ADMINISTRATION D'UNE BASE DE DONNEES

I-ADMINISTRATION D'UNE BASE DE DONNEES

Les principales tâches d'un administrateur d'une base de données sont les suivantes:

* Installation et mise à jour du noyau serveur et des outils d'application
* Planification des ressources de mémorisation des données
* Organisation des structures logiques et physiques des données
* Création et gestion des utilisateurs et de leurs droits d'accès (privilèges)
* Gestion et optimisation des performances du système
* Gestion de la sécurité du système: gestion des accès concurrents
* Gestion de la sécurité du système: sauvegardes, restaurations et archivages de la base
* Gestion de bases de données réparties

Les autres utilisateurs assurent les tâches suivantes:

Développeurs d'application:
* Conception et réalisation d'une application
* Conception de la structure de la base de données
* Evaluation des besoins en ressources de mémorisation
* Optimisation des performances de l'application
* Etablissement des mesures de sécurité

Utilisateurs d'application
* Saisie, modification et suppression de données
* Génération des états de sortie

Pour réaliser les tâches qui lui sont dévolues, l'administrateur de la base dispose

* **de deux comptes spéciaux**: **SYS** et **SYSTEM** créés en même temps que la base de données.

SYS est propriétaire des tables et des vues du dictionnaire de données. Ces tables et ces vues, essentielles pour le fonctionnement du serveur, ne peuvent être modifiées que par Oracle7 lui-même.
SYSTEM crée les tables et les vues qui fournissent les informations nécessaires à l'administration du sytème; il crée les tables et les vues utilisées par les outils Oracle.

SYS et **SYSTEM** possèdent le rôle OSDBA; ils disposent de tous les privilèges Système qui leur permettent d'exécuter toutes les opérations nécessaires au fonctionnement d'une instance.

* **d'outils spécifiques**:

a/ SERVER MANAGER:outil d'administration et de contrôle de la base de données qui permet de

* démarrer et arrêter une instance
* monter, démonter, ouvrir et fermer une instance
* contrôler en temps réel l'utilisation et les performances du serveur
* réaliser des sauvegardes et des restaurations
* exécuter des commandes SQL et PL/SQL

Server manager est appelé par les commandes:

* **svrmgrl** : mode caractères
* **svrmgrm**: mode graphique (motif)

svrmgrl permet d'exécuter les commandes suivantes:

Démarrage d'une instance

STARTUP [RESTRICT] [FORCE] [PFILE=filename]
[NOMOUNT
| MOUNT [EXCLUSIVE | {PARALLEL | SHARED} [RETRY]]
| OPEN [RECOVER] [dbname]
[EXCLUSIVE | {PARALLEL | SHARED} [RETRY]]]

Arrêt d'une instance

SHUTDOWN [NORMAL | IMMEDIATE | ABORT | dbname]

Activation du module Monitor

MONITOR { FILE | PROCESS | IO | LATCH | LOCK | ROLLBACK
| SESSION | STATISTIC | TABLE }

Activation ou désactivation de l'archivage automatique

ARCHIVE LOG {{STOP|LIST}|{START|NEXT|<n>|ALL}[TO 'destination']}

Restauration dune base ou de tablespaces

RECOVER { [DATABASE [UNTIL {CANCEL | CHANGE integer | TIME
date}]
[USING BACKUP CONTROL FILE]]
| [TABLESPACE ts-name [,tsname]]
| [DATAFILE 'filename' [,'filename']]}

Connexion à la base

CONNECT {[username [/password]]|[INTERNAL]} ['@'instance-spec]

Déconnexion de la base

DISCONNECT

Affectation de valeurs à des variables système
SET options: ARRAYSIZE, AUTORECOVERY, CHARWIDTH, COMPATIBILITY CYCLE, DATEWIDTH, ECHO, FETCHROWS, HISTORY, INSTANCE, LABWIDTH, LINES, LOGSOURCE, LONGWIDTH, MAXDATA, NUMWIDTH, RETRIES, SERVER OUTPUT, SPOOL, STOPONERROR, TERM, TERMOUT, TIMING

Affichage des valeurs de variables système
SHOW options: same as SET plus ALL, ERRORS, LABEL, PARAMETERS, SGA and VAR

Sortie de Server Manager
EXIT

Introduction d'un commentaire dans un script SQL
REMARK

Exécution d'une commande PL/SQL
EXECUTE pl/sql_block

Desciption d'un objet de la base
DESCRIBE {table_name | view_name | proc_name | package_name | function_name }

Exécution d'une commande système
HOST [os_command]

Impression de la valeur d'une variable définie avec la commande VARIABLE
PRINT variable

Activation ou désactivation d'un fichier de spooling
SPOOL [filename | OFF]

Déclaration d'une variable, utilisable avec les commandes EXECUTE ou PRINT
VARIABLE type name

Exécution de script SQL ou PL/SQL
@ script name

b/ SQL*LOADER: cet utilitaire permet de

* charger dans la bases des données, ayant des formats divers, provenant de fichiers externes.
* manipuler des champs de données avant leur insertion dans la base (contrôles de validité)
* distribuer des enregistrements d'un fichier dans plusieurs tables
* transformer plusieurs enregistrements physiques en un enregistrement logique

c/ EXPORT et **IMPORT** Utilities: utilitaires réalisant:

* l'archivage de données
* le transfert de données entre bases Oracle
* le stockage de données dans des fichiers externes à la base
* le stockage des définitions d'objets (tables, clusters, index) avec ou sans les données
* la sauvegarde des seules tables modifiées depuis le dernier export (export incrémental ou cumulatif)
* la restauration de données accidentellement supprimées

d/ ENTERPRISE MANAGER

Outil graphique d'Administration de Bases de Données permettant de réaliser, à partir d'un poste de travail Windows NT, les tâches suivantes :

-Administration, diagnostic, optimisation de plusieurs bases
-Distribution de software à des postes clients
-Programmation de Jobs s'exécutant à intervalles réguliers
-Gestion d'évènements à travers le réseau

SQL*LOADER,EXPORT-IMPORT et ENTERPRISE MANAGER sont présentés au chapitre VIII.

II-AUTHENTIFICATION D'UN ADMINISTRATEUR

L'administrateur d'une base de données doit réaliser des opérations particulières comme l'ouverture ou la fermeture de l'instance ; il doit pour cela bénéficier de privilèges spéciaux.
L'identification et l'authentification d'un administrateur répondent ainsi à des règles de sécurité très strictes ; elles peuvent se faire de deux manières différentes :

-Authentification Système
-Utilisation d'un fichier Password

Authentification Système

Sur la plupart des systèmes d'exploitation, l'authentification système impose de placer le login OS de l'administrateur dans un groupe spécial (groupe dba sous Unix).
Le paramètre d'initialisation remote_login_password doit être égal à NONE.

Fichier Password

Le fichier password est utilisé pour authentifier les utilisateurs possédant les privilèges SYSOPER ou SYSDBA qui permettent d'exécuter, sous svrmgrl, les commandes suivantes :

SYSOPER: STARTUP, SHUTDOWON, ALTERDATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVELOG, RECOVER

SYSDBA : STARTUP, SHUTDOWON, ALTERDATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVELOG, RECOVER avec l'option WITH ADMIN OPTION et CREATE DATABASE

Un fichier password est créé à l'aide de la commande
orapwd FILE=filename,PASSWORD=password,ENTRIES=max_users
(password :valeur du password pour sys et internal)

Le paramètre d'initialisation remote_login_password doit être égal à EXCLUSIVE

L'atribution des privilèges SYSOPER ou SYSDBA (ou des rôles OSOPER ou OSDBA) à un utilisateur ajoute cet utilisateur au fichier password. Elle se fait par un utilisateur ayant ces privilèges (SYS ou SYSTEM connectés sous svrmgrl avec le privilège SYSDBA).

Svrmgrl
SVRMGRL> connect systeme/password as SYSDBA

La liste des utilisateurs possédant les privilèges SYSDBA ou SYSOPER peut être visualisée à l'aide de la commande

```
Select username, sysdba,sysoper from v$pwfile_user
```

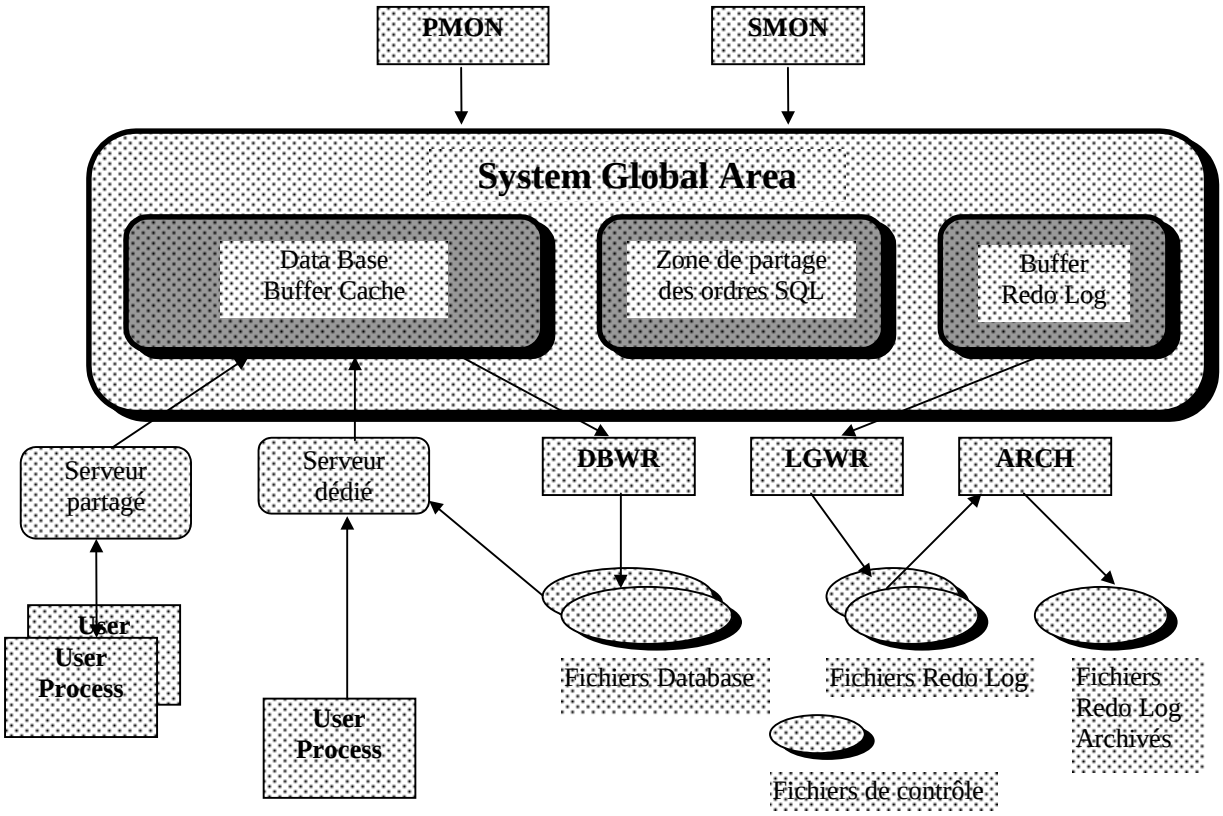
ARCHITECTURE D'UNE BASE DE DONNEES

Une instance est constituée de trois types d'éléments:

-**System Global Area** (SGA): ensemble des buffers nécessaires à la gestion des transactions

-**Process**: ensemble des processus Système et des processus Utilisateurs

-**Files**: ensembles des fichiers contenant les informations



I-SYSTEM GLOBAL AREA

Oracle crée et utilise des structures mémoire rassemblées dans la System Global Area (SGA), partagées par les différents utilisateurs.
La SGA et les processus background constituent une **instance**; l'espace mémoire nécessaire à la SGA est alloué au démarrage d'une instance et est restitué à la fermeture de cette instance.
Les données de la SGA sont partagées par l'ensemble des utilisateurs connectés à un moment donné; elles sont divisées en plusieurs types de buffers:

* **Data base Buffer Cache**: Il contient les blocs de données, les blocs d'index,des blocs contenant les ROLLBACK SEGMENTS et des blocs pour la gestion du système, les plus récemment utilisés; il peut contenir des données modifiées qui n'ont pas encore été enregistrées sur disque.

* **Redo Log Buffer**: Il contient les redo entries (toutes les données avant leur mise à jour,toutes les modifications effectuées sur ces données,la trace de toutes les transactions validées ou non encore validées),ensemble des modifications réalisées sur la base; ces redo entries sont mémorisées sur un redo log file, qui pourra être utilisé en cas de panne.

* **Zone de partage des ordres SQL**: cette zone est utilisée pour mémoriser,analyser et traiter les ordres SQL soumis par les utilisateurs

II-LES PROCESSUS

Une base Oracle contient deux types de processus:

* User Process
* Oracle Process

Un **User Process** est créé et maintenu pour exécuter le code d'un programme applicatif (ex:application Oracle Forms) ou d'un outil Oracle (ex: Server Manager); le User process communique avec les Process Server à travers le programme interface.

Les **Oracle Process** sont divisés en deux catégories:

*** Process Server** qui prennent en charge les demandes des utilisateurs
Le Process Server est responsable de la communication entre la SGA et le Process User; il analyse et exécute les ordres SQL, lit les fichiers DATABASE et ramène les blocs de données en SGA, retourne le résultat au Process User.
Oracle peut être configuré de deux façons:
- avec un ***dedicated server***, un server process traite les requêtes d'un seul user process
- avec un ***multi-threaded*** server, plusieurs user processes se partagent un petit nombre de server processes, minimisant le nombre de server processes et maximisant l'utilisation des ressources système.

* **Background Process** qui prennent en charge les mécanismes internes d'Oracle

Ils ont pour nom ora_processname_*SID*

DataBase Writer (DBWR):écrit les blocs modifiés de la base dans les fichiers Datafile, d'une manière désynchronisée par rapport aux transactions, en utilisant une LRU list
Log Writer (LGWR): écrit le contenu du buffer Redo Log de la SGA dans le fichier Redo Log en ligne lors d'un COMMIT
Checkpoint (CKPT): signale au DBWR la nécessité d'un CHECKPOINT et trace cet évènement dans les fichiers de contrôle et dans les en-têtes des fichiers Datafile. Il est facultatif; s'il est absent il est suppléé par LGWR.
System Monitor (SMON): il rétablit la cohérence du système après un incident et libère les ressources utilisées par le système
Process Monitor (PMON): il récupère les anomalies des process USER; il supprime les process en erreur, annule les transactions non validées, libère les verrous, libère les ressources utilisées dans la SGA. Il contrôle également les dispatchers et les process serveurs.
Archiver (ARCH): il recopie les fichiers redo log pleins sur un fichier archive pour pallier une perte éventuelle d'un fichier DATABASE (optionnel,existe en mode ARCHIVELOG uniquement)
Recoverer(RECO): il est utilisé pour résoudre les transactions interrompues par une panne dans un système de bases de données distribuées
Dispatcher(Dnnn): processus présent dans une configuration multi-threaded. Il y a au moins un de ces processus pour chaque protocole de communication. Il dirige les requêtes d'un utilisateur vers un serveur partagé et lui renvoie ses requêtes.
Lock(LCKn): de 1 à 10 processus de verrouillage peuvent être utilisés lorsque Oracle Parallel Server est installé.

Le programme Interface

C'est un mécanisme par lequel un programme utilisateur communique avec le server process; il est utilisé comme une méthode de communication standard entre un client et Oracle. Il agit comme un mécanisme de communication en formattant les données, transférant les données, interceptant et retournant les erreurs. Il réalise les conversions de données, en particulier entre différents types d'ordinateurs ou avec des données de programmes externes.

III-LES FICHIERS

Il existe quatre types de fichiers
* Fichiers Datafile
* Fichiers Redo Log
* Fichiers Control
* Fichiers Archivage

Fichiers Datafile
Ils contiennent toutes les données de la base; toutes les structures logiques et physiques y sont stockées (tables, index, rollback segments). Ils possèdent les caractéristiques suivantes:

* un fichier Datafile peut être associé à une seule base de données
* les fichiers Datafile ont un ensemble de caractéristiques qui permet de leur allouer automatiquement une extension en cas de dépassement de capacité
* un ou plusieurs Datafiles forment une unité logique appelée tablespace, présentée au chapître 3.

* un fichier Datafile est constitué d'un ensemble de blocs dont la taille dépend du sytème d'exploitation.

Les nouvelles données et les données modifiées ne sont pas nécessairement écrites immédiatement sur un fichier Datafile; afin d'optimiser les performances du sytème, elles sont mémorisées dans la SGA et sont écrites périodiquement sur les fichiers Datafile par le process DBWR.

Fichiers Redo Log

Ils contiennent toutes les données modifiées et sont utilisés en cas de perte des fichiers Datafile; ils sont au minimum deux et ont un fonctionnement circulaire. Ils peuvent être utilisés de façon unique ou multiplexée:

* *Façon unique*: un seul fichier Redo Log est en service à un moment donné; quand un fichier est plein, le deuxième est mis en service; les modifications ne sont stockées qu'une seule fois.
* *Façon multiplexée*: plusieurs groupes de plusieurs fichiers Redo Log sont en service et mis à jour simultanément.

Fichiers Control file

Chaque base possède au moins un fichier de contrôle. Il est hautement recommandé de le multiplexer pour des raisons de sécurité. Ils contiennent la description physique de la base:

* nom de la base
* nom et chemin d'accès des fichiers Dafile et Redo Log
* date et heure de création de la base
* informations concernant la cohérence de la base (checkpoint)

Ils sont utilisés au démarrage d'une instance et pour la restauration si nécessaire. Ils sont modifiés à chaque modification structurelle de la base.

Fichiers Archivage

Ils contiennent des copies des fichiers Redo Log (mode ARCHIVELOG uniquement)

Fichier init*SID*.ora

Il contient les paramètres de fonctionnement d'une instance et un paramètre identifiant le ou les fichiers CONTROL; il est utilisé à la création ou au démarrage d'une instance (SID: nom de la base)

.

Les différents paramètres définissent

* les limites des ressources de la base
* les nombres maximum d'utilisateurs ou de process simultanés
* les noms des fichiers et des répertoires utilisés par le système

L'ensemble de ces paramètres est mémorisé dans la vue v$parameter; il est présenté en Annexe C. Tous les fichiers sont stockés dans le répertoire ORACLE_HOME/dbs, ORACLE_HOME contenant le répertoire d'installation de Oracle.

IV-LE DICTIONNAIRE DE DONNEES

Le dictionnaire de données est formé par un ensemble de tables système contenant toutes les informations sur les structures logiques et physiques de la base:

*noms des utilisateurs
*privilèges et rôles de chaque utilisateur
*noms et caractéristiques des objets de la base (tables, vues, snapshots, index, clusters, synonyms, séquences, procédures,fonctions, packages, triggers, etc..)
*contraintes d'intégrité
*ressources allouées
*activité de la base
*etc....

Seul Oracle peut mettre à jour les tables du dictionnaire de données. Il contient des vues accessibles aux utilisateurs à l'aide de l'ordre SELECT. Il est conservé dans le tablespace SYSTEM; il est la propriété de l'utilisateur SYS.
Les classes de vues:

USER_...: informations sur tous les objets dont l'utilisateur est propriétaire
ALL_.....: informations sur tous les objets accessibles par l'utilisateur connecté
DBA_....: informations sur tous les objets de la base(utilisable uniquement par les utilisateurs ayant le privilège SELECT ANY TABLE)
V$.....: informations sur dynamic performance tables décrivant l'état actuel du système (locks,rollback segments,control files,etc....).
Les vues les plus couramment utilisées possèdent un synonyme simple.

L'annexe A présente l'ensemble de ces vues; le contenu de chacune d'elles est fourni dans Oracle7 Server Reference.

Le dictionnaire de données a deux usages principaux:

* vérification de chaque requête DDL (syntaxe et privilèges)
* informations sur la structure de la base

V-ORGANISATION DES REPERTOIRES

Le serveur Oracle7 est installé dans le répertoire défini dans la variable ORACLE_HOME; il contient les sous-répertoires suivants:

bin: exécutables de tous les produits
dbs: fichiers: init*sid*.ora datafiles logfiles controlfiles
rdbms: outils d'administration
lib: bibliothèques des produits Oracle
otrace: Oracle trace
ows: Oracle WebServer
guicommon2: bibliothèques,fichiers,scripts et messages utilisés par les produits d'interface
network: SQL*NET version 2
et pour chaque produit installé
product_name: admindemo doc install lib mesg

Le serveur Oracle7 utilise également des répertoires admin,arch,bg,core,db,user pour accueillir en particulier les fichiers trace contenant le détail de l'exécution des processus; les localisations de ces répertoires sont définies par les valeurs des paramètres background_core_dump, core_dump_dest, log_archive_dest, use_dump_dest du fichier init*SID*.ora.

VI-QUESTIONS

I/ A l'aide des commandes sytème, rechercher le nom de chaque instance Oracle démarrée sur le système

II/ Rechercher l'ensemble des processus actifs d'une instance

III/ A l'aide de l'outil svrmgrm
- trouver le nom des fichiers Datafile et Log file de la base IUP
- trouver les caractéristiques principales de la SGA
- afficher les paramètres qui ont servi au démarrage de l'instance

IV/ Afficher le nom et la description des vues du dictionnaire de données;repérer les vues contenant les informations relatives à l'architecture de la base de données.

V/ A l'aide de ces vues, retrouver les informations suivantes:
- taille des différents buffers de la SGA
- répertoire de mémorisation des fichiers datafile,log file et control file
- taille des fichiers datafile
- état et nom des fichiers log file et control file

CREATION D'UNE BASE DE DONNEES

I-CREATION D'UNE BASE DE DONNEES

La création d'une base de données comporte les étapes suivantes:

a/ Sauvegarde des bases existantes: cette opération est facultative mais recommandée

b/ Création des fichiers paramètres: Chaque instance est démarrée à l'aide d'un fichier paramètre **init*SID*.ora** ou ***SID*** est le nom de la base; un modèle de fichier paramètre est fourni avec la distribution du noyau (init.ora).Le fichier paramètre de la base devra indiquer au minimum les valeurs des paramètres suivants: db_name, db_domain, control_files, db_block_size, db_block_buffers, processes, rollback_segments; par défaut, Oracle recherche ce fichier dans ORACLE-HOME/dbs.

c/ Démarrage d'une base de données

Le démarrage d'une base de données se fait en trois étapes:

- démarrage de l'instance
- montage de la base
- ouverture de la base

Le démarrage de l'instance déclenche l'allocation de l'espace pour la SGA et la création des background processes; aucun fichier datafile ou logfile n'est associé à l'instance.
Une instance est identifiée par son nom, qui est mémorisé dans la variable ORACLE_SID
ex: ORACLE_SID=test; export ORACLE_SID
Il est également nécessaire de charger la variable ORACLE_HOME avec le nom du répertoire racine du noyau
ex: ORACLE_HOME=/net4/oracle ; export ORACLE_HOME

Le montage de la base associe une base avec une instance; l'instance ouvre les fichiers Control file; cette option permet à l'administrateur de réaliser différentes opérations telles que restauration ou sauvegarde, les autres utilisateurs n'ayant pas accès à la base.

L'ouverture de la base rend la base disponible pour les opérations des utilisateurs; Oracle ouvre les fichiers Datafile et les fichiers redo log on line

Ces trois étapes sont réalisées à l'aide de la commande STARTUP de l'outil svrmgrl

Démarrage de l'instance:
STARTUP NOMOUNT pfile = …/init*SID*.ora
Montage de la base
STARTUP MOUNT pfile = …/init*SID*.ora
Ouverture de la base
STARTUP OPEN pfile = …/init*SID*.ora

Le paramètre pfile indique le répertoire de stockage du fichier init*SID*.ora; par défaut, Oracle recherche ce fichier dans le répertoire ORACLE_HOME/dbs.
Le passage d'une étape à la suivante se fait à l'aide des commandes de l'outil svrmgrl (server manager en mode caractères) et avec le mot clé INTERNAL
Sous UNIX, un utilisateur peut utiliser svrmgrl s'il appartient à un groupe d'administration de la base (DBA par défaut); les groupes sont définis dans le fichier /etc/group.

