

Chapitre 2 :

Architecture et Fonctions d'un



Système Relationnel
de Base de Données

Partie II. Fonction d'un SGBD

Création/ Modification / Suppression du Schéma



2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

- La structure de données primaire dans les systèmes relationnels est la relation de base. La forme générale de la requête permettant la création d'une relation ou table dans SQL est la suivante :



CREATE TABLE <nom de table>

(définition d'attribut [, définition d'attribut]...) ;

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

CREATE TABLE <nom de table>
(définition d'attribut [, définition d'attribut]...) ;

Où définition d'attribut est :

<nom-attribut (type-donnée [, NONULL])>



- **CHAR (n)** n précise la longueur maximum, par défaut 254
- **DATE** format mm/jj/aa
- **DECIMAL(x,y)** x nombre total de chiffres et y détermine la place de la virgule en partant de la droite
- **FLOAT (x,y)**
- **INTEGER**
- **LOGICAL**

2.2. Fonctions des SGBD relationnels

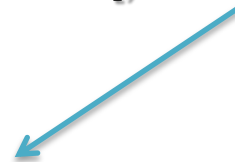
2.2. 1. Création / Modification des tables

**CREATE TABLE <nom de table>
(définition d'attribut [, définition d'attribut]...) ;**

Où définition d'attribut est :



<nom-attribut (type-donnée [, NONNULL])>



De plus, on peut imposer à la valeur d'un attribut de n'être jamais indéfinie (NONNULL), évitant ainsi certaines incohérences dans une expression arithmétique dans laquelle une des opérandes est indéfinie.


2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

**CREATE TABLE <nom de table>
(définition d'attribut [, définition d'attribut]...);**

Exemple :

Notre base de données approvisionnement de l'exemple 1 sera décrite à l'aide de cette commande de la manière suivante :

CREATE TABLE FOURNISSEUR 
(NF CHAR(5) NONNULL, NOM CHAR(20), CODE INTEGER, VILLE
(CHAR (15)));

CREATE TABLE PIECE
(NP CHAR(5) NONNULL, NOM CHAR(20), MATERIAU CHAR(6),
POIDS INTEGER, VILLE CHAR (15));

CREATE TABLE FOURNITURE
(NF CHAR (5) NONNULL, NP CHAR(5) NONNULL, QTE
INTEGER);

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

CREATE TABLE <nom de table>
(définition d'attribut [, définition d'attribut]...) ;

Les ordres **CREATE TABLE** peuvent être émis à **tout moment** dans la vie de la base et non pas comme les systèmes classiques seulement à la phase de création du module ou du schéma de la base.



Pour les **clés primaires**, il est recommandé de les définir **dès la création de table car les données sont stockées selon l'ordre de cette clé**. La commande de la création de table est dans ce cas :

CREATE TABLE <nom de table> (définition d'attribut [, définition d'attribut]..., **CONSTRAINT nom_contrainte PRIMARY KEY (nom_Att1,...,nom_Attn)**);

Les attributs Att1, ...Attn représentent les attributs qui composent la clé primaire.

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Une autre possibilité des SGBD relationnels est de modifier à tout moment le schéma des relations existantes.

Par modification nous entendons soit :

- **Supprimer** une relation par la commande :
DROP TABLE <nom table> ;



Modification du schéma



2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

- **Ajouter** un nouveau constituant afin de prendre en compte une information qui n'avait pas été prévue au départ. Ceci est possible dans SQL avec l'ordre ALTER TABLE de la forme général :

ALTER TABLE <nom table>

ADD (< nom attribut> type de donnée) ;

EXEMPLE :

ALTER TABLE FOURNITURE **ADD** (DATE CHAR (6));

Cette requête ajoute l'attribut DATE à la relation FOURNITURE. Tous les n-uplets de la relation vont être étendus de trois à quatre champs. La valeur du nouveau champ est indéfinie dans tous les cas.



