SGBD I

Partie II : Le Modèle Relationnel

Les bases du modèle relationnel ont été définies par CODD (mathématicien du centre de recherche IBM San-José) en 1969.

L'objectif au départ était de combler les inconvénients des modèles hiérarchique et réseau en assurant une meilleure indépendance entre les données et les programmes qui les utilisent et en offrant des règles pour résoudre les problèmes de redondance et de cohérence des données.

Le souci par la suite fut de développer un langage de manipulation de données reposant sur des bases solides. L'algèbre relationnelle fut alors inventée par CODD. Elle a introduit les opérations et les règles qui régissent et permettent la consultation des données d'une base de données et a finalement conduit IBM à la création du langage SQL (Structured Querry Language).

SQL fut normalisé en 1986. Cette première version ne permettait pas la description des données de la base de données mais uniquement leur manipulation. Il a fallu attendre 1992 pour qu'une deuxième version plus complète de SQL dite SQL2 soit normalisée.

En 1998, une nouvelle version SQL3 a vu le jour. Elle intègre le modèle Objet - Relationnel .

L'ALGEBRE RELATIONNELLE

L'algèbre relationnelle offre un ensemble d'opérateurs permettant de manipuler les données contenues dans les relations. On distingue entre les opérateurs ensemblistes, spécifiques et dérivés.

1. Les opérateurs ensemblistes :

- L'union : L'union entre deux relations R1 et R2 est une nouvelle relation U contenant l'ensemble des t-uplets appartenant aux relations R1 et R2 sans répétition

U = R1 UNION R2

Exemple

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

→
→

R3 = R1 UNION R2	
Attribut1	Attribut2
1	a
2	b
3	c
4	d

- L'intersection : L'intersection entre deux relations R1 et R2 crée une nouvelle relation I contenant les t-uplets commun entre R1 et R2

I = R1 INTER R2

Exemple

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

→

R3 = R1 INTER R2	
Attribut1	Attribut2
2	b

- La différence : La différence entre R1 et R2 est une nouvelle relation D contenant tous les t-uplets de R1 sauf ceux qui appartiennent à R2

D=R1 - R2

Exemple :

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

→ →

R3 = R1 - R2	
Attribut1	Attribut2
1	a

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

→ →

R3 = R2 - R1	
Attribut1	Attribut2
3	c
4	d

Remarque : L'Union, l'Intersection et la Différence doivent mettre en liaison des relations incluent le même nombre d'attributs.

- Le produit Cartésien : Le produit cartésien entre R1 et R2 est une nouvelle relation PC contenant la combinaison de tous les t-uplets provenant de R1 et de R2

PC=R1 x R2

Exemple

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

→ →

R3 = R1 x R2			
R1.Attribut1	R1.Attribut 2	R2.Attribut1	R2.Attribut2
1	a	2	b
1	a	3	c
1	a	4	d
2	b	2	b
2	b	3	c
2	b	4	d

2. Les opérateurs spécifiques :

- La sélection (la restriction) : La sélection appliquée à une relation R crée une nouvelle relation S contenant tous les t-uplets de R obéissent aux conditions de sélection

S=S(Condition)R

Exemple

R1		
Attribut1	Attribut2	Attribut3
1	a	yy
2	b	xx
3	c	yy
4	d	gg

→ →

R2= S(Attribut3='yy')R1		
Attribut1	Attribut2	Attribut3
1	a	yy
3	c	yy

- La projection : La projection crée une nouvelle Relation P à la base de quelques attributs d'une Relation R. Tous les t-uplets seront conservés

P=[Attribut1, Attribut2,...]R

Exemple

R1		
Attribut1	Attribut2	Attribut3
1	a	yy
2	b	xx
3	c	yy
4	d	gg

→
→

R2= [Attribut1,Attribut3]R1	
Attribut1	Attribut3
1	yy
2	xx
3	yy
4	gg

- La jointure : crée une nouvelle relation J en faisant correspondre les attributs et les t-uplets de deux relations R1 et R2 selon une condition dite de jointure. Il s'agit en fait du produit cartésien de R1 et de R2 sur lequel on applique une sélection selon la condition de jointure.

J=R1[Condition de Jointure]R2

Exemple

R1		
Attribut1	Attribut2	Attribut3
1	a	1
2	b	1
3	c	2
4	d	1

R2	
Attribut3	Attribut5
1	x
2	y
3	z

→
→

R3 = R1[R1.Attribut3=R2.Attribut3]R2				
Attribut1	Attribut2	R1.Attribut3	R2.Attribut3	Attribut5
1	a	1	1	x
2	b	1	1	x
3	c	2	2	y
4	d	1	1	x

3. Les opérateurs dérivés :

- L'intersection : est un opérateur ensembliste et aussi un opérateur dérivé qui peut être déduit de la différence. Il permet la création d'une nouvelle relation I à partir de deux relations R1 et R2 auxquelles on aurait appliqué l'opérateur de différence.