En mode caractères l'appel à svrmgrl se fait par la commande
svrmgrl
SVRMGR> Connect internal
Connected

d/ Création de la base à l'aide de la commande CREATE DATABASE

Cette commande exécute les opérations suivantes:
- création des fichiers Datafile
- création des fichiers Control file
- création des fichiers Redo Log file
- création du tablespace SYSTEM et du SYSTEM rollback segment
- création du dictionnaire de données dans le tablespace SYSTEM
- création des utilisateurs SYS/CHANGE_ON_INSTALL et SYSTEM/MANAGER
- spécification de l'ensemble de caractères utilisé pour stocker les données dans la base
- MOUNT et OPEN de la base

Syntaxe de la commande CREATE DATABASE

CREATE DATABASE database
DATAFILE filespec
[AUTOEXTEND OFF]
ON [NEXT integer K] [MAXSIZE UNLIMITED]
M integer K
M
LOGFILE [GROUP integer] filespec

[CONTROLFILE REUSE]
[MAXLOGFILES integer]
[MAXLOGMEMBERS integer]
[MAXLOGHISTORY integer]
[MAXDATAFILES integer]
[MAXINSTANCES integer]
[CHARACTER SET 'US7ASCII']
[ARCHIVELOG]
NOARCHIVELOG
[EXCLUSIVE]

filespec
'filename' SIZE integer K/M [REUSE]

e/ Création des vues du dictionnaire de données

Elle se fait à l'aide des scripts SQL fournis dans le répertoire ORACLE_HOME/rdbms/admin

- sous SYS: catalog.sql vues et synonymes publics
  catproc.sql procédures système
  cataudit.sql audit

- sous SYSTEM, et pour chaque administrateur de la base
  catdbsyn.sql synonymes sur les vues DBA_*

f/ Activation de sqlplus

- sous SYSTEM exécuter la procédure pupbld.sql qui se trouve dans ORACLE_HOME/sqlplus/admin

g/ Restauration des bases sauvegardées en a/

II-ACTIONS SUR L'ETAT D'UNE BASE

Démarrage d'une base

STARTUP [FORCE] [RESTRICT] [PFILE=nom_fich] [OPEN
MOUNT
NOMOUNT]

FORCE: fermeture de l'instance si ouverte, puis démarrage
RESTRICT: pour les utilisateurs dont le privilège SYSTEM correspond à RESTRICTED SESSION
PFILE: nom du fichier init.ora à utiliser
NOMOUNT: démarrage de l'instance
MOUNT: démarrage de l'instance et ouverture des fichiers CONTROL
OPEN: démarrage complet de la base

Modification de l'état d'une base

ALTER DATABASE nom_base MOUNT
OPEN

Arrêt d'une base

SHUTDOWN	ABORT	arrêt brutal de tous les process
	IMMEDIAT	annulation des transactions en cours
	NORMAL	avec attente de deconnexion

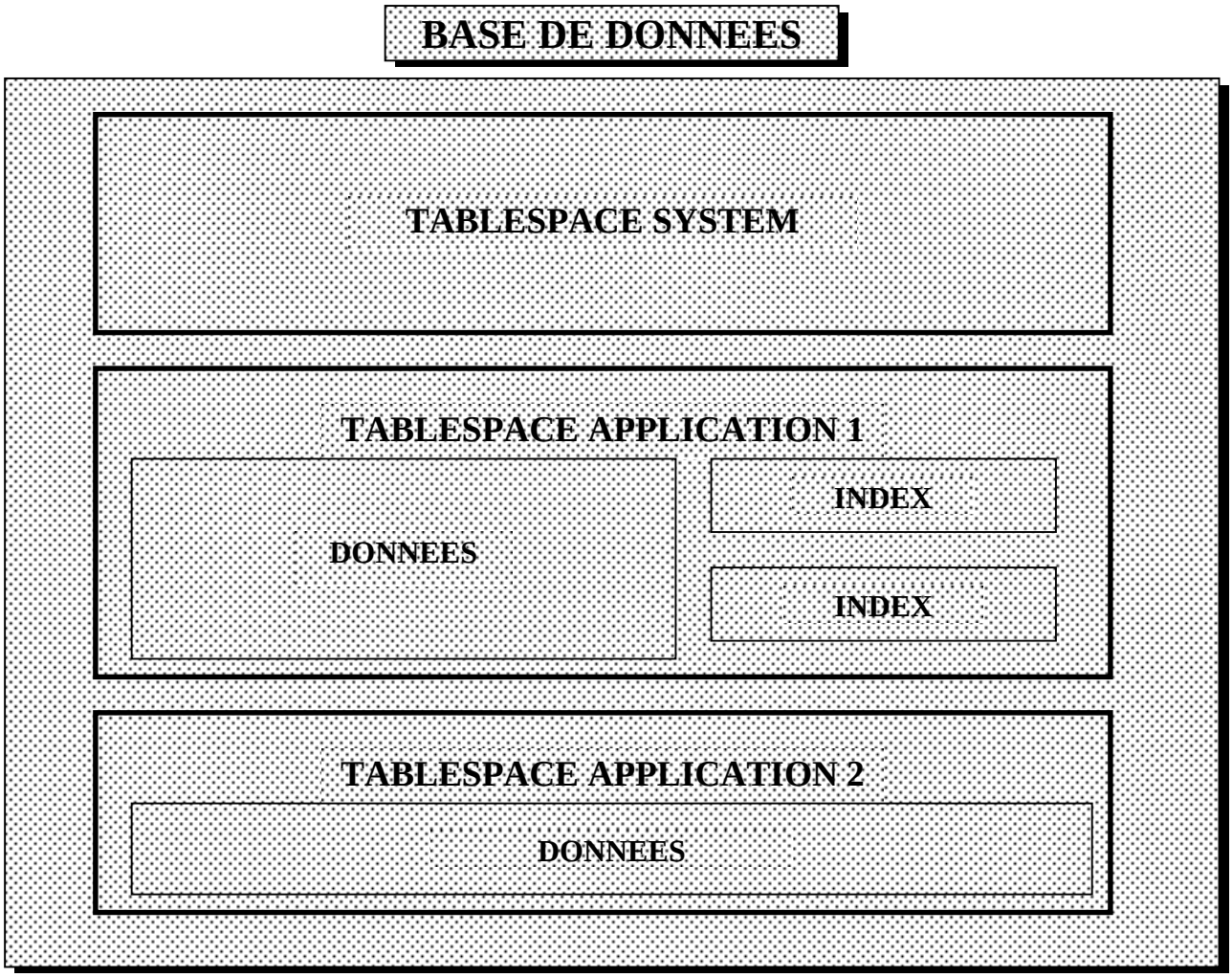
III-LES TABLESPACES

Les données d'une base Oracle sont mémorisées dans une ou plusieurs unités logiques appelées **tablespaces** et physiquement dans des fichiers associés à ces tablespaces.

L'administrateur de la base peut utiliser le concept de tablespace pour

* contrôler l'allocation d'espace disque
* assigner des quotas de ressource disque aux utilisateurs
* contrôler la disponibilité des données en rendant les tablespaces online ou offline
* constituer des unités de sauvegarde ou de restauration partielle de la base
*répartir les zones de stockage entre plusieurs disques pour accroître les performances

Chaque base contient au moins un tablespace appelé SYSTEM, qui est automatiquement créé par l'ordre CREATE DATABASE; ce tablespace SYSTEM contient toujours les tables du dictionnaire de données, les procédures, les fonctions, les packages, les triggers et le rollback segment SYSTEM.



L'administrateur de la base peut créer d'autres tablespaces à l'aide de la commande CREATE TABLESPACE et attribuer aux utilisateurs des droits d'accès à ces tablespaces.

CREATE TABLESPACE tablespace
DATAFILE filespec
[AUTOEXTEND OFF]
ON [NEXT integer K] [MAXSIZE UNLIMITED]
M integer K
M
[DEFAULT STORAGE storage_clause]
[ONLINE]
OFFLINE
[PERMANENT]
TEMPORARY

Un tablespace peut être online ou offline. L'administrateur peut rendre un tablespace offline pour:

* rendre une partie de la base non accessible, alors qu'un accès normal continue sur les autres tablespaces
* faire la sauvegarde des informations contenues dans ce tablespace
* rendre une application et ses tables innaccessibles pendant la maintenance de l'application

Un tablespace est constitué d'un ou plusieurs fichiers physiques qui contiennent les différents types de segments(données, index, rollback); la taille d'un tablespace peut être augmentée en lui affectant un nouveau fichier à l'aide de la commande ALTER TABLESPACE.

IV-LES ROLLBACK SEGMENTS

Une base de données contient un ou plusieurs ROLLBACK SEGMENTS; un rollback segment enregistre les actions d'une transaction qui peuvent être annulées en cas d'incident afin de remettre la base de données dans un état cohérent.
Un rollback segment est constitué de plusieurs « entrées », chacune d'elles contenant le nom du fichier et le numéro de block modifiés par la transaction ainsi que le contenu du bloc de données avant la transaction.
Le rollback segment SYSTEM est créé lors de la création de la base dans le tablespace SYSTEM; il n'est utilisé que pour les transactions portant sur les données du dictionnaire (commandes du langage de définition).
Un ou plusieurs autres rollback segments doivent exister pour les transactions portant sur des données utilisateur; leur nombre est fonction du débit transactionnel (cf Organisation physique des données).

Un rollback segment est créé à l'aide de la commande

CREATE ROLLBACK SEGMENT rollback_name

TABLESPACE tablespace_name
STORAGE clause
A sa création, un rollback segment est *offline*; il doit être mis *online* à l'aide de la commande
ALTER ROLLBACK SEGMENT rolback segment_name online
pour être utilisé pendant la session.
Pour être en permanence *online* un rollback segment doit figurer dans le fichier init.ora de l'instance.
Un rollback segment est supprimé par la commande DROP ROLLBACK SEGMENT rollback segment_name; il devra avoir été mis offline auparavant.
Les caractéristiques des rollback segments sont mémorisées dans la vue sys.dba_rollback_segs du dictionnaire de données.

Une transaction peut être orientée dans un rollback segment avec la commande

SET TRANSACTION USE ROLLBACK SEGMENT nom_rollback

c'est alors la première commande de la transaction

V-QUESTIONS

I/ Etudiez les scripts fournis en Annexe D et déterminer la structure de la base créée par CretdbGEN.sql.
Inspirez vous de ces scripts pour créer votre propre base.

II/ Création d'une instance

Sur la station qui vous a été désignée (où vous appartenez au groupe dba), réalisez, dans l'ordre indiqué, les opérations suivantes pour créer une nouvelle instance

A/ *Définition de l'environnement de travail*

-Placez vous dans le groupe dba à l'aide de la commande newgrp dba

-Exécutez la commande umask 002 pour permettre à Oracle d'écrire dans vos répertoires

-Positionnez les variables d'environnement
ORACLE_HOME=/net4/oracle
ORACLE_SID=<login> (<login> représentera toujours votre nom d'utilisateur Oracle)

-Créez le répertoire /oracle/<login>/scripts
cd /oracle
mkdir -p /login>/scripts
et recopiez les scripts de /net4/oracle/data/GENERIC/scripts dans ce répertoire

-Créez les répertoires d'accueil des fichiers trace suivants:
/oracle/<login>/admin
/oracle/<login>/arch
/oracle/<login>/bg
/oracle/<login>/core
/oracle/<login>/db
/oracle/<login>/user

-Désactivez l'instance IUP à l'aide de la commande **unset TWO_TASK**

B/ *Création de la base <login>*

Créez la base de données <login> ayant les paramètres suivants:

- tous les fichiers seront mémorisés dans /oracle/<login>
- les fichiers REDO LOG seront désignés par log<login>1(2).dbf et auront une taille de 200K (2 groupes de deux fichiers)
- le fichier Datafile sera mémorisé dans le même répertoire et aura une taille de 10M. son nom sera sys<login>.dbf

C/ *Création des vues du dictionnaire de données*

Sous svrmgrl,exécutez les procédures cataloguées permettant de créer les vues du dictionnaire de données

D/Activez sqlplus

E/Démarrez l'instance créée; vérifiez que les processus Oracle sont activés

III/ Retrouvez à l'aide des vues du dictionnaire de données les caractéristiques physiques (tablespaces, rollback segments,...) de cette base

IV /Sur la base que vous venez de créer précédemment, créez un tablespace ayant les caractéristiques suivantes:

Nom du tablespace:	<login>
Nom du fichier associé:	<login>.dbf
Localisation du fichier associé:	/oracle/<login>
Taille du fichier associé:	200K

V/ Dans le tablespace créé, créez la table T_<login> (col1 number(3),col2 char(10)).

VI/ Assurez-vous que la table a été créée dans le bon tablespace.

VII/ Insérez une ligne dans la table T_<login>; Que se passe-t-il?

VIII/ Créez le rollback segment RBS_<login> dans le tablespace <login>. Essayez à nouveau d'insérer une ligne dans la table T. Que se passe-t-il? Comment peut-on remédier au problème?

IX/ Ecrivez une requête SQL qui affiche toutes les caractéristiques des rollback segments de la base

X/ Créez un deuxième rollback segment RBS_<login>2; Insérerz deux lignes dans la table T_<login> et faite en sorte que l' insertion se déroule à l'aide du rollback segment RBS_<login> pour la première puis de RBS_<login>2 pour la deuxième; vérifiez cette propriété à l'aide des tables virtuelles du dictionnaire de données v$rollname et v$rollstat. (valeur du champ Xacts de la vue v$rollstat)

ORGANISATION PHYSIQUE D'UNE BASE DE DONNEES

I- ORGANISATION GENERALE

Une base de données Oracle est physiquement constituée par un ensemble de fichiers où sont stockées les données; elle est divisée en unités logiques appelées tablespaces (cf chapitre 3).
Le niveau le plus fin de granularité est le **bloc** (appelé aussi bloc logique, bloc Oracle ou page); il correspond à un nombre spécifique de bytes, défini à la création de la base. La taille d'un bloc est un multiple de la taille d'un bloc du système d'exploitation; sa valeur est donnée par le paramètre db_block_size.
Un ensemble de blocs contigüs forme un **extent**, contenant un type particulier d'informations (table, index,…).

Un **segment** est un ensemble d'extents alloués pour un type spécifique d'informations, stockées dans le même tablespace; on distingue les types de segments suivants :

-segment de données: Chaque table non liée à un cluster ou chaque cluster est stocké dans un segment de données créé par les commandes CREATE TABLE ou CREATE CLUSTER.

-segment d'index: l'index est stocké dans un segment index créé par la commande CREATE INDEX; tous les extents alloués à un segment index lui sont conservés aussi longtemps que l'index existe; lorsque la table associée ou l'index sont supprimés, l'espace est utilisé pour d'autres usages dans le tablespace.

-rollback segment: Chaque base de données contient un ou plusieurs rollback segments; un rollback segment contient les actions d'une transaction qui pourrait être annulée en cas d'incident; il est utilisé pour assurer la cohérence des lectures, pour détruire certaines transactions ou pour restaurer la base de données.

-segment temporaire: il est utilisé pour mémoriser temporairement des informations pendant des requêtes de tris ou contenant une clause group by par exemple; il est stocké dans un tablespace créé à cet effet.

-segment de démarrage: il est créé à la création de la base; il contient les définitions des objets du dictionnaire de données et est chargé à l'ouverture de la base.

Caractéristiques physiques d'un élément de mémorisation
Les caractéristiques physiques des tables, index, clusters, rollback segments et tablespaces sont définies par la clause **Storage clause** contenant les paramètres suivants:

-INITIAL	integer	K/M:	taille en bytes du premier extent
-NEXT	integer	K/M:	taille en bytes du prochain extent

-MINEXTENTS	integer	nb d'extents alloués à la création (minimum 2 pour les rollback segments)
-MAXEXTENTS	integer	nb maximum d'extents
-PCTINCREASE	integer	% d'augmentation entre 2 extents (par defaut 50)
-OPTIMAL	integer K/M	taille optimale d'un rollback segment

exemple:
CREATE TABLE dept(deptno number(2), dname varchar2(14), loc varchar2(13))
STORAGE (initial 100K next 50K minextents 1 maxextents 50 pctincrease 5)

Les commandes CREATE TABLE ou CREATE CLUSTER contiennent également les paramètres suivants:

-PCTFREE integer: % d'espace réservé dans chaque bloc pour des modifications (update) ultérieures (10% par défaut)

-PCTUSED integer: % minimum d'espace utilisé dans un bloc (40% par défaut) pour insérer de nouveaux enregistrements après des suppressions ; (la somme PCTFREE + PCTUSED doît être inférieure à 100).

-INITRANS integer: nb initial d'entrées transactions allouées à un bloc (1-255); chaque transaction qui modifie un bloc demande une entrée dans le bloc

-MAXTRANS integer: nb maximum de transactions concurrentes qui peuvent modifier un bloc alloué à une table (1-255)

Les caractéristiques des différents segments sont fournies par la vue DBA_SEGMENTS.

II- LES TABLES

Les tables, indépendantes ou faisant partie d'un cluster, sont mémorisées dans les segments de données des fichiers Datafile. La figure suivante présente l'organisation physique de ces fichiers et la structure des informations à l'intérieur d'un bloc physique.

Un bloc physique comprend trois parties:

-**ENTETE**: informations générales sur la structure du bloc (type d'information, propriétaire, date de création,…)

-**DONNEES**: zone de mémorisation des données (enregistrements)

-**DISPO**: partie du bloc destinée à mémoriser les modifications d'enregistrements en minimisant le nombre de chaînages ; sa dimension est définie à partir des paramètres PCTFREE et PCTUSED.

Selon l'activité transactionnelle sur la table, il faut:

- augmenter PCTFREE si beaucoup de modifications augmentent la longueur des données.

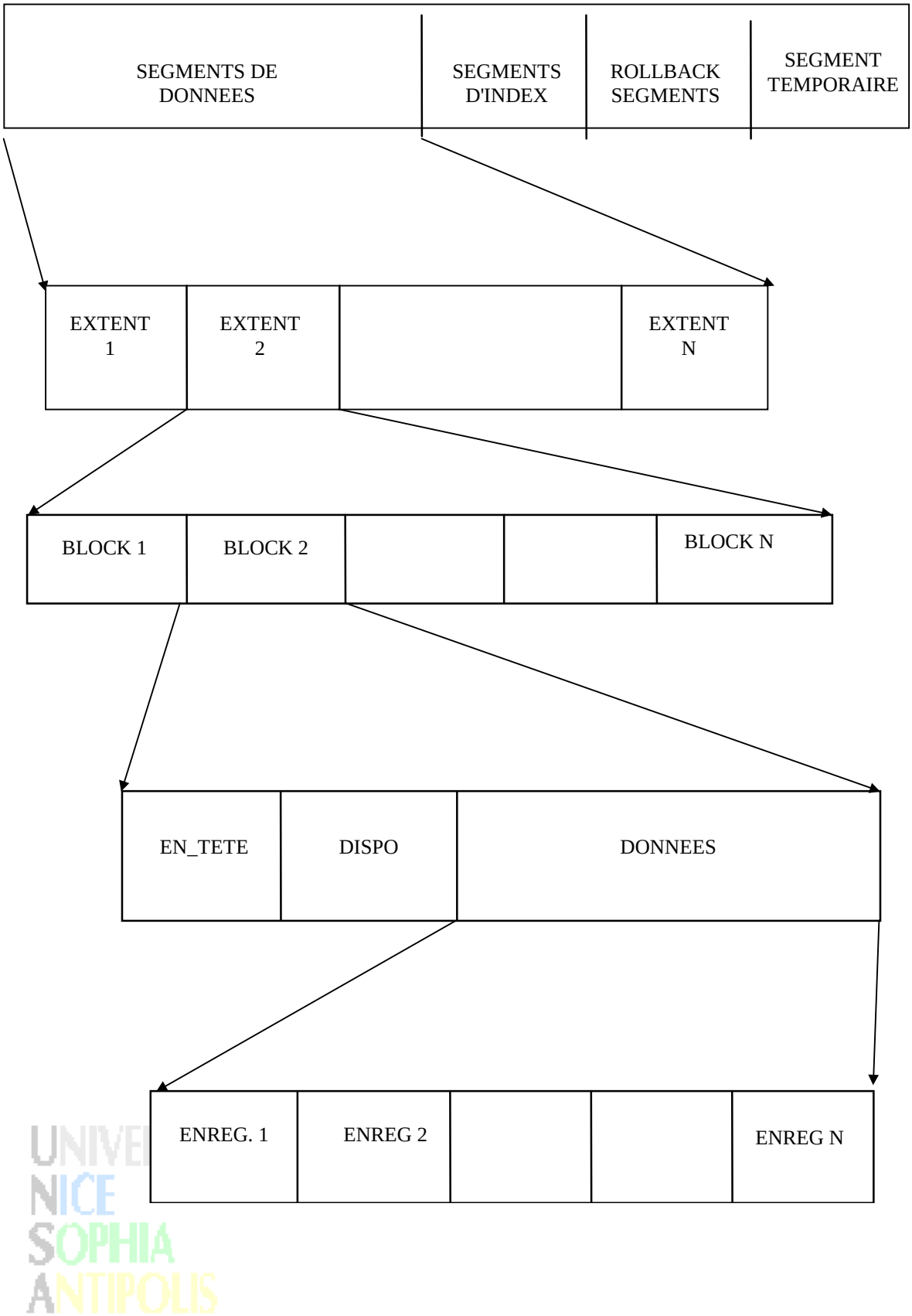
Une valeur élevée de PCTFREE privilégie les opérations de mise à jour et est adaptée à une base active, elle implique la réservation d'un espace plus importante; une valeur faible de PCTFREE est adaptée à une base stable et favorise les interrogations.

- augmenter PCTUSED pour favoriser les performances en balayage complet de la table.

Une valeur élevée de PCTUSED permet une occupation plus efficace de l'espace, augmente le coût des mises à jour.

- La somme PCTFREE + PCTUSED doit être <= 100.

FICHIER DATAFILE



Dimensionnement d'une table

En-tête de bloc
La taille de l'en-tête de bloc est donné par la formule:

EN_TETE= KCBH +UB4 + KTBBH + (INITRANS-1)*KTBIT + KDBH

où KCBH, UB4, KTBBH, KTBIT, KDBH sont fournies dans la vue V$TYPE_SIZE (cf Annexe E) et INITRANS est le nombre initial d'entrées allouées à la table

La taille DB_BLOCK_SIZE du bloc est donnée dans la vue V$PARAMETER

Données
L'espace disponible en dehors de l'en-tête (DISPO +DONNEES) est

LIBRE = DB_BLOCK_SIZE - EN_TETE

et l'espace alloué aux données dans un bloc est donc

DONNEES = CEIL(LIBRE * (1-PCTFREE/100)) - KDBT

où KDBT est donné dans la vue V$TYPE_SIZE

Structure d'un enregistrement

EN_TETE_LIGN	LG_COL	COL	LG_COL	COL	

En-tête d'enregistrement : EN_TETE_LIGNE=UB1*3 (donné par V$TYPE_SIZE)
Pour un enregistrement chaîné, l'en-tête-ligne contient l'information de chaînage sous la forme d'une adresse
Chaque colonne de chaque enregistrement est précédée d'une zone LG_COL contenant la longueur de la colonne
LG_COL: champ contenant la longueur de la colonne
1 si taille col < 250
3 sinon

COL : taille de la colonne

Taille des colonness selon le type de données:

VARCHAR2, CHAR, LONG:	1 octet par caractère
DATE:	7 octets
NUMBER(x,y):	1 + (longueur moyenne de x)/2 + 1*
	* si négatif
NULL	1 octet
ROWID:	6 octets

***Taille moyenne d'un enregistrement*:**

$$LG\_ENREG = EN\_TETE\_LIGNE + \Sigma (LG\_COL_i + COL_i)$$

taille totale des colonnes en incluant les longueurs

Nombre d'enregistrements par bloc:

NB_ENREGS_BLOC =FLOOR(DONNEES / LG_ENREG)

Nombre de blocs nécessaires pour mémoriser les données de la table

NB_BLOCS = CEIL (NB_ENREGS /NB_ENREGS_BLOC)

où NB_ENREGS est le nombre estimé d'enregistrements dans la table

III- LES INDEX

Un index est une structure de données supplémentaire qui permet à l'optimiseur d'accélérer les recherches dans une table; il est organisé en B-arbre, c'est à dire de manière à mettre le même temps (même nombre d'accès disque) pour atteindre n'importe quel enregistrement de la table.
Un index est créé implicitement à la création d'une table avec les contraintes PRIMARY KEY et UNIQUE ou explicitement par la commande CREATE INDEX ; il peut être construit sur une ou plusieurs colonnes.