2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

- Ajouter un nouveau constituant afin de prendre en compte une information qui n'avait pas été prévue au départ. Ceci est possible dans SQL avec l'ordre ALTER TABLE de la forme général :

ALTER TABLE <nom table>

ADD (< nom attribut> type de donnée) ;

- **Modifier** une colonne existante :

ALTER TABLE <nom table>



MODIFY (< nom attribut> type de donnée) ;

Par exemple modifier la taille du nom dans la relation fournisseur, s'écrit :

ALTER TABLE Fournisseur

MODIFY NOM (CHAR (40)) ;

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

- Ajouter un nouveau constituant afin de prendre en compte une information qui n'avait pas été prévue au départ. Ceci est possible dans SQL avec l'ordre ALTER TABLE de la forme général :

ALTER TABLE <nom table>

ADD (< nom attribut> type de donnée) ;

- **Modifier** une colonne existante :

ALTER TABLE <nom table>

MODIFY (< nom attribut> type de donnée) ;



- **Supprimer** une colonne existante :

ALTER TABLE <nom table>

Drop column (< nom attribut>) ;

Par exemple Supprimer la colonne ville dans la relation fournisseur, s'écrit :

```
ALTER TABLE Fournisseur
Drop column Ville;
```

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

- Ajouter un nouveau constituant afin de prendre en compte une information qui n'avait pas été prévue au départ. Ceci est possible dans SQL avec l'ordre ALTER TABLE de la forme général :

ALTER TABLE <nom table>

ADD (< nom attribut> type de donnée) ;

- **Modifier** une colonne existante :

ALTER TABLE <nom table>

MODIFY (< nom attribut> type de donnée) ;



Par exemple modifier la taille du nom dans la relation fournisseur, s'écrit :

ALTER TABLE Fournisseur

MODIFY NOM (CHAR (40)) ;

Pour les contraintes je peux écrire:

ALTER TABLE <nom table>

ADD/ Modify constraint (< nom contrainte> type de contrainte
(Prédicat))



Insertion / Modification / Suppression des données

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Après la création des tables, l'utilisateur peut maintenant procéder au chargement des données en utilisant l'instruction INSERT de SQL. Pour insérer une pièce (par exemple, la pièce P7 qui est un clou) dans la relation PIECE nous écrivons :



```
INSERT INTO PIECE (NP, NOM, MATERIAU, POIDS, VILLE)  
values ('P7', 'clou', ?, ?, ?);
```

Les trois derniers champs de la pièce insérée sont indéfinis, cela est possible car ces champs n'ont pas été définis avec l'option NONNULL.

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Pour l'insertion d'un grand nombre de n-uplets dans la relation, il est évident que la commande d'insertion s'avère inadéquate. De plus, les informations correspondantes se trouvent généralement à l'extérieur de la base de données, dans un fichier de saisie ou dans une autre base.



2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Pour l'insertion d'un grand nombre de n-uplets dans la relation, il est évident que la commande d'insertion s'avère inadéquate. De plus, les informations correspondantes se trouvent généralement à l'extérieur de la base de données, dans un fichier de saisie ou dans une autre base.



Pour résoudre ce problème, les SGBD offrent plusieurs solutions, par exemple, en utilisant une fonction de chargement en invoquant une commande spéciale du système et en lui précisant la relation à charger, le support des informations sources et les divers paramètres décrivant le format des données. Ou bien à l'aide d'un programme de lecture écriture écrit dans un langage hôte.

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

On peut également copier certaines lignes à partir d'une autre relation.

Exemple :



```
INSERT INTO FOURNISSEUR-TUNIS (NF, NOM)
SELECT NF, NOM FROM FOURNISSEUR WHERE VILLE =
        'Tunis' ;
```

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

On peut également copier certaines lignes à partir d'une autre relation.

Exemple :



```
INSERT INTO FOURNISSEUR-TUNIS (NF, NOM)
```

```
SELECT NF, NOM FROM FOURNISSEUR WHERE VILLE =  
        'Tunis' ;
```

Le nombre de colonnes et le type des colonnes dans le INSERT doivent correspondre à ceux du résultat du SELECT

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Pour la **modification** d'une relation nous utilisons la commande UPDATE.

UPDATE <nom de table>

SET <nom attribut1> = nouvelle valeur, [nom attribut2 = nouvelle valeur, ...]

WHERE <prédicat> ;



Par exemple, pour changer le code des fournisseurs qui sont localisés à Alger, on écrit :

```
UPDATE FOURNISSEUR  
SET CODE = CODE + 20  
WHERE VILLE = 'ALGER' ;
```

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Pour la **modification** d'une relation nous utilisons la commande UPDATE.



UPDATE <nom de table>

SET <nom attribut1> = nouvelle valeur, [nom attribut2 = nouvelle valeur, ...]

