le cache de données d'Adaptive Server est géré comme une chaîne de buffers MRU/LRU (most recently used/least recently used), c'est-à-dire sur le principe d'utilisation la plus récente/utilisation la plus ancienne. A mesure que les pages du cache "vieillissent", elles accèdent à une zone de vidage dans laquelle toute page modifiée en mémoire est écrite sur le disque. Il existe toutefois quelques exceptions, notamment :

- Les caches configurés avec une stratégie de cache LRU souple utilisent la zone de vidage décrite ci-dessus, mais ne sont pas gérés selon le principe MRU/LRU.

  En général, les pages dans la zone de vidage ne sont pas modifiées, soit les E/S sur ces pages ont été réalisées. Quand une tâche ou une requête doit extraire une page de l'extrémité LRU, celle-ci ne doit pas avoir été modifiée. Dans le cas contraire, la requête doit attendre la fin des E/S sur la page avant de pouvoir l'extraire, ce qui nuit aux performances.

- Une stratégie spéciale fait vieillir les pages d'index et les **pages OAM** plus lentement que les pages de données. Ces pages sont interrogées fréquemment dans certaines applications et le fait de les conserver dans le cache réduit de manière non négligeable les lectures sur disque.

  Pour plus d'informations, reportez-vous au *Guide d'administration système*.

- Adaptive Server peut choisir d'utiliser la stratégie de remplacement LRU, qui ne remplace pas les autres pages du cache par celles qui ne sont utilisées qu'une fois dans toute la requête.

- Le processus de point de reprise assure que si Adaptive Server doit être redémarré, le processus de reprise peut être achevé dans un délai raisonnable.

  Quand le processus de point de reprise estime que le délai de récupération du nombre de modifications apportées à une base de données sera plus long que la valeur configurée du paramètre recovery interval, il parcourt le cache en écrivant des pages modifiées sur le disque.

- La récupération n'utilise que le cache de données par défaut qui l'accélère.

- Une tâche de gestion interne (housekeeper) écrit également des pages modifiées sur le disque, lors des moments d'inactivité entre deux processus utilisateur.

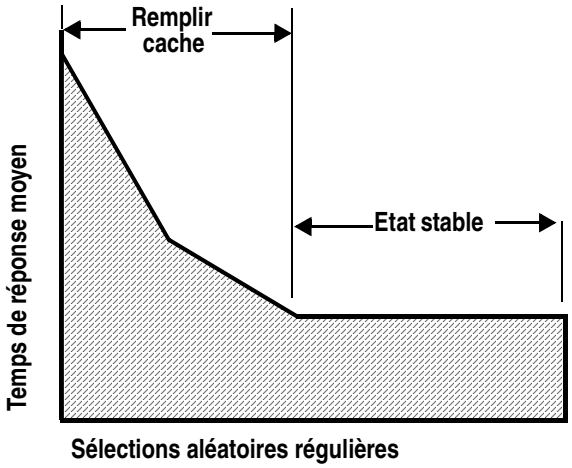
Impact du cache de données sur les extractions

La figure 14-3 illustre l'impact de la mise en mémoire des données sur une série d'instructions select aléatoires exécutées pendant une certaine période. Si le cache est initialement vide, il est sûr que la première instruction select requiert des E/S disque. Vérifiez que la taille du cache de données s'adapte au nombre de transactions attendues sur la base de données.

Au fur et à mesure que les requêtes sont exécutées et que le cache se remplit, il est de plus en plus probable qu'une ou plusieurs requêtes d'accès aux pages peuvent être satisfaites par le cache, réduisant par-là même le temps de réponse moyen de l'ensemble des recherches.

Une fois le cache rempli, il existe une probabilité fixe et durable de trouver la page souhaitée dans le cache.

Figure 14-3 : Impact des sélections aléatoires sur le cache de données



Si le cache est plus petit que le nombre total de pages consultées dans toutes les bases de données, une instruction donnée risque de devoir effectuer des E/S disque. Le cache ne réduit pas le temps de réponse maximal possible (certaines requêtes peuvent continuer de nécessiter l'exécution d'E/S physiques pour toutes les pages souhaitées). Toutefois, la mise en cache diminue la probabilité qu'une requête nécessite le temps de réponse maximal (la plupart des requêtes sont susceptibles de trouver au moins certaines pages recherchées dans le cache).

Impact des modifications de données sur le cache

Le comportement du cache pour les transactions de mise à jour est plus complexe que pour les extractions.

Il existe toujours une période initiale remplie par le cache. Puis, en raison du fait que les pages du cache sont en cours de modification, il arrive un moment où le cache doit commencer à écrire ces pages sur disque avant de pouvoir charger les autres pages. Dans le temps, la quantité d'écriture et de lecture se stabilise, et il existe une certaine probabilité que les transactions suivantes demandent une lecture de disque et une autre probabilité qu'elles causent une écriture sur disque.

La période stable est interrompue par les points de reprise, ce qui provoque l'écriture sur disque par le cache de toutes les pages modifiées.

Performances du cache de données

Vous pouvez observer les performances du cache de données en examinant le **cache hit ratio**, soit le pourcentage de demandes de page auxquelles répond le cache.

Un taux de 100 % est peu courant mais il indique que votre cache de données est aussi important que les données, ou du moins suffisamment grand pour contenir toutes les pages de vos tables et index fréquemment utilisés.

Si le taux de présence dans le cache est faible, celui-ci est probablement trop petit pour le chargement des applications courantes. Les très grandes tables avec un accès de page aléatoire ont généralement un faible taux de présence dans le cache.

Test des performances du cache de données

Prenez en compte le comportement du cache de données et du cache des procédures lorsque vous mesurez les performances d'un système. Lorsqu'un test commence, le cache peut être dans l'un des états suivants :

- Vide
- traité en totalité de manière aléatoire
- traité partiellement de manière aléatoire
- déterministe

Un cache vide ou traité en totalité de manière aléatoire fournit des résultats de test répétables, car le cache demeure dans le même état d'un test à l'autre.

Un cache traité partiellement de manière aléatoire ou déterministe contient des pages laissées par des transactions venant d'être exécutées. Ces pages peuvent être le résultat d'un test antérieur. Dans ce cas, si les étapes suivantes du test requièrent ces pages, aucune E/S disque ne sera nécessaire.

Une telle situation peut aboutir à des résultats fort différents de ceux d'un test purement aléatoire et fausser les estimations de performances.

La meilleure stratégie pour effectuer des tests est de commencer par un cache vide ou de s'assurer que toutes les étapes du test ont accès à des parties aléatoires de la base de données. Pour un test plus précis, exécutez une combinaison des requêtes qui respecte la combinaison planifiée des requêtes utilisateur sur votre système.

Taux de présence dans le cache pour une seule requête

Pour visualiser le taux de présence dans le cache pour une seule requête, utilisez set statistics io on pour afficher le nombre de lectures logiques et physiques, et set showplan on pour afficher la taille des E/S utilisée par la requête.

Pour calculer le taux de présence dans le cache, utilisez la formule suivante :

$$\text{Taux de présence dans le cache} = \frac{\text{Lectures logiques} - (\text{Lectures logiques} * \text{Pages par E/S})}{\text{Lectures logiques}}$$

Avec statistics io, les lectures physiques sont indiquées en unités de taille des E/S. Une requête qui utilise des E/S de 16 k lit 8 pages par opération d'E/S.

Si statistics io comptabilise 50 lectures physiques, 400 pages ont été lues. Exécutez showplan pour afficher la taille des E/S utilisée par une requête.

Informations sur le taux de présence dans le cache issues de *sp_sysmon*

sp_sysmon fournit des informations sur les présences et non-présences dans le cache pour :

- tous les caches d'Adaptive Server
- le cache de données par défaut
- tous les caches configurés par l'utilisateur

Le rapport du serveur précise le nombre total de recherches dans le cache ainsi que le pourcentage de présence et de non-présence dans le cache.

Pour plus d'informations, reportez-vous à la section "Récapitulatif des statistiques relatives aux applications (toutes applications)", page 1001.

Pour chaque cache, le rapport indique le nombre de recherches, de présences et de non-présences dans le cache, ainsi que le nombre de fois qu'un buffer recherché a été trouvé dans la zone de vidage.

Pour plus d'informations, reportez-vous à la section "Gestion des caches par cache", page 1055.

Configuration du cache de données pour améliorer les performances

Lors de son installation, Adaptive Server possède un seul cache de données par défaut, avec une zone mémoire de 2 k, une partition de cache et un seul verrou d'attente.

Pour améliorer les performances, vous pouvez ajouter des caches de données et y lier des bases de données ou des objets de base de données :

1 Pour limiter les conflits de verrouillage du cache de données par défaut, divisez le cache par *n*, où *n* correspond à 1, 2, 4, 8,16, 32 ou 64. En cas de conflits de verrouillage avec une partition de cache, ceux-ci devraient diminuer de x/*n*, où *n* est le nombre de partitions.

2 Quand un verrou d'attente particulier au niveau de la partition de cache est *saturé*, envisagez de fractionner le cache par défaut en caches nommés.

3 Si les conflits perdurent, envisagez de fractionner le cache nommé en partitions de caches nommés.

Vous pouvez configurer des zones de buffers de 4 k, 8 k et 16 k issues de la taille de la page logique dans les caches de données définis par l'utilisateur et les caches de données par défaut, ce qui permet à Adaptive Server d'effectuer des E/S étendues. En outre, les caches dimensionnés de façon à stocker toute une table ou tout un index peuvent utiliser une stratégie de cache LRU souple pour limiter l'overhead.

Vous pouvez également diviser le cache de données par défaut ou un cache nommé en partitions, afin de réduire les conflits de verrous d'attente.

La configuration du cache de données peut améliorer les performances des manières suivantes :

- Vous pouvez configurer des caches de données nommés suffisamment grands pour contenir des tables et des index prioritaires.

  Cela empêche les autres activités du serveur de se disputer l'espace du cache et accélère les requêtes qui utilisent ces tables, étant donné que les pages requises sont toujours trouvées dans le cache.

  Vous pouvez configurer ces caches pour utiliser une stratégie de remplacement LRU souple, qui limite l'overhead de cache.

- Vous pouvez lier une table "stratégique" (table très demandée par les applications utilisateur) à un cache, et les index sur la table à d'autres caches, afin d'augmenter la concurrence d'accès.

- Vous pouvez créer un cache de données nommé de taille suffisamment grande pour contenir les "pages stratégiques" d'une table, si un pourcentage important de requêtes ne font référence qu'à une partie de la table.

  Par exemple, si une table contient des données sur une année mais que 75 % des requêtes font appel aux données du mois le plus récent (soit environ 8 % de la table), configurez un cache d'environ 10 % de la taille de la table ; vous disposerez ainsi de suffisamment d'espace pour conserver dans le cache les pages les plus souvent utilisées et celles les moins souvent utilisées.

- Vous pouvez attribuer des tables ou des bases de données utilisées dans les systèmes d'aide à la décision (DSS) à des caches spécifiques ayant des E/S étendues configurées.

Cela empêche les applications DSS de se disputer l'espace du cache avec les applications de traitement de transaction en ligne (OLTP). Généralement, les applications DSS accèdent à une grande quantité de pages séquentielles et les applications OLTP accèdent à relativement peu de pages aléatoires.

- Vous pouvez lier tempdb à son propre cache pour lui empêcher d'entrer en conflit avec les autres processus utilisateur.

  Une définition correcte de la taille du cache de tempdb peut permettre de conserver la plupart des activités de tempdb en mémoire pour de nombreuses applications. Si ce cache est suffisamment grand, l'activité de tempdb peut éviter les E/S disque.

- Les pages de texte peuvent être liées à des caches nommés, afin d'améliorer les performances de l'accès au texte.

- Vous pouvez lier un journal de base de données à un cache, ce qui réduit également les conflits vis à vis de l'espace du cache et de son accès.

- Lorsque des modifications sont apportées à un cache par un processus utilisateur, un **verrou d'attente** refuse l'accès au cache à tout autre processus.

  Bien que les verrous d'attente ne restent actifs que pendant de courtes périodes, ils peuvent ralentir les performances des systèmes multiprocesseur dont les taux de transaction sont élevés. Lorsque vous configurez plusieurs caches, chacun d'entre eux est contrôlé par un verrou d'attente distinct, ce qui augmente la concurrence d'accès sur les systèmes ayant plusieurs CPU.

  Dans un cache unique, l'ajout de partitions de caches crée plusieurs verrous d'attente afin de réduire ultérieurement les conflits. Les conflits de verrous ne sont pas un problème sur les serveurs à moteur unique.

La plupart de ces utilisations des caches de données nommés ont un impact très important sur les systèmes multiprocesseur ayant des taux de transaction élevés ou des requêtes DSS fréquentes et plusieurs utilisateurs. Certains d'entre eux peuvent améliorer les performances sur les systèmes ayant une seule CPU lorsqu'ils conduisent à une meilleure utilisation de la mémoire et réduisent les E/S.

Commandes de configuration des caches de données nommés

Les commandes de configuration des caches et des zones sont présentées dans le tableau 14-1.

Tableau 14-1 : Commandes utilisées pour configurer les caches

Commande	Fonction
sp_cacheconfig	Crée ou supprime des caches nommés et définit la taille et le type du cache, la stratégie de cache ainsi que le nombre de partitions de caches local. Fournit des informations sur la taille des caches et des zones.
sp_poolconfig	Crée et supprime les zones d'E/S et modifie leur taille, la taille de vidage et la limite de prélecture asynchrone.
sp_bindcache	Lie les bases de données ou les objets de base de données à un cache.
sp_unbindcache	Supprime le lien de la base de données ou de l'objet de base de données spécifié au cache.
sp_unbindcache_all	Supprime le lien de toutes les bases de données et objets de base de données liés à un cache spécifié.
sp_helpcache	Fournit des informations de synthèse sur les caches de données et énumère les bases de données et les objets de base de données qui sont liés à un cache. Signale également le volume d'overhead requis par un cache.
sp_sysmon	Génère des statistiques permettant d'optimiser la configuration du cache, y compris les conflits relatifs aux verrous d'attente sur le cache, l'utilisation du cache et les masques d'E/S disque.

Pour une description complète de la configuration des caches nommés et de la liaison d'objets aux caches, reportez-vous au *Guide d'administration système*. Seul un administrateur système peut configurer des caches nommés et lier des objets de base de données à ces caches.

Optimisation des caches nommés

La création de caches de données nommés et de zones de mémoire, ainsi que la liaison de bases de données et d'objets de base de données aux caches, peuvent améliorer ou diminuer considérablement les performances d'Adaptive Server. Exemple :

- Un cache peu utilisé réduit les performances.

  Si vous allouez 25 % de votre cache de données à une base de données qui prend en cache un très faible pourcentage de l'activité de requêtes sur votre serveur, les E/S augmentent dans les autres caches.

- Une zone inutilisée diminue les performances.

  Si vous ajoutez une zone de 16 k mais qu'aucune de vos requêtes ne l'utilise, vous avez retiré de l'espace de la zone de 2 k. Le taux de présence dans le cache de la zone de 2 ko est réduit et le nombre d'E/S augmente.

- Une zone trop utilisée diminue les performances.

  Si vous configurez une petite zone de 16 k et que presque toutes vos requêtes l'utilisent, les taux d'E/S augmentent. Le cache de 2 k devient sous-utilisé, tandis que les pages tournent rapidement sur une zone de 16 k. Le taux de présence dans le cache dans la zone de 16 k sera très faible.

- Si vous équilibrez l'utilisation de vos zones dans un cache, les performances peuvent augmenter considérablement.

  Les requêtes utilisant des zones de 16 k et de 2 k peuvent connaître des taux de présence dans le cache améliorés. Le grand nombre de pages souvent utilisé par les requêtes qui effectuent des E/S de 16 k ne vide pas les pages de 2 k du disque. Les requêtes qui utilisent une zone de 16 k effectueront approximativement un-huitième du nombre d'E/S requis par les requêtes effectuant des E/S de 2 k.

Lorsque vous optimisez les caches nommés, mesurez toujours les performances courantes, effectuez les modifications nécessaires dans votre configuration et mesurez les conséquences de ces modifications pour une charge de travail similaire.

Objectifs de la configuration des caches

Les différents objectifs de la configuration des caches sont :

- Réduire les conflits de verrous d'attente sur les serveurs à plusieurs moteurs.
- Améliorer les taux de présence dans le cache et/ou limiter les E/S disque. De plus, l'amélioration des taux de présence dans le cache pour les requêtes peut réduire les conflits de verrouillage, puisque les requêtes qui ne requièrent pas d'E/S physiques maintiennent généralement des verrous pour des périodes plus courtes.
- Réduire le nombre de lectures physiques, grâce à l'utilisation efficace des E/S étendues.
- Réduire les écritures physiques, car les pages récemment modifiées ne sont pas vidées du cache par d'autres processus.

- Réduire l'overhead du cache et le temps de latence du bus du processeur sur les systèmes SMP, lorsque la stratégie LRU souple est appliquée correctement.
- Réduire les conflits de verrous d'attente au niveau du cache sur les systèmes SMP, lorsque des partitions de cache sont utilisées.

En plus des commandes telles que showplan et statistics io qui permettent une optimisation par requête, vous devez utiliser un outil de contrôle des performances tel que sp_sysmon pour comprendre la manière dont plusieurs applications et requêtes partagent l'espace du cache lorsqu'elles sont exécutées simultanément.