CREATE INDEX nom_index ON nom_table(col1[,col2])
Storage clause

Création d'un index

Un index augmente la performance des requêtes qui sélectionnent un petit nombre d'enregistrements (moins de 25%) des enregistrements de la table.
Le choix des colonnes à indexer peut se faire à partir des critères suivants:

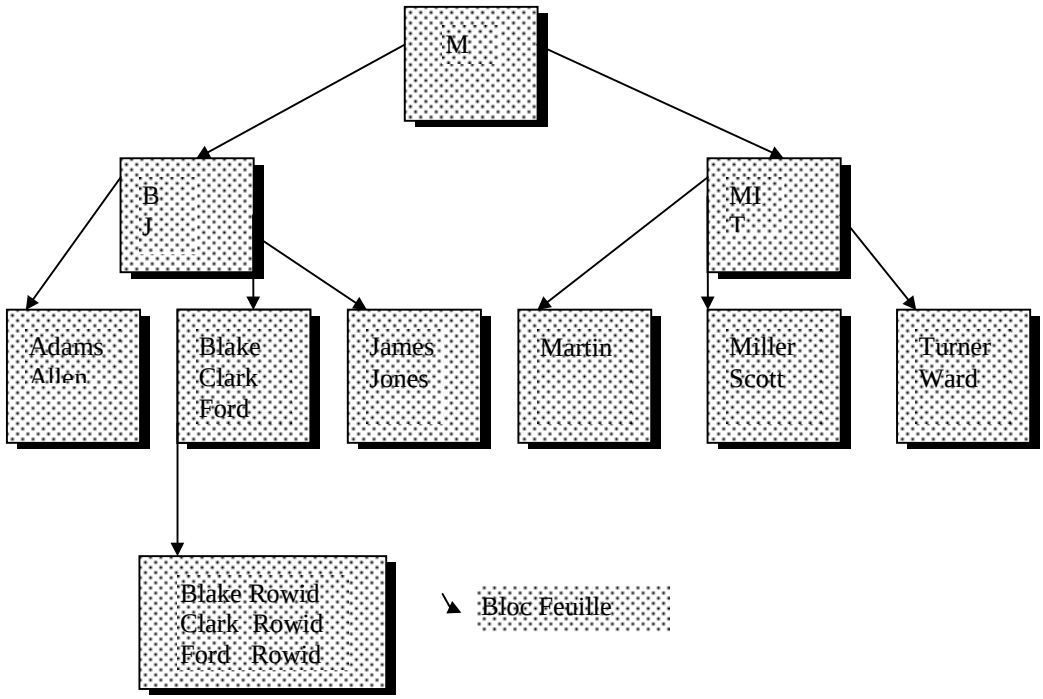
-colonne fréquemment utilisée dans une clause WHERE
-colonne fréquemment utilisée dans une jointure
-colonne ayant une bonne « sélectivité », c'est à dire colonne dont peu d'enregistrements ont la même valeur
-colonne rarement modifiée
-colonne n'aparaissant pas dans une clause WHERE avec des fonctions ou des opérateurs

Un index peut être « composé », c'est à dire construit sur plusieurs colonnes, pour augmenter la « sélectivité » par exemple.

Il ne faut pas oublier qu'un index, s'il peut augmenter la performance des ordres SELECT, diminue les performances des ordres INSERT, UPDATE, DELETE et occupe une place non négligeable dans la base.

Un index est composé de deux parties:

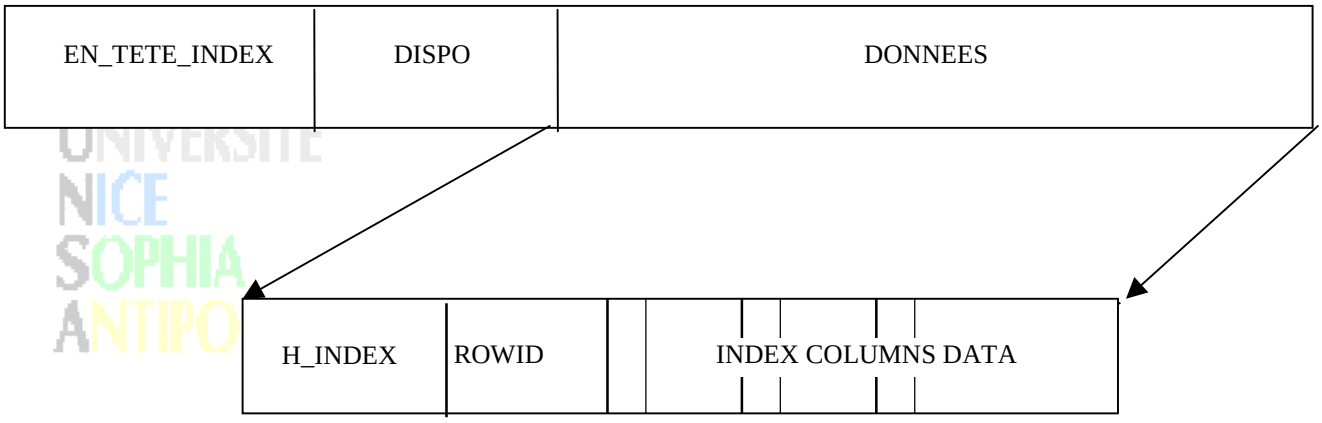
- les blocs supérieurs contiennent des informations qui pointent sur les blocs inférieurs (feuilles)
- les feuilles contiennent une valeur du champ indexé et le ROWID de l'enregistrement correspondant



Dimensionnement d'un index

Un bloc index est, comme un bloc table, composé:

- d'un en-tête comportant une partie fixe et une partie variable
- d'un espace PCTFREE
- d'un espace de stockage des clés



En-tête de bloc
EN_TETE_INDEX = 113 + 24*INITRANS
Pour un index INITRANS = 2 par défaut

Espace disponible pour les données dans un bloc
DONNEES = (DB_BLOCK_SIZE - EN_TETE_INDEX) *(1 - (PCTRFREE/100))

Taille moyenne d'une entrée d'index
INDEX = H_INDEX + ROWID_LENGTH +F +V

H_INDEX: 2
ROWID_LENGTH: 6
F longueur totale (colonne + zone longueur)des colonnes de l'index de longueur inférieure ou égale à 127; pour ces colonnes, la taille de la zone longueur est 1
V: longueur totale(colonne + zone longueur)des colonnes de l'index de longueur supérieure à 127; pour ces colonnes, la taille de la zone longueur est 2

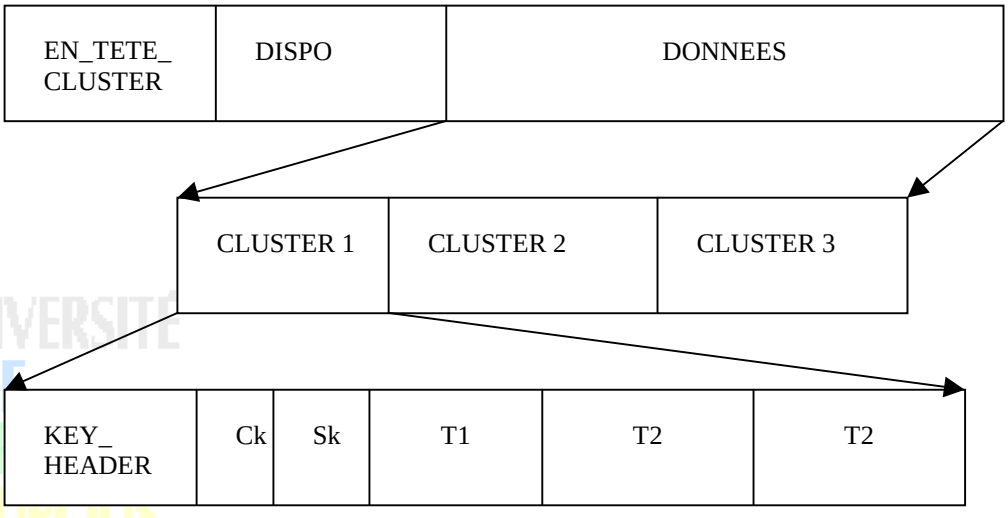
Nombre de blocs pour l'index

NB_BLOCKS_INDEX = 1.05* (NB_NOT_NULL_ROWS) / FLOOR (DONNEES / INDEX)

IV-LES CLUSTERS

Un cluster est une structure physique utilisée pour stocker des tables sur lesquelles doivent être effectuées de nombreuses requêtes avec opération de jointure. Un cluster ne doit pas être installé sur une table fréquemment utilisée isolément.
Dans un cluster, les enregistrements de plusieurs tables ayant même valeur du champ servant à la jointure (clé du cluster) sont mémorisés dans un même bloc physique ; la valeur du paramètre SIZE de la commande CREATE CLUSTER donne le nombre maximum de clusters qui peuvent être mémorisés dans un bloc.

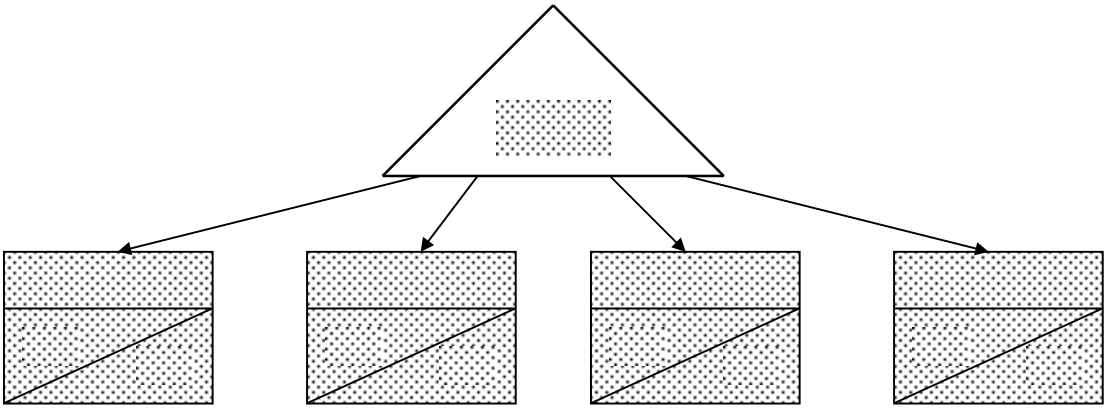
Structure d'un cluster



Ce cluster contient un enregistrement de la table T1 et deux enregistrements de la table T2.

Il existe deux types de clusters:
- les clusters indexés
- les clusters hash

Cluster indexé
Un index est créé sur la clé du cluster



Cluster hash

Une fonction de hachage est appliquée à la clé du cluster et renvoie une valeur qui est utilisée pour localiser l'enregistrement; Oracle fournit une fonction de hachage interne qui produit un minimum de collisions dans la plupart des cas. Il est possible d'utiliser une fonction de hachage particulière en la spécifiant dans la clause CREATE CLUSTER.

Création d'un cluster

Un cluster sera créé
-entre des tables souvent accédées à l'aide d'un ordre SELECT comportant une opération de jointure entre ces tables; si l'opération est une équi-jointure, il sera intéressant de créer un hash cluster.
-si les enregistrements ayant la même valeur de cluster key peuvent être mémorisés dans un seul bloc physique

On choisira un cluster indexé si la taille de segment et le nombre de clés est difficile à prévoir, un hash cluster si ces informations sont faciles à prévoir.
On choisira un cluster indexé pour les jointures, un hash cluster pour les requêtes dont la sélection porte sur la clé de hash.

On ne créera pas de cluster
-si les opérations de jointure sont rares, si les traitements séquentiels de l'une des tables sont fréquents ou si la valeur de la cluster key peut être modifiée
-si les tables doivent s'accroître de manière importante

Dimensionnement d'un cluster

Pour un cluster il faut calculer
-la taille de l'en-tête EN_TETE_CLUSTER dans un bloc
-la taille de l'espace disponible dans un bloc pour le stockage des lignes DONNEES
-la taille moyenne d'un enregistrement de cluster CLUSTER

En-tête
EN_TETE_CLUSTER = KCBH + UB4 + KTBBH + KTBIT*(INITTRANS-1) + KDBH

Les tailles de toutes les variables sont données dans V$TYPE_SIZE

Espace disponible dans un bloc, hors en-tête
DISPO= DB_BLOCKSIZE - EN_TETE_CLUSTER
Espace disponible pour les données

DONNEES=DISPO*(1-PCTFREE/100) - 4*(NB_TABLES + 1)*ROWS_IN_BLOCK

NB_TABLES: nombre de tables dans le cluster
ROWS_IN_BLOCK: nombres d'enregistrements dans un bloc du cluster

Espace minimum nécessaire pour stocker les enregistrements de la table Tn du cluster

Sn = ROWHEADER + Fn + Vn
avec
ROWHEADER = 4
Fn: longueur totale (colonne + zone longueur) des colonnes de la table Tn de longueur inférieure ou égale à 250; pour ces colonnes, la taille de la zone longueur est 1
Vn: longueur totale (colonne + zone longueur) des colonnes de la table Tn de longueur supérieure à 250; pour ces colonnes, la taille de la zone longueur est 3

Taille moyenne d'un enregistrement cluster

CLUSTER = ((R1*S1) + (R2*S2) + ...+(Rn*Sn)) + KEY_HEADER + Ck + Sk +2Rt

avec
Rn: nombre moyen d'enregistrements de la table n associés à une clé du cluster
Sn: taille moyenne d'un enregistrement de la table n associé à une clé du cluster
KEY_HEADER=19
Ck: longueur de colonne pour la clé du cluster
Sk: taille moyenne de la valeur de la clé du cluster
Rt: Nombre total d'enregistrements associés à une clé du cluster (R1 + R2 +...+Rn)

Nombre de clés de cluster par bloc physique

NB_KEYS_BLOCK = FLOOR((DONNEES + 2Rt)/ (CLUSTER+ 2Rt))

Nombre de blocks pour le cluster

NB_BLOCKS = CEIL (NB_KEYS / NB_KEYS_BLOCK)

NB_KEYS: nombre de clés de cluster à estimer à la définition des tables

V-LES ROLLBACK SEGMENTS

A la création de la base, le rollback segment SYSTEM est créé dans la tablespace SYSTEM. Si la base doît avoir d'autres tablespaces, elle doît posséder au moins deux autres rollback segments dans le tablespace SYSTEM. Le rollback segment SYSTEM est créé avec les paramètres par défaut asssociés à la tablespace; il ne peut pas être détruit.
Une instance utilise toujours le rollback segment SYSTEM en complément d'autres rollback segments, si nécessaire. Cependant, s'il existe plusieurs rollback segments, Oracle essaie d'utiliser le rollback segment SYSTEM uniquement pour des transactions spéciales.
La taille totale des rollback segments doît être calculée à partir de la taille des transactions les plus fréquentes. En général, des transactions courtes sont plus performantes avec plusieurs petits rollback segments alors que les transactions plus longues ,batch par exemple, seront plus efficaces avec de plus grands rollback segments.
Si toutes les transactions sont courtes, les rollback segments seront assez petits pour être mémorisés en mémoire centrale; s'ils sont assez petits, ils pourront être mémorisés dans la SGA selon l'algorithme LRU et le nombre d'opérations d'entréee/sortie sera nettement diminué.
Le principal inconvénient des petits rollback segments est d'accroître la probabilité d'errreur « snapshot too old » (bases de données distribuées).
Quand il y a des transactions courtes et des transactions longues, les performances peuvent être optimisées en affectant certains rollback segments à certaines transactions à l'aide de la commande SET TRANSACTION USE ROLLBACK SEGMENT rollback_segment.
Dans le cas général, la taille de chaque rollback segment doît être environ 10% la taille de la plus grosse table étant donné que la plupart de ses instructions SQL affectent 10% ou moins d'une table; la taille optimale du rollback segment peut être précisée par le paramètre OPTIMAL de la STORAGE clause lors de la création du rollback segment.
L'espace alloué à un rollback segment doît être réparti entre des extents de même taille; la performance optimale au niveau des entrées/sorties est observée si chaque rollback segment est composé de 10 à 20 extents.
Le nombre total de rollback segments est lié au nombre de transactions simultanées envisagées:

Nombre de transactions simultanées: n	Nombre de rollback segments
n <= 16	4
16 <= n >= 32	8
32 <= n	n/ 4 et < 50

Les informations sur les rollback segments se trouvent dans les vues DBA_ROLLBACK_SEGS, V$ROLLNAME, V$ROLLSTAT.

VI-QUESTIONS

I/ La base de données "Gestion des Commandes" est constituée des tables suivantes:

CUSTOMER

custid	not null	number(6)
name	not null	varchar2(45)
address		varchar2(40)
city		varchar2(30)
state		varchar2(2)
zip		varchar2(9)
area		number(3)
phone		varchar2(9)
repid	not null	number(4)
creditlimit		number(9,2)
comments		long

ORD

ordid	not null	number(4)
orderdate		date
commplan		varchar2(1)
custid		number(6)
shipdate		date
total		number(8,2)

ITEM

ordid	not null	number(4)
itemid	not null	number(4)
prodid		number(6)
actualprice		number(8,2)
qty		number(8)
itemtot		number(8,2)

PRODUCT

prodid	not null	number(6)
descrip		varchar2(30)

PRICE

prodid	not null	number(6)
stdprice		number(8,2)
minprice		number(8,2)
startdate		date
enddate		date

La base de données devra accueillir les volumes suivants:

CUSTOMER	:	2000 enregistrements
ORD	:	50000 enregistrements
ITEM	:	200000 enregistrements (5 item en moyenne par ord)
PRODUCT	:	1000 enregistrements
PRICE	:	4000 enregistrements (4 price en moyenne par product)

Des clés primaires sont définies pour les tables CUSTOMER,ORD et PRODUCT.
Des index sont créés sur les champs CUSTOMER.name et PRODUCT.descrip.
Des clusters (indexés) sont créés entre les tables ORD et ITEM d'une part, PRODUCT et PRICE d'autre part.
Evaluer l'espace nécessaire pour mémoriser ces informations; le volume du tablespace SYSTEM est évalué à 10M.

II/ Retrouver l'espace disponible pour tous les tablespaces de la base.

III/ Faire un état de la base avec l'occupation de l'espace par tablespace et par propriétaire d'objet, fournissant les informations suivantes: type,nom, bytes, blocks et extents.

GESTION DES UTILISATEURS ET DES PRIVILEGES

I-LES UTILISATEURS

Le contrôle d'accès à Oracle se fait par l'association du nom et du mot de passe de l'utilisateur. L'utilisation de la base de données sera autorisée si

* l'utilisateur peut se connecter (privilège CREATE SESSION)
* l'utilisateur a un espace de travail suffisant sur disque (développeur)

L'administration de la sécurité sur la base de données est réalisée grâce à la création des utilisateurs et à la gestion de leurs droits d'accès. Chaque base de données possède sa propre liste d'utilisateurs. Le contrôle des droits d'accès à la base de données se fait par rapport à un ensemble de caractéristiques prédéfinies:

* Informations d'authentification (login et password)
* Tablespaces accessibles
* Quotas sur les tablespaces
* Tablespace par défaut
* Tablespace temporaire
* Privilèges et rôles
* Ressources SYSTEME (PROFILE)

La commande **CREATE USER** permet à l'administrateur de

* donner un nom et un mot de passe à l'utilisateur
* lui assigner un tablespace par défaut
* lui assigner un tablespace temporaire
* identifier la liste des tablespaces pour lesquels il aura des droits d'accès
* délimiter ses ressources disque sur chaque tablespace
* délimiter ses ressources système

CREATE USER user IDENTIFIED BY password / EXTERNALLY

[DEFAULT TABLESPACE tablespace_name]
[TEMPORARY TABLESPACE tablespace_name]
[QUOTA integer K/M / UNLIMITED ON tablespace_name]

[PROFILE profile_name]

Un utilisateur créé avec la clause identified EXTERNALLY se connectera à la base de données à l'aide de son compte système.
La commande **ALTER USER** permet à l'administrateur de

* changer le mot de passe de l'utilisateur
* de modifier ses droits et ses quotas sur les tablespaces
* de changer le rôle de l'utilisateur (un utilisateur peut appartenir à plusieurs rôles)

La commande ALTER USER permet à l'utilisateur de modifier uniquement son mot de passe (identified by password).

ALTER USER USER IDENTIFIED BY password/EXTERNALLY

[DEFAULT TABLESPACE tablespace_name]
[TEMPORARY TABLESPACE tablespace_name]
[QUOTA integer K/M / UNLIMITED ON tablespace_name]

[PROFILE profile_name]
[DEFAULT ROLE role_name/ ALL EXCEPT role_name/NONE]

La commande **DROP USER** permet à l'administrateur de supprimer un utilisateur et,éventuellement tous les objets dont il est propriétaire (option CASCADE).

DROP USER user_name [CASCADE]

Toutes les caractéristiques des utilisateurs sont répertoriées dans le dictionnaire de données
* USER_USERS: informations sur l'utilisateur courant
* ALL_USERS: informations sur tous les utilisateurs de la base
* DBA_USERS: toutes les informations sur tous les utilisateurs de la base
* USER_TS_QUOTAS: informations sur les quotas de l'utilisateur courant
* DBA_TS_QUOTAS: informations sur les quotas de tous les utilisateurs

II- LES PRIVILEGES

La gestion des privilèges permet

*de donner aux utilisateurs le droit d'exécuter certaines commandes
*d'interdire ou d'autoriser la consultation, la mise à jour et la suppression des données
*d'interdire ou d'autoriser l'accès à des fonctions système
*d'interdire ou d'autoriser l'accès à des commandes de changement de structure de la base
*de créer et de supprimer des privilèges pour un utilisateur particulier, pour un rôle particulier ou pour tous les utilisateurs(PUBLIC)

Il y a deux types de privilèges:

* le **privilège SYSTEM** qui donne le droit d'exécuter des actions sur un certain type d'objet

*le **privilège OBJET** qui donne le droit d'accès à une table, une vue, une séquence, une procédure, une fonction ou un package.

Privilèges SYSTEM

Les privilèges SYSTEM définissent les types d'opérations disponibles pour l'utilisateur.
Ces privilèges sont attribués par la commande GRANT.

GRANT system_privilege TO user/role/public [WITH ADMIN OPTION]

L'option WITH ADMIN OPTION donne le droit de redistribuer les privilèges reçus (cette option est interdite si le privilège est attribué à un rôle). Ces privilèges sont supprimés par la commande REVOKE.

REVOKE system_privilege FROM user/role/public

La liste des privilèges SYSTEMest fournie en annexe B.

Privilèges OBJET

Ils donnent le droit d'accès à une table, une vue,une séquence, une procédure, une fonction ou un package. Ils sont différents selon les types d'objet.

Ils sont attribués par la commande GRANT et supprimés par la commande REVOKE.

GRANT object_privilege ON object_name TO user/role/public
[WITH GRANT OPTION]

Recevoir un privilège avec la clause WITH GRANT OPTION permet de redistribuer le privilège reçu à d'autres utilisateurs, également avec la clause WITH GRANT OPTION.
Retirer les privilèges à un utilisateur se répercute en cascade sur les autres utilisateurs.
On peut attribuer des privilèges et un role avec la clause WITH GRANT OPTION.

Privilège	Droit	Table	Vue	Séquence	Procédure Fonction Package
ALTER	Modifier objets	OUI		OUI	
DELETE	Supprimer lignes		OUI		
EXECUTE	Exécuter				OUI
INDEX	Créer index	OUI			
INSERT	Insérer lignes	OUI	OUI		
REFERENCES	Référencer une colonne dans table	OUI			
SELECT	Lire lignes	OUI	OUI	OUI	
UPDATE	Modifier lignes	OUI	OUI		

Ces privilèges sont supprimés à l'aide de la commande REVOKE.

III- LES RÔLES

Un rôle est un ensemble de privilèges donnés à des utilisateurs ou à d'autres rôles permettant de gérer plus facilement les droits d'accès aux données; ils permettent de définir des groupes d'utilisateurs ayant les mêmes privilèges.
Il existe des rôles prédéfinis à la création de la base:

* **EXP_FULL_DATABASE**: possibilité d'utiliser EXPORT
* **IMP_FULL_DATABASE**: possibilité d'utiliser IMPORT
* **RESOURCE**:
CREATE CLUSTER, PROCEDURE, SEQUENCE, TABLE, TRIGGER
* **CONNECT**:
* **DBA**: administrateur de base de données (tous les privilèges with admin option + **EXP_FULL_DATABASE et IMP_FULL_DATABASE** rôles)

Ces trois derniers roles sont créés pour la compatibilité avec les versions précédentes d'Oracle.