I= R1 INTER R2 = R1-(R1-R2)

I= R1 INTER R2 = R2-(R2-R1)

Exemple :

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

R1 - R2	
Attribut1	Attribut2
1	a

→
→

R1 INTER R2 = R1 - (R1 - R2)	
Attribut1	Attribut2
2	b

R1	
Attribut1	Attribut2
1	a
2	b

R2	
Attribut1	Attribut2
2	b
3	c
4	d

R2 - R1	
Attribut1	Attribut2
3	c
4	d

→
→

R1 INTER R2 = R2 - (R2 - R1)	
Attribut1	Attribut2
2	b

- La division : La division d'une relation R1 par R2 construit une nouvelle relation DV contenant les t-uplets qui concaténés à ceux de R2 constituent un t-uplet de R1

DV = R1 DIV R2

Exemple

R1		
Attribut1	Attribut2	Attribut3
1	a	x
1	a	y
2	b	x
3	c	x
3	c	y
3	c	z
4	d	y

R2
Attribut3
x
y

→
→

R3 = R1 DIV R2	
Attribut1	Attribut2
1	a
3	c

R1		
Attribut1	Attribut2	Attribut3
1	a	1
2	b	1
3	c	2
4	d	1
5	a	1
1	c	2

R2	
Attribut2	Attribut3
a	1
c	2

→
→

R3 = R1 DIV R2
Attribut1
1

- La Jointure externe : La jointure présentée dans les opérateurs spécifiques est dite interne car elle n'inclut dans la relation créée que les t-uplets de R1 ayant une correspondance dans R2 selon la condition de jointure et ceux de R2 ayant une correspondance dans R1 selon la condition de jointure. Il existe un autre opérateur appelé Jointure externe.

  La jointure externe inclut dans la relation de jointure créée entre R1 et R2 tous les t-uplets de R1 et de R2 en affectant des valeurs NULL pour les attributs n'ayant pas de valeurs.

  On distingue :

  - La jointure externe à gauche : Inclut dans la relation de jointure créée entre R1 et R2 tous les t-uplets de R1 et seulement ceux de R2 pour lesquels les champs joints sont égaux.

    JE =R1 + [Condition de jointure] R2

  - La jointure externe à droite : Inclut dans la relation de jointure créée entre R1 et R2 tous les t-uplets de R2 et seulement ceux de R1 pour lesquels les champs joints sont égaux.

    JE =R1 [Condition de jointure] + R2

  - La jointure externe complète : Inclut dans la relation de jointure créée entre R1 et R2 tous les t-uplets de R1 et tous les t-uplets de R2

    JE =R1 + [Condition de jointure] + R2

Exemples :

R1		
Attribut1	Attribut2	Attribut3
1	a	1
2	b	2
3	c	2
4	d	3
5	e	7

R2	
Attribut3	Attribut5
1	x
2	y
3	z
4	u
5	v
6	w

R3 = R1+[R1.Attribut3=R2.Attribut3]R2				
Attribut1	Attribut2	R1.Attribut3	R2.Attribut3	Attribut5
1	a	1	1	x
2	b	2	2	y
3	c	2	2	y
4	d	3	3	z
5	e	7	-	-

R3 = R1[R1.Attribut3=R2.Attribut3]+R2				
Attribut1	Attribut2	R1.Attribut3	R2.Attribut3	Attribut5
1	a	1	1	x
2	b	2	2	y
3	c	2	2	y
4	d	3	3	z
-	-	-	4	u
-	-	-	5	v
-	-	-	6	w

R3 = R1+[R1.Attribut3=R2.Attribut3]+R2				
Attribut1	Attribut2	R1.Attribut3	R2.Attribut3	Attribut5
1	a	1	1	x
2	b	2	2	y
3	c	2	2	y
4	d	3	3	z
5	e	7	-	-
-	-	-	4	u
-	-	-	5	v
-	-	-	6	w

LE LANGAGE SQL :

SQL (Structured Querry Language) dans sa version 2 de 1992 est un langage structuré regroupant un ensemble d'instructions permettant la création, la structuration et l'interrogation des bases de données relationnelles. SQL offre trois catégories de langages :

- *Langage de Manipulation de Données (LMD)* : Offre les instructions pour ajouter, supprimer et modifier des t-uplets ainsi que pour appliquer les différentes opérations précédemment traitées dans l'algèbre relationnelle (sélection, projection, jointure, union,...)

- *Langage de Description de Données (LDD)* : Offre les instructions permettant la création de la base de données : Base de données, Tables, Attributs, Clés primaires, Clés étrangères, index...

- *Langage de Contrôle de Données (LCD)* : Offre les instructions permettant de définir des permissions pour les utilisateurs

Remarque : Pour la suite de ce cours nous allons utiliser pour l'ensemble des exemples la base de données Parfumerie :

- Parfum (**Codpar**, NomPar, PrixVente, DateSortie, NumLabo)
- Laboratoire (**NumLabo**, DesignationLabo)
- Produit (**CodPro**, NomPro, UniteMesurePro, PrixUnitairePro)
- Composition (**CodPar**, **CodPro**, QuantiteUtilisee)

Cette base de données correspond au système d'information d'un artisan parfumeur qui propose un ensemble de parfums à la vente. Ces parfums ont été réalisés dans des laboratoires. Chaque laboratoire est identifié par un numéro et par une désignation. Les parfums créés sont codifiés. On leur affecte alors un nom, un prix de vente, la date de leur création et le numéro de laboratoire où ils ont été créés. L'artisan souhaitant aussi mémoriser la composition de chaque parfum, on enregistre les produits qui composent chaque parfum avec pour produit la quantité utilisée.

1. Langage de Manipulation de Données (LMD) :

Interrogation des données :

L'interrogation des données se fait à l'aide de l'instruction Select. La syntaxe complète pour cette instruction est :

Select *Attribut1, Attribut2,...*
From *Relation1, Relation2,...*
Where *Condition*
Group By *Attribut_Regroupement1, Attribut_Regroupement2*
Having *Condition_Regroupeme*
Order By Critère_Tri1 Asc| Desc, Critère_Tri2 Asc| Desc

- Select : Indique la liste des attributs que va retourner la requête

  Remarque : Pour éliminer les répétitions de **t-uplets** utilisez le mot clé Distinct

  Select Distinct Col1, Col2...