Collecte de données, planification et mise en oeuvre

La première étape de développement d'un plan d'utilisation des caches consiste à allouer autant de mémoire que possible au cache de données :

- Déterminez la quantité maximale de mémoire que vous pouvez allouer à Adaptive Server. Définissez le paramètre de configuration 'max memory' sur cette valeur.
- Une fois que tous les paramètres de configuration qui utilisent la mémoire d'Adaptive Server ont été configurés, la différence entre 'max memory' et la valeur d'exécution de 'total logical memory' constitue une mémoire supplémentaire à disposition des autres paramètres de configuration et/ou des caches des procédures et des caches de données. Si vous avez suffisamment configuré tous les autres paramètres de configuration, vous pouvez décider d'allouer cette mémoire supplémentaire aux caches de données. Notez que la configuration d'un cache de données exige un redémarrage.
- Notez que si vous allouez toute la mémoire supplémentaire aux caches de données, il se peut qu'aucune mémoire ne soit disponible pour la reconfiguration d'autres paramètres de configuration. Cependant, si votre système dispose de mémoire supplémentaire, la valeur 'max memory' peut être accrue de façon dynamique et d'autres paramètres de configuration dynamiques, tels que 'procedure cache size', 'user connections', etc., peuvent être accrus.
- Utilisez vos outils de contrôle des performances pour établir des performances de référence et vos objectifs d'optimisation.

Déterminez la quantité de mémoire que vous pouvez allouer aux caches de données comme susmentionné. Incluez la taille de ou des caches déjà configurés, tels que le cache de données par défaut et tout cache nommé.

Décidez la taille des caches de données en fonction des objets et applications existants. Notez que l'ajout de nouveaux caches ou l'accroissement des paramètres de configuration qui utilisent de la mémoire ne réduit pas la taille du cache de données par défaut. Une fois que vous avez décidé la quantité de mémoire à allouer aux caches de données et la taille de chaque cache, ajoutez de nouveaux caches et augmentez ou diminuez la taille des caches de données existants.

- Evaluez les besoins du cache en analysant les masques d'E/S et évaluez les besoins en zones grâce à l'analyse des plans d'exécution de requête et des statistiques d'E/S.
- Configurez d'abord les choix les plus simples qui gagneront le plus en performances :
    - Choisissez une taille pour le cache de tempdb.
    - Choisissez une taille pour tout cache de journal et optimisez la taille des E/S du journal.
    - Choisissez une taille pour les tables ou les index que vous souhaitez conserver entièrement dans le cache.
    - Ajoutez des zones d'E/S étendues pour les caches d'index ou de données, en fonction des besoins.
- Une fois ces tailles définies, examinez les masques d'E/S restants, les conflits de cache et les performances des requêtes. Configurez les caches proportionnellement à l'activité d'E/S pour les objets et les bases de données.

Gardez vos objectifs de performances présents à l'esprit lorsque vous configurez les caches :

- Si votre objectif principal en configurant des caches est de réduire les conflits de verrouillage, l'augmentation du nombre de partitions dans les caches fortement utilisés peut suffire.

  Le transfert de quelques objets fortement consommateurs d'E/S pour séparer les caches réduit également les conflits de verrous et améliore les performances.
- Si votre objectif principal est d'améliorer le temps de réponse en améliorant le taux de présence dans le cache de certaines requêtes ou applications, la création de caches pour les tables et les index utilisés par ces requêtes doit être guidée par une compréhension approfondie des méthodes d'accès et des besoins en E/S.

Evaluation des besoins en cache

En général, votre objectif est de configurer des caches proportionnellement au nombre d'accès aux pages de ces caches par vos requêtes et de configurer des zones dans les caches proportionnellement au nombre de pages utilisées par les requêtes qui choisissent des E/S de cette taille de zone.

Si vos bases de données et leurs journaux se trouvent sur des devices logiques séparés, vous pouvez estimer les proportions des caches à l'aide de sp_sysmon ou des commandes du système d'exploitation pour examiner les E/S physiques par device.

Pour plus d'informations sur le résultat de sp_sysmon présentant des E/S disque, reportez-vous à la section "Gestion des E/S disque", page 1068.

E/S étendues et performances

Vous pouvez configurer le cache par défaut et n'importe quel cache nommé créé pour des E/S étendues, en divisant un cache en plusieurs zones. La taille d'E/S par défaut est de 2 k, soit une page de données d'Adaptive Server.

Pour certaines requêtes où les pages sont stockées et interrogées séquentiellement, Adaptive Server peut lire jusqu'à 8 pages de données en une seule E/S. La majorité du temps E/S étant utilisé pour le positionnement et la recherche physique sur le disque, les E/S étendues peuvent réduire considérablement le temps d'accès au disque. La pratique la plus courante est de configurer une zone de 16 k dans le cache de données par défaut.

Certains types de requête Adaptive Server sont susceptibles de tirer parti des E/S étendues. Leur identification peut vous aider à déterminer la taille appropriée des caches de données et des zones mémoire.

Dans les exemples suivants, soit la base de données, soit la table, l'index ou la chaîne de pages (utilisée pour les colonnes de type text, image et les colonnes Java hors de la ligne) doit être lié à un cache de données nommé comprenant des zones de mémoire de grande taille, ou bien le cache de données par défaut doit disposer de zones d'E/S étendues. Les types de requête qui ont avantage à utiliser des E/S étendues sont :

- Les requêtes qui balaient des tables entières. Exemple :

```
select title_id, price from titles
select count(*) from authors
    where state = "CA"   /* no index on state */
```

- Les requêtes à intervalle sur des tables ayant des index clusterisés. Exemple :

  ```
  where indexed_colname >= value
  ```

- Les requêtes qui lisent le niveau feuille d'un index (lectures avec et sans correspondance). S'il existe un index non clusterisé sur type, price, cette requête pourrait utiliser des E/S étendues au niveau feuille de l'index, puisque toutes les colonnes utilisées dans la requête se trouvent bien dans l'index :

  ```
  select type, sum(price)
      from titles
      group by type
  ```

- Les requêtes qui joignent des tables entières ou de grandes portions de tables. Des tailles d'E/S différentes peuvent être utilisées sur les tables d'une jointure.

- Les requêtes qui sélectionnent des colonnes de type text ou image ou des colonnes Java hors de la ligne. Exemple :

  ```
  select au_id, copy from blurbs
  ```

- Les requêtes qui génèrent des produits cartésiens. Exemple :

  ```
  select title, au_lname
       from titles, authors
  ```

  Cette requête requiert le balayage complet d'une table puis de l'autre table à la recherche de chaque ligne de la première table. Les stratégies de cache suivent les mêmes principes pour ces requêtes que pour les jointures.

- Les requêtes telles que select into qui allouent plusieurs pages.

- Commande create index.

- Les opérations bulkcopy sur des tables sans index clusterisés, incluant les copies entrantes et sortantes.

- Les commandes update statistics, dbcc checktable et dbcc checkdb.

Optimiseur et choix relatifs aux caches

Si le cache pour une table ou un index contient une zone de 16 ko, l'optimiseur choisit la taille d'E/S à utiliser pour les pages de données et d'index de niveau feuille en fonction du nombre de pages à lire et des taux de clusterisation de la table ou de l'index.

Les connaissances de l'optimiseur sont limitées à la requête en cours d'analyse et à certaines statistiques sur la table et le cache. Il ne dispose pas d'informations sur le nombre d'autres requêtes utilisant simultanément le même cache de données. Il ne dispose pas non plus de statistiques sur le fait que le stockage des données de la table est fragmenté de telle sorte que les E/S étendues ou la prélecture asynchrone serait moins efficace.

Dans certains cas, cette combinaison de facteurs peut aboutir à des E/S trop importantes. Par exemple, les utilisateurs peuvent se retrouver avec des E/S plus importantes et des performances plus faibles si des requêtes simultanées donnant lieu à des jeux de résultats importants utilisent une zone mémoire très petite.

Choix d'une bonne combinaison de tailles d'E/S pour un cache

Vous pouvez configurer jusqu'à quatre zones dans un cache de données mais, dans la plupart des cas, les caches pour les objets individuels fonctionnent mieux avec des zones de 2 k et de 16 k seulement. Un cache contenant une base de données dont le journal n'est pas lié à un cache séparé doit aussi disposer d'une zone configurée de manière à correspondre à la taille des E/S du journal configurée pour la base de données (souvent, la taille d'E/S du journal la plus adaptée est 4 k).

Réduction des conflits de verrous avec les partitions de caches

Sur les systèmes SMP, à mesure que le nombre de moteurs et des tâches augmente, les conflits de verrous d'attente sur le cache de données augmentent également. Chaque fois qu'une tâche doit accéder au cache pour y rechercher une page ou relier une page à la chaîne LRU/MRU, elle maintient le verrou d'attente pour éviter que d'autres tâches ne modifient le cache en même temps.

Avec plusieurs moteurs et utilisateurs, les tâches passent leur temps à attendre l'accès au cache. L'ajout de partitions dans le cache crée des zones protégées chacune par son propre verrou d'attente. Lorsqu'une page doit être lue ou recherchée dans le cache, une fonction de hachate est appliquée à l'ID de base de données et à l'ID de page afin d'identifier la partition qui contient la page.

Le nombre de partitions de cache est toujours une puissance de 2. Chaque fois que vous l'augmentez, vous réduisez le conflit de verrous d'attente d'environ la moitié (1/2). Si ce dernier est supérieur à 10 ou 15 %, augmentez le nombre de partitions dans le cache. L'exemple suivant crée 4 partitions dans le cache de données par défaut :

```
sp_cacheconfig "default data cache",
"cache_partition=4"
```

Vous devez redémarrer le serveur pour que les modifications au niveau du partitionnement du cache prennent effet.

Pour plus d'informations sur la configuration des partitions de cache, reportez-vous au *Guide d'administration système*.

Pour plus d'informations sur le contrôle des conflits de verrous d'attente sur le cache avec sp_sysmon, reportez-vous à la section "Conflits de verrouillage d'attente dans le cache", page 1055.

Chaque zone du cache est partitionnée en une chaîne de pages LRU/MRU distincte, chaque page ayant son marqueur de vidage.

Stratégies et principes de remplacement dans le cache

L'optimiseur d'Adaptive Server utilise deux stratégies de remplacement du cache dans le but de conserver les pages souvent utilisées dans le cache et de vider du cache les pages les moins souvent utilisées. Pour certains caches, vous pouvez envisager de définir les règles de remplacement du cache pour limiter l'overhead.

Stratégies

Les stratégies de remplacement déterminent l'endroit où la page est placée dans le cache après avoir été lue sur le disque. L'optimiseur décide de la stratégie de remplacement à utiliser pour chaque requête. Les deux stratégies sont les suivantes :

- la stratégie de lecture-élimination (ou remplacement MRU), qui lie les derniers buffers lus au marqueur de vidage dans la zone ;
- la stratégie de remplacement LRU, qui lie les derniers buffers lus à l'extrémité MRU.

Les stratégies de remplacement peuvent modifier le taux de présence dans le cache pour votre combinaison de requêtes :

- Les pages lues dans le cache selon la stratégie de lecture-élimination y restent beaucoup moins longtemps que les requêtes lues à l'extrémité MRU du cache. Si une même page doit être lue à nouveau (par exemple, si la même requête est ré-exécutée peu de temps après), elle le sera probablement depuis le disque.
- Les pages lues dans le cache selon la stratégie de lecture-élimination ne déplacent pas celles qui résident déjà dans le cache avant le marqueur de vidage. Cela signifie que les pages qui se trouvent dans le cache avant le marqueur de vidage ne seront pas remplacées par celles qui ne seront lues qu'une seule fois par une requête.

Pour plus d'informations sur la spécification d'une stratégie de cache dans les requêtes ou sur la définition de valeurs pour des tables, reportez-vous aux sections "Spécification de la stratégie de caches", page 456 et "Contrôle des E/S étendues et des stratégies de cache", page 457.

Principes

L'administrateur système peut spécifier si un cache doit être géré comme une liste de pages chaînées MRU/LRU (*principe strict*) ou s'il est possible d'utiliser le *principe de remplacement LRU souple*. Les deux principes de remplacement sont :

- Le principe de remplacement strict remplace les plus anciennes pages utilisées dans la zone et lie les pages lues en dernier au début (extrémité MRU) de la chaîne de pages.
- Le principe de remplacement souple tente d'éviter de remplacer une page utilisée récemment, mais sans l'overhead de maintenir les buffers dans l'ordre LRU/MRU.

Le principe de remplacement du cache strict est appliqué par défaut. Le principe de remplacement souple doit être utilisé uniquement lorsque les deux conditions suivantes sont vérifiées :

- les buffers sont peu ou pas remplacés dans le cache,
- les données ne sont pas ou peu mises à jour.

Le principe LRU souple évite l'overhead nécessaire pour mettre à jour le cache dans l'ordre MRU/LRU. Sur les systèmes SMP, où les copies des pages en cache peuvent résider directement dans des caches sur les CPU, le remplacement LRU souple peut réduire le débit sur le bus qui relie les CPU.

Si vous avez créé un cache destiné à contenir la totalité, ou la plupart, de certains objets et que le taux de présence dépasse 95 %, le principe de remplacement souple peut améliorer légèrement les performances.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Configuration du remplacement LRU souple pour les journaux des bases de données

Les pages de journal sont remplies d'enregistrements et immédiatement enregistrées sur disque. Lorsque les applications comprennent des triggers, des mises à jour différées ou des annulations de transactions, certaines pages de journal peuvent être lues, mais ce sont généralement des pages utilisées très récemment, qui se trouvent encore dans le cache.

Etant donné que l'accès à ces pages contenues dans le cache provoque leur déplacement à l'extrémité MRU du cache utilisant le principe de remplacement strict, les performances des caches de journal pourront bénéficier du remplacement LRU souple.

Remplacement LRU souple pour les tables de consultation et les index

Les caches définis par l'utilisateur dont la taille est établie pour qu'ils puissent contenir des index et des tables de consultation fréquemment sollicitées sont particulièrement adaptés à l'application du principe LRU souple. Si c'est le cas pour un cache mais si vous trouvez que le taux de présence y est légèrement plus bas que le taux conseillé de 95 %, déterminez si une légère augmentation de la taille du cache peut fournir suffisamment d'espace pour que celui-ci contienne la totalité de la table ou de l'index.

Recommandations relatives au cache de données nommé

Ces recommandations peuvent améliorer les performances à la fois sur les serveurs comportant une seule CPU et sur les serveurs multiprocesseurs :

- Adaptive Server écrit les pages du journal en fonction de la taille logique d'une page. Des pages de journal plus grandes peuvent réduire le taux des écritures à commit partagé pour les pages du journal.

  Le partage de commits se produit quand, au lieu d'exécuter plusieurs commits distincts, Adaptive Server exécute un batch de commits en une seule fois. Les caches de journaux utilisateur par processus sont dimensionnés en fonction de la taille de la page logique et du paramètre de configuration user log cache size. La taille par défaut du cache des journaux utilisateur correspond à une page logique.

  Pour les transactions qui génèrent plusieurs enregistrements du journal, le temps nécessaire pour vider le cache des journaux utilisateur est légèrement supérieur pour des tailles de page logique supérieures. Cependant, étant donné que les tailles des caches des journaux sont également supérieures, Adaptive Server n'a pas besoin d'exécuter autant de vidages des caches des journaux sur la page des journaux pour les transactions longues.

  Pour plus d'informations, reportez-vous au guide *Utilitaires*.

- Créez un cache nommé pour tempdb et configurez-le pour des E/S de 16 k utilisables par des requêtes select into et des tris.

- Créez un cache nommé pour les journaux si vos bases de données sont fréquemment utilisées. Configurez les zones de ce cache afin qu'elles correspondent à la taille des E/S du journal définie à l'aide de sp_logiosize.

  Pour plus d'informations, reportez-vous à la section "Choix de la taille des E/S du journal de transactions", page 347.

- Si une table ou son index est de petite taille et constamment utilisé, créez un cache pour cet objet ou pour quelques objets.

- Pour les caches dont le taux de présence excède 95 %, choisissez le principe de remplacement de cache LRU souple, surtout si vous utilisez plusieurs moteurs.

- Conservez des tailles de cache et de zones proportionnelles aux objets et aux requêtes qui utilisent ce cache :
  - Si 75 % des tâches effectuées sur votre serveur ne concernent qu'une seule base de données, celle-ci doit se voir allouer environ 75 % du cache de données dans un cache spécialement créé pour cette base, des caches créés pour ses tables et ses index les plus actifs ou le cache de données par défaut.
  - Si environ 50 % des tâches de votre base de données peuvent utiliser des E/S étendues, configurez approximativement 50 % du cache dans une zone mémoire de 16 ko.
- Il est préférable de considérer le cache comme une ressource partagée plutôt que d'essayer de gérer les besoins en cache de chaque table et chaque index.

  Démarrez l'analyse et le test du cache au niveau de la base de données, en vous concentrant sur les tables et les objets avec des besoins importants en matière d'E/S ou avec des priorités élevées pour les applications, ainsi que sur les tables et les objets à usage spécifique, tels que tempdb et les journaux de transactions.
- Sur les serveurs SMP, utilisez plusieurs caches pour éviter un conflit au niveau du verrou d'attente du cache :
  - Utilisez un cache séparé pour le journal de transactions des bases de données les plus sollicitées et utilisez des caches séparés pour les tables et les index pour lesquels l'accès est fréquent.
  - Si le conflit de verrous d'attente dépasse 10 % sur le cache, divisez ce dernier en plusieurs caches ou utilisez des partitions de cache.

    Utilisez sp_sysmon périodiquement pendant les heures de pointe pour détecter des conflits liés au cache.