Un role est créé par la commande CREATE ROLE et supprimé par DROP ROLE.

```
CREATE ROLE role_name IDENTIFIED BY password
```

Les privilèges SYSTEM attribués à un role sont définis par la commande GRANT; un role est donné à un utilisateur par la commande GRANT; un utilisateur peut appartenir à plusieurs rôles, le nombre de rôles étant limité par le paramètre d'initialisation (fichier init.ora) MAX_ENABLED_ROLES.
Dans ce cas, l'utilisateur doit posséder un role par défaut défini par la commande ALTER USER;il peut changer de rôle, s'il en a reçu le droit, avec la commande

```
SET  ROLE role_name IDENTIFIED BY password
```

IV- LES PROFILES

Un profile est défini par un ensemble de paramètres qui permettent de limiter les consommations de ressources d'un utilisateur: temps CPU, opérations d'entrées-sorties, temps d'innocupation, temps d'occupation, espace mémoire, sessions courantes.
Les profiles permettent de restreindre les grosses consommations de ressources des utilisateurs, d'être sûr que les utilisateurs sont déconnectés lorsqu'ils ont quitté leur poste de travail, de regrouper les utilisateurs ayant les mêmes fonctions et les mêmes charges de travail.
Il existe un profile DEFAULT, assigné par défaut à tous les utilisateurs, initialement sans limites.
Un profile est défini par la commande CREATE PROFILE et peut être affecté à un utilisateur par les commandes CREATE USER ou ALTER USER.

```
CREATE PROFILE profile_name LIMIT
SESSIONS_PER_USER              integer/unlimited/default
CPU_PER_SESSION
CPU_PER_CALL
```

CONNECT_TIME
IDLE_TIME
LOGICAL_READS_PER_SESSION
LOGICAL_READS_PER_CALL
COMPOSITE_LIMIT
PRIVATE_SGA integer K/M / UNLIMITED /DEFAULT

Ces limites de ressource sont prises en compte de façon permanente si le paramètre d'environnement RESOURCE_LIMIT = TRUE ; elles peuvent être temporairement prises en compte à l'aide de la commande ALTER SYSTEM SET RESSOURCE_LIMIT=TRUE.

Le dictionnaire de données contient toutes les informations sur chaque utilisateur et chaque profil; ces informations sont conservées dans les vues suivantes:

- ALL_USERS, USER_USERS, DBA_USERS
- USER_TS_QUOTAS, DBA_TS_QUOTAS
- USER_RESOURCE_LIMITS, DBA_PROFILES, RESOURCE_COST
- V$SESSION,V$SESSTAT, V$STATNAME

V- LE SCHEMAS

A chaque utilisateur est associé un schéma: ensemble des objets accessibles à cet utilisateur: tables, index, vues, séquences, synonymes, clusters, database links, procédures et packages.
A chaque schéma correspond un espace logique de stockage dans un tablespace de la base de données; il n'y a pas de relation entre un schéma et un tablespace: un tablespace peut contenir plusieurs schémas et un schéma peut être situé sur plusieurs tablespaces.
Un schéma est défini par la requête CREATE SCHEMA qui garantit la création de plusieurs tables vues et droits en une seule opération.

CREATE SCHEMA AUTHORIZATION schema_name
CREATE TABLE commande
CREATE VIEW commande
GRANT commande

VI-QUESTIONS

I/ Créer les tablespaces APPLI,TEMP et RBS avec les mêmes caractéristiques que le tablespace <login>

II/ Créer les utilisateurs suivants:

Utilisateur	Tablespace par défaut	Tablespace temporaire
ADMIN	APPLI	TEMP
DEVEL	<login>	TEMP
UTIL1	<login>	TEMP
UTIL2	<login>	TEMP

Vérifier que tous les utilisateurs ont été créés dans les bons tablespaces

III/ Modifier les quotas des utilisateurs de la manière suivante:

Utilisateur	Quota à assigner	Tablespace
ADMIN	Aucun	SYSTEM
	Illimité	APPLI
DEVEL	Aucun	SYSTEM
	100K	<login>
UTIL1	Aucun	Tous
UTIL2	Aucun	Tous

Vérifiez ces modifications

IV/ Connectez-vous sous les comptes ainsi créés; que se passe-t-il?

V/ Créer les utilisateurs suivants avec les privilèges indiqués

Utilisateur/ mot de passe	Tablespace par défaut	Tablespace temporaire	Privilèges
ADMIN_APPLI/ ADMIN_APPLI	APPLI	TEMP	CREATE SESSION CREATE TABLE CREATE VIEW CREATE SYNONYM CREATE ROLE
ADMIN_SYS/ ADMIN_SYS	APPLI	TEMP	CREATE SESSION CREATE USER, ALTER USER CREATE ANY TABLE, ALTER ANY TABLE, DROP ANY TABLE SELECT ANY TABLE CREATE ANY INDEX

Tous ces privilèges seront attribués avec la clause WITH ADMIN OPTION

En vous plaçant sous le compte SYSTEM
a/ Donner aux utilisateurs ADMIN,DEVEL,UTIL la possibilité de se connecter à la base
b/ Donner à ADMIN la possibilité de créer des tables, des vues et des synonymes

VII/ Charger votre répertoire les fichiers /u/profs/jcg/dept.sql et /u/profs/jcg/emp.sql
Sans changer les privilèges SYSTEM des utilisateurs,
A partir du fichier dept.sql, créer la table dept de façon à ce qu'elle appartienne à ADMIN
A partir du fichier emp.sql, créer la table emp de façon à ce qu'elle appartienne à DEVEL

VIII/ Dans quels tablespaces ont été créées ces deux tables?

IX/ Donner à UTIL1 la possibilité de visualiser en une seule requête le nom des employés (emp.ename) et leur lieu de travail (dept.loc) (sans pouvoir visualiser les autres champs des tables emp et dept).

X/ Donner à UTIL2 la possibilité de mettre à jour, supprimer et d'insérer des enregistrements dans la table emp.

XI/ Donner à UTIL2 la possibilité de mettre à jour uniquement la colonne LOC de la table dept.

XII/ Ecrire les requêtes permettant de vérifier ces possibilités

XIII/ Vérifier dans le dictionnaire de données la liste des privilèges de UTIL1 et UTIL2.

XIV/ Quelles conclusions pouvez-vous tirer sur les "rôles" des différents utilisateurs

XV/ Attribuer à l'utilisateur <login> les caractéristiques suivantes:
Tablespace par défaut: <login>
Tablespace temporaire: temp
quota sur <login>: 200K

XVI/ Connectez vous sur un compte adéquat et créez les rôles R1 et R2.

XVII/ Sans lui donner explicitement les privilèges, faites en sorte que <login> puisse créer des tables, des vues et/ou des synonymes.

XVIII/ Vérifier que <login> peut effectivement créer des tables et des synonymes

XIX/ Sans vous déconnecter du compte <login>, faites en sorte qu'il ne puisse plus créer de synonymes mais toujours créer des tables

XX/ Sans vous déconnecter du compte <login>, faites en sorte qu'il ne puisse plus créer de tables mais toujours créer des synonymes

XXI/ Sans vous déconnecter du compte <login>, rétablissez la situation de départ

XXII/ Donner à <login> le rôle R1 par défaut

XXIII/ Rechercher dans le dictionnaire de données, la liste des rôles existants ainsi que les privilèges attribués à chaque rôle

XXIV/ Faire en sorte que UTIL1 et UTIL2 ne puissent ouvrir que 2 sessions simultanément et ne puissent pas rester connectés plus de 2 minutes.

MECANISMES TRANSACTIONNELS

La gestion des accès concurrents a pour objectif d'assurer la sérialisation de transactions multiples voulant accéder simultanément aux mêmes données; l'exécution en parallèle de ces transactions fournit le même résultat que leur exécution en série mais avec de meilleures performances.
La gestion des accès concurrents est basée sur les concepts d'intégrité des données, de concurrence des transactions et de consistance des données; elle utilise essentiellement la technique de verrouillage des données.

-*intégrité des données*: l'intégrité des données est assurée si les contraintes d'intégrité définies au moment de la création des tables sont respectées à l'issue de l'exécution des transactions.

-*concurrence des transactions*: la gestion de la concurrence des transactions a pour but d'assurer l'intégrité des données lorsque plusieurs transactions accèdent simultanément aux mêmes données.

-*consistance des données*: la consistance des données est assurée lorsque la transaction utilise, tout le temps de son exécution, la valeur de ces données au début de la transaction, même si d'autres transactions essaient de modifier tout ou partie de ces données.

I- TRANSACTIONS ET ACCES CONCURRENTS

Une base de données est dans un état cohérent si toutes les valeurs contenues dans la base vérifient toutes les contraintes d'intégrité définies sur la base.
Une transaction est un ensemble d'ordres de mise à jour, INSERT, UPDATE ou DELETE, qui font passer la base d'un état initial cohérent à un état final cohérent; elle se termine par un ordre (explicite ou implicite) de validation (COMMIT) ou d'annulation (ROLLBACK).

Transactions concurrentes: Il y a concurrence d'accès à une donnée quand celle-ci doît être modifiée simultanément par au moins deux transactions.

a/ Lecture cohérente

La lecture cohérente assure qu'une transaction non encore validée n'affecte en rien la visualisation des données pour l'ensemble des utilisateurs:

- l'utilisateur initiateur de la transaction visualisera la donnée **après** sa modification
- les autres utilisateurs visualiseront la donnée **avant** sa modification

b/ Lecture incohérente ou impropre

La même transaction utilise des données validées et des données non validées

exemple 1:

T1	**T2**
Début transaction	
	Début transaction
UPDATE comptes SET solde = 25000 WHERE num_compte = 7;	

```
                                    SELECT solde
                                    FROM comptes
                                    WHERE num_compte = 7;
ROLLBACK;
```

exemple 2:

T1	**T2**
Début transaction	
	Début transaction
UPDATE comptes SET solde = solde - 200 WHERE num_compte = 7;	
	SELECT sum(solde) FROM comptes WHERE num_compte in (7,16);
UPDATE compte SET solde =solde + 200 WHERE num_compte = 16; COMMIT;	

Dans les deux cas T2 lit une valeur impropre.

c/ Lectures non reproductibles

Dans une même transaction, deux requêtes identiques ne donnent pas le même résultat, celui_ci ayant été modifié entre temps par une autre transaction

T1	**T2**
Début transaction	
	Début transaction
SELECT Points FROM resultat WHERE num_cours = 5 and num_etudiant = 7;	
	UPDATE resultat SET Points =Points+2 WHERE num_cours = 5 and num_etudiant = 7;
SELECT Points FROM resultat WHERE num_cours = 5 and num_etudiant = 7; COMMIT;	

La lecture de points n'est pas reproductible.

d/ Perte de mise à jour
Une transaction peut modifier des données non validées

T1	**T2**
Début transaction	
	Début transaction
SELECT nb_places_dispo INTO dispo FROM vol WHERE num_vol = 10 AND date_vol = ‘1-MAR-96’;	
	SELECT nb_places_dispo INTO dispo FROM vol WHERE num_vol = 10 AND date_vol = ‘1-MAR-96’;

IF dispo >= 3 THEN
« enregistrer la réservation »;
UPDATE vol
SET nb_places_dispo = nb_places_dispo - 3
WHERE num_vol =10
AND date_vol = ‘1-MAR-96’;
END IF;	

	IF dispo >= 3 THEN
	« enregistrer la réservation »;
	UPDATE vol
	SET nb_places_dispo = nb_places_dispo - 3
	WHERE num_vol =10
	AND date_vol = ‘1-MAR-96’;
	END IF;

COMMIT;	
	COMMIT;

La valeur de nb_places_dispo est erronée.

Par la commande
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE/READ COMITTED,
Oracle permet deux niveaux d'isolation des transactions les unes par rapport aux autres.

READ COMITTED: mode de fonctionnement par défaut d'Oracle; il évite les lectures incohérentes ou les pertes de mise à jour mais pas les lectures non répétitives.
SERIALIZABLE: cette option empêche une transaction de modifier une donnée mise à jour par une autre transaction non validée; on évite ainsi les anomalies de lecture non répétitive. Ce mode est pénalisant car il limite le fonctionnement en parallèle des transactions.

II- LES VERRROUS ORACLE

Pour gérer les concurrences d'accès aux données, Oracle utilise des mécanismes de verrouillage et la notion de transaction.
Il n'existe pas de concurrence d'accès aux données en LECTURE: un utilisateur qui lit une donnée n'interferera pas avec une transaction; une transaction n'interferera pas avec une opération de lecture de la même donnée.
Il existe une concurrence d'accès en MISE A JOUR: la première transaction qui accède à la donnée est prioritaire et positionne un **verrou** sur les ressources accédées(*table* ou *ligne*).
Deux types de verrous:

Les verrous exclusifs: la première transaction qui verrouillera la ressource de façon exclusive sera la seule à pouvoir la modifier.
Les verrous partagés: ils assurent le partage des ressources en fonction du type d'opération effectué sur ces ressources.

Les verrous sont tous maintenus sur les ressources jusqu'à la fin de la transaction; ils sont libérés quand la transaction est validée ou annulée.

Oracle détecte le phénomène d'attente mutuelle (DEADLOCK); il résoud automatiquement les DEADLOCK en annulant toujours l'ordre qui les a provoqués (STATEMENT LEVEL ROLLBACK).
Pour gérer la lecture cohérente, Oracle utilise automatiquement les « images avant » de toutes les données modifiées. Un utilisateur peut gérer le mécanisme de lecture cohérente au niveau de la transaction en utilisant explicitement l'ordre:

SET TRANSACTION READ ONLY

Dans ce cas, seules les opérations de lecture sont autorisées, les autres utilisateurs peuvent modifier les ressources, les données validées par d'autres transactions ne seront pas visibles. L'ordre SET TRANSACTION READ ONLY est obligatoirement la première instruction de la transaction; les seules autres instructions autorisées dans la transaction sont SELECT (sans la clause FOR UPDATE), COMMIT, ROLLBACK ou une non-DML instruction (SET ROLE, ALTER SYSTEM, LOCK TABLE).

Les différentes catégories de verrous

Les verrous DDL(data dictionary locks)
Ils sont posés sur les structures des objets(tables) et sont utilisés pour éviter les modifications de structure pendant les requêtes:

Le verrou exclusif: est posé sur l'objet si aucun autre type de verrou (DDL et DML) n'est pas déja acquis.
Le verrou partagé: est posé sur l'objet si l'un des ordres suivants est utilisé:
AUDIT, NOAUDIT, GRANT, REVOKE, COMMENT,
CREATE(REPLACE)(TABLE/VIEW/SYNONYM/FUNCTION,PROCEDURE/PACKAGE)
.

Le verrou de parsing: posé sur chaque objet référencé dans un ordre SQL et utilisé pour déterminer si l'analyse stockée dans la zone de partage des ordres SQL ne serait pas obsolète du fait d'un changement de structure de l'objet accédé.

Les verrous DML

Tout ordre modifiant une donnée positionne:

-un verrou de type exclusif sur la ligne contenant la donnée
-un verrou de type exclusif ou partagé sur la table

Les cinq verrous DML, du plus libéral au plus restrictif, sont:

- RS: Row Share
- RX: Row eXclusive
- S: Share
- SRX: Share Row eXclusive
- X: eXclusive

ROW SHARE (RS)

Un verrou de type **ROW SHARE (RS)**interdit l'accès, aux autres utilisateurs, des enregistrements sélectionnés pour une mise à jour ultérieure. Il

-verrouille les enregistrements concernés et attend la mise à jour
-autorise la visualisation de tous les enregistrements de la base, y compris ceux de la table concernée par la transaction en cours
-autorise l'insertion, la mise à jour et la suppression de tous les enregistrements non verrouillés, y compris dans la table concernée par la transaction en cours.
-est compatible avec les verrous RS,RX,S et SRX
-permet de se prémunir contre la pose d'un verrou X
-se pose de manière explicite par
SELECT....FROM table.....FOR UPDATE OF colonne
LOCK TABLE table IN ROW SHARE MODE [NOWAIT]

L'option NOWAIT rend le contrôle à la transaction si la table fait déja l'objet d'une instruction LOCK de la part d'une autre transaction; si cette clause est omise, la transaction est mise en attente jusqu'à la libération des verrous installés sur la table.
Un verrou de type ROW SHARE permet des accès concurrents à la table; il interdit à d'autres transactions de placer un verrou EXCLUSIVE sur la table.

ROW EXCLUSIVE (RX)

Un verrou de type **ROW EXCLUSIVE (RX)** est mis en place automatiquement par Oracle avant l'exécution d'un ordre INSERT,UPDATE,DELETE; il permet à l'utilisaeur de modifier certains enregistrements tout en laissant d'autres utilisateurs modifier d'autres enregistrements de la même table. Il

-verrouille les enregistrements concernés et effectue la mise à jour dans la table
-autorise la visualisation de tous les enregistrements de la base, y compris ceux de la table concernée par la transaction en cours
-autorise l'insertion, la modification et la suppression de tous les enregistrements non verrouillés, y compris dans la table concernée par la transaction en cours
-est compatible avec les verrous RS et RX

-permet de se prémunir contre la pose de verrous S, SRX et X
-se pose de manière implicite dans les ordres INSERT, UPDATE, DELETE
-se pose de manière explicite pour utiliser le verrou X
LOCK TABLE table IN ROW EXCLUSIVE MODE [NOWAIT]

Un verrou de type ROW EXCLUSIVE a le même effet que ROW SHARE et interdit en plus le verrouillage en mode SHARE par d'autres transactions.

SHARE (S)

Un verrou de type **SHARE (S)** est employé lorsque la transaction utilise la table en interrogation uniquement et exige que cette table ne soit pas modifiée par d'autres transactions. Il

-empêche toutes les insertions, modifications et suppressions d'enregistrements dans la table concernée si un autre verrou de type SHARE est déja positionné sur la table
-autorise la visualisation dans le but de verrouiller certains enregistrements dans la table concernée par la transaction en cours
-est compatible avec un autre verrou RS ou S
-permet de se prémunir contre la pose de verrous RX, SRX et X
-se pose de manière explicite par
LOCK TABLE table IN SHARE MODE [NOWAIT]

exemple:

```
LOCK TABLE dept IN SHARE MODE

UPDATE emp
SET sal = sal *1.1
WHERE deptno in (SELECT deptno FROM dept WHERE loc = 'DALLAS')

UPDATE budget
SET total = total * 1.1
WHERE deptno in (SELECT deptno FROM dept WHERE loc = 'DALLAS')

COMMIT
```

Plusieurs transactions peuvent installer un verrou de type SHARE sur la même table en même temps.

SHARE ROW EXCLUSIVE (SRX)

Un verrou de type **SHARE ROW EXCLUSIVE (SRX)** est employé pour visualiser une table entière; il permet aux autres transactions de visualiser les lignes mais leur interdit de verrouiller la table en mode SHARE ou de faire des mises à jour. Il

-empêche toutes insertions, modifications et suppressions d'enregistrements dans la table concernée par la transaction en cours

-autorise la visualisation dans le but de verrouiller certains enregistrements dans la table concernée par la transaction en cours
-est compatible avec un autre verrou RS
-permet de se prémunir contre la pose de verrous RX, S, SRX, X
-se pose de manière explicite par
LOCK TABLE table IN SHARE ROW EXCLUSIVE MODE [NOWAIT]

EXCLUSIVE (X)

Un verrou de type EXCLUSIVE est employé lorsque la transaction exige un accès immédiat à la table pour réaliser une opération de mise à jour; il autorise les interrogations mais interdit toute autre action.

-empêche toutes insertions, modifications et suppressions d'enregistrements dans la table concernée par la transaction en cours
-autorise la visualisation de tous les enregistrements de la base, y compris dans la table concernée par la transaction en cours
-n'est compatible avec aucun autre verrou
-se pose de manière explicite par

LOCK TABLE table IN EXCLUSIVE MODE [NOWAIT]

Ce mode est très contraignant pour les autres utilisateurs; le déblocage de la table doit être très rapide; un verrouillage en mode EXCLUSIVE doit être suivi rapidement d'un COMMIT ou d'un ROLLBACK.

III-QUESTIONS

I/Montrer que les trois cas d'incohérence, lecture impropre, lecture non reproductible et perte de mise à jour ne peuvent pas se produire avec les verrous posés automatiquement par Oracle.
Pour cela vous simulerez deux transactions essayant de mettre simultanément à jour les mêmes données en ouvrant deux transactions sqlplus

-l'une par l'utilisateur ADMIN, propriétaire de la table dept
-l'autre par l'utilisateur DEVEL, propriétaire de la table emp

Les deux utilisateurs auront les privilèges d'interrogation et de mise à jour sur les deux tables.

II/Donner aux deux utilisateurs DEVEL et ADMIN le privilège LOCK ANY TABLE.
En utilisant uniquement la table admin.dept, montrer les diverses possibilités offertes par les différents types de verrous.
ADMIN verrouillant la table dept successivement dans les modes ROW SHARE, ROW EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE et EXCLUSIVE , examinez, dans chaque cas, les diverses possibilités pour DEVEL de poser des verrous sur cette table ou d'effectuer des mises à jour sur un enregistrement que ADMIN aura verrouillé ou non auparavant ex :les transactions essaieront d'effectuer la mise à jour suivante:
update admin.dept set loc = 'NEW YORK' where deptno=20.

OPTIMISATION DES TRAITEMENTS

I-L'OPTIMISATION DES TRAITEMENTS

L'obtention de bonnes performances d'une application dépend d'une bonne définition des structures(logiques et physiques) de mémorisation de l'information et de l'efficacité du traitement des ordres SQL.

L'allocation appropriée de ressources mémoire peut avoir un large impact sur les performances; ces ressources sont les zones de traitement SQL et PL/SQL, le dictionnaire de données et le buffer; une bonne définition de ces ressources entrainera une amélioration des performances par la réduction de l'analyse des ordres SQL ou PL/SQL et de la pagination.

La répartition des données sur les disques permettra de diminuer le nombre d'entrées/sorties. Il est conseillé de mémoriser sur des disques différents:

- les fichiers data files et les fichiers Redo Log
- les tables de données
- les tables et les index

Une amélioration des performances sera également obtenue par une définition appropriée de la taille des rollback segments, de l'architecture des serveurs partagés et des buffers Redo Log et des zones mémoire réservées aux opérations de tri.

L'optimisation des requêtes SQL peut être obtenue en

* créant des index appropriés
* créant des clusters pour optimiser les opérations de jointure
* choisissant un mode de traitement des requêtes
* comparant plusieurs solutions SQL pour la même requête

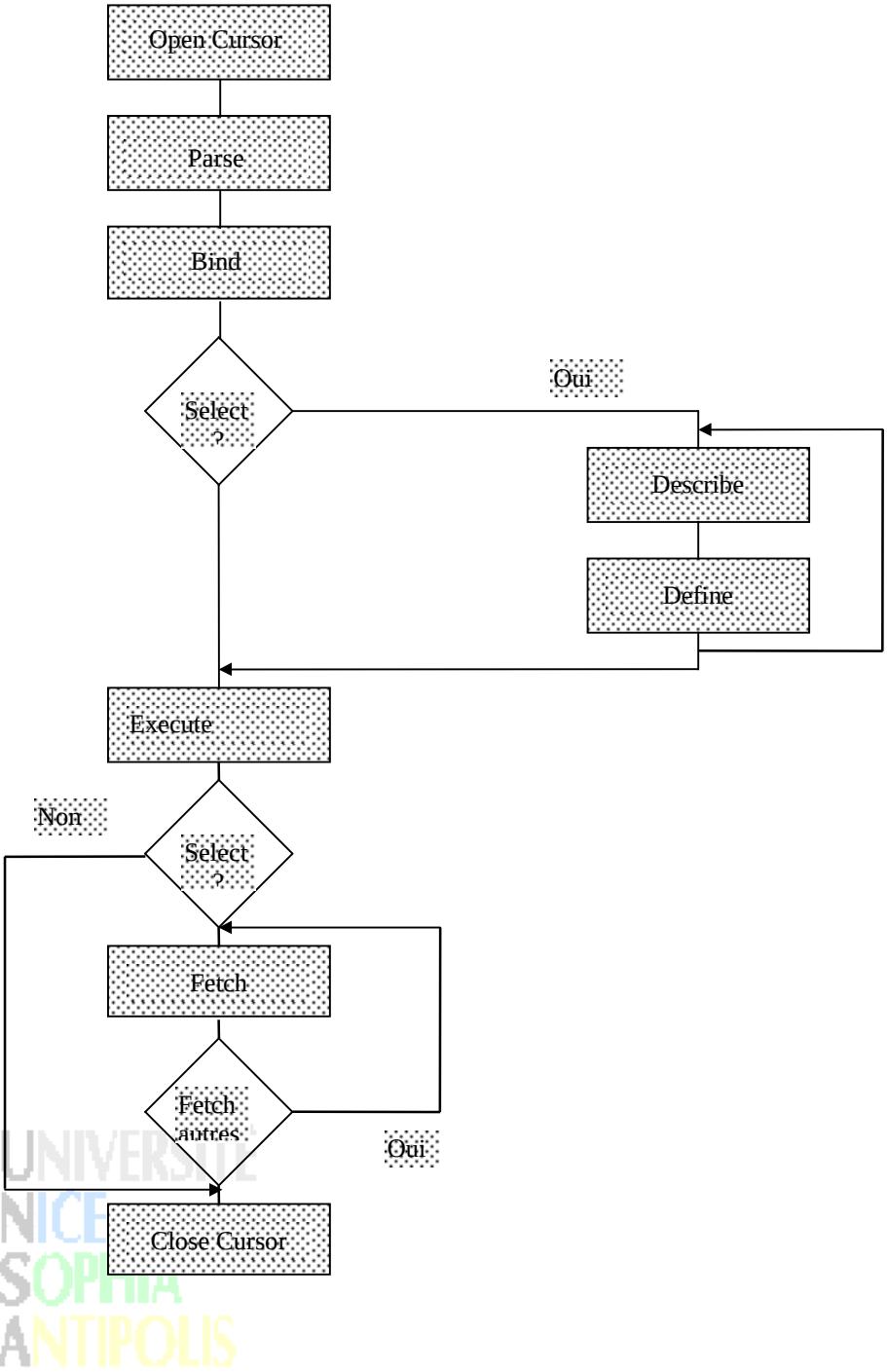
La création d'index ou de cluster est étudiée dans le chapitre 'ORGANISATION PHYSIQUE D'UNE BASE DE DONNEES'.