- From : Indique la liste des relations dont les attributs seront utilisés par la requête soit comme attributs de retour, de regroupement ou de tri, soit pour exprimer des conditions de sélection ou de jointure
- where : Exprime des conditions de sélection ou de jointure.
- Group by : Effectue des regroupements sur certains attributs. En général, il est utilisé pour obtenir des statistiques et calculs en exploitant les fonctions offertes par SQL : Sum (Somme), Min(Minimum), Max(Maximum), Count(Nombre), Avg(Moyenne).

  Remarque : Pour ne travailler que sur les valeurs distinctes

  Select Fonction(Distinct(Colonne))...

- Having : Exprime des conditions sur les éléments de regroupement

- Order by : Spécifie des critères de tri

Les conditions peuvent être exprimée à l'aide des opérateurs =, >, >= ,<, <=, Between, in, like, is Null, exists, any, all, union, intersect, minus et peuvent être combinées à l'aide des opérateurs logiques : and, or et not

- Between : Permet de tester l'appartenance de la valeur d'une colonne à un intervalle :

  Where Colonne Between Valeur1 and Valeur2
  Ce qui est équivalent à :
  Where Colonne >= Valeur1 and Colonne >= Valeur2

- In : Teste si les valeurs d'une colonne appartiennent à un groupe de valeur. Le groupe de valeurs pouvant s'exprimer à l'aide de valeurs constantes ou d'une requête select

  Where Attribut in (Val1,Val2,Val3,...)
  Ou
  Where Attribut in (Select...From...Where...)

- Like : Permet une comparaison entre une chaîne de caractères et les valeurs d'une colonne. Elle utilise deux caractères spéciaux % et _. % remplace plusieurs caractères alors que _ remplace un seul caractère

Exemples :

  Where Attribut Like 'A%' : Attribut commence par A
  Where attribut Like 'A_' : Attribut se compose de deux lettres et commence par A
  Where attribut Like 'A__' : Attribut se compose de trois lettres et commence par A

- Any : Compare une colonne avec une liste de valeurs fournies par une sous requête. Si une des valeurs de la liste rend la condition vraie alors la ligne est sélectionnée
- All : compare une colonne avec une liste de valeurs fournies par une sous requête. Si toutes les valeurs de la liste rendent la condition vraie alors la ligne est sélectionnée
- Exists : Est évalué à vrai si la sous-requête qui le suit donne au moins une ligne en retour
- Union : Permet de réaliser l'union entre deux ensembles

  (Select...From...Where...)
  Union
  (Select...From...Where...)

- Intersect : Permet de réaliser l'intersection entre deux ensembles
- Minus : Permet de réaliser la différence entre deux ensembles

Pour représenter une jointure interne entre deux tables la table Table1 et la table Table2 on utilise :

- Select Table1. Colonne1, Table2.Colonne2
- From Table1, Table2
- Where Table1.ColonneJointure1=Table2.ColonneJointure2
- Dans les nouvelles versions de SQL, elle se présente :
- Select Table1. Colonne1, Table2.Colonne2
- From Table1 Inner Join Table2 On Table1.ColonneJointure1=Table2.ColonneJointure2
- Dans une jointure interne seuls seront représentés les t-uplets qui existent à la fois dans Table1 et dans Table2
-

- Pour exprimer une jointure externe à droite entre la table Table1 et la table Table2 on utilise :
- Select Table1. Colonne1, Table2.Colonne2
- From Table1 Right Outer Join Table2
- On Table1.ColonneJointure1 = Table2.ColonneJointure2
- Dans cette jointure, l'ensemble des t-uplets de Table2 vont s'afficher et seulement ceux de Table1 pour lesquels les champs joints sont égaux
-
- Pour exprimer une jointure externe à gauche entre la table Table1 et la table Table2 on utilise :
- Select Table1. Colonne1, Table2.Colonne2
- From Table1 Left Outer Join Table2
- On Table1.ColonneJointure1=Table2.ColonneJointure2
- Dans cette jointure, l'ensemble des t-uplets de Table1 vont s'afficher et seulement ceux de Table2 pour lesquels les champs joints sont égaux
-
- Pour exprimer une jointure totale externe entre la table Table1 et la table Table2 on utilise :
- Select Table1. Colonne1, Table2.Colonne2
- From Table1 Full Left Outer Join Table2
- On Table1.ColonneJointure1=Table2.ColonneJointure2
- Dans cette jointure, l'ensemble des t-uplets de Table1 et de table 2 vont s'afficher. S'il y 'a des correspondances elles s'affichent sinon des valeurs null s'affichent à la place.