    Pour plus d'informations, reportez-vous à la section "Conflits de verrouillage d'attente dans le cache", page 1055.
  - Définissez un remplacement LRU souple pour les caches dont le taux de présence est supérieur à 95 %, tels que ceux qui sont configurés pour contenir une table ou un index entier.

Définition de la taille des caches pour des objets spécifiques, pour *tempdb* et pour les journaux de transaction

La création de caches pour tempdb, pour les journaux de transactions et pour quelques tables ou index que vous souhaitez conserver intégralement dans le cache peut limiter les conflits de verrouillage et améliorer le taux de présence dans le cache.

Définition de la taille des caches pour des tables ou des index spécifiques

Vous pouvez utiliser sp_spaceused pour déterminer la taille des tables ou index que vous souhaitez conserver entièrement dans le cache. Si vous connaissez la vitesse d'augmentation de la taille de ces tables, prévoyez de l'espace dans le cache pour leur croissance. Pour visualiser la taille de tous les index pour une table, utilisez :

```
sp_spaceused nom_table, 1
```

Etude des besoins en cache pour *tempdb*

Considérez votre utilisation de tempdb :

- Estimez la taille des tables temporaires et des tables de travail générées par vos requêtes.

  Considérez le nombre de pages générées par les requêtes select into.

  Ces requêtes peuvent utiliser des E/S de 16 k, vous pouvez donc vous aider de ces informations pour dimensionner une zone de 16 k pour le cache de tempdb.

- Estimez la durée des tables temporaires et des tables de travail.
- Estimez la fréquence d'exécution des requêtes qui créent des tables temporaires et des tables de travail.

  Essayez d'estimer le nombre d'utilisateurs simultanés, notamment pour les requêtes qui génèrent des jeux de résultats très importants dans tempdb.

Avec ces informations, vous pouvez effectuer une estimation approximative du nombre d'E/S simultanées dans tempdb. En fonction de vos autres besoins en cache, vous pouvez choisir de dimensionner tempdb de façon que presque toutes les activités de tempdb puissent avoir lieu dans le cache et que peu de tables temporaires soient réellement écrites sur disque.

Dans la plupart des cas, les deux premiers méga-octets de tempdb sont stockés sur le device master, avec de l'espace supplémentaire sur un autre device logique. Vous pouvez utiliser sp_sysmon pour contrôler ces devices afin de pouvoir déterminer les taux d'E/S physiques.

Etude des besoins en cache pour les journaux de transactions

Sur les systèmes SMP avec des taux de transactions élevés, la liaison du journal de transactions à son propre cache peut limiter grandement les conflits de verrouillage du cache de données par défaut. Dans de nombreux cas, le cache peut être très petit.

La page en cours du journal de transactions étant écrite sur disque lors de la validation des transactions, ne cherchez pas à éviter les écritures lorsque vous dimensionnez le cache ou la zone de ce journal. Essayez plutôt de dimensionner le journal de façon à réduire le nombre de fois que les processus relisant des pages du journal doivent accéder au disque du fait que ces pages ont été supprimées du cache.

Les processus d'Adaptive Server devant lire les pages du journal sont :

- Les triggers qui utilisent les tables inserted et deleted, qui sont créées à partir du journal de transactions lorsque le trigger interroge les tables.
- Les mises à jour, les insertions et les suppressions différées, puisque ces actions nécessitent une nouvelle lecture du journal pour appliquer les modifications aux tables ou aux index.
- Les transactions qui sont annulées, puisqu'il faut accéder aux pages du journal pour annuler les modifications.

Lorsque vous dimensionnez un cache pour un journal de transactions :

- Examinez la durée des processus qui requièrent une nouvelle lecture des pages du journal.

  Estimez la durée des mises à jours différées et des triggers les plus longs.

  Si certaines de vos transactions longues sont annulées, vérifiez leur temps d'exécution.
- Estimez le taux de croissance du journal au cours de cette période.

  Vous pouvez vérifier la taille de votre journal de transactions régulièrement à l'aide de sp_spaceused pour estimer la vitesse de croissance du journal.

Utilisez ces deux estimations pour dimensionner le cache du journal. Par exemple, si la mise à jour différée la plus longue est de 5 minutes et que le journal de transactions pour la base de données augmente à la vitesse de 125 pages par minute, 625 pages sont allouées au journal au cours de l'exécution de cette transaction.

Si quelques transactions ou requêtes ont un temps d'exécution spécialement long, vous pouvez dimensionner le journal en fonction du temps moyen et non du temps maximum.

Choix de la taille des E/S du journal de transactions

Lorsqu'un utilisateur effectue des opérations qui nécessitent une journalisation, les enregistrements du journal sont d'abord stockés dans un "cache du journal de l'utilisateur" jusqu'à ce que certains événements vident les enregistrements du journal de l'utilisateur à la page du journal de transactions courante dans le cache. Les enregistrements d'un journal sont vidés :

- lorsqu'une transaction se termine,
- lorsque le cache du journal de l'utilisateur est plein,
- lorsque la transaction modifie les tables dans une autre base de données,
- lorsqu'un autre processus doit écrire une page référencée dans le cache du journal de l'utilisateur,
- lors de certains événements système.

Pour économiser les écritures disque, Adaptive Server conserve des pages du journal de transactions partiellement remplies pendant un temps très court pour permettre aux enregistrements de plusieurs transactions d'être écrites sur disque simultanément. Ce processus s'appelle *validation de groupe*.

Dans des environnements possédant des taux de transactions élevés ou des transactions qui créent des enregistrements importants dans le journal, les pages de journal de transactions de 2 k se remplissent rapidement et une grande proportion des écritures du journal sont dues à des pages de journal pleines plutôt qu'à des validations de groupe.

La création d'une zone de 4 k pour le journal de transactions peut réduire considérablement le nombre d'écritures du journal dans ces environnements.

sp_sysmon fournit des informations sur le rapport entre le nombre d'écritures du journal de transactions et le nombre d'allocations de journal de transactions. Vous devez essayer d'utiliser des E/S de journal de 4 ko si toutes ces conditions sont vérifiées :

- votre base de données utilise des E/S de journal de 2 k,
- le nombre d'écritures du journal par seconde est élevé,
- le nombre moyen d'écritures par page de journal est légèrement supérieur à 1.

Voici un échantillon de résultats qui montre qu'une taille d'E/S plus grande peut améliorer les performances :

```
                          per sec     per xact    count  % of total
Transaction Log Writes       22.5        458.0     1374         n/a
Transaction Log Alloc        20.8        423.0     1269         n/a
Avg # Writes per Log Page     n/a          n/a  1.08274         n/a
```

Pour plus d'informations, reportez-vous à la section "Ecritures dans le journal de transactions", page 1023.

Configuration de grandes tailles d'E/S de journal

La taille des E/S du journal pour chaque base de données est consignée dans le journal d'erreurs du serveur lorsqu'Adaptive Server démarre. Vous pouvez aussi utiliser sp_logiosize.

Pour consulter la taille de la base de données courante, exécutez sp_logiosize sans aucun paramètre. Pour consulter la taille de toutes les bases de données sur le serveur et le cache utilisé par le journal, utilisez :

```
sp_logiosize "all"
```

Pour que vous puissiez fixer à 4 k (valeur par défaut) la taille des E/S du journal pour une base de données, cette base doit être en cours d'utilisation. La commande suivante fixe la taille à 4 ko :

```
sp_logiosize "default"
```

Par défaut, Adaptive Server fixe à 4 ko la taille des E/S du journal pour les bases de données utilisateur. Si aucune zone de 4 ko n'est disponible dans le cache utilisé par le journal, la valeur utilisée pour les E/S est 2 ko.

Si une base de données est liée à un cache, tous les objets qui ne sont pas explicitement liés à d'autres caches utilisent le cache de cette base. La table syslogs rentre dans cette catégorie.

Pour lier syslogs à un autre cache, vous devez d'abord placer la base de données en mode mono-utilisateur à l'aide de sp_dboption, puis utiliser la base de données et exécuter sp_bindcache. Par exemple :

```
sp_bindcache pubs_log, pubtune, syslogs
```

Conseils complémentaires d'optimisation pour les caches du journal

Pour une optimisation approfondie après la configuration d'un cache pour le journal, consultez le résultat de sp_sysmon. Consultez notamment :

- le cache utilisé par le journal,
- le disque sur lequel le journal est stocké,
- le nombre moyen d'écritures par page du journal.

Lorsque vous consultez la section concernant le cache du journal, vérifiez les valeurs "Cache Hits" et "Cache Misses" pour déterminer si la plupart des pages requises pour les opérations différées, les triggers et les annulations se trouvent dans le cache.

Dans la section "Disk Activity Detail", consultez le nombre de "lectures" effectuées pour connaître la fréquence d'accès au disque des tâches qui doivent relire le journal.

Détermination de la taille des zones de données en fonction des plans d'exécution de requête et des E/S

Divisez un cache en zones en tenant compte de la proportion des E/S effectuées par les requêtes qui utilisent les tailles d'E/S correspondantes. Si la plupart de vos requêtes peut tirer parti d'E/S de 16 ko et que vous configurez un très petit cache de 16 ko, il se peut que vous constatiez une dégradation des performances.

La plupart de l'espace de la zone de 2 k n'est pas utilisé et la zone de 16 k a un taux élevé de rotation. Le taux de présence dans le cache est considérablement réduit.

Le problème sera plus grave avec des requêtes de jointure à boucle imbriquée, qui doivent lire de façon répétitive la table interne sur le disque.

Le choix des tailles de zones adéquates requiert :

- une connaissance approfondie de la combinaison d'applications et de la taille des E/S que vos requêtes peuvent utiliser,
- une étude et une optimisation consciencieuses à l'aide d'outils de contrôle pour vérifier l'utilisation du cache, les taux de présence et les E/S disque.

Vérification de la taille des E/S pour les requêtes

Vous pouvez examiner les plans d'exécution de requête et les statistiques d'E/S afin de déterminer quelles requêtes sont susceptibles d'effectuer de nombreuses E/S et la quantité d'E/S qu'elles effectuent. Ces informations peuvent servir de base pour l'estimation de la quantité d'E/S de 16 ko que ces requêtes devraient effectuer avec une zone mémoire de 16 ko. Les E/S sont effectuées en fonction des tailles des pages logiques, pour une page de 2 k les E/S sont par tailles de 2 k, pour une page de 8 k elles sont par tailles de 8 k, comme indiqué :

Tailles des pages logiques	Zone de mémoire
2 ko	16 ko
4 ko	64 k
8 ko	128 k
16 ko	256 k

Un autre exemple, une requête qui balaie une table et effectue 800 E/S physiques en utilisant une zone de 2 k devrait effectuer environ 100 E/S de 8 k chacune.

Reportez-vous à la section "E/S étendues et performances", page 337 pour une liste des types de requêtes.

Pour tester vos estimations, vous devez configurer les zones, puis exécuter vos requêtes individuelles et votre combinaison de requêtes cible afin de déterminer des tailles de zones optimales. Le choix d'une bonne taille de départ pour votre premier test avec des E/S de 16 ko dépend d'une bonne connaissance des types de requêtes dans votre combinaison d'applications.

Cette estimation est d'autant plus importante si vous configurez une zone de 16 ko pour la première fois sur un serveur de production actif. Faites la meilleure estimation possible des utilisations simultanées du cache.

Quelques recommandations :

- Si vous voyez que la plupart des E/S proviennent de requêtes ponctuelles utilisant des index pour accéder à un petit nombre de lignes, configurez une zone de 16 ko relativement petite, d'environ 10 à 20 % de la taille du cache.
- Si vous estimez qu'un pourcentage élevé des E/S utilisera la zone de 16 k, configurez de 50 à 75 % du cache pour des E/S de 16 k.

Les requêtes qui utilisent des E/S de 16 k comprennent toute requête qui balaie des tables, utilise l'index clusterisé pour des recherches séquentielles et order by, et les requêtes qui effectuent des balayages avec ou sans correspondance sur les index non clusterisés.

- Si vous n'êtes pas sûr de la taille des E/S qui seront utilisées par vos requêtes, configurez environ 20 % de l'espace du cache dans une zone de 16 ko, et utilisez showplan et statistics i/o pendant l'exécution de vos requêtes.

  Examinez dans les résultats de showplan le message "Using 16 K I/O". Vérifiez les résultats de statistics i/o pour voir combien d'E/S ont été effectuées.

- Si vous pensez que votre combinaison d'applications typique utilise à la fois des E/S de 16 k et des E/S de 2 k, configurez de 30 à 40 % de votre espace de cache pour les E/S de 16 k.

  Votre valeur optimale peut être plus grande ou plus petite, en fonction de la combinaison et de la taille des E/S choisies par la requête.

  Si des E/S à la fois de 2 k et de 16 k accèdent à de nombreuses tables, Adaptive Server ne peut pas utiliser d'E/S de 16 k, si le cache de 2 k contient une page de l'extent. Il effectuera des E/S de 2 ko sur les autres pages de l'extent dont la requête a besoin. Cela s'ajoute aux E/S dans le cache de 2 k.

Après la configuration des E/S de 16 k, vérifiez l'utilisation du cache et contrôlez les E/S des devices concernés, en utilisant sp_sysmon ou Adaptive Server Monitor. Utilisez également showplan et statistics io pour observer vos requêtes.

- Recherchez particulièrement les requêtes de jointure où une table interne utilise des E/S de 16 k, et où la table est régulièrement balayée selon la stratégie de lecture-élimination (MRU).

  Cela arrive quand des deux tables ne rentre totalement dans le cache. Si l'augmentation de la taille de la zone de 16 ko permet à la table interne de rentrer totalement dans le cache, les E/S peuvent être réduites de manière significative. Vous pourriez également penser à lier les deux tables à des caches séparés.

- Recherchez les E/S de 16 ko excessives par rapport à la taille des tables dans les pages.

Par exemple, si vous avez une table de 8000 pages et si un balayage de table faisant intervenir des E/S de 16 k effectue bien plus de 1000 E/S pour lire cette table, vous pouvez améliorer les résultats en recréant l'index clusterisé de cette table.

- Recherchez les fois où des E/S étendues sont refusées. Cela arrive souvent lorsque des pages se trouvent déjà dans la zone de 2 ko, de telle sorte que cette zone sera utilisée pour le reste des E/S de la requête.

  Pour une liste complète des raisons pour lesquelles les E/S étendues ne peuvent pas être utilisées, reportez-vous à la section "Omission de la spécification de prefetch", page 454.

Configuration de la taille de vidage des buffers

Vous pouvez configurer la zone de vidage pour chaque zone de chaque cache. Si la taille de vidage est trop élevée, Adaptive Server risque d'effectuer des écritures non nécessaires. Si elle est trop basse, Adaptive Server ne pourra peut-être pas trouver de buffer non modifié à l'extrémité de la chaîne des buffers et devra attendre l'achèvement des E/S avant de continuer. Généralement, la valeur par défaut des tailles de vidage est correcte et ne doit être ajustée que dans de grandes zones ayant des taux importants de modification de données.

Adaptive Server alloue des zones de buffers en unités de pages logiques. Par exemple, sur un serveur qui utilise des pages logiques de 2 k, 8 Mo sont alloués au cache de données par défaut. Par défaut, cela représente approximativement 4096 buffers.

Si vous allouez toujours 8 Mo au cache de données par défaut sur un serveur qui utilise une taille de page logique de16 k, le cache de données par défaut est d'environ 512 buffers. Sur un système sollicité, cette petite quantité de buffers peut comporter la présence permanente d'un buffer dans la zone de vidage, causant un ralentissement des tâches demandant des buffers non modifiés.

En général, pour obtenir les mêmes caractéristiques de gestion des buffers sur des tailles de pages supérieures, telles que des tailles de pages logiques de 2 k, vous devez adapter la taille des caches à la taille la plus grande de la page. Autrement dit, si vous augmentez la taille de votre page logique de 4 fois, la taille de vos cache et zone doivent augmenter également de 4 fois.

Les requêtes exécutant des E/S étendues, des lectures et écritures basées sur extents et ainsi de suite tirent parti des tailles les plus grandes de pages logiques. Cependant, étant donné que les catalogues continuent à présenter des pages verrouillées, le conflit et le blocage au niveau des pages des catalogues système sont accrus.

La copie de ligne et de colonne pour les tables DOL comporte un ralentissement supérieur en cas de colonnes larges. L'allocation de la mémoire afin de supporter les lignes et les colonnes larges ralentissent marginalement le serveur.

Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Overhead de la configuration des zones et de la liaison d'objets

Il est préférable d'effectuer la configuration des zones de mémoire et la liaison des objets aux caches pendant les périodes de faible activité, parce qu'elle risque de réduire les performances dans un système de production.

Overhead de la configuration des zones

Lorsqu'une zone est créée, supprimée ou modifiée, les plans de toutes les procédures stockées et de tous les triggers qui utilisent des objets liés au cache sont recompilés lors de leur exécution suivante. Si une base de données est liée au cache, cela affecte tous ses objets.

Le déplacement de buffers d'une zone à une autre nécessite un certain overhead.

Overhead de la liaison au cache

Lorsque vous liez ou déliez un objet, toutes les pages de l'objet contenues alors dans le cache sont vidées sur le disque (si elles sont modifiées) ou supprimées du cache (si elles ne sont pas modifiées) pendant le processus de liaison.

La prochaine fois que les pages sont nécessaires à des requêtes utilisateur, elles doivent être lues à nouveau depuis le disque, ce qui ralentit les performances des requêtes.

Adaptive Server pose un verrou exclusif sur la table ou l'index pendant le vidage du cache ; la liaison peut donc ralentir l'accès d'autres utilisateurs aux objets. Le processus de liaison devra peut-être attendre l'achèvement des transactions pour poser le verrou.