II- LES PHASES DE TRAITEMENT DES REQUÊTES

Le traitement de tout ordre SQL Oracle nécessite plusieurs phases successives: PARSE, BIND,DESCRIBE, DEFINE,EXECUTE et FETCH.

PARSE consiste à transmettre au serveur la chaîne de caractères constituant l'ordre SQL; le serveur décompose cette chaîne afin d'identifier les objets de la base de données qui vont être sollicités. A ce moment-là, les droits de l'utilisateur sur les objets concernés sont contrôlés. Un plan d'exécution est ensuite choisi en fonction des connaissances statistiques du serveur sur les objets mais aussi des éventuels 'hint' à destination de l'optimiseur.
BIND identifie les zones mémoire, taille et type correspondant aux variables de l'ordre SQL (variables de la clause WHERE, variables contenant les valeurs à insérer pour l'INSERT ou encore les variables de remplacement pour l'UPDATE).

DESCRIBE récupère dans le dictionnaire de données les descriptions précises des colonnes manipulées dans l'ordre SQL (dans le cas de SQL dynamique uniquement).
DEFINE est la phase d'identification des zones mémoire, de leur taille et de leurs types pour chacune des colonnes devant être rapatriées lorsqu'il s'agit d'un ordre SELECT.
EXECUTE permet au client de demander la constitution d'une liste de tuples dans le cas du SELECT ou simplement l'exécution de l'ordre SQL dans les autres cas, en communiquant au serveur les valeurs des variables définies dans la phase BIND précédente.
FETCH permet au client de recevoir tout ou partie des données de la liste constituée dans l'étape précédente, pour les ordres SELECT.



Les 6 phases de traitement d'un ordre SQL sont effectuées à l'initiative du client qui pilote le serveur; pour cela, le client fait réserver, sur le serveur, une zone mémoire appelée 'curseur' (OPEN CURSOR en PL/SQL). Les phases de PARSE,DESCRIBE,EXECUTE et FETCH nécessitent un échange entre le client et le serveur tout comme la demande de réservation de curseur sur le serveur; il faut donc en théorie 5 échanges entre le client et le serveur pour ramener un tuple. Pour diminuer ce nombre, Oracle utilise la technologie ARRAY qui rapporte en seul FETCH plusieurs tuples.
Généralement, les ordres SQL effectués dans une application sont toujours les mêmes, seules changent les valeurs des variables. Il n'est donc pas nécessaire de refaire toutes les phases de traitement de l'ordre SQL à chaque nouvelle exécution, pas plus que de libérer le curseur sur le serveur pour en réallouer un nouveau par la suite. Il suffit simplement de refaire les phases de BIND,d'EXECUTE et éventuellement de FETCH et donc de ne générer qu'un seul échange au lieu de deux, afin de ne pas consommer inutilement de la CPU sur le serveur pour des opérations déja réalisées.

III- LES MODES DE TRAITEMENT D'UNE REQUÊTE

Pour exécuter un ordre SQL (phase EXECUTE), Oracle effectue plusieurs étapes qui consistent à retrouver des enregistrements dans la base ou à préparer l'ordre suivant; la combinaison d'étapes choisie par l'optimiseur est appelée *execution plan*
Le tableau suivant présente les différentes opérations possibles:

OPERATION	OPTION	DESCRIPTION
AGGREGATE	GROUP BY	Recherche d'un tuple résultat d'une fonction de groupe
AND-EQUAL		Intersection d'ensembles éliminant les doublons
CONNECT BY		Recherche de tuples dans un ordre hiérarchique
CONCATENATION		Union-All de deux ensembles
COUNTING		Dénombrement des tuples sélectionnés
FILTER		Sélection de tuples
FIRST ROW		Premier tuple sélectionné
FOR UPDATE		Recherche et verrouillage de tuples
INDEX	UNIQUE SCAN	Recherche d'un tuple à partir d'un index (ROWID)
	RANGE SCAN	Recherche d'un ou plusieurs tuples à partir d'un index en ordre croissant
	RANGE SCAN DESCENDING	Recherche d'un ou plusieurs tuples à partir d'un index en ordre décroissant
INTERSECTION		Intersection entre 2 ensembles de tuples, en éliminant les doublons
MERGE JOIN		Jointure
	OUTER	Jointure externe
MINUS		Différence entre deux ensembles
NESTED LOOPS		Requêtes imbriquées
	OUTER	Requêtes imbriquées avec jointure externe
PROJECTION		Opération interne
REMOTE		Recherche dans une base distante

SEQUENCE		Opération invoquant la valeur d'une séquence
SORT	UNIQUE	Tri éliminant les doublons
	GROUP BY	Requête avec GROUP BY
	JOIN	Tri avant jointure
	ORDER BY	Requête avec ORDER BY
TABLE ACCESS	FULL	Parcours séquentiel
	CLUSTER	Recherche basée sur la valeur d'une clé d'un cluster indexé
	HASH	Recherche basée sur la valeur d'une clé d'un hash cluster
	BY ROWID	Recherche basée sur la valeur de ROWID
UNION		Union de deux ensembles
VIEW		Requête invoquant une vue

Pour choisir un plan d'exécution,l'optimiseur utilise une des deux approches suivantes: rule-based ou cost-based.

- **rule-based**: l'optimiseur examine les possibilités d'accès dans l'ordre de priorité suivant:

Ordre de priorité	Méthode d'accès
1	Single row by ROWID
2	Single row by cluster join
3	Single row by hash cluster key with unique or primary key
4	Single row by unique or primary key
5	Cluster join
6	Hash cluster key
7	Indexed cluster key
8	Composite index
9	Single-column indexes
10	Bounded range search on indexed column
11	Unbounded range search on indexed column
12	Sort-merge join
13	MAX or MIN of indexed column
14	ORDER BY on indexed column
15	Full table scan

-**cost-based**: l'optimiseur détermine l'ordre des opérations en fonction des statistiques contenues dans le dictionnaire de données sur les tables, colonnes, index et clusters; ces statistiques sont obtenues à l'aide de la commande ANALYZE; il procède en 3 étapes:

- génération d'un ensemble de plans d'exécution basés sur les chemins d'accès possibles
- estimation des coûts de chaque plan basée sur la distribution des données et les caractéristiques de stockage des tables, clusters et index
- choix du plan de moindre coût

Le choix de la méthode d'optimisation est obtenu à l'aide du

* paramètre OPTIMIZER_MODE dans le fichier init.ora
  - COST: coût estimé du nombre de lectures logiques
  - RULE: utilisation des règles de base

* paramètre OPTIMIZER_GOAL de la commande ALTER SESSION
  -CHOOSE: optimiseur statistique si le dictionnaire de données contient des statistiques
  -ALL_ROWS: optimiseur statistique par utilisation des informations statistiques: capacité de traitement optimale
  -FIRST_ROWS: optimiseur statistique par utilisation des informations statistiques: temps de réponse optimal
  -RULE: règles de base

Avantages de l'optimiseur statistique

- la recherche du plan d'exécution pour une requête est excellente ou bien meilleure que celui choisi par l'optimisation basée sur les règles pour de nombreuses requêtes(spécialement pour de grandes requêtes avec des jointures multiples)
- le réglage manuel de la syntaxe des requêtes n'est pas nécessaire dans beaucoup de cas (augmentation de la productivité)
- il permet au développeur de l'application de choisir entre l'optimisation de coûts ou de règles
  °les requêtes actuelles peuvent tourner sans être changées
  °l'utilisateur doit faire un choix entre les deux méthodes et prendre la plus efficace

L'optimisation basée sur les coûts sera choisie pour toutes les nouvelles applications et pour certaines requêtes spécifiques.
L'optimisation basée sur les règles sera choisie:
-pour les requêtes sur les tables où la taille et la distribution des données changent souvent
-pour les requêtes avec des jointures qui peuvent être réglées manuellement
-quand la création de statistiques n'est pas possible

IV- La commande EXPLAIN PLAN

La commande EXPLAIN PLAN décrit le plan choisi par l'optimiseur pour exécuter l'ordre SQL.

```
EXPLAIN PLAN
      set statement_id ='Nom_req'
FOR   select ename,job,sal,dname
      from emp,dept
      where emp.deptno = dept.deptno
      and not exists (select * from salgrade
                      where emp.sal between losal and hisal)
```

Le résultat de la commande EXPLAIN PLAN est mémorisé dans une table **plan_table** qui contient les colonnes suivantes:

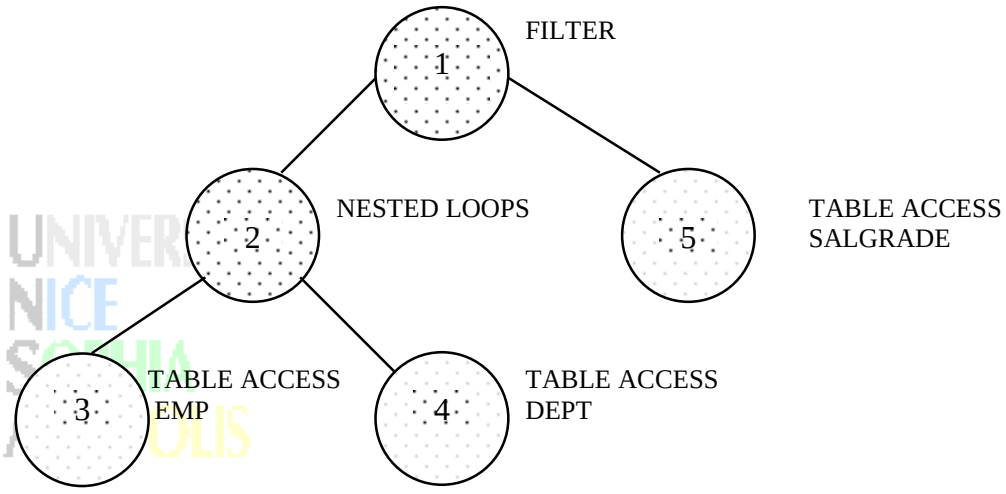
statement_id	varchar2(30)	nom de la requête
timestamp	date	date de la requête
remarks	varchar2(80)	
operation	varchar2(30)	
option	varchar2(30)	
object_node	varchar2(30)	nom du database link
object_owner	varchar2(30)	
object_name	varchar2(30)	nom de la table ou de l'index
object_instance	numeric	
object_type	varchar2(30)	
search_columns	numeric	
id	numeric	numéro de l'étape
parent_id	numeric	numéro de l'étape précédente
position	numeric	
other	long	

Pour la requête 'Nom_req',la commande EXPLAIN PLAN fournira le résultat suivant:

```
Select operation,options,object_name,id,parent_id,position
From plan_table
Where statement_id = 'Nom_req' Order by id
```

Operation	Options	Object_name	ID	Parent_ID	Position
Select statement			0		5
Filter			1	0	0
Nested loops			2	1	1
Table access	Full	Emp	3	2	1
Table access	Full	Dept	4	2	2
Table access	Full	Salgrade	5	1	3

correspondant au schéma suivant:



Les opérations symbolisées par un cercle gris indiquent un accès à la base de données, celles symbolisées par un cercle clair indiquent un traitement en mémoire.

V- La commande ANALYZE

La commande ANALYZE permet de
- collecter(ou de supprimer) des statistiques sur une table, un index ou un cluster
- valider la structure d'un objet de la base
- identifier les chaînages dans une table ou un cluster

Pour une **table**, ANALYZE fournit le nombre de lignes,le nombre de blocs contenant des données,le nombre de blocks alloués à la table jamais utilisés, l'espace libre moyen par bloc (en bytes), le nombre de lignes chaînées, la longueur moyenne d'une ligne (avec en-tête), le nombre de valeurs distinctes pour chaque colonne,la valeur minimum et la valeur maximum pour chaque colonne.
Pour un **cluster**, ANALYZE fournit le nombre moyen de blocs de données par valeur de clé de cluster (avec les données).
Pour un **index**, ANALYZE fournit la profondeur de l'arborescence,le nombre de blocs feuilles, le nombre de valeurs distinctes de la clé, le nombre moyen de blocs feuilles par valeur de clé,le nombre moyen de blocs de données pointé par valeur de clé, taux des données triées dans le segment de données, par rapport à la clé d'index.

Ces statistiques apparaissent dans les vues relatives à ces objets: *_TABLES, *_TAB_COLUMNS, *_CLUSTERS, *_INDEXES. Par défaut, l'analyse se fait sur 1064 lignes; si cette valeur représente plus de la moitié des lignes ou des clés, l'analyse se fait sur la totalité.

VI- LES MESURES DE PERFORMANCES

La **SQL trace facility** fournit des informations sur les ordres SQL. Elle génère les statistiques suivantes:
- -nombre d'opérations de type PARSE,EXECUTE et FETCH (count)
- -temps CPU (CPU) et temps total (elapsed) en secondes
- -nombre de lectures logiques (query pour SELECT
  current pour INSERT,UPDATE ou DELETE)
- -nombre de lectures physiques (disk)
- -nombre d'enregistrements traités (rows)

Le fichier trace est mémorisé dans le répertoire défini par le paramètre user_dump_dest; les statistiques de temps (cpu,elapsed) seront calculées si le paramètre timed_statistics est positionné à true.

La SQL trace facility est activée par la commande

```
ALTER SESSION
SET sql_trace = TRUE
```

Les informations fournies sont mémorisées dans un fichier du répertoire désigné par le paramètre USER_DUMP_DEST du fichier init.ora

Elles sont exploitées et mises en forme par la commande système TKPROF

TKPROF filename1 filename2 [SORT = option] [EXPLAIN=user/password]

filename1: nom du fichier fourni par trace_facility; l sous unix filename1 a pour valeur ora_*spid*.trc, *spid* étant le numéro du processus exécutant la session sqlplus. Sa valeur est obtenue par la commande SQL
Select spid from v$process where username = lower(USER)

filename2: nom du fichier de sortie; vous choisirez *<login>.trc*

SORT= option: présentation ordonnée des résultats
EXPLAIN=user/password: entraine l'exécution de la commande EXPLAIN PLAN pour chaque ordre SQL du fichier trace.

Exemple:

TKPROF: Release 7.1.6.2.0 - Production on Thu Feb 8 14:55:34 1996

Trace file: ora_14689.trc
Sort options: default
****
count = number of times OCI procedure was executed
cpu = cpu time in seconds executing
elapsed = elapsed time in seconds executing
disk = number of physical reads of buffers from disk
query = number of buffers gotten for consistent read
current = number of buffers gotten in current mode (usually for update)
rows = number of rows processed by the fetch or execute call
****
Recherche séquentielle:

```
select nomemp
from personnel
where nomemp like '_K%'
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.06	0.06	0	0	0	0
Execute	1	0.00	0.0	0	0	0	0
Fetch	12	0.40	0.48	463	475	4	166
total	14	0.46	0.54	463	475	4	166

Misses in library cache during parse: 1
Optimizer hint: CHOOSE
Parsing user id: 8

Recherche en utilisant un index

```
select nomemp
from personnel
where numemp =8765
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.03	0.03	0	0	0	0
Execute	1	0.00	0.0	0	0	0	0
Fetch	1	0.00	0.00	1	3	0	1
total	3	0.03	0.03	1	3	0	1

Misses in library cache during parse: 1
Optimizer hint: CHOOSE
Parsing user id: 8

VII-QUESTIONS

I/ - Dans la base IUP, vous disposez des tables JCG.EMPLOYE,JCG.DEPART,JCG.VILLE qui ont pour synonymes publics PERSONNEL,SERVICE et CITE. La commande ANALYZE a été appliquée à ces tables et à leurs index.
Afficher , pour chaque table, les valeurs des paramètres PCTFREE et PCTUSED, le nombre d'enregistrements, le nombre de blocs occupés, le nombre de blocs vides et la longueur moyenne d'un enregistrement.
Pour les index construits sur les tables EMPLOYE et DEPART, vous afficherez le nom de l'index, le nom de la table, le nombre de blocs feuilles et le nombre de clés distinctes.

II/- Créer la table plan_table à partir du fichier /u/profs/jcg/cr_plantable.sql

III/- Appliquer la commande EXPLAIN PLAN aux requêtes suivantes:

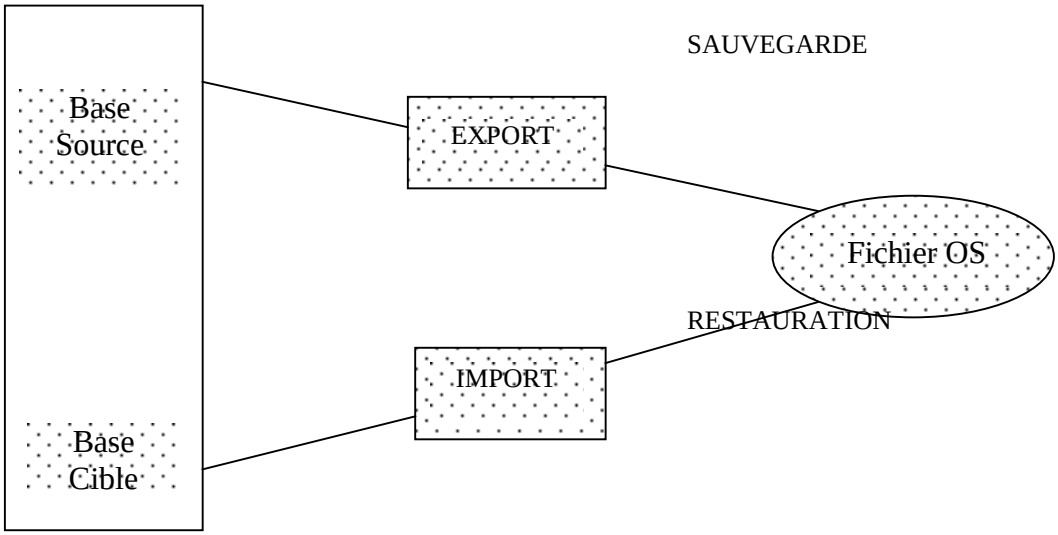
a/ Nom des employés qui ont un 'K' en deuxième position
b/ Nom de l'employé ayant le numéro 195
c/ Nom des employés ayant un numéro < 195
d/ Nom des employés ayant un numéro > 195
Vous traiterez ces requêtes
* avec l'option CHOOSE du paramètre OPTIMIZER_GOAL
* avec l'option RULE du paramètre OPTIMIZER_GOAL
e/ Nombre d'employés par département (service)
f/ Nom des employés qui travaillent à NICE; vous traiterez cette requête à l'aide
* d'une opération de jointure
* d'une sous-requête
g/ Nom des villes qui n'ont pas de département; vous traiterez cette requête à l'aide
* de l'opérateur NOT IN
* de l'opérateur MINUS

Expliquer les résultats obtenus

IV/ L'option trace_facility étant activée, exécuter ces requêtes .Appliquer la commande TKPROF (le nom du fichier trace vous sera fourni) et commenter les résultats obtenus dans le fichier<*login*>.trc.

OUTILS D'ADMINISTRATION

I- EXPort/IMPort



EXPort est utilisé pour

-sauvegarder la définition des tables (avec ou sans les données)
-transférer les données entre deux versions d'Oracle
-changer le propriétaire des données
-transférer les données entre deux deux machines
-réorganiser la base de données
-fusionner les données de plusieurs bases

EXPort peut être utilisé suivant 3 modes:

-mode TABLE: seules les tables appartenant à l'utilisateur peuvent être exportées
-mode USER: tous les objets appartenant à l'utilisateur peuvent être exportés
-mode FULL: tous les objets de la base peuvent être exportés
(rôle EXP_FULL_DATABASE)

En mode FULL, il existe 3 types d'EXPort:
-COMPLETE: toutes les données sont exportées
-CUMULATIVE: seules les tables modifiées depuis le dernier EXPort CUMULATIVE ou COMPLETE sont exportées
-INCREMENTAL: seuls les objets modifiés depuis le dernier EXPort sont exportés

L'utilisation d'EXPort peut se faire

**en mode interactif*
Appel par *$ exp username*
Tous les autres paramètres sont demandés interactivement par le système

**en mode commande*
Appel par *$ exp username/password*
tables=(T1,T2)
grants=Y
indexes= Y

**en mode batch*
Appel par *$ exp parfile= filename*
filename est un fichier contenant tous les paramètres

PARAMETRES

PARAMETRES	**Interactif**	**Défaut**	**Signification**
USERID	Y	OPS$	username et password
BUFFER	Y	4096	taille du buffer de l'export
FILE	Y	expdat.dmp	fichier résultat de l'export
COMPRESS	Y	Y	compression des extensions
GRANTS	Y	Y	export des privilèges
ROWS	Y	Y	export des données
TABLES	Y	Y	tables concernées par l'export
INDEXES	N	Y	export des index
CONSTRAINTS	N	Y	export des contraintes d'intégrité
LOG	N		nom du fichier trace de l'export
FULL	N	Y	export total
OWNER	N	util courant	utilisateur courant
RECORDLENGTH	N	OS dep	taille des enregistrements du fichier export
INCTYPE	N		type d'export (complete, incremental, cumulative)
RECORD	N	Y	utilisé uniquement pour incremental
PARFILE	N		fichier de paramètres en mode batch
CONSISTENT	N	N	lecture cohérente
ANALYZE	N		calculs statistiques

Interactif= Y: paramètre utilisable en mode interactif

IMPort permet de charger des données à partir d'un fichier créé par EXPort. On utilise IMPort pour

-restaurer la définition des tables (avec ou sans les données)
-récupérer les données d'une autre base
-restaurer une base perdue

Certains privilèges sont nécessaires pour pouvoir utiliser IMPort
-privilèges SYSTEM: create table, tablespace quota ou unlimited tablespace
-privilèges OBJET: insert table, alter table

Comme pour EXPort, l'utilisation d 'IMPort peut se faire:

**en mode interactif*
Appel par *$ imp username*

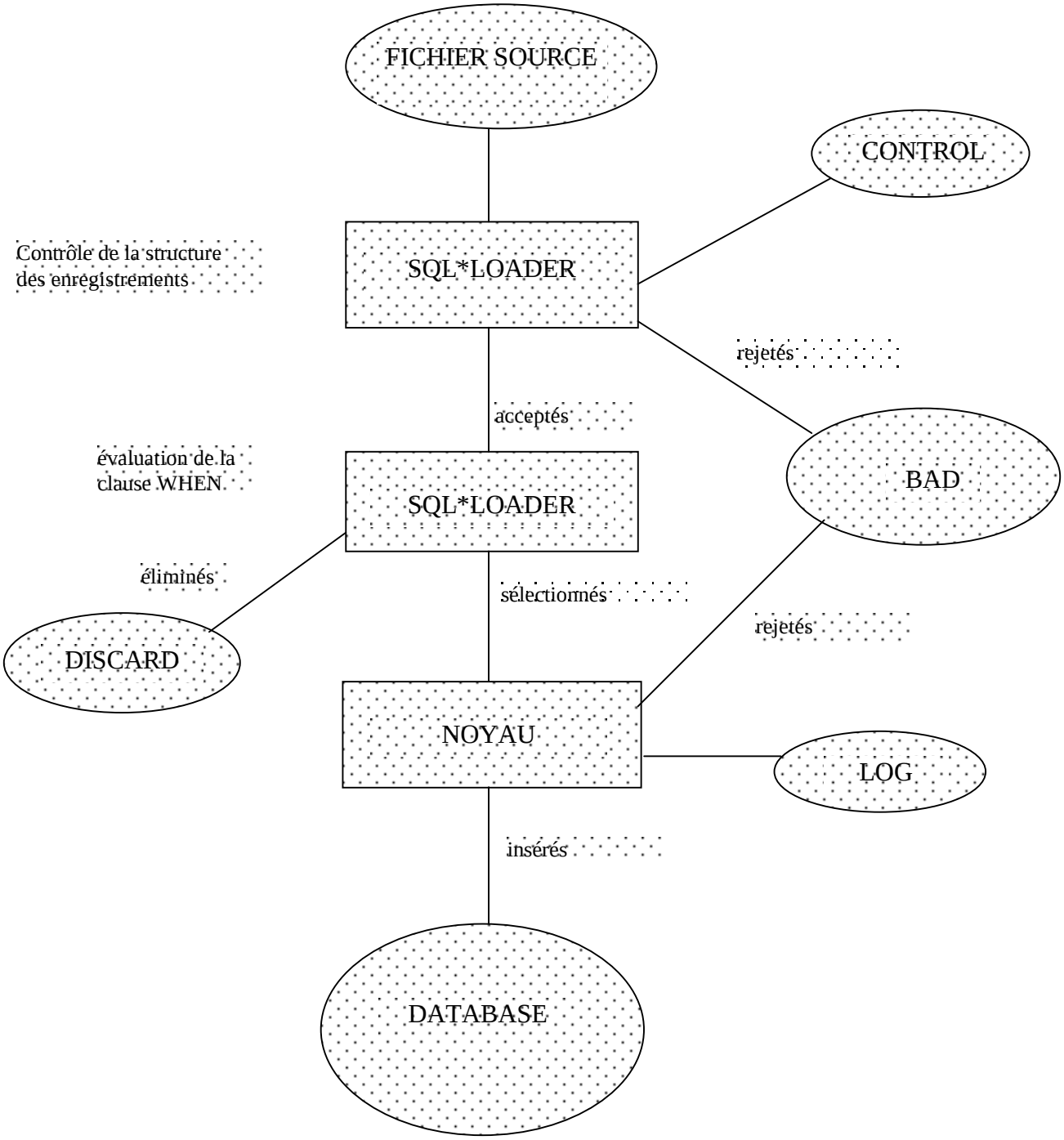
**en mode commande*
Appel par *$ imp username/password FILE= filename.dmp TABLES=(T1,T2)*
**en mode batch*
Appel par *$ imp parfile=filename*

PARAMETRES

PARAMETRES	Interactif	Défaut	Signification
USERID	Y	OPS$	username et password
BUFFER	Y	4096	taille du buffer de l'export
FILE	Y	expdat.dmp	fichier résultat de l'export
SHOW	Y	N	simulation de l'import
GRANTS	Y	Y	import des privilèges
ROWS	Y	Y	import des données
TABLES	Y	Y	tables à importer
INDEXES	N	Y	export des index
IGNORE	Y	Y	ignorer les erreurs éventuelles
LOG	N		nom du fichier trace de l'import
FULL	Y	N	import total
FROMUSER	N		utilisateur propriétaire
RECORDLENGTH	N	OS dep	taille des enregistrements du fichier import
INCTYPE	N		type d'export (restore,system)
DESTROY	N	N	réutilisation des fichiers de la base
PARFILE	N		fichier de paramètres en mode batch
COMMIT	N	N	commit après chaque tableau d'enregistrements importés
INDEXFILE	N		nom du fichier contenant la description des index

II-SQL*LOADER

SQL*LOADER est un outil d'administration utilisé pour charger les données d'un ou plusieurs fichiers séquentiels dans une ou plusieurs tables. La figure suivante présente son mode de fonctionnement.



SQL*LOADER permet de

-charger des données à partir d'un ou plusieurs fichiers de différents types
-charger des enregistrements de longueur fixe ou variable
-manipuler des données à l'aide de fonctions SQL avant de les insérer dans la base
-traiter des données de différents types, y compris DATE, BINARY, PACKED DECIMAL, ZONED DECIMAL
-charger plusieurs tables simultanément, en sélectionnant les enregistrements
-regrouper plusieurs enregistrements physiques en un enregistrement logique
-traiter un enregistrement physique comme plusieurs enregistrements logiques
-générer des clés séquentielles et uniques
-fournir des messages d'erreur
-utiliser un dispositif d'accès direct pour charger rapidement les données dans les fichiers database

SQL*LOADER utilise les fichiers suivants:

en entrée
-DATA: des données à charger dans la base
-CONTROL: description du programme de chargement et éventuellement des enregistrements du fichier DATA

en sortie
-BAD: liste des enregistrements ne correspondant pas à la description contenue dans le fichier CONTROL et rejetés par le noyau Oracle
- DISCARD: liste des enregistrements rejetés à la suite d'un test utilisateur dans le fichier CONTROL
-LOG: statistiques sur le déroulement de l'opération

Structure du fichier CONTROL

LOAD DATA	Instruction obligatoire
INFILE nom du fichier DATA (.dat) *	données dans le fichier de contrôle après BEGINDATA
[BADFILE 'badfile_name] [DISCARDFILE 'discardfile_name']	
[CONCATENATE n] [CONTINUEIF]	création d'un enregistrement logique à partir de plusieurs enregistrements physiques
INSERT/APPEND/REPLACE/TRUNCATE	INSERT par défaut (la table cibledoit être vide)
INTO TABLE table_name	

[structure de la table]

[FIELDS TERMITATED BY ' délimiteurs
[OPTIONALLY ENCLOSED BY ']

[WHEN condition] transfert éventuel vers le fichier discard
[TRAILING NULLCOLS] mettre à null les colonnes manquantes

[SORTED INDEXES] utilisation d'un chemin direct

[BEGINDATA début des données si INFILE *
données]

Le fichier log

Il contient des informations statistiques relatives au déroulement du programme SQL*LOADER

exemple:
SQL*Loader: Release 7.3.2.1.0 - Production on Fri Jan 24 10:46:44 1997

Control File: ulcase1.ctl
Data File: ulcase1.ctl
Bad File: ulcase1.bad
Discard File: none specified

(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array: 64 rows, maximum of 65536 bytes
Continuation: none specified
Path used: Conventional

Table "JCG"."DEPART", loaded from every logical record.
Insert option in effect for this table: INSERT

Column Name	Position	Len	Term	Encl	Datatype
DEPTNO	FIRST	*	,	O(')	CHARACTER
DNAME	NEXT	*	,	O(')	CHARACTER
LOC	NEXT	*	,	O(')	CHARACTER

Table "JCG"."DEPART":
7 Rows successfully loaded.

```
 0 Rows not loaded due to data errors.
 0 Rows not loaded because all WHEN clauses were failed.
 0 Rows not loaded because all fields were null.


Space allocated for bind array:                49920 bytes(64 rows)
Space allocated for memory besides bind array:    97276 bytes

Total logical records skipped:          0
Total logical records read:             7
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Fri Jan 24 10:46:44 1997
Run ended on Fri Jan 24 10:46:50 1997

Elapsed time was:     00:00:05.59
CPU time was:         00:00:00.87
```

Lancement de SQL*LOADER

Un programme SQL*LOADER est contenu dans un fichier control qui a obligatoirement le suffixe ctl

L'appel à SQL*LOADER se fait par la commande **sqlldr** qui peut avoir les paramètres suivants:

-userid=	login/password
-control=	nom du fichier control (.ctl)
-log=	nom du fichier log (.log)
-bad=	nom du fichier bad (.bad)
-data=	nom du fichier data (.dat)
-discard=	nom du fichier discard (.dsc)

Ces paramètres peuvent également être définis dans le fichier control

SQL*LOADER peut s'exécuter selon deux possibilités:

- chemin "traditionnel"
- chemin "direct"

Chemin "traditionnel"
Oracle utilise le chargement en tableau (ARRAY) et le verbe INSERT; il contrôle les contraintes d'intégrité et exécute les triggers Base de données; il conserve les traces d'insertion dans un fichier LOG et met à jour les index existants.
Chemin "direct"
Ce chemin est choisi en positionnant le paramètre DIRECT à TRUE; il y a inhibition du contrôle des contraintes d'intégrité référentielle et des triggers Base de données; seules restent actives les contraintes NOT NULL, UNIQUE et PRIMARY KEY.

III-ORACLE ENTERPRISE MANAGER

Outil graphique d'Administration de Bases de Données permettant de réaliser, à partir d'un poste de travail Windows NT, les tâches suivantes :

-Administration, diagnostic, optimisation de plusieurs bases
-Distribution de software à des postes clients
-Programmation de Jobs s'exécutant à intervalles réguliers
-Gestion d'évènements à travers le réseau

Oracle Enterprise Manager est formé de trois composants de base :

- ***La Console***: ensemble de fenêtres, menus, barres d'outils et palettes permettant de réaliser les différentes tâches du DBA. Elle est constituée par 4 *fenêtres* :

* *Navigator*: parcours hiérarchisé des objets composanrt la base (Network, database(users, roles,profiles), group, listeners, nameservers, nodes
**Map*: gestion d'un groupe d'objets
**Event Management*:: création et enregistrement des évènements, visualisation de l'état des services devant être gérés, visualisation d'informations sur des évènements survenus
**Job Control System*::création, exécution suppression de jobs

-***Common Services***:

**Job Control System*: Programmation et exécution de jobs sur des sites distants, à travers le réseau (ex: rapport édité tous les dimanche soir)
**Event Management System*: Gestion d'évènements particuliers (ex:gestion d'une alerte si un tablespace n'a plus de d'espace disponible)
**Repository*: Ensemble de tables mémorisant la configuration et des informations sur l'état de la base
**Intelligent Agents and Communication Daemon*:: Processus assurant la gestion des tâches telles que exécution de jobs ou pilotage d'évènements
**Discovery cache*: le daemon de communication peut localiser des services par lecture de fichiers de topologie ou en interrogeant le server de noms SQL*Net

-***Integrated Applications*** : ensembles d'applications spécialisées dans des tâches spécifiques

Database Administration Applications

**Instance Manager*: démarrage et arrêt d'une base, examen des paramêtres (fichier init.ora),
gestion des sessions

*Schema manager**:*** création, édition et visualisation des objets suivants: clusters, database links, functions et procedures, indexes, packages, refresh group, sequences, snapshots et snapshots logs, synonyms, tables, triggers, views

**Security Manager*: création, modification et suppression des utilisateurs, roles et profiles

**Storage Manager*: gestion des tablespaces et des rollback segments, ajout et renommination des fichiers Datafile

**SQL Worksheet*: écriture et exécution dynamique des requêtes SQL, des programmes PL/SQL, des commandes Oracle Enterprise Manager et des scripts

**Software Manager*: automatisation des processus de distribution des softwares à travers une architecture Client/Serveur, désinstallation des produits, effacement ou installation de packages, distribution de fichiers

**Backup Manager*: réalisation de backup de tablespaces, création de scripts de backup,administration de redo log

**Data Manager*: transfert de données à l'aide de Export, Import, Load

Oracle Enterprise Manager Performance Pack

**Oracle Expert*: Optimisation des performances de la base

**Oracle Locks Manager*: Gestion des verrous

**Oracle Performance Manager*: Statistiques d'utilisation sur les utilisateurs, tablespaces, redo log, buffers, cache, I/O

**Oracle TopSession*: Statistiques sur l'activité de la base au cours d'une session

**Oracle Tablespace Manager*:Gestion de l'espace disque: tablespaces, fichiers

**Oracle Trace*

IV-QUESTIONS

I/ A l'aide des utilitaires EXPort et IMPort, transférer les données de votre table dept de la base <login> dans la table dept de la base IUP; écrire une procédure qui puisse être lancée automatiquement.

II/Le répertoire ~oracle/sqlplus/help/admin contient les fichiers nécessaires à l'installation de l'aide SQL-PL/SQL et SQL*PLUS. Après avoir examiné le contenu de ces fichiers, réaliser l'installation de ce Help;.

SAUVEGARDE ET RESTAURATION D'UNE BASE DE DONNEES

Une des plus importantes tâches de l'administrateur de la base de données est la planification et la mise en place des procédures de sauvegarde et de restauration de la base. Cette planification est guidée par plusieurs critères:

-est-il acceptable de perdre des données et, si oui, dans quelles limites?
-la base doit-elle être accessible en permanence?
-quel est le délai acceptable de restauration au niveau de l'utilisation de la base?
-quelle est la fréquence d'évolution des données?
-les modifications physiques de la base sont-elles nombreuses?

I-LES TYPES DE PANNES

Erreur accidentelle: suppression de tables ou d'autres objets; de telles erreurs peuvent être évitées grâce aux dispositifs de confidentialité d'Oracle et à des contrôles stricts au niveau des programmes utilisateurs.

Panne sur une commande SQL: en présence d'un problème d'allocation d'extensions ou autres, Oracle renvoie un code d'erreur et annule les commandes de cette transaction pour rétablir la cohérence de la base de données

Panne d'un processus: si un processus s'arrête brutalement sans avoir validé ou annulé les transactions en cours, le processus PMON détecte cet état, se connecte automatiquement sur le compte disparu pour valider ou annuler la transaction et libérer les ressources (verrous) détenues par ce processus.

Panne réseau: une panne réseau peut interrompre l'exécution normale d'une application cliente et provoquer une panne de processus; cette panne sera résolue par le processus PMON de la même manière que précédemment. En cas de transaction répartie, une fois que le problème réseau est résolu, le processus RECO de chaque noeud participant à la transaction résout le même problème de son coté.

Panne d'instance: une panne matériel ou logiciel peut empêcher une base Oracle de continuer à fonctionner. La base s'arrête dans un état incohérent: certaines modifications validées n'ont pas encore été transmises aux fichiers de données mais existent sur les journaux de reprise; inversement certaines modifications non validées ont déja été transmises aux fichiers de données (mais leur image avant existe dans le segments Rollback). Au démarrage, Oracle détecte automatiquement que l'arrêt précédent ne provient pas d'une fermeture (shutdown) propre et le processus SMON effectue le "Roll-forward" de l'ensemble des modifications présentes dans les journaux de reprise, suivi de l'annulation des transactions non validées.

Panne disque: une cause matériel peut empêcher de lire et d'écrire sur les fichiers concernés par les transactions en cours. Le cas le plus fréquent est l'erreur I/O sur l'un des fichiers de la base (données, reprise, contrôle) ; une restauration appropriée dépend du type de fichier perdu.

Les opérations d'Oracle après une panne affectant le journal de reprise ou le fichier de contrôle dépendent de la présence de fichiers miroir. En présence d'un miroir du fichier reprise en ligne endommagé, Oracle continue à fonctionner sans interruption; dans le cas contraire, les opérations d'Oracle s'arrêtent et peuvent provoquer des pertes de données.
Oracle s'arrête si le fichier Control est endommagé, qu'il y ait fichier miroir ou non.

Les pannes sur fichier de données se situent à deux niveaux:

-Oracle détecte une erreur de lecture sur un fichier de données, un code d'erreur est renvoyé à l'utilisateur et Oracle continue à fonctionner

-Oracle détecte une erreur d'écriture : si le fichier de reprise en ligne plein est archivé, une erreur est retournée par le fichier de trace et le tablespace endommagé est mis automatiquement offline; si le fichier de reprise en ligne plein n'est pas archivé, le processus DBWR s'arrête et l'instance se bloque.

II-LES STRUCTURES DE SAUVEGARDE

Les segments Rollback
Ils servent à mémoriser les anciennes valeurs de données modifiées pour permettre au système de revenir en arrière (Rollback) avec une annulation de transaction

Les fichiers de contrôle
Le fichier de contrôle est un petit fichier binaire associé à une instance et nécessaire à son démarrage. Il contient des informations demandées à chaque accès à la base: nom de la base, estampille de la création de la base, noms et localisations des différents journaux, numéro de séquence du journal ouvert, informations sur le checkpoint pour garder la prochaine entrée dans le journal.
Le fichier de contrôle est mis à jour chaque fois qu'il y a modification de la structure des fichiers de la base. Oracle 7 permet l'ouverture et l'écriture de plusieurs fichiers de contrôle concurrents pour la même base de données, offrant ainsi un miroir de ce fichier. Ces fichiers doivent être listés par l'initialisation du paramètre CONTROL_FILES du fichier INIT.ORA en respectant la procédure suivante:

-arrêt de la base en mode Normal ou Immediate
-copie du fichier control_file1 dans un fichier control_file2 situé sur un autre disque
-modification de init.ora CONTROL_FILES= control_file1,control_file2
-redémarrage de la base

Le premier fichier est nécessaire pour la lecture, les autres sont seulement écrits chaque fois qu'il est nécessaire de mettre à jour le fichier de contrôle.
Une instance devient inopérante quand le premier fichier de contrôle n'est plus disponible. Il faut donc fermer la base et la redémarrer pour que le système pointe sur le fichier suivant. Il est recommandé d'avoir un minimum de deux fichiers de contrôle, localisés si possible sur des disques distincts.
Si une panne disque affecte les fichiers de données de la base, et si un recouvrement incomplet recouvrant une certaine période est désiré, c'est la sauvegarde du fichier de contrôle correspondant à l'intention souhaitée qui est utilisée et pas nécessairement le fichier de contrôle courant.

Les journaux de reprise (REDO LOG)
Ils conservent une trace des enregistrements modifiés dans la base. Chaque instance doît avoir au moins deux journaux de reprise indépendants des fichiers de données. Il existe deux types de fichiers de reprise: les journaux en ligne et les journaux archivés

Le journal de reprise en ligne
Les journaux de reprise sont remplis avec des entrées de reprise enregistrant les données qui peuvent être utilisées pour reconstruire toutes les modifications faites sur la base de données, y compris les segments Rollback. Ces entrées de reprise sont mémorisées d'une façon circulaire dans le buffer de reprise de la SGA et sont écrites sur le fichier de reprise par le processus d'arrière-plan LGWR.
Au moins deux journaux de reprise sont nécessaires pour assurer le fonctionnement de la base; l'un deux peut être disponible quand l'autre est plein et est en attente d'être archivé ou d'être réutilisé. Chaque fois qu'un journal est plein, il est désactivé et le prochain journal est activé et devient le journal courant de l'instance. Le point qui correspond à la fin de l'écriture du journal courant et au début de l'écriture sur le prochain journal est dit *interrupteur journal*. Ce point peut être forcé, par l'administrateur, par la fermeture du journal de reprise courant pour archiver un tablespace par exemple.

```
ALTER SYSTEM SWITCH LOGFILE;
```

Chaque journal de reprise possède un numéro de séquence journal qui s'incrémente chaque fois que l'interrupteur journal se produit. Le fichier de contrôle mémorise le plus récent numéro de séquence journal.

CHECKPOINT

Un autre évènement nommé checkpoint se produit quand le processus DBWR enregistre sur les fichiers de données tous les buffers modifiés de la SGA, qu'ils soient validés ou non. Cet évènement garantit que tous les blocs de segments de données qui changent fréquemment, soient écrits dans les fichiers de données à intervalles réguliers.
Un checkpoint complet garantit que toutes les données modifiées depuis le dernier checkpoint sont réellement écrites sur le disque. Il peut se produire pour tous les fichiers de données et plus particulièrement dans les situations suivantes:

-à chaque interrupteur journal
-à chaque valeur des paramètres: LOG_CHECKPOINT_INTERVAL et LOG_CHECKPOINT_TIMOUT du fichier init*SID*.ora
-à chaque sauvegarde d'un tablespace activé
-chaque fois qu'un tablespace est désactivé
-à chaque fermeture de la base en mode NORMAL ou IMMEDIATE
-à chaque commande ALTER SYSTEM CHECKPOINT GLOBAL

Oracle offre la possibilité de créer des fichiers miroir sur les journaux de reprise; écrits par LGWR d'une façon concurrente, ils garantissent une sécurité supplémentaire et une disponibilité de haut niveau en présence d'un problème sur un journal de reprise. L' ensemble de ces journaux est appelé un groupe et chaque journal du groupe est appelé un membre du groupe. Un seul groupe est actif à la fois, et tous les journaux membres de ce groupe sont mis à jour concurrement par le processus LGWR.

Le journal de reprise archivé

Pour permettre le recouvrement de toutes les transactions validées sans perte d'information, les fichiers de reprise pleins doivent être archivés dès qu'ils sont inactifs. Le choix d'archivage des journaux dépend essentiellement de la politique de sauvegarde et de restauration souhaitée.

Le mécanisme d'archivage des groupes de journaux de reprise pleins est exécuté:

-*automatiquement* par le processus ARCH en l'activant par le paramètre LOG_ARCHIVE_START=true
LOG_ARCHIVE_DEST= destination de duplication des fichiers de reprise
LOG_ARCHIVE_FORMAT= formatage du nom de duplication des fichiers de reprise

l'instance devra être arrétée, démarrée dans le statut MOUNT,changée de mode de fonctionnement (ALTER DATABASE ARCHIVELOG) et ouverte (ALTER DATABASE OPEN)

ou par la commande

ALTER SYSTEM ARCHIVE LOG START

-*manuellement* par l'administrateur, mais cette possibilité n'est pas recommandée puisqu'il n'existe aucun moyen de savoir à quel moment les journaux de reprise seront disponibles à l'archivage.

Les vues V$ARCHIVE etV$LOG donnent des informations sur la liste des journaux archivés; la commande ARCHIVE LOG LIST (svrmgr) donne les informations suivantes: mode de journalisation de la base de données, archivage automatique, destination des fichiers à archiver, ancien numéro de séquence du journal de reprise en ligne, prochain numéro de séquence du journal à archiver, numéro de séquence courant.

III-LES TYPES DE SAUVEGARDE

- Sauvegarde complète

La sauvegarde de tous les fichiers de la base (données, journaux, contrôle) doit être réalisée après une fermeture propre (dans un état cohérent) de la base (shutdown normal). Si la base opère en mode NOARCHIVELOG et si une panne disque affecte quelques fichiers, seule la sauvegarde la plus récente peut être utilisée pour restaurer la base. Toutes les données modifiées ou saisies depuis cette dernière sauvegarde doivent être ressaisies.

Si la base est en mode ARCHIVELOG, la sauvegarde la plus récente est utilisée comme partie de recouvrement de la base; celui-ci peut être complété par par l'application des journaux en ligne ou archivés pour restaurer les données modifiées entre la sauvegarde et la panne.

- Sauvegarde partielle

Une sauvegarde partielle consiste à sauvegarder les fichiers d'un tablespace, un des fichiers de données ou les fichiers de contrôle; cette sauvegarde peut être réalisée avec une base ouverte ou fermée. La sauvegarde partielle est utile uniquement pour une base qui fonctionne en mode ARCHIVELOG, car les journaux de reprise archivés permettent un recouvrement de la base à partir des fichiers restaurés.

- Sauvegarde des fichiers de données

Un ou plusieur fichiers de données peuvent être sauvegardés indépendamment des autres fichiers; il est préférable de réaliser cette opération avec une base fermée.

- Sauvegarde du fichier de contrôle

La sauvegarde du fichier de contrôle doit être faite chaque fois qu'il y a modification dans la structure des fichiers. Les fichiers de contrôle miroir assurent la sécurité en cas de perte.

Stratégies de sauvegarde

La stratégie de sauvegarde est fonction des deux critères suivants:

- Acceptabilité: Peut-on accepter de perdre des données en cas de panne?

Si la réponse est non, la base doit opérer obligatoirement en mode ARCHIVELOG pour sauvegarder les journaux de reprise pleins.
Si la réponse est oui, la base pourra opérer en mode NOARCHIVELOG et la fréquence des sauvegardes dépendra de la tolérance de perte (1 jour ou 1 semaine par exemple).

- Disponibilité: Quelle est la durée de panne tolérée?