Exemples :

La liste de tous les parfums (code et nom)

```
Select CodPar, NomPar
From Parfum                                    (Projection)
```

La liste de tous les parfums créés dans le laboratoire n° 1

```
Select *
From Parfum
where NumLabo=1                                (Sélection)
```

Le numéro du laboratoire où a été créé le parfum portant le code 'P2'

```
Select NumLabo
from Parfum
where codpar="P2"                        (Projection + Sélection)
```

Le numéro et la désignation du laboratoire où a été créé le parfum portant le code 'P2'

```
Select Laboratoire.NumLabo, DesignationLabo
from Parfum, Laboratoire
where Parfum.numLabo=Laboratoire.numLabo and codpar="P2"
                                ( Projection + Sélection + Jointure )
```

La liste des produits entrant dans la composition du parfum 'Fleur d'été' avec pour chaque produit le code, le nom, l'unité de mesure et la quantité utilisée

```
Select Produit.CodPro, NomPro, UniteMesurePro, QuantiteUtilisee
from Produit, Composition, Parfum
where  Produit.codpro=Composition.CodPro  and  Composition.codPar=
Parfum.CodPar and NomPar="Fleur d'été"
```

La liste des produits qui n'entrent pas dans la composition du parfum "Marrakech". Pour chaque produit, on affiche le code et le nom

```
Select CodPro, NomPro
From Produit
where CodPro not in (
Select CodPro
From Composition, Parfum
```

```
Where Composition.CodPar=Parfum.CodPar and NomPar= "Marrakech" )
```

La liste des noms de parfums qui ont été créés dans le même laboratoire où a été créé le parfum "Layali" (Il ne faut pas afficher "Layali")

```
Select Par1.NomPar
From Parfum as Par1, Parfum as Par2
where Par1.NumLabo=Par2.NumLabo and Par2.NomPar="Layali" and
Par1.Nompar <>"Layali"
```

Les couples de parfums (Code Parfum , Nom Parfum, Code Parfum, Nom Parfum, Désignation Laboratoire) qui ont été créés dans le même laboratoire. Il faut naturellement éviter les doublons et les égalités. Si par exemple le couple (P1,"Layali", P2, "Marrakech", "Senteurs du Maroc") s'affiche, il ne faut pas afficher le couple (P2, "Marrakech", P1, "Layali", "Senteurs du Maroc").De même (P2, "Marrakech", P2, "Marrakech", "Senteurs du Maroc") n'est pas considéré comme un couple

```
Select Par1.CodPar, Par1.NomPar, Par2.CodPar, Par2.nomPar,
DesignationLabo From Parfum as Par1, Parfum as Par2, Laboratoire
where Par1.NumLabo=Par2.NumLabo and Par1.NumLabo=Laboratoire.NumLabo
and Par1.CodPar< Par2.CodPar
```

La liste des parfums triés par ordre décroissant des noms

```
Select *
From Parfum
Order By Desc NomPar
```

Remarque :

Si on ne spécifie pas de sens de tri, le tri est considéré comme croissant

Order By Critère_Tri1 Asc| Desc, Critère_Tri2 Asc| Desc

La liste des cinq premiers parfums de la liste

```
Select top 5 * from article
```

La liste des laboratoires pour lesquels il n'existe aucun parfum

```
Select designationLabo from laboratoire
where not exists(select * from Parfum where
NumLabo=Laboratoire.numlabo)
```

Remarque :

La liaison entre les sous-requêtes et la requête principal se fait par NumLabo)

Le nombre total de parfums

```
Select count(CodPar)
From Parfum
```

Le nombre total de parfums créés dans le laboratoire "Senteurs du Maroc"

```
Select count(CodPar)
From Parfum, Laboratoire
where Parfum.Numlabo=Laboratoire.Numlabo and
DesignationLabo="Senteurs du Maroc"
```

Le nombre de parfums par laboratoire (numéro de laboratoire)

```
Select NumLabo, count(CodPar)
From Parfum
Group By NumLabo
```

Le nombre de parfums par laboratoire (Numéro et désignation du laboratoire)

```
Select Laboratoire.NumLabo, DesignationLabo, count(CodPar)
From Parfum, Laboratoire
where Parfum.Numlabo=Laboratoire.Numlabo
Group By Laboratoire.NumLabo, DesignationLabo
```

Le prix de reviens du parfum portant le code P6

```
Select Sum(PrixUnitairePro*Quantiteutilisee)
```

```
from Produit, Composition
where Produit.codpro=Composition.CodPro and CodPar="P6"
```

Le prix de reviens de chaque parfum avec pour chaque parfum le code

```
Select CodPar, Sum(PrixUnitairePro*Quantiteutilisee)
from Produit, Composition
where Produit.codpro=Composition.CodPro
Group by CodPar
```

Le prix de reviens de chaque parfum avec pour chaque parfum le code, le nom et la désignation du laboratoire où il a été créé

```
Select Parfum.CodPar,NomPar,DesignationLabo, Sum(PrixUnitairePro*
Quantiteutilisee)
from Produit, Composition, Parfum, Laboratoire
where Produit.codpro=Composition.CodPro and Composition.CodPar =
Parfum.CodPar and Parfum.numLabo=laboratoire.numLabo
Group by Parfum.CodPar, NomPar, DesignationLabo
```

Les numéros de laboratoire où ont été créés plus de cinq parfums

```
Select NumLabo
From Parfum
Group By NumLabo
Having Count(CodPar) >5
```

Les noms des parfums créés dans le laboratoire n°1 dont le prix de reviens est >300

```
Select NomPar
from Produit, Composition, Parfum
where Produit.codpro=Composition.CodPro and Composition.CodPar =
Parfum.CodPar and NumLabo= 1
Group by Parfum.CodPar, NomPar
Having Sum(PrixUnitairePro*Quantiteutilisee)>300
```