Remarque Le fait de lier des objets aux caches et de les délier les retire de la mémoire, ce qui peut être utile pour l'optimisation des requêtes pendant le développement et les tests.

Si vous devez vérifier les E/S physiques pour une table particulière, et si vos efforts d'optimisation précédents ont placé des pages dans le cache, vous pouvez délier et relier l'objet. Au prochain accès à cette table, toutes les pages utilisées par la requête devront être lues dans le cache.

Les plans de toutes les procédures stockées et de tous les triggers qui utilisent les objets liés sont recompilés à leur exécution suivante. Si une base de données est liée au cache, cela affecte tous ses objets.

Maintien des performances du cache de données pour les E/S étendues

Lorsque des tables sans index, des index clusterisés ou des index non clusterisés viennent d'être créés, ils montrent des performances optimales lors de l'utilisation d'E/S étendues. Avec le temps, les effets d'annulations, de ruptures de pages et de libérations et réallocations de pages répétées peuvent provoquer une augmentation du coût des E/S. optdiag fournit l'information "Large I/O efficiency" concernant les tables et les index.

Lorsque cette valeur est égale à 1 ou proche de 1, l'E/S étendue est très performante. A mesure que cette valeur diminue, des E/S supplémentaires sont nécessaires pour accéder aux pages de données et l'E/S étendue risque alors de placer dans le cache des pages qui ne sont pas nécessaires à la requête.

Dans ce cas, lorsque l'efficacité des E/S étendues baisse ou que l'activité dans la zone augmente du fait de l'accroissement du nombre d'E/S de 16 k, prévoyez de recréer des index.

Lorsque l'efficacité des E/S étendues diminue, vous pouvez :

- Exécuter reorg rebuild sur les tables verrouillées au niveau des pages de données seulement. Vous pouvez aussi exécuter reorg rebuild sur l'index des tables verrouillées au niveau des pages de données seulement.
- Pour les tables verrouillées au niveau de toutes les pages, supprimez les index puis créez-les de nouveau.

Pour plus d'informations, reportez-vous à la section "Exécution de reorg sur des tables et des index", page 390.

Diagnostic d'un nombre d'E/S excessifs

Une requête qui effectue des E/S étendues peut nécessiter plus de lectures que prévues pour plusieurs raisons :

- Le cache utilisé par la requête comprend un cache de 2 ko et de nombreux autres processus ont déplacé des pages de la table dans ce cache de 2 ko.

  Si Adaptive Server effectue des E/S de 16 k et trouve qu'une des pages dont il a besoin se trouve déjà dans le cache de 2 k, il effectuera des E/S de 2 k sur les autres pages de l'extent qui sont nécessaires à la requête.

- Le premier extent de chaque unité d'allocation stocke la page d'allocation, de sorte que si une requête a besoin d'accéder aux 255 pages de l'extent, elle doit effectuer des E/S de 2 ko sur les 7 pages qui partagent l'extent avec la page d'allocation.

  Les 31 autres extents pourront être lus avec des E/S de 16 ko. Le nombre minimal de lectures pour une unité complète d'allocation sera donc toujours 38, et non 32.

- Dans des index non clusterisés et des index clusterisés sur des tables verrouillées au niveau des pages de données seulement, un extent peut stocker à la fois des pages de niveau feuille et des pages de l'index de niveau élevé. Des E/S de 2 ko sont effectuées sur les niveaux supérieurs des index et sur les pages de niveau feuille si quelques-unes d'entre elles sont nécessaires à la requête.

  Lorsqu'une couverture par index de niveau feuille procède à des E/S de 16 k, il se peut que certaines pages de certains extents se trouvent dans le cache de 2 k. Le reste des pages de l'extent est lu par des E/S de 2 k.

Utilisation de *sp_sysmon* pour vérifier les performances des E/S étendues

Les résultats de sp_sysmon pour chaque cache de données comprennent des informations qui peuvent vous aider à déterminer l'efficacité des E/S étendues :

- La section "Utilisation des E/S étendues", page 1050 indique le nombre d'E/S étendues effectuées et refusées et fournit des statistiques récapitulatives.
- "Détail des E/S étendues", page 1062 indique le nombre total de pages qui ont été lues dans le cache par une E/S étendue et le nombre de pages auxquelles on a réellement accédé pendant qu'elles étaient dans le cache.

Vitesse de reprise

Au fur et à mesure que les utilisateurs modifient des données dans Adaptive Server, seul le journal de transactions est écrit immédiatement sur disque, afin d'assurer la reprise de ces données ou transactions. Les pages de données et d'index modifiées restent dans le cache de données jusqu'à ce que l'un des événements suivants les écrive sur disque :

- Le processus de point de reprise s'active, détermine que les pages de données et d'index modifiées pour une base de données spécifique doivent être écrites sur disque et écrit toutes les pages modifiées dans chaque cache utilisé par la base de données.

  La combinaison de la valeur de recovery interval et du taux de modifications des données sur votre serveur détermine le nombre de fois où le processus de point de reprise écrit les pages modifiées sur disque.
- Au fur et à mesure que les pages se déplacent dans la zone de vidage de buffer du cache, les pages modifiées sont automatiquement écrites sur disque.
- Adaptive Server dispose de cycles CPU et d'une capacité d'E/S disque supplémentaires entre les transactions utilisateur et la tâche de l'housekeeper utilise ce temps pour écrire les buffers modifiés sur disque.

- La reprise n'est effectuée que sur le cache de données par défaut.
- Un utilisateur émet une commande checkpoint.

La combinaison des points de reprise, de l'housekeeper et des écritures commencées au marqueur de vidage a deux avantages majeurs :

- De nombreuses transactions peuvent modifier une page dans le cache ou lire la page dans le cache mais une seule écriture physique sera effectuée.
- Adaptive Server effectue de nombreuses écritures physiques lorsque les E/S ne causent pas de conflits avec les processus utilisateur.

Optimisation de l'intervalle de reprise

L'intervalle de reprise par défaut sur Adaptive Server est de 5 minutes par base de données. La modification de cet intervalle peut nuire aux performances car il peut influer sur le nombre de fois où Adaptive Server écrit des pages sur disque.

Le tableau 14-2 illustre l'influence de la modification de l'intervalle de reprise sur votre système.

Tableau 14-2 : Influence de l'intervalle de reprise sur les performances et sur le temps de reprise

Valeur	Influence sur les performances	Influence sur la reprise
Faible	Peut entraîner plus de lectures et d'écritures et diminuer la vitesse de traitement. Adaptive Server va écrire plus souvent des pages modifiées sur le disque. Les "pointes" des E/S de point de reprise seront plus petites.	La période de reprise sera très courte.
Elevée	Réduit les écritures et améliore le débit du système. Les pointes d'E/S de point de reprise seront plus élevées.	La reprise automatique risque d'être plus longue au démarrage. Adaptive Server devra peut-être ré-appliquer un grand nombre d'enregistrements du journal de transactions aux pages de données.

Pour plus d'informations sur la définition de l'intervalle de reprise, reportez-vous au *Guide d'administration système*. La procédure sp_sysmon fait état du nombre et de la durée des points de reprise.

Pour plus d'informations, reportez-vous à la section "Gestion de la restauration", page 1065.

Influence de la tâche "housekeeper" sur le temps de reprise

La tâche "housekeeper" Adaptive Server commence automatiquement à nettoyer les buffers modifiés pendant les cycles de veille du serveur. Si la tâche peut vider toutes les zones de buffers actives dans les caches configurés, elle active le processus de point de reprise. Cela peut résulter en des points de reprises plus rapides et en des temps de reprise de base de données plus courts.

Les administrateurs système peuvent utiliser le paramètre de configuration housekeeper free write percent pour optimiser ou désactiver la tâche du gestionnaire (housekeeper). Ce paramètre spécifie le pourcentage maximum d'augmentation d'écritures de bases de données résultant des écritures effectuées par la tâche du gestionnaire de tâches.

Pour plus d'informations sur l'optimisation de la tâche "housekeeper" et de l'intervalle de reprise, reportez-vous à la section "Gestion de la restauration", page 1065.

Audit et performances

Un audit trop important peut diminuer les performances de la façon suivante :

- Les enregistrements d'audit sont d'abord écrits dans une file d'attente en mémoire, puis dans la base de données sybsecurity. Si la base de données partage un disque avec d'autres bases de données très utilisées, les performances peuvent baisser.
- Lorsque la file d'attente d'audit en mémoire est saturée, les processus utilisateur qui génèrent les enregistrements d'audit se mettent en veille. Pour plus d'informations, reportez-vous à la figure 14-4, page 359.

Définition de la taille de la file d'attente d'audit

La taille de la file d'attente d'audit peut être définie par un responsable de la sécurité du système. La configuration par défaut est la suivante :

- Un seul enregistrement d'audit requiert entre 32 et 424 octets.

  Cela signifie qu'une seule page de données stocke entre 4 et 80 enregistrements.
- La taille par défaut de la file d'attente d'audit est de 100 enregistrements, ce qui nécessite environ 42 ko.

  La taille minimale est de 1 enregistrement, la taille maximale est de 65 335 enregistrements.

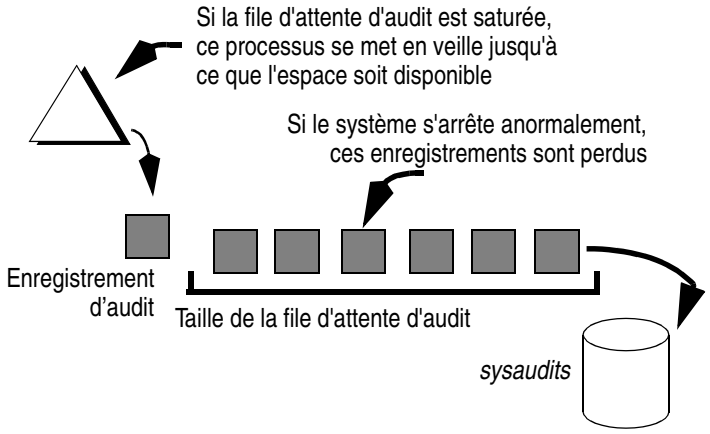
Vous devez faire des compromis pour définir la taille de la file d'attente d'audit, comme le montre la figure 14-4.

Si celle-ci est importante, vous évitez la mise en veille de processus utilisateur, mais vous risquez de perdre les enregistrements d'audit en mémoire, en cas de panne système. Le nombre maximal d'enregistrements qui peuvent être perdus correspond au nombre maximal d'enregistrements qui peuvent être enregistrés dans la file d'attente d'audit.

Si la sécurité est votre préoccupation première, définissez une petite file d'attente. Si vous pouvez risquer de perdre plus d'enregistrements d'audit et si vous avez besoin de performances élevées, définissez une file d'attente plus importante.

L'augmentation de la taille de la file d'attente d'audit en mémoire prélève de la mémoire sur la quantité de mémoire totale allouée au cache de données.

Figure 14-4 : Compromis en matière d'audit et de performances



Recommandations concernant les performances des audits

- Un audit trop important ralentit les performances générales du système. N'effectuez un audit que sur les événements que vous avez besoin d'analyser.
- Si possible, placez la base de données sysaudits sur son propre device. Dans le cas contraire, placez-la sur un device qui n'est pas utilisé pour vos applications les plus stratégiques.

CHAPITRE 15

Détermination de la taille des tables et des index

Ce chapitre explique comment déterminer la taille courante des tables et des index et évaluer la taille des tables en vue de planifier l'espace.

Il contient les sections suivantes :

Sujet	Page
Pourquoi la taille des objets est importante pour l'optimisation des requêtes	361
Outils pour la détermination de la taille des tables et des index	363
Conséquences des modifications de données sur la taille des objets	363
Utilisation de optdiag pour afficher la taille des objets	364
Utilisation de la procédure sp_spaceused pour afficher la taille d'un objet	365
Evaluation de la taille des objets à l'aide de sp_estspace	367
Evaluation de la taille des objets à l'aide des formules de calcul	369

Pourquoi la taille des objets est importante pour l'optimisation des requêtes

Pour comprendre les requêtes et le fonctionnement du système, il est essentiel que vous connaissiez la taille des tables et des index. Au cours des différentes étapes de l'optimisation, vous devez disposer des informations sur la taille pour pouvoir :

- Interpréter les résultats renvoyés par statistics io pour un plan d'exécution de requête spécifique. Le chapitre 33, "Utilisation des statistiques pour améliorer les performances", explique comment utiliser statistics io pour obtenir des informations sur les E/S exécutées.

- Comprendre le plan d'exécution de requête choisi par l'optimiseur. L'optimiseur Adaptive Server évalue, en se fondant sur le coût, les E/S physiques et logiques requises par chacune des méthodes d'accès disponibles et choisit la méthode la plus rentable. Si vous avez des doutes sur un plan d'exécution de requête particulier, exécutez la commande dbcc traceon(302) pour connaître les raisons qui ont conduit l'optimiseur à faire ce choix. Ce résultat comporte des évaluations du nombre de pages.
- Déterminer l'emplacement des objets, en fonction de leur taille et des E/S qu'ils sont susceptibles de subir. Vous pouvez obtenir des gains de performances en répartissant les objets de base de données sur plusieurs devices physiques afin que les opérations d'écriture et de lecture soient réparties uniformément. L'emplacement des objets est traité au chapitre 5, "Gestion de l'emplacement physique des données".
- Comprendre les variations des performances. Si la taille des objets augmente, leurs caractéristiques de performances risquent de changer. Prenons par exemple une table utilisée intensivement qui est mise à 100 % en mémoire cache. Si la taille de cette table devient trop importante pour le cache, les requêtes qui utilisent la table peuvent subir une baisse soudaine des performances. Ce problème se pose en particulier pour les jointures nécessitant un grand nombre de balayages.
- Élaborer des planifications de capacité. Que vous soyez sur le point de concevoir un nouveau système ou de planifier une extension d'un système existant, vous devez déterminer l'espace requis afin d'être en mesure de prévoir les besoins en mémoire et en disques physiques.
- Interpréter les résultats renvoyés par Adaptive Server Monitor et sp_sysmonsur les E/S physiques.

Outils pour la détermination de la taille des tables et des index

Adaptive Server comprend plusieurs outils fournissant des informations sur les tailles des tables ou index ou qui permettent de prévoir leurs tailles futures :

- L'utilitaire optdiag affiche les tailles et bien d'autres statistiques sur les tables et les index. Pour plus d'informations sur l'utilisation de optdiag, reportez-vous au chapitre 36, "Affichage des tables de statistiques avec optdiag"..
- La procédure système sp_spaceused fournit des informations sur la taille actuelle d'une table existante et de tous les index.
- La procédure système sp_estspace peut prévoir la taille d'une table et de ses index, à partir d'un nombre de lignes fourni comme paramètre.

Vous pouvez également calculer la taille des tables et index en utilisant les formules fournies dans ce chapitre. Les commandes sp_spaceused et optdiag fournissent l'utilisation effective de l'espace. Les autres méthodes présentées dans ce chapitre fournissent des estimations de la taille. Pour les tables partitionnées, la procédure système sp_helpartition indique le nombre de pages enregistrées sur chacune des partitions de la table. Pour plus d'informations, reportez-vous à la section "Informations sur les partitions", page 107.

Conséquences des modifications de données sur la taille des objets

Au cours du temps, les effets des modifications des données distribuées au hasard sur un groupe de tables tend à créer des pages et des pages d'index qui sont en moyenne remplies à environ 75 %. Les principales opérations en cause sont les suivantes :

- Lorsque vous insérez une ligne dans une page pleine comportant un index clusterisé, la page est divisée en deux pages remplies à environ 50 pour cent.
- Lorsque vous supprimez des lignes de tables sans index ou avec index clusterisé, l'espace utilisé dans la page diminue. Vous pouvez obtenir des pages contenant très peu de lignes, voire une seule ligne.

- Lorsque vous insérez des lignes dans des tables disposant d'index clusterisés, après avoir effectué des suppressions et des fractionnements de page, les pages fractionnées ou les pages dont certaines lignes ont été supprimées se remplissent.

Les fractionnements de page se produisent également lorsque vous insérez des lignes dans des pages d'index pleines ; ainsi les pages d'index ne sont souvent remplies qu'à 75 pour cent, sauf si les index sont supprimés et recréés régulièrement.

Utilisation de *optdiag* pour afficher la taille des objets

La commande optdiag affiche des statistiques sur les tables, les index et les colonnes, y compris la taille des tables et des index. Si vous devez optimiser les requêtes, optdiag est le meilleur outil vous permettant d'afficher toutes les statistiques nécessaires. L'exemple suivant concerne la table titles de la base de données pubtune :

```
Propriétaire de la table :              "dbo"

Statistiques de la table :              "titles"

     Nombre de pages de données :       662
     Nombre de pages vides :            10
     Nombre de lignes de données :      4986.0000000000000000
     Nombre de lignes redirigées :      18.0000000000000000
     Nombre de lignes supprimées :      87.0000000000000000
     Nombre de CR de pages :            86.0000000000000000
     OAM + nombre d'allocations de page :5
     Pages de données du premier extent :3
     Taille des lignes de données :     238.8634175691937287
```

Pour plus d'informations, reportez-vous au chapitre 36, "Affichage des tables de statistiques avec optdiag".

Avantages de *optdiag*

Les avantages de optdiag sont les suivants :

- optdiag permet d'afficher les statistiques relatives à toutes les tables d'une base de données ou à une table particulière.
- Les résultats fournis par optdiag contiennent des informations supplémentaires utiles pour comprendre le coût des requêtes, tels que hauteur d'index et longueur moyenne de la ligne.
- optdiag est fréquemment utilisé pour d'autres tâches d'optimisation ; il est par conséquent conseillé de conserver ces rapports à portée de la main.