Une base fortement disponible fonctionnera en mode ARCHIVELOG. Les journaux de reprise en ligne pleins sont archivés automatiquement ou manuellement; couplés avec les journaux en ligne, ils permettront un recouvrement complet de la base jusqu'au moment de la panne.

La stratégie de sauvegarde comportera les étapes suivantes:

-sauvegarde de la base entière lors de sa création
-sauvegardes partielles pouvant mettre à jour la sauvegarde de la base
-les sauvegardes des fichiers de données activés et désactivés peuvent être utilisés pour mettre à jour les sauvegardes des fichiers de données. Les fichiers des tablespaces les plus utilisés doivent être sauvegardés fréquemment pour réduire le temps de recouvrement en cas de panne.
-sauvegarde du fichier de contrôle pour chaque modification de structure de la base.
-les journaux de reprise en ligne n'ont pas besoin d'être sauvegardés.

IV-EXECUTION D'UNE OPERATION DE SAUVEGARDE

- détermination des fichiers à sauvegarder

Select name from v$datafile pour les fichiers de données
Select name from v$logfile pour les journaux de reprise en ligne
Select name from v$controlfile pour les fichiers de contrôle

- désactivation de ces fichiers

*arrêt de la base en cas de sauvegarde complète
*alter tablespace tablespace_name begin backup pour une sauvegarde partielle d'un tablespace activé

*alter tablespace tablespace_name offline normal pour la sauvegarde d'un tablespace désactivé

*alter database backup controlefile to 'nom_fich' reuse pour la sauvegarde de fichier de contrôle avec base de données ouverte

-exécution de la sauvegarde

*commande de l'operating system

*commande EXPORT

-réactivation des objets désactivés

		Acceptabilité des pertes		
		Une journée	**Quelques heures**	**Aucune**
Fréquence d'arrêt	**1 fois par jour**	BACKUP ou EXPORT	BACKUP/jour config REDOLOGS	BACKUP online & ARCHIVELOG
	1 fois par semaine	BACKUP/semaine + EXPORT/jour	BACKUP/semaine + EXPORT/jour config REDOLOGS	BACKUP online & ARCHIVELOG
	Jamais	EXPORT/jour	BACKUP online & ARCHIVELOG	BACKUP online & ARCHIVELOG

V-RESTAURATION D'UNE BASE DE DONNEES

Les buffers Database sont écrits sur disque uniquement lorsque c'est nécessaire, en utilisant l'algorithme LRU (Last Recently Used); les fichiers de données pevent ainsi contenir des blocs de données modifiées par des transactions non validées et ne pas contenir des blocs de données modifiées par des transactions validées (ces données modifiées sont contenues dans le journal de reprise)
Pour résoudre ces problèmes, Oracle réalise la restauration de la base en deux étapes:

-la première étape d'un recouvrement consiste à appliquer aux fichiers de données « l'image avant » du journal de reprise (roll-forward); cette opération consiste à enregistrer toutes les modifications contenues dans des journaux en ligne et archivés sur les fichiers de données et sur les rollback segments. Après cette opération, les fichiers de données contiennent toutes les modifications, validées ou non.

-la seconde étape consiste à appliquer sur les fichiers de données « l'image arrière » à partir des rollback segments; cette étape annule l'action des transactions non validées.

Recouvrement après une panne d'instance

Le recouvrement d'une instance restaure la base dans l'état cohérent qu'elle possédait juste avant la panne. Le noyau Oracle exécute automatiquement les étapes suivantes:

-déroulement de l'image avant « Roll Forward »
-déroulement de l'image après « Roll Back »
-libération des ressources et des verrous maintenus par la transaction au moment de la panne
-en cas de base de données réparties, résolution des transactions participant à la validation à deux phases

Les actions à réaliser par l'administrateur sont:
-fermer la base avec l'option Normal (SHUTDOWN NORMAL)
-ouvrir la base avec STARTUP

Recouvrement après une panne disque

Le mode de recouvrement après panne disque dépend du mode d'archivage dans lequel opérait la base avant la production de la panne.
En mode NOARCHIVELOG, les journaux de reprise pleins sont réutilisés sans être archivés. Le recouvrement consiste alors en une restauration de la sauvegarde complète la plus récente; les modifications réalisées après cette dernière sauvegarde sont alors perdues.
En mode ARCHIVELOG, le recouvrement peut permettre de restaurer la base jusqu'à la dernière transaction validée, juste avant la production de la panne.
Plusieurs cas de figure peuvent alors se présenter selon les types de fichiers perdus et selon la disponibilité souhaitée de la base au moment du recouvrement.

Recouvrement complet

Base de données fermée
-la base ne peut être ouverte en mode normal
-un ou plusieurs fichiers de données du tablespace SYSTEM sont endommagés
-un fichier de données contenant les rollback segments est endommagé

Il faut alors
-réparer ou changer le disque
-restaurer la sauvegarde la plus récente des fichiers endommagés
-se connecter comme INTERNAL
-démarrer une nouvelle instance sans ouvrir la base (STARTUP NOMOUNT)
-renommer et localiser les fichiers de données s'ils ne sont pas restaurés à l'endroit d'origine
-activer tous les fichiers de données avec la commande
ALTER DATABASE DATAFILE nom_fich ONLINE
-soit démarrer le recouvrement de toute la base par la commande RECOVER DATABASE soit démarrer le recouvrement d'un fichier endommagé par RECOVER DATAFILE
-ouvrir la base avec la commande ALTER DATABASE nom_base OPEN

Base de données ouverte, tablespace désactivé
-la base peut être ouverte en mode normal
-les fichiers du tablespace SYSTEM ou contenant les rollback segments ne sont pas endommagés

- des fichiers de données sont endommagés

Il faut alors

-désactiver les tablespaces endommagés

-réparer ou changer le disque

-restaurer les fichiers endommagés

-renommer et localiser les fichiers de données s'ils ne sont pas restaurés à l'endroit d'origine

-lancer le recouvrement des tablespaces endommagés par les commandes RECOVER TABLESPACE ou RECOVER DATAFILE

-le noyau Oracle applique les journaux de reprise archivés et en ligne pour produire l'image avant « Roll Forward »

-les tablespaces endommagés sont restaurés et peuvent être mis en ligne par la commande ALTER TABLESPACE nom_tablespace ONLINE

Fichier de contôle endommagé

Le recouvrement peut être complet si la sauvegerde du fichier de contrôle reflète la structure de la base au moment de la panne.

Recouvrement incomplet

Le recouvrement incomplet peut être utilisé dans les cas suivants:

-annuler des opérations jusqu'à un point donné

-plusieurs groupes de journaux sont endommagés et ne peuvent être appliqués au recouvrement

-recouvrement à un point donné dans le passé

-suite à une perte accidentelle d'une table, pouvoir utiliser un recouvrement à certain moment

-un journal de reprise en ligne peut être indisponible à cause d'une panne système; les entrées journaux qui sont écrites sur les fichiers de données sont valides, les autres ne le sont ps; une partie de ce journal peut être utilisée pour un recouvrement allant à une période T par exemple.

Il faut alors

-fermer la base avec l'option ABORT

-démarrer une nouvelle instance avec l'option MOUNT

-mettre tous les fichiers de données en ligne

ALTER TABLSPACE DATAFILE nom_fich ONLINE

- ancer une des procédures de recouvrement suivantes:

RECOVER DATABASE UNTIL CANCEL

RECOVER DATABASE UNTIL TIME date

RECOVER DATABASE UNTIL CHANGE entier

En cas d'utilisation d'une sauvegarde d'un fichier de contrôle, l'option USING BACKUP CONTROLFILE doît être spécifiée.

Type de fichier perdu				Mode d'Archivage	
Data	**Redolog on line**	**Archive**	**Control file**	**ARCHIVELOG**	**NOARCHIVELOG**
X				Effectuer une restauration du fichier	Réappliquer un Backup à froid
	X			Reconstruire les fichiers perdus suivant le type de configuration	
		X		Refaire un backup de la base	Sans incidence
			X	Utiliser un fichier de contrôle miroir	
X		X		Restauration incomplète	Réappliquer un backup à froid

ANNEXE A: LE DICTIONNAIRE DE DONNEES

Informations sur tous les objets de la base(privilège SELECT ANY TABLE)

DBA_2PC_NEIGHBORS : information about incoming and outgoing connections for pending transactions
DBA_2PC_PENDING : info about distributed transactions awaitingrecovery
DBA_ANALYZE_OBJECTS :
DBA_AUDIT_EXISTS : Lists audit trail entries produced by AUDIT NOT EXISTS and AUDIT EXISTS
DBA_AUDIT_OBJECT : Audit trail records for statements concerning objects, specifically: table, cluster, view, index, sequence,[public] database link, [public]synonym, procedure, trigger, rollback segment, tablespace, role,user
DBA_AUDIT_SESSION : All audit trail records concerning CONNECT and DISCONNECT
DBA_AUDIT_STATEMENT : Audit trail records concerning grant, revoke, audit, noaudit and alter system
DBA_AUDIT_TRAIL : All audit trail entries
DBA_CATALOG : All database Tables, Views, Synonyms, Sequences
DBA_CLUSTERS : Description of all clusters in the database
DBA_CLUSTER_HASH_EXPRESSIONS : Hash functions for all clusters
DBA_CLU_COLUMNS : Mapping of table columns to cluster columns
DBA_COL_COMMENTS : Comments on columns of all tables and views
DBA_COL_PRIVS : All grants on columns in the database
DBA_CONSTRAINTS : Constraint definitions on all tables
DBA_CONS_COLUMNS : Information about accessible columns in constraint definitions
DBA_DATA_FILES : Information about database files
DBA_DB_LINKS : All database links in the database
DBA_DEPENDENCIES : Dependencies to and from objects
DBA_ERRORS : Current errors on all stored objects in the database
DBA_EXP_FILES : Description of export files
DBA_EXP_OBJECTS : Objects that have been incrementally exported
DBA_EXP_VERSION : Version number of the last export session
DBA_EXTENTS : Extents comprising all segments in the database
DBA_FREE_SPACE : Free extents in all tablespaces
DBA_FREE_SPACE_COALESCED : Statistics on Coalesced Space in Tablespaces
DBA_FREE_SPACE_COALESCED_TMP1 : Coalesced Free Extents for all Tablespaces
DBA_FREE_SPACE_COALESCED_TMP2 : Free Extents in Tablespaces
DBA_HISTOGRAMS : Histograms on columns of all tables
DBA_INDEXES : Description for all indexes in the database
DBA_IND_COLUMNS : COLUMNs comprising INDEXes on all TABLEs and CLUSTERs
DBA_JOBS : All jobs in the database
DBA_JOBS_RUNNING : All jobs in the database which are currently running, join v$lock and job$
DBA_OBJECTS : All objects in the database
DBA_OBJECT_SIZE : Sizes, in bytes, of various pl/sql objects
DBA_OBJ_AUDIT_OPTS : Auditing options for all tables and views
DBA_PRIV_AUDIT_OPTS : Current system privileges being audited across the system and by user

DBA_PROFILES : Display all profiles and their limits
DBA_RCHILD : All the children in any refresh group. This view is not a join.
DBA_REFRESH : All the refresh groups
DBA_REFRESH_CHILDREN : All the objects in refresh groups
DBA_RGROUP : All refresh groups. This view is not a join.
DBA_ROLES : All Roles which exist in the database
DBA_ROLE_PRIVS : Roles granted to users and roles
DBA_ROLLBACK_SEGS : Description of rollback segments
DBA_SEGMENTS : Storage allocated for all database segments
DBA_SEQUENCES : Description of all SEQUENCEs in the database
DBA_SNAPSHOTS : All snapshots in the database
DBA_SNAPSHOT_LOGS : All snapshot logs in the database
DBA_SOURCE : Source of all stored objects in the database
DBA_STMT_AUDIT_OPTS : Current system auditing options acrossthe system and by user
DBA_SYNONYMS : All synonyms in the database
DBA_SYS_PRIVS : System privileges granted to users and roles
DBA_TABLES : Description of all tables in the database
DBA_TABLESPACES : Description of all tablespaces
DBA_TAB_COLUMNS : Columns of user's tables, views and clusters
DBA_TAB_COMMENTS : Comments on all tables and views in the database
DBA_TAB_PRIVS : All grants on objects in the database
DBA_TRIGGERS : All triggers in the database
DBA_TRIGGER_COLS : Column usage in all triggers
DBA_TS_QUOTAS : Tablespace quotas for all users
DBA_UPDATABLE_COLUMNS : Description of dba updatable columns
DBA_USERS : Information about all users of the database
DBA_VIEWS : Text of all views in the database

Informations sur tous les objets dont l'utilisateur est propriétaire

USER_ARGUMENTS : Arguments in object accessible to the user
USER_AUDIT_OBJECT : Audit trail records for statements concerning objects, specifically: table, cluster, view, index, sequence, [public] database link, [public] synonym, procedure, trigger, rollback segment, tablespace, role, user
USER_AUDIT_SESSION : All audit trail records concerning CONNECT and DISCONNECT
USER_AUDIT_STATEMENT : Audit trail records concerning grant, revoke, audit, no audit and alter system
USER_AUDIT_TRAIL : Audit trail entries relevant to the user
USER_CATALOG : Tables, Views, Synonyms and Sequences owned by the user
USER_CLUSTERS : Descriptions of user's own clusters
USER_CLUSTER_HASH_EXPRESSIONS : Hash functions for the user's hash clusters
USER_CLU_COLUMNS : Mapping of table columns to cluster columns
USER_COL_COMMENTS : Comments on columns of user's tables and views
USER_COL_PRIVS : Grants on columns for which the user is the owner, grantor or grantee
USER_COL_PRIVS_MADE : All grants on columns of objects owned by the user
USER_COL_PRIVS_RECD : Grants on columns for which the user is the grantee
USER_CONSTRAINTS : Constraint definitions on user's own tables
USER_CONS_COLUMNS : Information about accessible columns in constraint definitions

USER_DB_LINKS : Database links owned by the user
USER_DEPENDENCIES : Dependencies to and from a users objects
USER_ERRORS : Current errors on stored objects owned by the user
USER_EXTENTS : Extents comprising segments owned by the user
USER_FREE_SPACE : Free extents in tablespaces accessible to the user
USER_HISTOGRAMS : Histograms on columns of user's tables
USER_INDEXES : Description of the user's own indexes
USER_IND_COLUMNS : COLUMNs comprising user's INDEXes or on user's TABLES
USER_JOBS : All jobs owned by this user
USER_OBJECTS : Objects owned by the user
USER_OBJECT_SIZE : Sizes, in bytes, of various pl/sql objects
USER_OBJ_AUDIT_OPTS : Auditing options for user's own tables and views
USER_REFRESH : All the refresh groups
USER_REFRESH_CHILDREN : All the objects in refresh groups, where the user owns the refresh group
USER_RESOURCE_LIMITS : Display resource limit of the user
USER_ROLE_PRIVS : Roles granted to current user
USER_SEGMENTS : Storage allocated for all database segments
USER_SEQUENCES : Description of the user's own SEQUENCEs
USER_SNAPSHOTS : Snapshots the user can look at
USER_SNAPSHOT_LOGS : All snapshot logs owned by the user
USER_SOURCE : Source of stored objects accessible to the user
USER_SYNONYMS : The user's private synonyms
USER_SYS_PRIVS : System privileges granted to current user
USER_TABLES : Description of the user's own tables
USER_TABLESPACES : Description of accessible tablespaces
USER_TAB_COLUMNS : Columns of user's tables, views and clusters
USER_TAB_COMMENTS : Comments on the tables and views owned by the user
USER_TAB_PRIVS : Grants on objects for which the user is the owner, grantor or grantee
USER_TAB_PRIVS_MADE : All grants on objects owned by the user
USER_TAB_PRIVS_RECD : Grants on objects for which the user is the grantee
USER_TRIGGERS : Triggers owned by the user
USER_TRIGGER_COLS : Column usage in user's triggers
USER_TS_QUOTAS : Tablespace quotas for the user
USER_UPDATABLE_COLUMNS : Description of updatable columns
USER_USERS : Information about the current user
USER_VIEWS : Text of views owned by the user

Informations sur tous les objets accessibles par l'utilisateur connecté

ALL_ARGUMENTS : Arguments in object accessible to the user
ALL_CATALOG : All tables, views, synonyms, sequences accessible to the user
ALL_CLUSTERS : Description of clusters accessible to the user
ALL_CLUSTER_HASH_EXPRESSIONS : Hash functions for all accessible clusters
ALL_COL_COMMENTS : Comments on columns of accessible tables and views
ALL_COL_PRIVS : Grants on columns for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
ALL_COL_PRIVS_MADE : Grants on columns for which the user is owner or grantor

ALL_COL_PRIVS_RECD : Grants on columns for which the user, PUBLIC or enabled role is the grantee
ALL_CONSTRAINTS : Constraint definitions on accessible tables
ALL_CONS_COLUMNS : Information about accessible columns in constraint definitions
ALL_DB_LINKS : Database links accessible to the user
ALL_DEF_AUDIT_OPTS : Auditing options for newly created objects
ALL_DEPENDENCIES : Dependencies to and from objects accessible to the user
ALL_ERRORS : Current errors on stored objects that user is allowed to create
ALL_HISTOGRAMS : Histograms on columns of all tables visible to user
ALL_INDEXES : Descriptions of indexes on tables accessible to the user
ALL_IND_COLUMNS : COLUMNs comprising INDEXes on accessible TABLES
ALL_JOBS : Synonym for USER_JOBS
ALL_OBJECTS : Objects accessible to the user
ALL_REFRESH : All the refresh groups that the user can touch
ALL_REFRESH_CHILDREN : All the objects in refresh groups, where the user can touch the group
ALL_SEQUENCES : Description of SEQUENCEs accessible to the user
ALL_SNAPSHOTS : Snapshots the user can look at
ALL_SOURCE : Current source on stored objects that user is allowed to create
ALL_SYNONYMS : All synonyms accessible to the user
ALL_TABLES : Description of tables accessible to the user
ALL_TAB_COLUMNS : Columns of user's tables, views and clusters
ALL_TAB_COMMENTS : Comments on tables and views accessible to the user
ALL_TAB_PRIVS : Grants on objects for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
ALL_TAB_PRIVS_MADE : User's grants and grants on user's objects
ALL_TAB_PRIVS_RECD : Grants on objects for which the user, PUBLIC or enabled role is the grantee
ALL_TRIGGERS : Triggers accessible to the current user
ALL_TRIGGER_COLS : Column usage in user's triggers or in triggers on user's tables
ALL_UPDATABLE_COLUMNS : Description of all updatable columns
ALL_USERS : Information about all users of the database
ALL_VIEWS : Text of views accessible to the user

Informations diverses
AUDIT_ACTIONS : Description table for audit trail action type codes.Maps action type numbers to action type names
CAT : Synonym for USER_CATALOG
CLU : Synonym for USER_CLUSTERS
COLS : Synonym for USER_TAB_COLUMNS
COLUMN_PRIVILEGES : Grants on columns for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
DICT : Synonym for DICTIONARY
DICTIONARY : Description of data dictionary tables and views
DICT_COLUMNS : Description of columns in data dictionary tables and views
DUAL :
GLOBAL_NAME : global database name
IND : Synonym for USER_INDEXES

INDEX_HISTOGRAM : statistics on keys with repeat count
INDEX_STATS : statistics on the b-tree
NLS_DATABASE_PARAMETERS : Permanent NLS parameters of the database
NLS_INSTANCE_PARAMETERS : NLS parameters of the instance
NLS_SESSION_PARAMETERS : NLS parameters of the user session
OBJ : Synonym for USER_OBJECTS
RESOURCE_COST : Cost for each resource
ROLE_ROLE_PRIVS : Roles which are granted to roles
ROLE_SYS_PRIVS : System privileges granted to roles
ROLE_TAB_PRIVS : Table privileges granted to roles
SEQ : Synonym for USER_SEQUENCES
SESSION_PRIVS : Privileges which the user currently has set
SESSION_ROLES : Roles which the user currently has enabled.
SM$VERSION : Synonym for SM_$VERSION
SYN : Synonym for USER_SYNONYMS
TABLE_PRIVILEGES : Grants on objects for which the user is the grantor, grantee ,owner, or an enabled role or PUBLIC is the grantee
TABS : Synonym for USER_TABLES

Informations sur dynamic performance tables

V$ACCESS : Objects that are currently locked and the sessions that are accessing them
V$ACTIVE_INSTANCES : Instances that have the database currently mounted
V$ARCHIVE : Information on archive logs for each thread in the database system
V$BACKUP : Backup status of all online datafiles
V$BGPROCESS : Background processes
V$CIRCUIT : Information about virtual circuits
V$COMPATIBILITY : Features in use by the database instance that may prevent downgrading to a previous release
V$COMPATSEG : Permanent features in use by the database that will prevent moving back to a earlier release
V$CONTROLFILE : Names of controlfiles
V$DATABASE : Database information from the control file
V$DATAFILE : Datafile information from the control file
V$DBFILE : Datafiles making up the database
V$DBLINK : All open database links
V$DB_OBJECT_CACHE : Database objects cached in the library cache
V$DB_PIPES : Pipes currently in the database
V$DISPATCHER : Information on the dispatcher processes
V$ENABLEDPRIVS : Privileges enabled
V$EVENT_NAME : Information about wait events
V$EXECUTION :
V$FILESTAT : Information about file read/write statistics
V$FIXED_TABLE : Dynamic performances tables, views and derived tables
V$FIXED_VIEW_DEFINITION : Definition of all fixed views (views beginning with v$)
V$INDEXED_FIXED_COLUMN : Columns in dynamic performances tables that are indexed
V$INSTANCE : State of the current instance
V$LATCH : Statistics for latches
V$LATCHHOLDER : Information about the current latch holders

V$LATCHNAME : information about decoded latch names
V$LATCH_CHILDREN : Statistics about child latches
V$LATCH_MISSES : Statistics about missed attemps to acquire a latch
V$LATCH_PARENT : Statistics about the parent latch
V$LIBRARYCACHE : Statistics about library cache performance and activity
V$LICENSE : Information about licence limits
V$LOADCSTAT : SQL*LOADER Statistics during the execution of a direct load
V$LOADTSTAT : SQL*LOADER Statistics during the execution of a direct load apply to the current table
V$LOCK : Locks currently held by Oracle7
V$LOCKED_OBJECT : Locks acquired by every transaction on the system
V$LOG : Log information from the control file
V$LOGFILE : information about redo log files
V$LOGHIST : Log history information from the control file
V$LOG_HISTORY : Archived log names for all logs in the log history
V$MLS_PARAMETERS : Trusted Oracle7 Server-specific initialization parameters
V$MTS : Information for tuning the multi-threaded server
V$MYSTAT : Statistics on the current session
V$NLS_PARAMETERS : Current values of NLS p arameters
V$NLS_VALID_VALUES : Valid values for NLS parameters
V$OBJECT_DEPENDENCY : Objects depended on by a package, procedure or cursor currently loaded in the shared pool
V$OPEN_CURSOR : Cursors that each user session currently has opened and parsed
V$OPTION :Options installed with the Oracle7 Server
V$PARAMETER : Information about initialization parameters
V$PQ_SESSTAT : Session statistics for parallel queries
V$PQ_SLAVE : Statistics for each of the active parallele query servers on an instance
V$PQ_SYSSTAT : System statistics for parallel queries
V$PQ_TQSTAT : Statistics on parallel query operations
V$PROCESS : Information about the currently active processes
V$PWFILE_USERS : Users who have been granted SYSDBA and SYSOPER privileges
V$QUEUE : Information on the multi-thread message queues
V$RECOVERY_LOG : Information about archived logs that are needed to complete recovery
V$RECOVER_FILE : Status of files needing media recovery
V$RECOVERY_FILE_STATUS : One row for each datafile for each RECOVER command
V$RECOVERY_STATUS : Statistics of the current recovery process
V$REQDIST : Statistics for the histogram of MTS>dispatcher request times
V$RESOURCE : Information about resources
V$ROLLNAME : Names of online rollback segments
V$ROLLSTAT : Rollback segment statistics
V$ROWCACHE : Statistics for dictionary activity
V$SESSION : Information for each current session
V$SESSION_CONNECT_INFO : Information about network connections for the current session
V$SESSION_CURSOR_CACHE : Information on cursor usage for the current session
V$SESSION_EVENT : Information on waits for an event by a session
V$SESSION_WAIT : resources or events for which active sessions are waiting
V$SESSTAT : User session statistics
V$SESS_IO : I/O statistics for each user session
V$SGA : Summary information on the System Global Area