Les noms des parfums créés dans le laboratoire "Senteurs du Maroc" dont le prix de reviens est >300 et sont composés à la base de plus de trois produits

```
Select NomPar
from Produit, Composition, Parfum, Laboratoire
where Produit.codpro=Composition.CodPro and Composition.CodPar =
Parfum.CodPar and Parfum.NumLabo=Laboratoire.NumLabo and
DesignationLabo="Senteurs du Maroc"
Group by Parfum.CodPar, NomPar
Having Sum(PrixUnitairePro*Quantiteutilisee)>300 and
count(Produit.CodPro)>3
```

Les parfums qui utilisent tous les produits

```
Les laboratoires où ont été fabriqué le plus de parfums
Select L.NumLabo,DesignationLabo
```

Exemples

La liste des parfums avec pour chaque parfum le code, le nom, le prix de vente et la désignation du laboratoire. On n'affiche que les laboratoires où ont été créés des parfums

```
SELECT Parfum.CodPar, Parfum.NomPar, Parfum.Prixvente, Laboratoire.DesignationLabo
FROM Laboratoire INNER JOIN Parfum ON Laboratoire.NumLabo = Parfum.NumLabo
```

CodPar	NomPar	Prixvente	DesignationLabo
1	P1	10	L1
2	P2	34	L1

La liste des parfums avec pour chaque parfum le code, le nom, le prix de vente et la désignation du laboratoire. On affiche tous les laboratoires

SELECT Parfum.CodPar, Parfum.NomPar, Parfum.Prixvente, Laboratoire.DesignationLabo
FROM Laboratoire LEFT OUTER JOIN Parfum ON Laboratoire.NumLabo = Parfum.NumLabo

CodPar	NomPar	Prixvente	DesignationLabo
1	P1	10	L1
2	P2	34	L1
			L2

La liste des parfums et des laboratoires. Si des correspondances existent elles s'affichent sinon des valeurs null s'affichent à la place.

SELECT Parfum.CodPar, Parfum.NomPar, Parfum.Prixvente, Laboratoire.DesignationLabo
FROM Laboratoire FULL OUTER JOIN Parfum ON Laboratoire.NumLabo = Parfum.NumLabo

La liste des produits entrant dans la composition du parfum portant le nom 'P1'

SELECT Produit.CodPro, Produit.NomPro, Parfum.NomPar
FROM Produit INNER JOIN (Parfum INNER JOIN Composition ON Parfum.CodPar = Composition.CodPar) ON Produit.CodPro = Composition.CodPro
WHERE (((Parfum.NomPar)="P1"))

Les sous-totaux et les statistiques

La clause compute est utilisée pour ajouter des sous-totaux à la fin d'une requête SQL. Elle est associée à des fonctions d'agrégation. A la différence du group by qui retourne des statistiques en masquant les données d'origine, compute laisse les données visibles.

Il existe une clause Compute qui est employée dans des instructions Select avec des fonctions d'agrégation (Sum, Avg,..) pour insérer des lignes supplémentaires faisant apparaître des sous totaux ou des statistiques.

Pour afficher un total général :

Select Col1, Col2, ...
From Table1, Table2,...
Where...
Compute Fonction1(Col_Regroupement), Fonction(Col_Regroupement)

Pour afficher des sous-Totaux :

Select Col1, Col2, ...
From Table1, Table2,...
Where...
Order by Col_Tri1, Col_Tri2,...
Compute Fonction(Col_Regroupement), Fonction(Col_Regroupement) by Col_Tri1, Col_Tri2,...ci

Exemples

La liste des parfums avec en bas la moyenne des prix de vente, le prix le plus élevé, le prix le plus bas

Select CodPar, NomPar, PrixVente

From parfum
Compute avg(prixvente),max(prixvente), min(prixvente)

	CodPar	NomPar	PrixVente
1	1	P1	10.0
2	2	P2	34.0
3	3	P3	44.0

	avg	min
1	29.333333333333332	10.0

La liste des parfums regroupés par laboratoire avec pour chaque laboratoire la moyenne des prix de vente, le prix le plus élevé, le prix le plus bas

Select CodPar,NomPar,PrixVente
From parfum
Order by Numlabo
Compute avg(prixvente),min(prixvente) by Numlabo

	CodPar	NomPar	PrixVente
1	1	P1	10.0
2	2	P2	34.0

	avg	min
1	22.0	10.0

	CodPar	NomPar	PrixVente
1	3	P3	44.0

	avg	min
1	44.0	44.0

Mise à jour des données :

Par mise à jour on désigne les différentes opérations de suppression, de modification et d'insertion des informations dans les relations.

Insertion d'un t-uplet :

La syntaxe utilisée est la suivante :

Insert into *Nom_Table* values (*valeur_Attribut1*, *valeur_Attribut2*,...)

Il existe une autre forme pour l'instruction insert où on précise les champs à remplir. L'indication de colonnes peut se faire dans n'importe quel ordre à condition que les valeurs attribuées soient indiquées dans le même ordre :

Insert into Nom_table (Col1, Col2,...) values (val1_Pour_Col1, val2_Pour_Col2....)

Un autre moyen de charger une table est d'utiliser une instruction select qui prend des informations à partir d'autres tables pour les insérer dans une table voulue

Insert into Nom_table Select Col1, Col2,... From Nom_Table1, Nom_Table2...Where...

Si on ne souhaite alimenter que quelques champs avec la même méthode :

Insert into Nom_table (Col1,Col2,...) Select ColOrig1, ColOrig2,... From Nom_Table1, Nom_Table2...Where...