Inconvénients de *optdiag*

Les inconvénients de optdiag sont les suivants :

- Il produit énormément de résultats et par conséquent, si vous n'avez besoin que d'une information, par exemple le nombre de pages de la table, d'autres méthodes sont plus rapides et ont un overhead inférieur.

Utilisation de la procédure *sp_spaceused* pour afficher la taille d'un objet

La procédure système sp_spaceused lit les valeurs enregistrées sur la page OAM d'un objet pour fournir un rapport rapide sur l'espace utilisé par l'objet.

```
                    sp_spaceused titles
name         rowtotal reserved   data      index_size  unused
------------ -------- ---------- --------- ----------- --------
titles       5000       1756 KB   1242 KB     440 KB     74 KB
```

Il se peut que la valeur rowtotal soit parfois inexacte ; tous les processus Adaptive Server ne mettent pas à jour cette valeur dans la page OAM. Les commandes update statistics, dbcc checktable et dbcc checkdb corrigent la valeur de rowtotal dans la page OAM. Le tableau 15-1 fournit une description des en-têtes du résultat de sp_spaceused.

Tableau 15-1 : résultat de sp_spaceused

Colonne	Signification
rowtotal	Indique une estimation du nombre de lignes. La valeur est lue dans la page OAM. Bien que cette estimation ne soit pas toujours exacte, elle est plus rapide et entraîne moins de conflits que select count(*).
reserved	Indique les pages réservées à la table et à ses index. Elle contient à la fois des pages utilisées et non utilisées dans les extents alloués aux objets. Il s'agit de la somme de data, index_size et unused.
data	Indique le nombre de kilo-octets des pages utilisées par la table.
index_size	Indique le nombre total de kilo-octets des pages utilisées pour les index.
unused	Indique le nombre de ko des pages inutilisées dans les extents alloués à l'objet, incluant les pages inutilisées pour les index de l'objet.

Si vous voulez disposer d'informations uniquement sur les index, exécutez la commande suivante :

```
                sp_spaceused titles, 1
 index_name           size       reserved   unused
 -------------------- ---------- ---------- ----------
 title_id_cix         14 KB      1294 KB    38 KB
 title_ix             256 KB     272 KB     16 KB
 type_price_ix        170 KB     190 KB     20 KB

name         rowtotal  reserved   data      index_size  unused
------------ --------  ---------- --------- ----------- --------
titles       5000        1756 KB   1242 KB     440 KB     74 KB
```

Pour les index clusterisés sur les tables APL, la valeur "size" correspond à l'espace utilisé pour les pages racines et les pages d'index intermédiaires. La valeur reserved (réservé) indique la taille d'index et les pages de données réservées et utilisées.

Dans la syntaxe sp_spaceused, "1" indique que les informations détaillées concernant l'index doivent être imprimées. Il n'existe pas de relations avec les identificateurs d'index ou avec d'autres informations.

Avantages de *sp_spaceused*

Les avantages de sp_spaceused sont les suivants :

- Elle fournit des informations rapides sans générer un nombre excessif d'E/S et de verrous car elle utilise uniquement des valeurs de la table et des pages OAM d'index pour renvoyer les résultats.
- Elle affiche l'espace réservé à l'expansion de l'objet mais non encore utilisé pour le stockage de données.
- Elle fournit des rapports détaillés sur la taille des index et la capacité de stockage des colonnes text et image et des colonnes Java hors ligne.

Inconvénients de *sp_spaceused*

Les inconvénients de sp_spaceused sont les suivants :

- Elle peut afficher des valeurs inexactes sur le total de lignes et l'utilisation de l'espace.
- Le résultat est en kilo-octets alors que la plupart des activités d'optimisation de requêtes utilisent les pages comme unité de mesure.

Evaluation de la taille des objets à l'aide de *sp_estspace*

sp_spaceused et optdiag rendent compte du volume d'espace effectivement occupé. sp_estspace peut vous aider à planifier la croissance future de vos tables et index. Cette procédure utilise les informations des tables de système (sysobjects, syscolumns et sysindexes) pour déterminer la longueur des lignes de données et d'index. Vous fournissez un nom de table et le nombre de lignes que vous envisagez pour la table et sp_estspace évalue la taille de la table et de tous les index existants. Elle ne prend pas en compte la taille effective des données dans les tables.

Utilisation de sp_estspace :

- créez la table si elle n'existe pas.
- créez tous les index sur cette table.
- exécutez la procédure, en fournissant une estimation du nombre de lignes que contiendra la table.

Le résultat indique le nombre de pages et d'octets pour la table et chaque niveau d'index.

L'exemple suivant fournit une estimation de la taille de la table titles avec 500 000 lignes, un index clusterisé et deux index non clusterisés :

```
    sp_estspace titles, 500000
name                 type           idx_level Pages    Kbytes
-------------------- ------------   --------- -------- --------
titles               data           0         50002    100004
title_id_cix         clustered      0         302      604
title_id_cix         clustered      1         3        6
title_id_cix         clustered      2         1        2
title_ix             nonclustered   0         13890    27780
title_ix             nonclustered   1         410      819
title_ix             nonclustered   2         13       26
title_ix             nonclustered   3         1        2
type_price_ix        nonclustered   0         6099     12197
type_price_ix        nonclustered   1         88       176
type_price_ix        nonclustered   2         2        5
type_price_ix        nonclustered   3         1        2

Total_Mbytes
-----------------
  138.30

name                 type         total_pages  time_mins
-------------------- ------------ ------------ ------------
title_id_cix         clustered    50308        250
title_ix             nonclustered 14314        91
type_price_ix        nonclustered 6190         55
```

sp_estspace vous permet également de spécifier un facteur de remplissage, la taille moyenne des champs de longueur variable et des champs texte, ainsi que la vitesse des E/S. Pour plus d'informations, reportez-vous au *Manuel de référence d'Adaptive Server*.

Remarque Le temps de création d'index affiché par sp_estspace n'a pas de répercussion sur le tri parallèle.

Avantages de *sp_estspace*

L'utilisation de sp_estspace pour évaluer la taille des objets présente les avantages suivants :

- sp_estspace permet de prévoir rapidement et facilement la croissance de la table et de l'index.
- sp_estspace vous aide à évaluer le nombre de niveaux d'index.
- sp_estspace peut être utilisé pour l'évaluation de l'espace disque, de l'espace cache et de l'espace mémoire requis.

Inconvénients de *sp_estspace*

L'utilisation de sp_estspace pour évaluer la taille des objets comporte les inconvénients suivants :

- Les tailles renvoyées ne sont que des estimations et peuvent être différentes des tailles réelles à cause des facteurs de remplissage, des fractionnements de pages, de la taille réelle des champs de longueur variable et d'autres facteurs.
- La durée de création des index peut varier considérablement selon la vitesse du disque, l'utilisation des buffers d'E/S d'extent et la charge du système.

Evaluation de la taille des objets à l'aide des formules de calcul

Les formules de calcul suivantes permettent d'évaluer la taille future des tables et des index de la base de données. Le nombre d'octets d'overhead de chaque ligne pour les tables et index contenant des champs de longueur variable est supérieur à celui des tables contenant seulement des champs de longueur fixe ; deux types de formule de calcul sont donc nécessaires.

La méthode générale consiste à calculer le nombre d'octets des données et de l'overhead de chaque ligne, puis à diviser le total par le nombre d'octets disponibles dans une page de données. Chaque page a besoin d'overhead, ce qui limite le nombre d'octets disponibles pour les données :

- pour les tables APL, l'overhead de page est de 32 octets, ce qui laisse 2016 octets disponibles pour les données sur une page de 2 ko.
- pour les tables DOL, l'overhead de page est de 46 octets, ce qui laisse 2002 octets disponibles pour les données.

Pour obtenir une estimation très précise, *arrondissez au chiffre inférieur* le résultat des divisions permettant de calculer le nombre de lignes par page (les lignes ne sont jamais partagées sur plusieurs pages) et *arrondissez au chiffre supérieur* le résultat des divisions permettant de calculer le nombre de pages.

Facteurs pouvant modifier la taille de stockage

Utilisation des propriétés de gestion de l'espace définies pour une table ou un index. Reportez-vous aux sections "Effets des propriétés de gestion de l'espace", page 384 et "max_rows_per_page", page 386.

Les formules de calcul fournies ici utilisent la taille maximale pour les caractères de longueur variable et les données binaires. Si vous souhaitez plutôt utiliser la taille moyenne, reportez-vous à la section "Utilisation de tailles moyennes pour des champs variables", page 386.

Si votre table contient des données de type text ou image ou Java hors ligne, utilisez la valeur 16 (la taille du pointeur texte qui est stocké dans la ligne) dans vos calculs. Reportez-vous ensuite à la section "Pages LOB", page 387 pour calculer l'espace de stockage nécessaire pour les données text ou image dont vous disposez.

Dans les tables DOL, les index peuvent être de taille inférieure aux valeurs fournies par les formules à cause de deux facteurs :

- Les clés répétées ne sont stockées qu'une seule fois, suivies par une liste d'ID de ligne correspondant à la clé.
- Les clés d'un niveau différent du niveau feuille sont comprimées ; seul est stocké le minimum nécessaire pour distinguer la clé de ses voisines. Ceci est particulièrement efficace pour réduire la taille pour de longues clés de caractères.

Si le paramètre de configuration page utilization percent (pourcentage d'utilisation de la page) est inférieur à 100, Adaptive Server peut allouer de nouveaux extents avant de templir toutes les pages des extents déjà alloués. Cela ne modifie pas le nombre de pages qu'un objet utilise mais laisse des pages vides dans les extents alloués à l'objet. Pour plus d'informations, reportez-vous au *Guide d'administration système*.

Tailles de stockage des différents types de données

Le tableau 15-2 répertorie les tailles de stockage des différents types de données.

Tableau 15-2 : Tailles de stockage des types de données d'Adaptive Server

Type de données	Taille
char	Taille définie
nchar	Taille définie * @@*ncharsize*
unichar	n* @@*unicharsize (@@unicharsize equals 2)*
univarchar	nombre réel de caractères * @@*unicharsize*
varchar	Nombre réel de caractères
nvarchar	Nombre réel de caractères * @@*ncharsize*
binary	Taille définie
varbinary	Taille de données
int	4
smallint	2
tinyint	1
float	4 ou 8, selon la précision
double precision	8
real	4
numeric	2–17, selon la précision et l'échelle
decimal	2–17, selon la précision et l'échelle
money	8
smallmoney	4
datetime	8
smalldatetime	4
bit	1
text	16 octets + 2ko * nombre de pages utilisées
image	16 octets + 2ko * nombre de pages utilisées
timestamp	8

La taille de stockage d'une colonne numeric ou decimal dépend de sa précision. Le stockage minimal est de 2 octets pour une colonne de un ou deux chiffres. La taille de stockage augmente d'un octet chaque fois que deux chiffres de précision sont ajoutés, jusqu'à un maximum de 17 octets.

Toutes les colonnes définies comme NULL sont considérées comme étant de longueur variable car elles génèrent l'overhead associé aux colonnes de longueur variable.

Tous les calculs présentés dans les exemples ci-dessous se basent sur la taille maximale des données de type varchar, univarchar, nvarchar et varbinary, c'est-à-dire sur la taille définie pour les colonnes. En outre, les colonnes sont considérées comme NOT NULL. Si vous souhaitez utiliser plutôt des valeurs moyennes, reportez-vous à la section "Utilisation de tailles moyennes pour des champs variables", page 386.

Tables et index utilisés dans les formules

L'exemple illustre les calculs sur une table contenant 9 000 000 lignes.

- La somme de toutes les colonnes de longueur fixe est de 100 octets.
- La somme de toutes les colonnes de longueur variable est de 50 octets ; il y a 2 colonnes de longueur variable.

La table a deux index :

- Un index clusterisé sur une colonne de longueur fixe de 4 octets.
- Un index composé non clusterisé pour les colonnes suivantes :
  - Une colonne de longueur fixe de 4 octets
  - Une colonne de longueur variable de 20 octets

Des formules différentes sont nécessaires pour les tables APL et les tables DOL car elles ont des overheads différents pour la page et par ligne :

- Reportez-vous à la section "Calcul des tailles des tables et des index clusterisés pour les tables APL", page 373 pour les tables utilisant le verrouillage des pages complètes (APL).
- Reportez-vous à la section "Calcul des tailles des tables DOL", page 379 pour les formules à utiliser pour les tables en mode verrouillage des données seules (DOL).

Calcul des tailles des tables et des index clusterisés pour les tables APL

Les formules et les exemples pour les tables APL sont divisés en deux groupes :

- les étapes 1–6 présentent les calculs pour une table APL avec un index clusterisé, fournissant la taille de la table et la taille de l'arborescence de l'index.
- les étapes 7–12 présentent les calculs de l'espace nécessaire pour les index non clusterisés.

Ces formules indiquent comment calculer la taille des tables et des index clusterisés. Si votre table ne possède pas d'index clusterisé, ignorez les étapes 3, 4 et 5. Une fois le nombre de pages de données calculé (étape 2), passez à l'étape 6 pour ajouter le nombre de pages OAM.

Etape 1 : Calcul de la taille des lignes de données

Les lignes stockant des données de longueur variable requièrent davantage d'overhead que les lignes contenant uniquement des données de longueur fixe ; il existe donc deux formules pour calculer la taille d'une ligne de données.

Colonnes de longueur fixe uniquement

Utilisez la première formule de calcul si toutes les colonnes sont de longueur fixe et définies comme NOT NULL.

Formule	
4	(Overhead)
+	Somme des octets de toutes les colonnes de longueur fixe
	= Taille des lignes de données

Exemples de colonnes de longueur variable

Utilisez cette formule de calcul si la table contient des colonnes de longueur variable ou des colonnes autorisant les valeurs NULL.

La table de l'exemple contient des colonnes de longueur variable ; les calculs sont donc indiqués dans la colonne de droite.

Formule		Exemple
4	(Overhead)	4

Formule			Exemple
+		Somme des octets de toutes les colonnes de longueur fixe	+ 100
+		Somme des octets de toutes les colonnes de longueur variable	+ 50
		= Sous-total	154
+		(Sous-total / 256) + 1 (Overhead)	1
+		Nombre de colonnes de longueur variable + 1	3
+	2	(Overhead)	2
		= Taille des lignes de données	160

Etape 2 : Calcul du nombre de pages de données

Formule

2016 / Taille de ligne de données = Nombre de lignes de données par page

Nombre de lignes / Lignes par page = Nombre de pages de données nécessaires

Exemple

2016 / 160	=	12 lignes par page de données
9 000 000 / 12	=	750 000 pages de données

Etape 3 : Calcul de la taille des lignes d'index clusterisé

Les lignes d'index contenant des colonnes de longueur variable requièrent davantage d'overhead que les lignes d'index ne contenant que des valeurs de longueur fixe. Utilisez la première formule de calcul si toutes les clés sont de longueur fixe. Utilisez la seconde si les clés comportent des colonnes de longueur variable ou autorisent les valeurs NULL.

Colonnes de longueur fixe uniquement

L'index clusterisé de l'exemple n'a que des clés de longueur fixe.

Formule			Exemple
	5	(Overhead)	5
+		Somme des octets des clés d'index de longueur fixe	+ 4
		= Taille des lignes d'index clusterisés	9

Exemples de colonnes de longueur variable

	5	(Overhead)
+		Somme des octets des clés d'index de longueur fixe
+		Somme des octets des clés d'index de longueur variable
		= Sous-total
+		(Sous-total / 256) + 1 (Overhead)
+		Nombre de colonnes de longueur variable + 1
+	2	(Overhead)
		= Taille des lignes d'index clusterisés

Les résultats de la division (sous-total/256) sont arrondis au chiffre inférieur.

Etape 4 : Calcul du nombre de pages d'index clusterisé

Formule			Exemple		
(2016 / Taille des lignes d'index clusterisés) – 2	=	Nbre lignes d'index clusterisé par page	(2016 / 9) – 2	=	222
Nbre lignes / Nbre lignes d'index clusterisé par page	=	Nbre de pages d'index au niveau suivant	750 000 / 222	=	3379

Si le nombre de pages d'index au niveau suivant est supérieur à 1, répétez l'étape de division suivante, en utilisant le quotient comme dividende jusqu'à ce que le quotient soit égal à 1, ce qui signifie que le niveau racine de l'index est atteint :

Formule

Nbre pages d'index au dernier niveau	/	Nbre lignes d'index clusterisé par page	=	Nbre de pages d'index au niveau suivant

Exemple

3379 / 222	=	16 pages d'index (Niveau 1)
16 / 222	=	1 page d'index (Niveau 2)

Etape 5 : Calcul du nombre total de pages d'index

Ajoutez le nombre de pages de chaque niveau pour déterminer le nombre total de pages dans l'index :

Formule			**Exemple**	
Niveaux d'index	Pages		Pages	Lignes
2			1	16
1	+		+ 16	3379
0	+		+ 3379	750000
		Nombre total de pages d'index	3396	

Etape 6 : Calcul de l'overhead d'allocation et du nombre total de pages

Chaque table et index d'une table possède une table d'allocation d'objets (OAM). Chaque page de l'OAM peut contenir la configuration d'allocation de 2 000 à 63 750 pages d'index ou de données. Dans la plupart des cas, le nombre de pages OAM nécessaires est proche de la valeur minimale. Pour calculer le nombre de pages OAM de la table, utilisez les formules suivantes :

Formule			**Exemple**		
Nombre de pages de données réservées/ 63 750	=	Pages OAM minimales	750 000 / 63 750	=	12
Nombre de pages de données réservées / 2000	=	Pages OAM maximales	750 000 / 2000	=	376

Pour calculer le nombre de pages OAM de l'index, utilisez les formules suivantes :

Formule			**Exemple**		
Nombre de pages d'index / 63 750	=	Pages OAM minimales	3396/ 63 750	=	1
Nombre de pages d'index réservées / 2000	=	Pages OAM maximales	3396 / 2000	=	2

Nombre total de pages nécessaires

Ajoutez le nombre de pages OAM aux totaux précédents pour déterminer le nombre total de pages requises :

Formule			Exemple	
	Minimale	Maximale	Minimale	Maximale
Pages d'index clusterisé			3396	3379
Pages OAM	+	+	1	2
Pages de données	+	+	750000	750000
Pages OAM	+	+	12	376
Total			753409	753773

Etape 7 : Calcul de la taille des lignes d'index de niveau feuille

Les lignes d'index contenant des colonnes de longueur variable requièrent davantage d'overhead que les lignes d'index ne contenant que des valeurs de longueur fixe.