V$SGASTAT : Detailed information on the System Global Area
V$SHARED_POOL_RESERVED : Statistics on reserved pool
V$SHARED_SERVER : Information on the shared server processes
V$SORT_SEGMENT : Information about every sort segment in a given instance
V$SQL : Statistics on shared SQL area
V$SQLAREA : Statistics on shared SQL area
V$SQLTEXT : Text of SQL statements
V$SQLTEXT_WITH_NEWLINES : Text of SQL statements
V$SQL_BIND_DATA :
V$SQL_BIND_METADATA :
V$SQL_CURSOR :
V$SQL_SHARED_MEMORY :
V$STATNAME : Decoded statistics names for the statistics in the V$SESSTAT table
V$SYSSTAT : System statistics
V$SYSTEM_CURSOR_CACHE : Information on cursor usage for the current session
V$SYSTEM_EVENT : Information on total waits for an event
V$SYSTEM_PARAMETER : Information on system parameters
V$THREAD : Thread information from the control file
V$TIMER : Elapsed time in hundredths seconds
V$TRANSACTION : Active transactions i the system
V$TYPE_SIZE : Sizes of various database components for use in estimating data block capacity
V$VERSION : Version numbers of core library components in the Oracle Server
V$WAITSTAT : Block contention statistics

ANNEXE B: LES PRIVILEGES SYSTEME

ALTER ANY CLUSTER
ALTER ANY INDEX
ALTER ANY PROCEDURE
ALTER ANY ROLE
ALTER ANY SEQUENCE
ALTER ANY SNAPSHOT
ALTER ANY TABLE
ALTER ANY TRIGGER
ALTER DATABASE
ALTER PROFILE
ALTER RESOURCE
ALTER ROLLBACK SEGMENT
ALTER SESSION
ALTER SNAPSHOT
ALTER SYSTEM
ALTER TABLESPACE
ALTER USER
ANALYZE ANY
AUDIT ANY
AUDIT SYSTEM
BACKUP ANY TABLE
BECOME USER
COMMENT ANY TABLE
CREATE ANY CLUSTER
CREATE ANY INDEX
CREATE ANY PROCEDURE
CREATE ANY SEQUENCE
CREATE ANY SNAPSHOT
CREATE ANY SYNONYM
CREATE ANY TABLE
CREATE ANY TRIGGER
CREATE ANY VIEW
CREATE CLUSTER
CREATE DATABASE LINK
CREATE INDEX
CREATE PROCEDURE
CREATE PROFILE
CREATE PUBLIC DATABASE LINK
CREATE PUBLIC SYNONYM
CREATE ROLE
CREATE ROLLBACK SEGMENT
CREATE SEQUENCE
CREATE SESSION
CREATE SNAPSHOT
CREATE SYNONYM
CREATE TABLE
CREATE TABLESPACE
CREATE TRIGGER
CREATE USER
CREATE VIEW
DELETE ANY TABLE
DROP ANY CLUSTER
DROP ANY INDEX
DROP ANY PROCEDURE
DROP ANY ROLE
DROP ANY SEQUENCE
DROP ANY SNAPSHOT
DROP ANY SYNONYM
DROP ANY TABLE
DROP ANY TRIGGER
DROP ANY VIEW
DROP PROFILE
DROP PUBLIC
DROP DATABASE LINK
DROP SNAPSHOT
DROP TABLESPACE
DROP USER
EXECUTE ANY PROCEDURE
FORCE ANY TRANSACTION
FORCE TRANSACTION
GRANT ANY PRIVILEGE
GRANT ANY PROCEDURE
GRANT ANY ROLE
GRANT ANY SEQUENCE
GRANT ANY TABLE
GRANT ANY VIEW
INSERT ANY TABLE
LOCK ANY TABLE
MANAGE TABLESPACE
READUP
RESTRICTED SESSION
SELECT ANY SEQUENCE
SELECT ANY TABLE
UNLIMITED TABLESPACE
UPDATE ANY TABLE
WRITEDOWN
WRITEUP

ANNEXE C: LES PARAMETRES D'ENVIRONNEMENT

always_anti_join	Always use this anti-join when possible
async_read	Enable Async Read
async_write	Enable DBWR Async Write
audit_file_dest	Directory in which auditing files are to reside
audit_trail	Enable system auditing
background_core_dump	Core Size for Background Processes
background_dump_dest	Detached process dump directory
bitmap_merge_area_size	Maximum memory allow for BITMAP MERGE
blank_trimming	Blank trimming semantics parameter
cache_size_threshold	Maximum size of table or piece to be cached (in blocks)
ccf_io_size	Number of bytes per write when creating contiguous file
checkpoint_process	Create a seperate checkpoint process
cleanup_rollback_entries	No. of undo entries to apply per transaction cleanup
close_cached_open_cursors	Close cursors cached by PL/SQL at each commit
commit_point_strength	Bias this node has toward not preparing in a two-phase commit
compatible	Database will be completely compatible with this software version
compatible_no_recovery	Database will be compatible unless crash or media recovery is ne
control_files	Control file names list
core_dump_dest	Core dump directory
cpu_count	Number of cpu's for this instance
create_bitmap_area_size	Size of create bitmap buffer for bitmap index
cursor_space_for_time	Use more memory in order to get faster execution
db_block_buffers	Number of database blocks cached in memory
db_block_checkpoint_batch	Max number of blocks to checkpoint in a DB Writer IO
db_block_checksum	Store checksum in db blocks and check during reads
db_block_lru_extended_statistics	Maintain buffer cache LRU statistics for last N blocks discarded
db_block_lru_latches	Number of lru latches
db_block_lru_statistics	Maintain buffer cache LRU hits-by-position statistics (slow)
db_block_size	Size of database block in bytes
db_domain	Directory part of global database name stored with CREATE DATABA SE
db_file_multiblock_read_count	Db block to be read each IO
db_file_simultaneous_writes	Max simultaneous (overlaped) writes per db file
db_file_standby_name_convert	Datafile name convert pattern and string for standby database
db_files	Max allowable # db files
db_name	Database name specified in CREATE DATABASE
db_writers	Number of database writer processes
dblink_encrypt_login	Enforce password for distributed login always be encrypted
delayed_logging_block_cleanouts	Turn on delayed-logging block cleanouts feature
discrete_transactions_enabled	Enable OLTP mode
distributed_lock_timeout	Number of seconds a distributed transaction waits for a lock

distributed_recovery_connection_hold_time	Number of seconds RECO holds outbound connections open
distributed_transactions	Max. number of concurrent distributed transactions
dml_locks	Dml locks - one for each table modified in a transaction
enqueue_resources	Resources for enqueues
event	Debug event control - default null string
fixed_date	Fixed SYSDATE value
gc_db_locks	# DB locks (DFS)
gc_files_to_locks	Mapping between file numbers and hash buckets
gc_freelist_groups	# freelist groups locks in (DFS)
gc_lck_procs	Number of background parallel server lock processes to start
gc_releasable_locks	# releasable DB locks (DFS)
gc_rollback_locks	# Undo locks in (DFS)
gc_rollback_segments	# Undo Segments
gc_save_rollback_locks	# Save Undo locks in (DFS)
gc_segments	# Segment headers
gc_tablespaces	# tablespaces
global_names	Enforce that database links have same name as remote database
hash_area_size	Size of in-memory hash work area
hash_join_enabled	Enable/disable hash join
hash_multiblock_io_count	Number of blocks hash join will read/write at once
ifile	Include file in init.ora
instance_number	Instance number
job_queue_interval	Wakeup interval in seconds for job queue processes
job_queue_keep_connections	Keep network connections between execution of jobs
job_queue_processes	Number of job queue processes to start
license_max_sessions	Maximum number of non-system user sessions allowed
license_max_users	Maximum number of named users that can be created in the database
license_sessions_warning	Warning level for number of non-system user sessions
log_archive_buffer_size	Size of each archival buffer in log file blocks
log_archive_buffers	Number of buffers to allocate for archiving
log_archive_dest	Archival destination text string
log_archive_format	Archival destination format
log_archive_start	Start archival process on SGA initialization
log_block_checksum	Calculate checksum for redo blocks when writing
log_buffer	Redo circular buffer size
log_checkpoint_interval	# redo blocks checkpoint threshold
log_checkpoint_timeout	Maximum time interval between checkpoints in seconds
log_checkpoints_to_alert	Log checkpoint begin/end to alert file
log_file_standby_name_convert	Logfile name convert pattern and string for standby database
log_files	Max allowable # log files
log_simultaneous_copies	Number of simultaneous copies into redo buffer(# of copy latches)
log_small_entry_max_size	Redo entries larger than this will acquire the redo copy latch
max_commit_propagation_delay	Max age of new snapshot in .01 seconds
max_dump_file_size	Maximum size (blocks) of dump file
max_enabled_roles	Max number of roles a user can have enabled
max_rollback_segments	Max. number of rollback segments in SGA cache
max_transaction_branches	Max. number of branches per distributed transaction

mts_dispatchers	Specifications of dispatchers
mts_listener_address	Address(es) of network listener
mts_max_dispatchers	Max number of dispatchers
mts_max_servers	Max number of servers
mts_multiple_listeners	Are multiple listeners enabled?
mts_servers	Number of servers to start up
mts_service	Service supported by dispatchers
nls_currency	NLS local currency symbol
nls_date_format	NLS Oracle date format
nls_date_language	NLS date language name
nls_iso_currency	NLS ISO currency territory name
nls_language	NLS language name
nls_numeric_characters	NLS numeric characters
nls_sort	NLS linguistic definition name
nls_territory	NLS territory name
open_cursors	Max # cursors per process
open_links	Max # open links per process
optimizer_mode	Optimizer mode
optimizer_percent_parallel	Optimizer percent parallel
oracle_trace_collection_name	Oracle TRACE default collection name
oracle_trace_collection_path	Oracle TRACE collection path
oracle_trace_collection_size	Oracle TRACE collection file max. size
oracle_trace_enable	Oracle TRACE instance wide enable/disable
oracle_trace_facility_name	Oracle TRACE default facility name
oracle_trace_facility_path	Oracle TRACE facility path
os_authent_prefix	Prefix for auto-logon accounts
os_roles	Retrieve roles from the operating system
parallel_default_max_instances	Default maximum number of instances for parallel query
parallel_max_servers	Maximum parallel query servers per instance
parallel_min_percent	Minimum percent of threads required for parallel query
parallel_min_servers	Minimum parallel query servers per instance
parallel_server_idle_time	Idle time before parallel query server dies
partition_view_enabled	Enable/disable partitioned views
post_wait_device	Name of post-wait device
pre_page_sga	Pre-page sga for process
processes	User processes
recovery_parallelism	Number of server processes to use for parallel recovery
reduce_alarm	Reduce Alarm
remote_dependencies_mode	Remote-procedure-call dependencies mode parameter
remote_login_passwordfile	Password file usage parameter
remote_os_authent	Allow non-secure remote clients to use auto-logon accounts
remote_os_roles	Allow non-secure remote clients to use os roles
resource_limit	Master switch for resource limit
rollback_segments	Undo segment list
row_cache_cursors	Number of cached cursors for row cache management
row_locking	Row-locking
sequence_cache_entries	Number of sequence cache entries
sequence_cache_hash_buckets	Number of sequence cache hash buckets
serializable	Serializable
session_cached_cursors	Number of cursors to save in the session cursor cache

sessions	User and system sessions
shadow_core_dump	Core Size for Shadow Processes
shared_pool_reserved_min_alloc	Minimum allocation size in bytes for reserved area of shared pool
shared_pool_reserved_size	Size in bytes of reserved area of shared pool
shared_pool_size	Size in bytes of shared pool
snapshot_refresh_interval	Wakeup interval in seconds for job queue processes
snapshot_refresh_keep_connections	Keep network connections between execution of jobs
snapshot_refresh_processes	Number of job queue processes to start
sort_area_retained_size	Size of in-memory sort work area retained between fetch calls
sort_area_size	Size of in-memory sort work area
sort_direct_writes	Use direct write
sort_read_fac	Multi-block read factor for sort
sort_spacemap_size	Size of sort disk area space map
sort_write_buffer_size	Size of each sort direct write buffer
sort_write_buffers	Number of sort direct write buffers
spin_count	Number of times to spin on latch miss
sql92_security	Require select privilege for searched update/delete
sql_trace	Enable SQL trace
temporary_table_locks	Temporary table locks
text_enable	Enable text searching
thread	Redo thread to mount
timed_statistics	Maintain internal timing statistics
transactions	Max. number of concurrent active transactions
transactions_per_rollback_segment	Number of active transactions per rollback segment
use_ism	Enable Shared Page Table - ISM
use_post_wait_driver	Use post-wait driver
use_readv	Use readv for multi-block read
user_dump_dest	User process dump directory
utl_file_dir	Utl_file accessible directories list

ANNEXE D: SCRIPTS DE CREATION D'UNE BASE

```
# FICHIER initGEN.ora
#
# $Header: init.ora 7.14 94/04/07 11:39:16 nsingh Osd<unix> $ init.ora Copyr (c) 1991
Oracle
#
##############################################################################
###
# Example INIT.ORA file
#
# This file is provided by Oracle Corporation to help you customize
# your RDBMS installation for your site.  Important system parameters
# are discussed, and example settings given.
#
# Some parameter settings are generic to any size installation.
# For parameters that require different values in different size
# installations, three scenarios have been provided: SMALL, MEDIUM
# and LARGE.  Any parameter that needs to be tuned according to
# installation size will have three settings, each one commented
# according to installation size.
#
# Use the following table to approximate the SGA size needed for the
# three scenarious provided in this file:
#
#                -------Installation/Database Size------
#                 SMALL         MEDIUM         LARGE
#  Block        2K    4500K          6800K         17000K
#  Size         4K    5500K          8800K         21000K
#
# To set up a database that multiple instances will be using, place
# all instance-specific parameters in one file, and then have all
# of these files point to a master file using the IFILE command.
# This way, when you change a public
# parameter, it will automatically change on all instances.  This is
# necessary, since all instances must run with the same value for many
# parameters. For example, if you choose to use private rollback segments,
# these must be specified in different files, but since all gc_*
# parameters must be the same on all instances, they should be in one file.
#
# INSTRUCTIONS: Edit this file and the other INIT files it calls for
# your site, either by using the values provided here or by providing
# your own.  Then place an IFILE= line into each instance-specific
# INIT file that points at this file.
```

```
###############################################################################
####

#include database configuration parameters
ifile = /home/oracle/data/GENERIC/scripts/cnfgGEN.ora

#rollback_segments = (rs_GEN_r01, rs_GEN_r02, rs_GEN_r03, rs_GEN_r04)
db_name = GEN
db_files = 100

db_file_multiblock_read_count = 8                    # SMALL
# db_file_multiblock_read_count = 16                 # MEDIUM
# db_file_multiblock_read_count = 32                 # LARGE

db_block_buffers = 200                               # SMALL
# db_block_buffers = 550                             # MEDIUM
# db_block_buffers = 3200                            # LARGE

shared_pool_size = 3500000                           # SMAL#L
# shared_pool_size = 6000000                         # MEDIUM
# shared_pool_size = 9000000                         # LARGE
# shared_pool_size = 12000000                        # LARGE
#shared_pool_size = 18000000                         # VERY LARGE

log_checkpoint_interval = 10000

processes = 50                                       # SMALL
# processes = 100                                    # MEDIUM
# processes = 200                                    # LARGE

dml_locks = 100                                      # SMALL
# dml_locks = 200                                    # MEDIUM
# dml_locks = 500                                    # LARGE

log_buffer = 8192                                    # SMALL
# log_buffer = 32768                                 # MEDIUM
# log_buffer = 163840                                # LARGE

sequence_cache_entries = 10                          # SMALL
# sequence_cache_entries = 30                        # MEDIUM
# sequence_cache_entries = 100                       # LARGE

sequence_cache_hash_buckets = 10                     # SMALL
# sequence_cache_hash_buckets = 23                   # MEDIUM
# sequence_cache_hash_buckets = 89                   # LARGE

# audit_trail = DB                                   # if you want auditing
# timed_statistics = true                            # if you want timed statistics
max_dump_file_size = 10240                           # limit trace file size to 5 Meg each
```

```
log_archive_start = true      # if you want automatic archiving
mts_dispatchers = "ipc, 1"
#mts_dispatchers = "tcp, 1"
#mts_max_dispatchers = 10
mts_servers =2
mts_max_servers = 10
mts_service = GEN
mts_listener_address = "(ADDRESS=(PROTOCOL=ipc) (KEY=GEN))"
#mts_listener_address =
"(ADDRESS=(PROTOCOL=TCP)(PORT=1525)(HOST=nom_machine))"
OPEN_CURSORS=300
GLOBAL_NAMES=FALSE
max_enabled_roles=40
```

 # FICHIER cnfgGEN.ora

```
#
# $Header: cnfg.orc 7001200.1 93/01/28 11:00:21 pku Osd<unix> $ Copyr (c) 1992 Oracle
#
# cnfg.ora - instance configuration parameters
# Parametres de configurations d'une instance
#

control_files   = (/home/oracle/data/GENERIC/cntrl1GEN.ctl,
                   /home/oracle/data/GENERIC/cntrl2GEN.ctl,
                   /home/oracle/data/GENERIC/cntrl3GEN.ctl)
# Below for possible future use...
#init_sql_files                 = (?/dbs/sql.bsq,
#                                  ?/rdbms/admin/catalog.sql,
#                                  ?/rdbms/admin/expvew.sql)
background_dump_dest                = /home/oracle/data/GENERIC/bg
user_dump_dest                      = /home/oracle/data/GENERIC/user
core_dump_dest                      = /home/oracle/data/GENERIC/core
#log_archive_dest               = /home/oracle/data/GENERIC/arch/arch.log
#db_block_size                      = <blocksize>

db_name                             = GEN
```

#Création d'une base de données: FICHIER CretdbGEN.sql

```
REM *
REM * Set terminal output and command echoing on; log output of this script.
REM *
#set termout on
#set echo on
spool GEN.lst

REM * Start the <GEN> instance (ORACLE_GEN here must be set to <GEN>).
```

```
REM *
connect internal
startup nomount pfile=/home/oracle/data/GENERIC/scripts/initGEN.ora

REM * Create the <dbname> database.
REM * SYSTEM tablespace configuration guidelines:
REM *   General-Purpose ORACLE RDBMS                    5Mb
REM *   Additional dictionary for applications      10-50Mb
REM * Redo Log File configuration guidelines:
REM *   Use 3+ redo log files to relieve ``cannot allocate new log..." waits.
REM *   Use ~100Kb per redo log file per connection to reduce checkpoints.
REM *
create database GEN
   maxinstances 1
   maxlogfiles  16
   maxdatafiles 100
   character set "US7ASCII"
   datafile
         '/home/oracle/data/GENERIC/systGEN.dbf' size   10M
   logfile
         group 1('/home/oracle/data/GENERIC/log1aGEN.dbf',
               '/home/oracle/data/GENERIC/log2aGEN.dbf') size 500k,
         group 2('/home/oracle/data/GENERIC/log1bGEN.dbf',
                  '/home/oracle/data/GENERIC/log2bGEN.dbf') size 500k,
         group 3('/home/oracle/data/GENERIC/log1cGEN.dbf',
                  '/home/oracle/data/GENERIC/log2cGEN.dbf') size 500k;


REM # install data dictionary views:
@/net4/oracle/rdbms/admin/catalog.sql

shutdown immediate
disconnect

connect internal
# SQLDBA bug: if GEN is numeric, leads to syntax error.
startup open GEN  pfile=/home/oracle/data/GENERIC/scripts/initGEN.ora
create rollback segment rs_GEN_r0 tablespace system
storage (initial 16k next 16k minextents 2 maxextents 10);
alter rollback segment rs_GEN_r0 online ;

REM * Create a tablespace for rollback segments.
REM * Rollback segment configuration guidelines:
REM *   1 rollback segments for every 4 concurrent xactions.
REM *   No more than 50 rollback segments.
REM *   All rollback segments the same size.
REM *   Between 2 and 4 homogeneously-sized extents per rollback segment.
REM *
create tablespace ts_GEN_rbs datafile
         '/home/oracle/data/GENERIC/rbsGEN.dbf'  size   2M
```

```
default storage (
        initial         128k
        next            128k
        pctincrease      0
        minextents       2
);

REM * Create a tablespace for temporary segments.
REM * Temporary tablespace configuration guidelines:
REM *   Initial and next extent sizes = k * SORT_AREA_SIZE, k in {1,2,3,...}.

create tablespace ts_GEN_temp datafile
     '/home/oracle/data/GENERIC/tempGEN.dbf'       size   550k
default storage (
     initial     256k
     next        256k
     pctincrease  0
);

alter tablespace ts_GEN_temp add datafile
     '/home/oracle/data/GENERIC/tempGEN2.dbf'       size   10M;


REM * Create a tablespace for database tools.
REM *
create tablespace tools datafile
        '/home/oracle/data/GENERIC/toolGEN.dbf' size   4M;

alter tablespace tools add datafile
        '/home/oracle/data/GENERIC/toolGEN2.dbf'         size   10M;

REM * Create a tablespace for miscellaneous database user activity.
REM *
create tablespace ts_GEN_users datafile
        '/home/oracle/data/GENERIC/usrGEN.dbf'  size   5M;

REM * Create rollback segments.
REM *
create rollback segment rs_GEN_r01 tablespace ts_GEN_rbs;
create rollback segment rs_GEN_r02 tablespace ts_GEN_rbs;
create rollback segment rs_GEN_r03 tablespace ts_GEN_rbs;
create rollback segment rs_GEN_r04 tablespace ts_GEN_rbs;

REM * Restart the instance to activate the the additional rollback segments.
REM *
shutdown immediate
disconnect

connect internal
# SQLDBA bug: if GEN is numeric, leads to syntax error.
```

```
startup open "GEN" pfile=/home/oracle/data/GENERIC/scripts/initGEN.ora
drop rollback segment rs_GEN_r0;
alter user sys temporary tablespace ts_GEN_temp;
#revoke resource from system;
#revoke resource on system from system;
#grant resource on tools to system;
alter user system default tablespace tools temporary tablespace ts_GEN_temp;

REM * For each DBA user, run DBA synonyms SQL script.  Don't forget that EACH
REM * DBA USER created in the future needs dba_syn.sql run from its account.
REM *
connect system/manager
@/net4/oracle/rdbms/admin/catdbsyn.sql
connect sys/change_on_install
@/net4/oracle/rdbms/admin/catproc.sql
@/net4/oracle/rdbms/admin/cataudit.sql

spool off
```

ANNEXE E: VUE V$TYPE_SIZE

SQL> select * from v$type_size;

COMP	TYPE	DESCRIPTION	SIZE
S	EWORD	EITHER WORD	4
S	EB1	EITHER BYTE 1	1
S	EB2	EITHER BYTE 2	2
S	EB4	EITHER BYTE 4	4
S	UWORD	UNSIGNED WORD	4
S	UB1	UNSIGNED BYTE 1	1
S	UB2	UNSIGNED BYTE 2	2
S	UB4	UNSIGNED BYTE 4	4
S	SWORD	SIGNED WORD	4
S	SB1	SIGNED BYTE 1	1
S	SB2	SIGNED BYTE 2	2
S	SB4	SIGNED BYTE 4	4
S	BOOLEAN	BOOLEAN	4
S	FLOAT	FLOAT	4
S	DOUBLE	DOUBLE	8
S	SIZE_T	SIZE_T	4
K	KDBA	DATABASE BLOCK ADDRESS	4
K	KTNO	TABLE NUMBER IN CLUSTER	1
K	KSCN	SYSTEM COMMIT NUMBER	8
K	KXID	TRANSACTION ID	8
K	KUBA	UNDO ADDRESS	8
KCB	KCBH	BLOCK COMMON HEADER	20
KTB	KTBIT	TRANSACTION VARIABLE HEADER	24
KTB	KTBBH	TRANSACTION FIXED HEADER	48
KDB	KDBH	DATA HEADER	14
KTE	KTECT	EXTENT CONTROL	44
KTE	KTETB	EXTENT TABLE	8
KTS	KTSHC	SEGMENT HEADER	8
KTS	KTSFS	SEGMENT FREE SPACE LIST	20
KTU	KTUBH	UNDO HEADER	16
KTU	KTUXE	UNDO TRANSACTION ENTRY	28
KTU	KTUXC	UNDO TRANSACTION CONTROL	100
KDX	KDXCO	INDEX HEADER	16
KDX	KDXLE	INDEX LEAF HEADER	32
KDX	KDXBR	INDEX BRANCH HEADER	24

35 rows selected.

.