Suppression d'un t-uplet ou d'un groupe de t-uplets :

Delete from Nom_Table where Condition_Suppression

Modification d'un t-uplet ou d'un groupe de t-uplets :

Update Nom_Table set Attribut1 = modification, Attribut2=modification,...From Nom_Table1, Nom_Table2,...where Condition

Exemples

Enregistrer les laboratoires n° 1 nommé "Senteurs du Maroc" et n° 2 nommé "Senteurs Indienne"

Insert into Laboratoire Values (1, "Senteurs du Maroc")

Supprimer la composition du parfum P4

Delete from composition where CodPar="P4"

Augmenter les prix des parfums créés dans la laboratoire "Senteurs du Maroc" de 5%

*Update parfum set PrixVente=Prixvente+(Prixvente*5)/100 From Laboratoire where Parfum.NumLabo = Laboratoire.NumLabo and DesignationLabo = "senteurs du Maroc"*

2. Langage de Description de Données (LDD) :

Les contraintes d'intégrité

Les contraintes d'intégrité permettent une meilleure exploitation de la base de données. Connaître les contraintes rend l'utilisation de la base de données meilleure et plus performante. Ces contraintes peuvent être de différentes natures :

- *Contrainte d'unicité de clé :* Chaque relation doit posséder une clé dite primaire qui identifie chaque t-uplet de manière unique.
- *Contrainte de domaine* : Pour chaque attribut (clé primaire comprise), il faudra définir et identifier l'ensemble des valeurs qu'il peut prendre
- *Contrainte d'intégrité référentielle* : Permet une bonne gestion des liens qui existent entre les relations. En fait un lien étant toujours défini d'une clé primaire vers une clé étrangères, les contraintes d'intégrité référentielle veillent à ce que les valeurs prises par la clé étrangère soient des valeurs qui existent pour la clé primaire.

Les contraintes d'intégrité référentielle peuvent être appliquées de deux manières Restricted ou cascades :

- Restricted : consiste à refuser toute modification qui nuirait l'intégrité référentielle
- cascades : consiste à accepter la modification et à réaliser un certain nombre d'opérations en parallèle sur l'ensemble des relations concernées soit en supprimant des t-uplets, soit en les modifications.

Les liens

Les liens entre les relations dans le modèle relationnel peuvent être de deux types :

Lien 1..n noté également 1..∞ : Une valeur de la clé primaire peut apparaître plusieurs fois en tant que clé étrangère

Lien 1..1 : Une valeur de la clé primaire peut apparaître une et une seule fois en tant que clé étrangère

Création de bases de données (Format réduit)

Create Database Parfumerie

Création de tables

Il existe deux manières de le faire. Une première consiste à déclarer les champs de la relation (la Table) puis à déclarer les contraintes en spécifiant sur quels attributs porte chaque contrainte et une deuxième qui consiste à déclarer les contraintes directement avec les attributs.

Méthode 1

Create Table *Nom_Table*

(

Attribut1 Type_Attribut1, Attribut2 Type_Attribut2,...

[Constraint *Nom_Contrainte* | Check (*condition*)
Unique (*nom_Attribut*)
Primary Key (*nom_Attribut_Clé1*, *Nom_Attribut_Clé2*...)
Foreign key(*Attribut_clé_étrangère*) references *Table_Origine* | **]**,

[Constraint...**]**,...

)

Méthode 2

Create Table *Nom_Table*

(

Attribut1 Type_Attribut1 **[**Constraint *Nom_Contrainte* | Not Null
Default *Valeur_Par_Defaut*
Check (*condition*)
Unique
Primary Key
Foreign key references *Table_Origine* | **] [**Constraint...**]** ,

Attribut1 *Type_Attribut1* **[**Constraint...**] [**Constraint...**]** ...,..., **[**Constraint...**]**

)

Remarques :

- Il est possible de créer des colonnes dans une table qui soient calculés à la base d'autres champs

  Create table (champ1 type..., champ2 type..., champN as expression)
- Il est possible de ne pas attribuer des noms à certaines contraintes en éliminant le mot clé Constraint mais à ce moment là il ne sera pas possible d'éliminer cette contrainte par code SQL de la table
- La contrainte Check permet le contrôle de :
  - La validité par rapport à des constantes ou des listes de constantes
  - La cohérence entre deux colonnes de la table

    Exp : constraint CK Check (date fin >=datedebut)
- Pour créer une colonne numéro auto :

  ...Nom_Colonne Type_Colonne IDENTITY (Valeur_Initiale, Pas)...

  Type colonne doit être dans ce cas de type numérique int, bigint, smallint, tinyint, decimal ou numeric

  Exemple :
  Create Table maTable (Col1 int identity(1,1) not null primary key, Col2 varchar(32))

Le champ col1 va démarrer avec la valeur 1 et sera incrémenté de 1 à chaque ordre insert

Conseil pratique : Pour créer les tables d'une base de données, on commence toujours par créer les tables ne contenant pas de clés étrangères puis on passe à la création des tables où les clés primaires de ces tables ont glissé comme clé étrangère car il n'est pas possible de créer un lien avec des tables qui n'existent pas.