Clés de longueur fixe uniquement

Utilisez cette formule de calcul si toutes les colonnes sont de longueur fixe et définies comme NOT NULL :

Formule	
7	(Overhead)
+	Somme de clés de longueur fixe
	= Taille des lignes de l'index de niveau feuille

Exemples de clés de longueur variable

Utilisez cette formule de calcul si l'index contient des clés de longueur variable ou des colonnes définies comme NULL :

Formule		Exemple	
9	(Overhead)		9
+	Longueur totale des clés de longueur fixe	+	4
+	Longueur totale des clés de longueur variable	+	20
+	Nombre de clés de longueur variable + 1	+	2
	= Sous-total		35
+	(Sous-total / 256) + 1 (overhead)	+	1
	= Taille des lignes de l'index de niveau feuille		36

Etape 8 : Calcul du nombre de pages de niveau feuille dans l'index

Formule			Exemple		
(2016 / Taille des lignes niveau feuille)	=	Nbre lignes d'index niveau feuille par page	2016 / 36	=	56
Nbre lignes de la table / Nbre lignes niveau feuille par page	=	Nbre de pages d'index au niveau suivant	9 000 000 / 56	=	160 715

Etape 9 : Calcul de la taille des lignes n'étant pas de niveau feuille

Formule		Exemple	
	Taille des lignes d'index de niveau feuille		36
+ 4	Overhead	+	4
	= Taille des lignes n'étant pas de niveau feuille		40

Etape 10 : Calcul du nombre de pages n'étant pas de niveau feuille

Formule			Exemple
(2016 / Taille des lignes n'étant pas de niveau feuille) – 2	=	Nbre de lignes d'index n'étant pas de niveau feuille par page	(2016 / 40) – 2 = 48

Si le nombre de pages au niveau feuille de l'étape 8 est supérieur à 1, répétez l'étape de division suivante, en utilisant le quotient comme dividende jusqu'à ce que le quotient soit égal à 1, ce qui signifie que le niveau racine de l'index est atteint :

Formule				
Nbre pages d'index au niveau précédent	/	Nbre de lignes d'index n'étant pas de niveau feuille par page	=	Nbre de pages d'index au niveau suivant

Exemple	
160715 / 48 = 3349	Pages d'index, niveau 1
3349 / 48 = 70	Pages d'index, niveau 2
70 / 48 = 2	Pages d'index, niveau 3
2 / 48 = 1	Page d'index, niveau 4 (niveau racine)

Etape 11 : Calcul du nombre total de pages d'index n'étant pas de niveau feuille

Ajoutez le nombre de pages de chaque niveau pour déterminer le nombre total de pages dans l'index :

Niveaux d'index		Pages		Lignes
4		1		2
3	+	2	+	70
2	+	70	+	3348
1	+	3349	+	160715
0	+	160715	+	9000000
	Nombre total de pages de données de 2 ko utilisées	164137		

Etape 12 : Calcul de l'overhead d'allocation et du nombre total de pages

Formule			Exemple		
Nbre de pages d'index réservées / 63 750	=	Pages OAM minimales	164137 / 63 750	=	3
Nbre de pages d'index / 2000	=	Pages OAM maximales	164137 / 2000	=	83

Nombre total de pages nécessaires

Ajoutez le nombre de pages OAM au total obtenu à l'étape 11 pour déterminer le nombre total de pages d'index :

Formule			Exemple	
	Minimale	Maximale	Minimale	Maximale
Pages d'index non clusterisé			164137	164137
Pages OAM	+	+	3	83
Total			164140	164220

Calcul des tailles des tables DOL

Ces formules et exemples indiquent comment calculer la taille des tables et des index. Cet exemple utilise les mêmes tailles de colonnes et index que le premier exemple. Pour les spécifications, reportez-vous à la section "Tables et index utilisés dans les formules", page 372.

Les formules pour les tables DOL sont divisées en deux groupes :

- les étapes 1–3 présentent les calculs pour une table DOL. L'exemple qui suit l'étape 3 illustre les calculs sur une table contenant 9 000 000 lignes.
- les étapes 4–8 présentent les calculs de l'espace nécessaire pour un index, suivis d'un exemple relatif à la table de 9 000 000 lignes.

Etape 1 : Calcul de la taille des lignes de données

Les lignes stockant des données de longueur variable requièrent davantage d'overhead que les lignes contenant uniquement des données de longueur fixe ; il existe donc deux formules pour calculer la taille d'une ligne de données.

Colonnes de longueur fixe uniquement

Utilisez cette formule de calcul si toutes les colonnes sont de longueur fixe et définies comme NOT NULL :

	6	(Overhead)
+		Somme des octets de toutes les colonnes de longueur fixe
		Taille des lignes de données

Remarque Les tables verrouillées au niveau des données uniquement doivent prévoir un espace suffisant pour stocker un ID de ligne de 6 octets. Si les lignes d'une table verrouillée au niveau des données uniquement comporte moins de 10 octets, chaque ligne passe automatiquement à 10 octets lorsqu'elle est insérée. Ceci n'intéresse que les pages de données et non pas les index et ne concerne pas les tables verrouillées au niveau de toutes les pages (APL).

Exemples de colonnes de longueur variable

Utilisez cette formule de calcul si la table contient des colonnes de longueur variable ou des colonnes autorisant les valeurs NULL :

Formule			Exemple	
	8	(Overhead)		8
+		Somme des octets de toutes les colonnes de longueur fixe	+	100
+		Somme des octets de toutes les colonnes de longueur variable	+	50
+		Nombre de colonnes de longueur variable * 2	+	4
		Taille des lignes de données		162

Etape 2 : Calcul du nombre de pages de données

Formule

2002 / Taille de ligne de données = Nombre de lignes de données par page

Nombre de lignes / Lignes par page = Nombre de pages de données nécessaires

Dans la première partie de cette étape, le nombre de lignes par pages est arrondi au-dessous :

Exemple

2002 / 162	=	12 lignes par page de données
9,000,000 / 12	=	750 000 pages de données

Etape 3 : Calcul de l'overhead d'allocation et du nombre total de pages

Overhead d'allocation

Chaque table et index d'une table possède une table d'allocation d'objets (OAM). L'OAM est enregistré dans les pages allouées à la table ou à l'index. Chaque page de l'OAM peut contenir la configuration d'allocation de 2 000 à 63 750 pages d'index ou de données. Dans la plupart des cas, le nombre de pages OAM nécessaires est proche de la valeur minimale. Pour calculer le nombre de pages OAM de la table, utilisez les formules suivantes :

Formule			Exemple		
Nbre de pages de données réservées / 63 750	=	Pages OAM minimales	750 000 / 63 750	=	12
Nombre de pages de données réservées / 2000	=	Pages OAM maximales	750 000 / 2000	=	375

Nombre total de pages nécessaires

Ajoutez le nombre de pages OAM aux totaux précédents pour déterminer le nombre total de pages nécessaires :

Formule			Exemple	
	Minimale	Maximale	Minimale	Maximale
Pages de données	+	+	750000	750000
Pages OAM	+	+	12	375
Total			750012	750375

Etape 4 : Calcul de la taille de la ligne d'index

Utilisez ces formules pour les index clusterisés et non clusterisés des tables DOL.

Les lignes d'index contenant des colonnes de longueur variable requièrent davantage d'overhead que les lignes d'index ne contenant que des valeurs de longueur fixe.

Clés de longueur fixe uniquement

Utilisez cette formule de calcul si l'index ne contient que des colonnes de longueur fixe et définies comme NOT NULL :

	9	(Overhead)
+		Somme de clés de longueur fixe
		Taille des lignes d'index

Exemples de clés de longueur variable

Utilisez cette formule de calcul si l'index contient des clés de longueur variable ou des colonnes autorisant les valeurs NULL :

Formule			Exemple	
	9	(Overhead)		9
+		Longueur totale des clés de longueur fixe	+	4
+		Longueur totale des clés de longueur variable	+	20
+		Nombre de clés de longueur variable * 2	+	2
		Taille des lignes d'index		35

Etape 5 : Calcul du nombre de pages de niveau feuille dans l'index

Formule

2002 / Taille de la ligne d'index = Nbre lignes par page

Nbre lignes de la table / Nbre lignes par page = Nbre de pages de niveau feuille

Exemple

2002 / 35 = 57 lignes d'index non clusterisé par page

9 000 000 / 57 = 157 895 pages de niveau feuille

Etape 6 : Calcul du nombre de pages non de niveau feuille dans l'index

Formule

Nombre de pages feuilles	/	Nbre de lignes d'index par page	=	Nbre de pages au niveau suivant

Si le nombre de pages d'index au niveau suivant est supérieur à 1, répétez l'étape de division suivante, en utilisant le quotient comme dividende jusqu'à ce que le quotient soit égal à 1, ce qui signifie que le niveau racine de l'index est atteint :

Formule

Nbre pages d'index au niveau précédent	/	Nbre de lignes d'index n'étant pas de niveau feuille par page	=	Nbre de pages d'index au niveau suivant

Exemple

157895 / 57 = 2771	Pages d'index, niveau 1
2770 / 57 = 49	Pages d'index, niveau 2
48 / 57 =1	Pages d'index, niveau 3

Etape 7 : Calcul du nombre total de pages d'index n'étant pas de niveau feuille

Ajoutez le nombre de pages de chaque niveau pour déterminer le nombre total de pages dans l'index :

Formule			**Exemple**	
Niveaux d'index	Pages		Pages	Lignes
3			1	49
2	+		49	2771
1	+		2771	157895
0	+		157895	9000000
		Nombre total de pages de 2 ko utilisées	160716	

Etape 8 : Calcul de l'overhead d'allocation et du nombre total de pages

Formule

Nombre de pages d'index / 63 750 = Pages OAM minimales

Nombre de pages d'index / 2000 = Pages OAM maximales

Exemple

160713 / 63 750 = 3 (minimum)

160713 / 2000 = 81 (maximum)

Nombre total de pages nécessaires

Ajoutez le nombre de pages OAM au total obtenu à l'étape 8 pour déterminer le nombre total de pages d'index :

Formule			**Exemple**	
	Minimale	Maximale	Minimale	Maximale
Pages d'index non clusterisé			160716	160716
Pages OAM	+	+	3	81
Total			160719	160797

Autres facteurs influant sur la taille des objets

Au delà des effets des modifications des données qui se produisent au cours du temps, d'autres facteurs peuvent influer sur la taille des objets et ses estimations :

- les propriétés de gestion d'espace
- le fait que les calculs utilisent la taille moyenne ou la taille maximale des lignes
- lignes de texte très courtes
- Utilisation de données text et image

Effets des propriétés de gestion de l'espace

Les valeurs de fillfactor, exp_row_size, reservepagegap et max_rows_per_page peuvent influer sur la taille des objets.

fillfactor

La valeur de fillfactor spécifiée avec create index est appliquée au moment où l'index est créé. Le fillfactor n'est pas maintenu au cours des insertions dans la table. Si un fillfactor a été enregistré pour un index à l'aide de sp_chgattribute, cette valeur est utilisée lorsque les index sont recréés à l'aide des commandes alter table...lock et reorg rebuild. La fonction principale de fillfactor est de prévoir de l'espace dans les pages d'index, de façon à réduire les fractionnements de page. Des valeurs très faibles de fillfactor peuvent augmenter de façon significative l'espace mémoire nécessaire pour une table ou un index.

Si fillfactor a la valeur par défaut de 0, le processus de gestion des index laisse la place pour deux lignes supplémentaires sur chaque page d'index lors de la création d'un nouvel index. Si vous donnez à fillfactor la valeur de 100 pour cent, il ne laisse plus de place pour ces lignes. Le seul effet de fillfactor sur les calculs des tailles se manifeste lors du calcul du nombre de pages d'index clusterisées et du nombre de pages non de type feuille. Au cours de ces deux calculs, 2 est soustrait du nombre de lignes par page. Eliminez –2 de ces calculs.

Les autres valeurs de fillfactor réduisent le nombre de lignes par page pour les pages de données et les pages d'index de type feuille. Pour calculer les valeurs correctes lors de l'utilisation de fillfactor, multipliez la taille de la page de données disponible (2016) par le fillfactor. Par exemple, si le fillfactor est de 75 pour cent, votre page de données pourra contenir 1471 octets. Utilisez cette valeur à la place de 2016 lors du calcul du nombre de lignes par page. Pour ces calculs, reportez-vous aux sections "Etape 2 : Calcul du nombre de pages de données", page 374 et "Etape 8 : Calcul du nombre de pages de niveau feuille dans l'index", page 378.

exp_row_size

La définition d'une taille de ligne prévue peut augmenter l'espace mémoire nécessaire. Si vos tables ont beaucoup de lignes qui sont plus courtes que la taille prévue, l'espace mémoire nécessaire pour la table augmente lorsque vous définissez cette valeur et exécutez reorg rebuild ou lorsque vous changez le mode de verrouillage. Cependant, l'utilisation de l'espace de la part des tables qui utilisaient max_rows_per_page devrait rester à peu près la même.

reservepagegap

La définition de reservepagegap pour une table ou un index laisse des pages vides sur les extents alloués à l'objet lors de l'exécution de commandes effectuant l'allocation des extents. Le fait de donner à reservepagegap une valeur faible augmente le nombre de pages vides et étale les données sur un plus grand nombre d'extents, de sorte que l'espace supplémentaire nécessaire est plus grand immédiatement après une commande telle que create index ou reorg rebuild. La redirection des lignes et les insertions dans la table remplissent les pages réservées. Pour plus d'informations, reportez-vous à la section "Réservation d'espace pour les lignes redirigées et les insertions", page 305.

max_rows_per_page

La valeur max_rows_per_page (spécifiée par create index, create table, alter table ou sp_chgattribute) limite le nombre de lignes d'une page de données.

Pour calculer les valeurs correctes lorsque vous utilisez max_rows_per_page, utilisez la plus petite des valeurs de max_rows_per_page et du nombre de lignes de données par page calculé aux sections "Etape 2 : Calcul du nombre de pages de données", page 374 et "Etape 8 : Calcul du nombre de pages de niveau feuille dans l'index", page 378.

Utilisation de tailles moyennes pour des champs variables

Toutes les formules utilisent la taille maximale des champs de longueur variable.

Le résultat de optdiag comprend la longueur moyenne des lignes de données et d'index. Vous pouvez utiliser ces valeurs comme longueur des lignes de données et d'index, si vous souhaitez plutôt utiliser des longueurs moyennes.

Lignes très courtes

Adaptive Server ne peut pas enregistrer plus de 256 lignes de données ou d'index dans une page. Même si les lignes sont extrêmement courtes, le nombre minimal de pages de données est :

Nombre de lignes / 256 =	Nombre de pages de données nécessaires

Pages LOB

Chaque colonne de type text ou image et chaque colonne Java hors ligne enregistre un pointeur de 16 octets dans la ligne de données avec le type de données varbinary(16). Chaque colonne initialisée nécessite au moins 2 ko (une page de données) d'espace mémoire.

Les C colonnes enregistrent des valeurs implicites nulles, ce qui signifie que le pointeur de texte dans la ligne de données conserve la valeur NULL et qu'aucune page de texte n'est initialisée avec cette valeur, ce qui économise 2 ko d'espace mémoire.

Si une colonne LOB est définie de façon à accepter des valeurs NULL et si la ligne est créée par une instruction insert comprenant NULL pour la colonne, la colonne n'est pas initialisée et l'espace n'est pas alloué.

Si une colonne LOB est modifiée d'une façon quelconque par l'instruction update, la page de texte est alors allouée. Naturellement, les insertions et mises à jour qui introduisent des données réelles initialisent la page. Si la colonne reçoit ensuite la valeur NULL, une seule page reste allouée.

Chaque page LOB peut stocker jusqu'à 1 800 octets de données. Pour évaluer le nombre de pages que cette entrée va utiliser, utilisez la formule suivante :

```
Longueur des données / 1800 = nombre de pages de 2 ko
```

Le résultat doit être arrondi par excès dans tous les cas ; par exemple, des données de 1801 octets ont besoin de deux pages de 2 ko.