Exemples:

Créer la base de données Parfumerie :

Create Database Parfumerie (Spécial SQL Server)

Create Table Laboratoire (NumLabo int constraint PK_Labo Primary Key, DesignationLabo VarChar(50) Constraint NN_DesLabo Not Null Constraint U_DesLabo Unique)

Create Table Produit (CodPro varchar(10) Constraint PK_Produit Primary Key, NomPro Varchar(50) Constraint NN_NomPro Not Null, UniteMesure VarChar(10) constraint CK_Unite Check (UniteMesure in ('l','kg','c')), PrixUnitairePro Real Constraint NN_PrixPro Not Null)

Create Table Parfum (CodPar varchar(10) Constraint PK_Parfum Primary Key, NomPar Varchar(50) Constraint NN_NomPar Not Null, Prixvente Real Constraint NN_PrixPar Not Null, NumLabo Int constraint FK_Parfum_Labo Foreign Key references Laboratoire)

Create Table Composition (CodPar varchar(10), CodPro varchar(10), QuantiteUtilisee Real, Constraint NN_Qte Not Null, Constraint CK_qte Check (QuantiteUtilisee >0), Constraint PK_Composition Primary Key (CodPar, CodPro), Constraint FK_Comp_Parfum Foreign Key(CodPar) references Parfum, Constraint FK_Comp_Produit Foreign Key(CodPro) references produit)

Suppression de tables

Drop table Nom_Table

Exemple

Supprimer la table composition

Drop table Composition

Remarque : Par défaut les contraintes référentielles créées sont de type Restricted. Pour spécifier des contraintes référentielles de type cascade la syntaxe serait :

... Constraint *Nom_Contrainte* Foreign key ...References *Table_Origine* **[** ON DELETE CASCADE **]**

[ON UPDATE CASCADE **]**

Modification de tables

Alter Table Nom_Table

Add Nom_Attribut Type_Attribut **[**Constraint *Nom_Contrainte* | Not Null / Default... / Check (...) / Unique / Primary Key / Foreign key ... | **] [**Constraint...**]**

Alter column nomcolonne_A_Changer nouveau type null ou not null

Drop Column Nom_Colonne

Drop Constraint Nom_Contrainte

Add Constraint	Primary Key (nom_Attribut_Clé1, Nom_Attribut_Clé2...) Unique (nom_Attribut) Check(condition) Foreign key(Attribut_clé_étrangère) references Table_Origine

Exemples

Supprimer la table Laboratoire

Drop table Laboratoire

La syntaxe est correcte mais la table ne sera pas supprimée car il existe au niveau de la table parfum une clé étrangère faisant référence à la clé primaire de la table laboratoire

Remarque : Si on supprime une table toutes les contraintes liées à cette table sont automatiquement supprimées

Supprimer l'attribut Numlabo de la table Parfum

Alter Table Parfum drop Column Numlabo

La syntaxe est correcte mais le champ ne sera pas supprimé car il existe une relation qui relie ce champ à une autre table. Il faut commencer par supprimer la relation.

Supprimer le lien qui existe entre la table Parfum et Laboratoire

Alter Table Parfum drop Constraint FK_Parfum_Labo

Une fois cette relation supprimée, il devient possible de supprimer le champ NumLabo

Recréer le champ NumLabo et recréer la relation entre la table parfum et la Table Laboratoire.

Alter Table Parfum Add NumLabo Int

Recréer la relation entre la table Parfum et la Table Laboratoire.

Alter Table Parfum Add constraint FK_Parfum_Labo Foreign Key references Laboratoire

Création d'index

Un index permet d'accélérer la recherche des informations en utilisant des pointeurs qui gardent la position de chaque élément mais les index doivent être utilisés soigneusement et surtout pour des tables contenant un grand nombre d'enregistrements car ils consomment les ressources mémoire.

Il existe deux catégories d'index : Non ordonnés et Ordonnés

Les index ordonné (Clustered ou Clustérisés): Modifient l'ordre des t-uplets sur le support de stockage pour accéder rapidement à l'information. On ne peut créer qu'un seul index ordonné par table et il faut le créer avant les index non ordonnés

Create Clustered Index Nom_Index On Nom_Table(Col1, Col2,...)

Les index non ordonnés : S'appuient sur l'ordre logique des t-uplets en retenant leur position pour retrouver l'information recherchée

Create Index Nom_Index On Nom_Table(Col1, Col2,...)

Remarque : Une clé primaire est un index ordonné par défaut. Pour créer une clé primaire non clustérisée. Il faut mettre l'instruction nonClustered après l'instruction Primary Key

Create table LaboratoireBis (NumLabo int Primary Key nonClustered, DesLabo varchar(10))
ou
Create table LaboratoireBis (NumLabo int, DesLabo varchar(10), Constraint Primary Key nonClustered (NumLabo))

Il est possible de créer un index sans doublons en lui associant le mot clé UNIQUE après Create ;

Un index sur une colonne est plus simple à utiliser qu'un index sur plusieurs colonnes et donc ce dernier ne sera utilisé que lorsque l'utilisateur a souvent besoin d'effectuer des recherches sur toutes ces colonnes à la fois.

Exemples

On recherche souvent un parfum par son nom. Créer un index dessus

Create Index Idx_Par On Parfum(Nom_Par)

Suppression d'index

Drop index *Nom_Table.Nom_Index*

Exemples

Supprimer l'index créé

Drop Index Parfum.Idx_Par

Sur les transactions

Une transaction est un bloc d'instructions qui devra s'exécuter dans sa totalité. Si une erreur survient, tout le bloc ne sera pas exécuté. C'est un moyen très efficace pour éviter les incohérences qui peuvent survenir suite à un arrêt inattendu du système. En SQL Server une transaction commence par Begin Transaction et se termine par Commit transaction
Remarque : Certaines instructions comme Create Database ne peuvent pas être incluses dans de transactions

Création d'une vue

Une vue est un jeu d'enregistrements (table virtuelle) créé à la base des tables existantes. Son objectif est de donner à certains utilisateurs une vue restreinte de la base de données en ne leur montrant que ce qui les concerne.

il est possible d'appliquer à une vue toutes les opérations de sélection, d'insertion, de modification et de suppression

Create View Nom_Vue As Select....