L'espace total nécessaire pour les données peut être légèrement supérieur à la valeur calculée car certaines pages LOB contiennent des informations sur les pointeurs à d'autres chaînes de pages de la colonne. Adaptive Server utilise ces informations pour effectuer un accès aléatoire avec prélecture des données lors de l'accès aux colonnes LOB. L'espace supplémentaire nécessaire pour enregistrer ces informations sur les pointeurs dépend de la taille totale et du type de données enregistrées dans la colonne. Utilisez les informations du tableau 15-3 pour évaluer les pages supplémentaires nécessaires pour les pointeurs des données des colonnes LOB.

Tableau 15-3 : Estimation des pages supplémentaires pour les pointeurs dans les colonnes LOB

Taille et type des données	Pages supplémentaires nécessaires pour les pointeurs des données
image de 400 ko	0 à 1 page
image de 700 ko	0 à 2 pages
image de 5 Mo	1 à 11 pages
texte de 400 ko, codé sur plusieurs octets	1 à 2 pages
texte de 700 ko, codé sur plusieurs octets	1 à 3 pages
texte de 5 Mo, codé sur plusieurs octets	2 à 22 pages

Avantages de l'utilisation de formules pour l'évaluation de la taille des objets

Les avantages de l'utilisation de formules sont les suivants :

- Vous disposez de plus de détails sur la définition interne des données et des index.
- Les formules apportent une grande souplesse dans la définition des tailles moyennes pour les colonnes de caractères ou de valeurs binaires.
- Lorsque vous calculez la taille de l'index, vous voyez le nombre de niveaux de chaque index, ce qui aide à évaluer les performances.

Inconvénients de l'utilisation de formules pour l'évaluation de la taille des objets

Les inconvénients de l'utilisation de formules sont les suivants :

- La validité des estimations dépend de celle des estimations de la taille moyenne des colonnes de longueur variable.
- Les calculs sont complexes et se déroulent sur plusieurs étapes qui peuvent introduire des erreurs si elles sont omises.
- La taille réelle d'un objet peut différer des résultats des calculs, selon le type d'utilisation.

CHAPITRE 16

Activités de maintenance et performances

Ce chapitre explique comment les activités de maintenance peuvent réduire les performances des autres activités d'Adaptive Server et indique comment optimiser les tâches de maintenance.

Ces dernières incluent notamment des opérations telles que la suppression et la création à nouveau d'index, l'exécution de contrôles dbcc et la mise à jour des statistiques d'index. Ces activités peuvent toutes s'exécuter parallèlement à d'autres traitements effectués sur le serveur.

Dans la mesure du possible, effectuez les tâches de maintenance quand l'utilisation de Adaptive Server est faible. Ce chapitre vous permet de déterminer l'impact de ces activités sur les applications et sur les performances globales d'Adaptive Server.

Sujet	Page
Exécution de reorg sur des tables et des index	390
Création et maintenance d'index	390
Création ou modification d'une base de données	396
Sauvegarde et restauration	397
Bulkcopy	399
Vérificateur de cohérence de base de données	403
Utilisation de dbcc tune (cleanup)	404
Détermination de l'espace disponible pour des activités de maintenance	404

Exécution de *reorg* sur des tables et des index

La commande reorg peut améliorer les performances sur des tables verrouillées au niveau des données seulement (tables DOL) en optimisant l'utilisation de l'espace reservé aux tables et aux index. Les options de la commande reorg et leurs fonctions sont les suivantes :

- reclaim_space – cears les suppressions validées et l'espace laissé par les mises à jour raccourcissant la longueur des lignes de données.
- forwarded_rows – renvoie les lignes redirigées aux pages d'accueil.
- compact – effectue les deux opérations précédentes.
- rebuild – reconstruit entièrement une table ou un index.

Pendant l'exécution de la commande reorg rebuild sur une table, cette table est verrouillée pendant le temps nécessaire à sa reconstruction ainsi qu'à celle de ses index. Vous devez donc programmer l'exécution de reorg rebuild sur une table à un moment où les utilisateurs n'ont pas besoin d'y accéder.

Toutes les autres commandes reorg, y compris reorg rebuild sur un index, ne bloquent qu'un petit nombre de pages à la fois et utilisent des transactions courtes et indépendantes pour effectuer leur travail. Vous pouvez exécuter ces commandes à tout moment. Les éventuels effets négatifs sur les performances ne concernent que les systèmes fortement consommateurs d'E/S.

Pour plus d'informations sur l'exécution des commandes reorg reportez-vous au *Guide d'administration système*.

Création et maintenance d'index

La création d'index réduit les performances en empêchant les autres utilisateurs de se servir d'une table. Le type de verrou dépend du type d'index :

- La création d'un index clusterisé requiert un verrou de table exclusif, verrouillant toutes les activités d'une table. Puisque les lignes d'un index clusterisé suivent l'ordre de la clé d'index, create clustered index réorganise les pages de données.

- La création d'un index non clusterisé requiert un verrou de table partagé, verrouillant les activités de mise à jour.

Configuration d'Adaptive Server pour accélérer le tri

Le paramètre de configuration définit le nombre de buffers utilisables dans le cache pour stocker des pages extraites des tables d'entrée. Par ailleurs, le tri parallèle peut tirer profit des E/S étendues dans le cache utilisé pour le tri.

Pour plus d'informations, reportez-vous à la section "Configuration des ressources pour le tri en parallèle", page 633.

Sauvegarde d'une base de données après la création d'un index

Lorsque vous créez un index, Adaptive Server écrit la transaction create index et les allocations de page dans le journal de transactions mais ne journalise pas les modifications effectives des pages d'index et de données. Pour restaurer une base de données que vous n'avez pas sauvegardée depuis la création de l'index, le processus create index complet est à nouveau exécuté pendant le chargement des sauvegardes du journal de transactions.

Si vous recréez régulièrement un index (par exemple, pour maintenir le facteur de remplissage fillfactor de l'index), vous pouvez programmer ces opérations peu de temps avant la sauvegarde régulière de la base de données.

Création d'un index sur des données triées

Si vous devez recréer un index clusterisé ou créer un index sur des données ayant été copiées dans le serveur via une opération bulkcopy et dans l'ordre de la clé d'index, utilisez l'option sorted_data de create index pour accélérer la création d'index.

Etant donné que, dans les index clusterisés, les lignes de données doivent suivre l'ordre indiqué par la clé d'index, la création d'un index clusterisé sans l'option sorted_data exige la réécriture des lignes de données dans un nouveau jeu de pages de données. Dans certains cas, Adaptive Server peut ne pas trier et/ou copier les lignes de données de la table. Les facteurs incluent le partitionnement de la table et des clauses on utilisées dans l'instruction create index.

Lors de la création d'un index sur une table non partitionnée, si vous utilisez sorted_data et l'une des clauses suivantes, vous devez copier les données, mais vous n'êtes pas obligé de les trier :

- ignore_dup_row
- fillfactor
- la clause on *nom_segment*, qui spécifie un autre segment que celui contenant les données de la table,
- la clause max_rows_per_page, qui spécifie une autre valeur que celle associée à la table.

Lorsque ces options et sorted_data sont incluses dans une instruction create index sur une table partitionnée, le tri est effectué et les données sont copiées, les pages de données étant réparties de façon équitable entre les partitions de la table.

Tableau 16-1 : Utilisation d'options pour la création d'un index clusterisé

Options	Table partitionnée	Table non partitionnée
Aucune option spécifiée	Tri parallèle ; copie les données en les répartissant de façon équitable sur les partitions ; crée un arbre d'index.	Tri parallèle ou non ; copie les données et crée un arbre d'index.
with sorted_data uniquement ou with sorted_data on *même_segment*	Crée uniquement un arbre d'index. N'effectue ni le tri ni la copie des données. Ne s'exécute pas en parallèle.	Crée uniquement un arbre d'index. N'effectue ni le tri ni la copie des données. Ne s'exécute pas en parallèle.
with sorted_data et ignore_dup_row ou fillfactor ou on *autre_segment* ou max_rows_per_page	Tri parallèle ; copie les données en les répartissant de façon équitable sur les partitions ; crée un arbre d'index.	Copie les données et crée l'arbre d'index. N'effectue pas le tri. Ne s'exécute pas en parallèle.

Dans le cas le plus simple, c'est-à-dire si vous utilisez sorted_data sans autre option sur une table non partitionnée, l'ordre des lignes est vérifié et l'arbre d'index est établi durant ce balayage unique.

Si vous devez copier les lignes de données sans effectuer de tri, un seul balayage de table permet de vérifier l'ordre des lignes, construire l'arbre d'index et copier les pages de données vers le nouvel emplacement.

Dans le cas de tables volumineuses pour lesquelles la construction de l'index requiert de nombreux passages, l'absence de tri réduit de façon non négligeable les E/S et l'utilisation de la CPU.

Lorsque la création d'un index clusterisé entraîne la copie des lignes de données, l'espace disponible doit correspondre à environ 120 % de l'espace de la table pour permettre la copie des données et le stockage des pages d'index.

Gestion des index et des statistiques sur les colonnes

Les valeurs d'histogramme et de densité d'un index ne sont pas mises à jour lorsque des lignes de données sont ajoutées ou supprimées. Le propriétaire de la base de données doit lancer une commande update statistics afin de garantir la mise à jour des statistiques. Lancez update statistics :

- après une suppression ou une insertion de lignes modifiant la typologie des valeurs clés dans l'index ;
- après l'ajout de lignes dans une table dont les lignes avaient précédemment été supprimées *avec* truncate table ;
- après la mise à jour de certaines valeurs des colonnes d'index.

Exécutez update statistics après toute insertion dans un index contenant des colonnes IDENTITY ou une valeur de clé croissante. Les colonnes de date ont souvent des clés qui augmentent régulièrement.

Il est particulièrement important d'exécuter la commande update statistics sur ce type d'index lorsque la colonne IDENTITY ou une autre clé croissante constitue la colonne principale de l'index. Après insertion, une fois que l'index a été créé, d'un certain nombre de lignes à la suite de la dernière clé, l'optimiseur peut uniquement vous indiquer que la valeur de recherche se trouve au-delà de la dernière ligne dans la page de statistiques.

Il n'a pas la possibilité de déterminer avec précision le nombre de lignes qui correspondent à une valeur donnée.

Remarque Les performances peuvent être considérablement diminuées si vous n'avez pas mis les statistiques à jour.

Pour plus d'informations, reportez-vous à la section chapitre 33, "Utilisation des statistiques pour améliorer les performances".

Reconstruction des index

La reconstruction des index requiert de l'espace dans les arbres B-tree. A mesure que s'effectuent les ruptures de pages et les suppressions de lignes, les index peuvent contenir de nombreuses pages qui ne comportent que quelques lignes. Si votre application effectue des balayages sur des index non clusterisés couvrant des requêtes et procède à des E/S étendues, la reconstruction des index non clusterisés préserve l'efficacité des E/S en réduisant la fragmentation.

Vous pouvez reconstruire des index en les supprimant puis en les recréant. Si la table est verrouillée au niveau des pages de données seulement (DOL), exécutez la commande reorg rebuild sur la table ou sur un index particulier.

Il convient de reconstruire des index dans les cas suivants :

- la répartition et l'utilisation des données ont considérablement changé ;
- de nombreuses insertions vont être effectuées ou viennent de l'être ;
- l'ordre de tri est modifié ;
- les requêtes qui utilisent des E/S étendues requièrent plus de lectures sur disque que prévu, ou l'utilitaire optdiag fait état de taux de clusterisation plus bas que d'habitude ;
- l'espace occupé dépasse les estimations car d'importantes modifications de données ont pratiquement rempli de nombreuses pages de données et d'index ;

- l'espace réservé par les propriétés de gestion de l'espace (fillfactor, expected row size et reserve page gap) a été rempli par des insertions et des mises à jour, ce qui a entraîné des ruptures de pages, des renvois de lignes et une fragmentation des données ;
- dbcc a identifié des erreurs dans l'index.

Si vous reconstruisez un index clusterisé ou si vous exécutez reorg rebuild sur une table DOL, tous les index non clusterisés sont recréés, puisque la création d'index clusterisé déplace les lignes vers d'autres pages.

Les index non clusterisés doivent être ensuite recréés afin de pointer sur les pages appropriées.

Dans de nombreux systèmes de bases de données, des périodes de forte ou de faible activité sont définies. Vous pouvez tirer parti des heures creuses, par exemple, pour :

- supprimer tous les index afin de rendre des insertions plus efficaces par bulkcopy.
- créer un nouveau groupe d'index afin de faciliter la génération d'un ensemble de rapports.

Pour plus d'informations sur les paramètres de configuration permettant d'accélérer la procédure de création d'index, reportez-vous à la section "Création et maintenance d'index", page 390.

Accélération de la création d'index avec *sorted_data*

Si les données sont déjà triées, vous pouvez utiliser l'option sorted_data pour la commande create index pour enregistrer l'heure de création d'index. Cette option est utilisable pour les index clusterisés et non clusterisés.

Pour plus d'informations, reportez-vous à la section "Création d'un index sur des données triées", page 391.

Création ou modification d'une base de données

La création ou la modification d'une base de données nécessite un grand nombre d'E/S, de sorte que d'autres opérations, également consommatrices d'E/S, risquent d'en souffrir. Lorsque vous créez une base de données, Adaptive Server y copie la base model puis initialise toutes les pages d'allocation ainsi que les pages de la base de données.

Les procédures suivantes peuvent accélérer la création d'une base de données ou en réduire l'impact sur les autres processus :

- Utilisez l'option for load de la commande create database pour restaurer une base de données, si vous pouvez exécuter la commande load database.

  Lorsque vous créez une base de données sans for load, elle copie la base de données model et toutes les unités d'allocation sont initialisées.

  Si vous utilisez l'option for load, la mise à zéro des unités d'allocation est retardée jusqu'à la fin du chargement. Ensuite, seules les unités d'allocation non modifiées sont initialisées. Si vous chargez une sauvegarde de base de données de grande taille, vous gagnez beaucoup de temps.

- Si possible, créez les bases de données pendant les heures creuses.

create database et alter database produisent des E/S parallèles et concurrentes lorsqu'elles effacent des pages de la base de données. Le nombre de devices est limité par le paramètre de configuration number of large i/o buffers. La valeur par défaut de ce paramètre est 6, ce qui autorise des E/S parallèles sur 6 devices simultanément.

Une seule commande create database ou alter database peut utiliser en même temps jusqu'à 8 de ces buffers. Ces derniers sont également utilisés par load database, par la mise en miroir des disques et par certaines commandes dbcc.

Avec la valeur par défaut 6, si vous spécifiez plus de 6 devices, les 6 premières écritures sont immédiatement lancées. Au fur et à mesure que les E/S de chaque device sont terminées, les buffers de 16 ko sont utilisés pour les autres devices répertoriés dans la commande. L'exemple suivant nomme 10 devices distincts :

```
create database hugedb
      on dev1 = 100,
      dev2 = 100,
```

```
        dev3 = 100,
        dev4 = 100,
        dev5 = 100,
        dev6 = 100,
        dev7 = 100,
        dev8 = 100
log on logdev1 = 100,
        logdev2 = 100
```

Pendant les opérations qui utilisent ces buffers, un message est envoyé au journal lorsque le nombre de buffers est dépassé. Pour la commande create database ci-dessus, ce message montre que create database a commencé par effacer les devices situés sur les 6 premiers disques en utilisant tous les buffers d'E/S étendues, puis a attendu que ces opérations soient terminées avant d'effacer les pages situées sur les autres devices :

```
CREATE DATABASE: allocating 51200 pages on disk 'dev1'
CREATE DATABASE: allocating 51200 pages on disk 'dev2'
CREATE DATABASE: allocating 51200 pages on disk 'dev3'
CREATE DATABASE: allocating 51200 pages on disk 'dev4'
CREATE DATABASE: allocating 51200 pages on disk 'dev5'
CREATE DATABASE: allocating 51200 pages on disk 'dev6'
01:00000:00013:1999/07/26 15:36:17.54 server No disk i/o buffers
are available for this operation. Le nombre total de buffers est
contrôlé par le paramètre de configuration "number of large i/o
buffer".
CREATE DATABASE: allocating 51200 pages on disk 'dev7'
CREATE DATABASE: allocating 51200 pages on disk 'dev8'
CREATE DATABASE: allocating 51200 pages on disk 'logdev2'
CREATE DATABASE: allocating 51200 pages on disk 'logdev2'
```

Lorsque la commande create database copie la base de données model, elle utilise des E/S de 2 ko.

Reportez-vous au *Guide d'administration système*.

Sauvegarde et restauration

Toutes les sauvegardes d'Adaptive Server sont effectuées par un serveur de sauvegarde. L'architecture de la sauvegarde est de type client/serveur, où les Adaptive Server sont clients d'un serveur de sauvegarde.

Sauvegardes locales

Adaptive Server envoie des instructions au Backup Server local, via des appels de procédure à distance, lui indiquant les pages à sauvegarder ou à charger, les devices de sauvegarde à utiliser, ainsi que d'autres options. Backup Server effectue toutes les E/S disque.

Adaptive Server n'envoie ni ne lit des données sauvegardées ou chargées, il n'envoie que des instructions.

Sauvegardes à distance

Backup Server gère également les sauvegardes sur des machines distantes. Pour les sauvegardes et restaurations à distance, un serveur de sauvegarde local effectue les E/S disque en relation avec le device de base de données et envoie, via le réseau, les données au serveur de sauvegarde distant qui les stocke sur le device de sauvegarde.

Sauvegardes en ligne

Vous pouvez effectuer des sauvegardes lorsque la base de données est active. Certes, elles réduisent les performances des autres transactions mais n'hésitez pas pour autant à sauvegarder les bases de données essentielles aussi souvent que nécessaire afin de garantir la fiabilité du système.

Pour obtenir des informations complètes sur les stratégies de sauvegarde et de restauration, reportez-vous au *Guide d'administration système*.

Utilisation de seuils pour éviter le manque d'espace de journalisation

Si votre base de données dispose d' un espace de journalisation limité et qu'occasionnellement vous atteignez le *seuil ultime*, fixez un deuxième seuil qui vous laisse suffisamment de temps pour effectuer une sauvegarde du journal de transactions. Un manque d'espace de journalisation peut avoir des conséquences graves sur les performances. Les utilisateurs ne pourront exécuter aucune commande de modification de données tant que l'espace ne sera pas libéré.

Striping de sauvegarde

Si vous effectuez des sauvegardes de journal incrémentielles, utilisez striping (répartition par bandes) pour améliorer les performances.

Diminution du temps de restauration

Vous pouvez réduire le temps de restauration en modifiant le paramètre de configuration recovery interval. La valeur par défaut de 5 minutes par base de données convient à la plupart des installations. Ne réduisez cette valeur que si les contraintes fonctionnelles exigent un temps de restauration plus court. Cette modification risque d'augmenter le nombre des E/S nécessaires.

Pour plus d'informations, reportez-vous à la section "Optimisation de l'intervalle de reprise", page 357.

La vitesse de restauration peut également être ralentie par la valeur du paramètre de configuration housekeeper free write percent. La valeur par défaut de ce paramètre permet à la tâche housekeeper du serveur d'écrire les buffers modifiés sur le disque pendant les cycles d'attente du serveur, à condition que les E/S disque n'augmentent pas de plus de 20 %.

Ordre de restauration

Pendant la restauration, les bases de données système sont restaurées les premières. Les bases de données utilisateur sont ensuite restaurées en fonction de l'ordre des ID de base de données.

Bulkcopy

L'opération bulkcopy dans une table sur Adaptive Server est plus rapide si cette table ne contient ni index ni trigger actif. Lorsque vous exécutez bulkcopy en mode rapide, Adaptive Server effectue des journalisations réduites.

Il ne journalise pas les modifications effectives de la base de données mais uniquement les allocations de pages. L'absence d'index à mettre à jour permet d'économiser le temps de mise à jour des index pour chaque insertion de données et celui de la journalisation des modifications des pages d'index.

Pour utiliser bulkcopy en mode rapide :

- Supprimez les index, puis recréez-les lorsque l'opération est terminée.
- Utilisez alter table...disable trigger pour désactiver les triggers durant la copie ; utilisez alter table...enable trigger une fois la copie terminée.
- Définissez l'option select into/bulkcopy/pllsort avec sp_dboption. N'oubliez pas de désactiver cette option à la fin de l'opération bulkcopy.

Pendant une opération bulkcopy en mode rapide, les règles habituelles ne sont pas appliquées mais les valeurs par défaut le *sont*.

Comme les modifications de données ne sont pas journalisées, vous devez exécuter dump database tout de suite après une opération bulkcopy en mode rapide. Une opération de ce type dans une base de données empêche l'utilisation de dump transaction, puisque les modifications de données, qui ne sont pas journalisées, ne peuvent pas être restaurées à partir d'une sauvegarde du journal de transactions.

Bulkcopy parallèle

En vue d'améliorer les performances, vous pouvez utiliser bulkcopy en mode rapide pour copier les données dans des tables partitionnées. Pour chaque session de bulkcopy, spécifiez la partition sur laquelle les données doivent résider.

Si votre fichier d'entrée est déjà trié, vous pouvez copier les données ordonnées dans des partitions. Ainsi, vous n'aurez pas à effectuer de tri lors de la création des index clusterisés.

Pour plus d'informations sur les procédures détaillées, reportez-vous à la section "Etapes de partitionnement des tables", page 113.

Batchs et bulkcopy

Si vous spécifiez une taille de batch pendant une opération bulkcopy en mode rapide, chaque nouveau batch doit commencer sur une nouvelle page de données, puisque seules les allocations de page (non les modifications de données) sont journalisées pendant une opération bulkcopy en mode rapide. La copie de 1000 lignes avec une taille de batch de 1 nécessite 1000 pages de données et 1000 enregistrements d'allocation dans le journal de transactions.

Si vous utilisez un batch de petite taille pour faciliter la détection des erreurs dans le fichier d'entrée, choisissez une taille de batch correspondant au nombre de lignes contenues dans une page de données.

Bulkcopy en mode lent

Si une table contient des index ou des triggers, une opération bulkcopy lente est automatiquement utilisée. Pour une opération bulkcopy en mode lent :

- La définition de select into/bulkcopy n'est pas obligatoire.
- Les règles habituelles ne sont pas appliquées et les triggers ne sont pas déclenchés mais les valeurs par défaut *sont* en vigueur.
- Toutes les modifications de données sont journalisées, ainsi que les allocations de pages.
- Les index sont mis à jour lorsque les lignes sont copiées et les modifications d'index journalisées.

Amélioration des performances de bulkcopy

Pour améliorer les performances des opérations bulkcopy, vous pouvez :

- Définir l'option trunc log on chkpt pour éviter que le journal de transactions ne soit rempli. Si votre base de données possède une procédure de seuil sauvegardant automatiquement le journal lorsqu'il est plein, vous économiserez le temps de sauvegarde des transactions.

  N'oubliez pas que chaque batch est une transaction différente. Par conséquent, si vous ne spécifiez pas de taille de batch, le fait de définir trunc log on chkpt ne vous aidera pas.

- attribuer au paramètre de configuration number of preallocated extents une valeur élevée si vous effectuez de nombreuses opérations bulkcopy volumineuses.

  Reportez-vous au *Guide d'administration système*.

- trouver la taille adéquate de paquet de réseau.

  Pour plus d'informations, reportez-vous à la section chapitre 2, "Réseaux et performances".

Remplacement de données dans une table volumineuse

Si vous remplacez toutes les données d'une table de grande taille, utilisez la commande truncate table plutôt que delete. En effet, truncate table effectue des journalisations réduites. Seules les libérations de pages sont journalisées.

La commande delete est complètement journalisée, c'est-à-dire que toutes les modifications de données le sont.

Les étapes de cette opération sont les suivantes :

1 Tronquez la table. Si cette dernière est partitionnée, vous devez supprimer les partitions pour pouvoir la tronquer.

2 Supprimez tous les index de la table.

3 Chargez les données.

4 Créez à nouveau les index.

Pour plus d'informations sur l'utilisation de bulkcopy avec des tables partitionnées, reportez-vous à la section "Etapes de partitionnement des tables", page 113.

Ajout d'un grand volume de données dans une table

Lorsque vous ajoutez 10 à 20 % (ou plus) de données dans une table volumineuse, supprimez les index non clusterisés, chargez les données, puis créez à nouveau les index non clusterisés.

Pour les tables de très grande taille, il peut s'avérer nécessaire de conserver l'index clusterisé en raison des contraintes d'espace. Adaptive Server doit effectuer une copie de la table lorsqu'il crée un index clusterisé. Dans la plupart des cas, lorsque les tables sont très grandes, le temps nécessaire pour effectuer une opération bulkcopy en mode lent avec index est inférieur au temps requis pour effectuer une opération bulkcopy en mode rapide et recréer l'index clusterisé.

Utilisation de partitions et de processus de bulkcopy multiples

Si vous chargez des données dans une table sans index, vous pouvez créer des partitions dans cette table et utiliser une session bcp pour chaque partition.

Pour plus d'informations, reportez-vous à la section "Utilisation de bcp parallèle pour copier des données dans les partitions", page 106.

Conséquences sur les autres utilisateurs

L'opération bulkcopy sur des tables de grande taille en entrée ou en sortie peut augmenter le temps de réponse des autres utilisateurs. Si possible :

- Effectuez les opérations bulkcopy aux heures de moindre activité serveur.
- Utilisez bulkcopy en mode rapide puisqu'il effectue moins de journalisations et moins d'E/S.

Vérificateur de cohérence de base de données

Effectuez régulièrement des vérifications de cohérence de base de données avec dbcc. Il est inutile de sauvegarder une base de données endommagée. Néanmoins, dbcc réduit les performances, étant donné que dbcc pose des verrous sur les objets qu'il vérifie.

Pour obtenir plus d'informations sur la manière de limiter les effets de dbcc sur les applications utilisateur, sur dbcc et le verrouillage, reportez-vous au *Guide d'administration système*.

Utilisation de *dbcc tune (cleanup)*

Adaptive Server effectue un contrôle supplémentaire de nettoyage de la mémoire afin de vérifier la cohérence finale après l'exécution de chaque tâche. Dans les environnements à très haut débit, vous pouvez légèrement améliorer les performances en passant outre cette vérification. Pour la désactiver, tapez la commande suivante :

```
dbcc tune(cleanup,0)
```

Le nettoyage final libère tout espace qu'une tâche pourrait garder. Si vous désactivez cette opération mais que vous obtenez des erreurs de mémoire, réactivez-la en entrant l'expression suivante :

```
dbcc tune(cleanup,0)
```

Détermination de l'espace disponible pour des activités de maintenance

Plusieurs opérations de maintenance ont besoin d'espace pour effectuer une copie des pages de données d'une table :

- create clustered index
- alter table...lock
- certaines commandes alter table qui ajoutent ou modifient des colonnes
- reorg rebuild sur une table

Ces commandes ayant généralement aussi besoin d'espace pour recréer des index, vous devez déterminer :

- la taille de la table et de ses index,
- l'espace disponible sur le segment où la table est stockée,
- les propriétés de gestion de l'espace définies pour la table et ses index.

Les sections suivantes décrivent les outils permettant d'obtenir des informations sur l'utilisation de l'espace et sa disponibilité.

Généralités sur les besoins en espace

Toutes les commandes qui copient des lignes d'une table recréent également tous les index de la table. Vous avez besoin d'espace pour effectuer une copie complète de la table et de ses index.

Ces commandes n'estiment pas l'espace nécessaire. Elles s'arrêtent simplement et affichent un message d'erreur lorsqu'elles manquent de place sur un segment utilisé par la table ou ses index. Pour les tables volumineuses, cet arrêt peut se produire plusieurs heures après le lancement de la commande.

Vous devez donc libérer de l'espace sur les segments utilisés par la table et ses index, selon les principes suivants :

- L'espace libre sur le segment de la table doit au moins être égal à :
  - la taille de la table, plus
  - environ 20 % de la taille de la table si celle-ci contient un index clusterisé et si vous passez d'un verrouillage APL (APL) à un verrouillage des pages de données seulement (DOL).
- L'espace libre sur les segments utilisés par les index non clusterisés doit être au moins égal à la taille des index.

Les index clusterisés de tables DOL comportent un niveau feuille au-dessus des pages de données. Si, dans une table possédant un index clusterisé, vous passez d'un verrouillage APL à un verrouillage DOL, l'index clusterisé résultant exige plus d'espace. L'espace supplémentaire requis dépend de la taille des clés d'index.

Outils servant à vérifier l'utilisation de l'espace et sa disponibilité

Notez toutefois que la copie d'une table et de ses index exige un espace égal à celui qu'occupent ces objets, plus 20 % environ. Toutefois :

- Si des modifications de données ont entraîné la création de plusieurs pages à demi-remplies, l'espace requis pour copier la table peut être inférieur à la taille actuelle de celle-ci.
- Si des propriétés de gestion de l'espace de la table ont changé ou si l'espace requis par fillfactor ou reservepagegap a été utilisé par des modifications de données, l'espace nécessaire pour copier la table peut être au contraire supérieur à la taille de celle-ci.

- L'ajout de colonnes ou l'attribution à des colonnes existantes de types de données consommateurs d'espace exigent davantage de place.

Un espace de journalisation est également nécessaire.

Vérification de l'espace utilisé par les tables et les index

Pour afficher la taille d'une table et ses index, utilisez la procédure suivante :

```
sp_spaceused titles, 1
```

Pour plus d'informations sur l'estimation de la taille des index clusterisés, reportez-vous à "Calcul des tailles des tables DOL", page 379.

Vérification de l'espace sur les segments

Les tables sont toujours copiées dans l'espace libre sur le segment où elles sont stockées, et les index sont également recréés sur le segment où ils se trouvent. Les commandes qui créent des index clusterisés peuvent spécifier un segment. La copie de la table et l'index clusterisé sont créés sur le segment cible.

Pour déterminer le nombre de pages disponibles sur un segment, utilisez sp_helpsegment. La dernière ligne de sp_helpsegment indique le nombre total de pages libres utilisables sur un segment.

La commande suivante affiche des informations relatives au segment default, où sont stockés les objets lorsqu'aucun segment n'est spécifié :

```
sp_helpsegment "default"
```

sp_helpsegment indique les noms des index sur le segment. Si vous ignorez le nom du segment pour une table, utilisez sp_help et le nom de la table. Les noms de segment des index sont également signalés par sp_help.

Vérification des besoins en espace pour les propriétés de gestion de l'espace

Après une modification significative des propriétés de gestion de l'espace, une table que vous copiez peut se révéler bien plus grande ou bien plus petite que l'originale. Les valeurs définies pour les propriétés de gestion de l'espace sont stockées dans les tables sysindexes et sont affichées à l'aide de sp_help et sp_helpindex. Le résultat suivant affiche les propriétés de gestion de l'espace pour la table titles :

```
exp_row_size reservepagegap fillfactor max_rows_per_page
------------ -------------- ---------- -----------------
         190             16         90                 0
```

sp_helpindex génère ce rapport :

```
index_name          index_description
    index_keys
    index_max_rows_per_page index_fillfactor index_reservepagegap
    ----------------------- ---------------- --------------------
title_id_ix          nonclustered located on default
    title_id
                         0               75                   0
title_ix             nonclustered located on default
    title
                         0               80                  16
type_price           nonclustered located on default
    type, price
                         0               90                   0
```

Propriétés de gestion de l'espace appliquées à la table

Durant la copie, les propriétés de gestion de l'espace de la table sont utilisées comme suit :

- Si une taille de ligne prévue est spécifiée pour la table et s'il y a passage d'un verrouillage APL à un verrouillage DOL, la taille de ligne prévue est appliquée aux lignes de données lorsqu'elles sont copiées.

  Si aucune taille de ligne prévue n'est définie mais s'il existe une valeur max_rows_per_page pour la table, une taille de ligne est calculée et appliquée.

  Sinon, la valeur par défaut définie par le paramètre de configuration default exp_row_size percent est utilisée pour chaque page allouée.

- La valeur de reservepagegap est appliquée lorsque des extents sont alloués à la table.
- Si sp_chgattribute a servi à sauvegarder une valeur de fillfactor pour la table, celle-ci s'applique aux nouvelles pages de données lorsque les lignes sont copiées.

Propriétés de gestion de l'espace appliquées à l'index

Lorsque les index sont reconstruits, les propriétés de gestion de l'espace les concernant sont appliquées, comme suit :

- Si sp_chgattribute a servi à enregistrer des valeurs de fillfactor pour les index, celles-ci sont appliquées au moment où les index sont recréés.
- Si des valeurs de reservepagegap sont définies pour des index, elles sont appliquées au moment où les index sont recréés.

Estimation de l'impact lié aux propriétés de gestion de l'espace

Le tableau 16-2 montre comment estimer l'impact des propriétés de gestion de l'espace.

Tableau 16-2 : Impact des propriétés de gestion de l'espace sur son utilisation

Propriété	Formule	Exemple
fillfactor	Requiert (100/fillfactor) * *nombre_pages* si les pages sont entièrement remplies	Un fillfactor de 75 requiert 1,33 fois le nombre de pages actuel ; une table de 1 000 pages passe alors à 1 333 pages.
reservepagegap	Augmente l'espace de 1/reservepagegap si les extents sont remplis	Un reservepagegap de 10 augmente l'espace utilisé de 10 % ; une table de 1 000 pages passe alors à 1 100 pages.
max_rows_per_page	Convertie en exp_row_size lors du passage au verrouillage DOL.	Pour plus d'informations, reportez-vous à la section tableau 16-3, page 409.
exp_row_size	L'augmentation dépend du nombre de lignes plus petites que exp_rowsize et de la longueur moyenne de ces lignes.	Si exp_row_size est égal à 10, et si 1 000 ont une longueur de 60, l'augmentation de l'espace est : (100 – 60) * 1000 ou 40 000 bytes, soit environ 20 pages supplémentaires.

Pour plus d'informations, reportez-vous à la section chapitre 13, "Définition des propriétés de gestion de l'espace".

Si, pour une table, la propriété max_rows_per_page est définie et si la table passe d'un verrouillage APL à un verrouillage DOL, la valeur est convertie en exp_row_size avant que la commande alter table...lock ne copie la table sur son nouvel emplacement.

La valeur exp_row_size est appliquée lors de la copie. Le tableau 16-3 précise la méthode de conversion des valeurs.

Tableau 16-3 : Conversion de max_rows_per_page en exp_row_size

Si *max_rows_per_page* vaut	**définissez *exp_row_size* à**
0	Une valeur en pourcentage définie par default exp_row_size percent
255	1, c'est-à-dire des pages entièrement remplies
1–254	La valeur la plus petite de : • la taille de ligne maximale • valeur max_rows_per_page

Insuffisance d'espace libre

Si l'espace ne suffit pas pour copier la table et recréer tous les index, essayez d'en libérer en supprimant les index non clusterisés sur la table. Sans index non clusterisés, il suffit de l'espace nécessaire à la table et à l'index clusterisé pour la copie.

Ne supprimez pas l'index clusterisé car il fournit l'ordre de copie des lignes. De plus, la recréation ultérieure pourrait nécessiter encore plus d'espace. Recréez les index non clusterisés une fois la commande de copie terminée.