Exemples

Créer la vue Parfums_Labo1 représentant la liste des parfums qui ont été créés dans le laboratoire n° : 1

Create View Parfums_Labo1 As

Select * From Parfum where NumLabo=1

Créer la vue Nbr_ParfumsLabo représentant le nombre de parfums par laboratoire (Désignation du laboratoire)

```
Create View Nbr_Parfums_labo As
    Select DesignationLabo, Count(CodPar)
    From Parfum, Laboratoire
    Where Laboratoire.NumLabo=Parfum.NumLabo
    Group By Laboratoire.NumLabo, DesignationLabo
```

Création d'une vue

Create View Nom_Vue [(Nom_Col1, NomCol2...)] As Select...

Remarque : Si l'utilisateur n'a pas spécifié de nom de colonne, la vue prend automatiquement les noms des champs parvenant des tables mais il faut faire attention à ce que le select ne retourne pas des champs en double ou des champs non calculés sans quoi il devient obligatoire

Modification de vue

une vue peut être modifiée sans passer par Drop View suivi de Create view ce qui obligerait à redéfinir les autorisations sur cette vue

Alter View Nom_Vue [(Nom_Col1, NomCol2...)] As Select...

Suppression de vue

Drop View *Nom_Vue*

Exemples

Supprimer la vue Parfums_Labo1

Drop View *Parfums_Labo1*

3. Langage de Contrôle de Données

Pour accéder à un serveur SQL Server, il faut être authentifié (il faut que SQL Server reconnaisse nos informations d'identification).

Deux Types d'authentification sont disponibles en SQL Server :

- Authentification Windows : SQL Server récupère le compte d'ouverture de session Windows. Ce mode d'authentification présente certains avantages :
  - La chaine de connexion envoyée par l'application ne contient pas les informations d'authentification
  - On bénéficie de la sécurité Active Directory
  - Plus facile puisque plus besoin de saisir les informations d'identification
  - Meilleur audit (contrôle)
- Authentification SQL Server : SQL Server utilise ses propres informations d'identification.

Rq : L'authentification Windows est plus efficace car elle repose sur plusieurs barrières de sécurité (sécurité du système d'exploitation et sécurité SQL Server).

SQL Server a deux niveaux de sécurité. Il cherche d'abord si la connexion a le droit d'accéder au serveur. Si c'est le cas, il cherche les droits que la connexion a sur les objets de la base de données.

Pour accéder au serveur, il faut donc disposer d'une connexion.

Pour créer une connexion type authentification Windows :

Create Login [Compte Windows] From Windows with Defaut_DataBase= [Nom_BD_Defaut] , Default_Language=[Langue_Defaut]

Pour créer une connexion type authentification Windows :

Create Login Nom_Logine with Password='MDP' Defaut_DataBase= [Nom_BD_Defaut] , Default_Language=[Langue_Defaut]

Une connexion donne accès mais pas encore à la base de données. L'étape suivante consiste à créer pour chaque base de données à laquelle la connexion doit accéder un utilisateur. Le concept d'utilisateur ici permet d'affecter à la connexion des droits sur une base de données.

Si la connexion donne accès à 3 bases de données SQL Server, on aura besoin de 3 utilisateurs.

Pour créer un utilisateur pour une connexion nommée XX sur une Base de Données YY, on utilise la syntaxe suivante :

Use YY

Create User Nom_Utilisateur for Login Nom_Login

Langage de Contrôle de Données (LCD) :

Il est possible d'affecter des permissions spécifiques aux utilisateurs sur les instructions (Create Table, Drop Table...) ou sur les objets (select, insert,...)

- Permissions sur les instructions

Grant Deny	All Create table Create view Drop Table ...	, Create table Create view Drop Table ...	,.. To	Public Nom_Utilisateur1 [With Grant Option], Nom_Utilisateur2 [With Grant Option],

- Permissions sur les objets

Grant Deny	All Select (Nom_Attribut1, Nom_Attribut2...) Insert Delete Update (Nom_Attribut1, Nom_Attribut2...)	’	Select ... Insert Delete Update...	On Nom_Table To Nom_Utilisateur1 [With Grant Option], Nom_Utilisateur2 [With Grant Option],

Le mot clé Grant sert à accorder des droits à l'utilisateur alors que le mot clé Deny permet de refuser un droit à un utilisateur.

Il existe un autre mot clé qui est Revoke. Ce mot est utilisé de la même manière que Grant et Deny sauf qu'il faut remplacer le mot clé to par from. Révoquer (Revoque) et refuser (Deny) une autorisation sont deux opérations différentes. Révoquer une autorisation consiste à supprimer un accord ou un refus (permission indéfinie), alors que refuser une autorisation interdit tout accès, même lorsque des autorisations d'accès ont été accordées.

Revoke enlève les droit accordé à un utilisateur, s'il avait un Deny celui-ci disparait, s'il avait un Grant celui-ci disparait. S'il n'a plus aucun Grant l'action n'est pas autorisée.

Remarques :

- N'oubliez pas de spécifier la base de données à laquelle appartient l'utilisateur à l'aide de l'instruction Use Nom_Base_de_données
- With Grant Option indique que l'utilisateur peut transférer les privilèges qu'il vient de recevoir à d'autres utilisateurs
- All permet d'affecter tous les droits