WHERE <prédicat> ;

Ou bien, pour modifier la ville des pièces en fer à l'identique de la ville du fournisseur F5, on écrit :

UPDATE PIECE

**SET VILLE = (SELECT VILLE
FROM FOURNISSEUR
WHERE NF = 'F5')**

WHERE MATERIAU = 'fer' ;

2.2. Fonctions des SGBD relationnels

2.2. 1. Création / Modification des tables

Et enfin pour effacer certains n-uplets d'une relation nous utilisons la commande :

DELETE FROM <nom table>
[WHERE <prédicat>] ;

L'instruction efface les tuples complets qui satisfont le prédicat.



par exemple, pour exprimer que les fournisseurs d'Alger ne fournissent plus de pièces, on écrit :

DELETE FROM FOURNITURE
WHERE NF IN **SELECT NF**
FROM FOURNISSEUR
WHERE VILLE = 'ALGER';



Création/ Suppression des chemins d'accès

2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

Le schéma relationnel de la base de données, tel qu'il apparaît dans notre exemple 1 est le seul que doivent connaître les usagers 'normaux' de la base de données gérée par notre SGBD. Il correspond à l'ensemble de tables de valeurs, sans description du stockage physique des n-uplets et des chemins d'accès qu'il faut utiliser pour retrouver les n-uplets d'une relation dans un ordre donné ou pour aller d'une relation à une autre.



2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

Le schéma relationnel de la base de données, tel qu'il apparaît dans notre exemple 1 est le seul que doivent connaître les usagers 'normaux' de la base de données gérée par notre SGBD. Il correspond à l'ensemble de tables de valeurs, sans description du stockage physique des n-uplets et des chemins d'accès qu'il faut utiliser pour retrouver les n-uplets d'une relation dans un ordre donné ou pour aller d'une relation à une autre.



Nous avons vu également qu'aucune requête relationnelle ne précise les chemins d'accès aux informations ou l'ordre dans lequel les opérateurs de produit s'appliquent. C'est à l'administrateur de demander au système la création des chemins d'accès sur les relations stockées et au système de choisir, grâce à l'optimiseur, parmi les chemins disponibles celui qui est le plus adéquat (indépendance physique).

2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

Un chemin d'accès défini sur une relation est appelé INDEX.

Un index est un “arrangement” des tuples d’une relation en fonction des valeurs d’un constituant ou d’un groupe de constituants.



2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès


Nous distinguons deux types d'index : index primaire et index secondaire.



2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès


Nous distinguons deux types d'index : index primaire et index secondaire.

L'index primaire est défini sur la clé de la relation : à une valeur de la clé, la fonction index ramènera un tuple. L'index primaire fournit un accès direct aux tuples d'une relation, c'est-à-dire que l'ordre des tuples sur la clé coïncide avec l'ordre physique. 

2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

Nous distinguons deux types d'index : index primaire et index secondaire.

L'index primaire est défini sur la clé de la relation : à une valeur de la clé, la fonction index ramènera un tuple. L'index primaire fournit un accès direct aux tuples d'une relation, c'est-à-dire que l'ordre des tuples sur la clé coïncide avec l'ordre physique. 

L'index secondaire peut être défini sur des constituants non clé : à une valeur de l'attribut de l'index, celui-ci ramènera un ou plusieurs tuples. L'index secondaire est un réarrangement logique des tuples d'une relation en fonction des valeurs d'un attribut ou groupe d'attributs.

2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

En SQL, la commande de création d'un index est de la forme :

```
CREATE [UNIQUE] INDEX <nom index> ON < nom relation>  
(nom attribut [ordre][,nom attribut [ordre]...] ) ;
```



Où **ordre** est soit ASC ou DESC et s'il n'est pas spécifié, l'ordre ASC est pris par défaut.

L'option **UNIQUE** permet d'une part de définir qu'un attribut (ou un groupe d'attributs) est clé discriminante de la relation et d'autre part de fournir un chemin d'accès direct aux n-uplets ayant une valeur de clé donnée.

2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

Exemple:

Pour définir les index primaire de notre base de données nous écrivons :



```
CREATE UNIQUE INDEX XF ON FOURNISSEUR (NF) ;  
CREATE UNIQUE INDEX XP ON PIECE (NP) ;  
CREATE UNIQUE INDEX XFP ON FOURNITURE (NF, NP) ;
```

La création d'un index secondaire selon la ville des fournisseurs serait :

```
CREATE INDEX XV ON FOURNISSEUR (VILLE) ;
```

2.2. Fonctions des SGBD relationnels

2.2. 2. Chemins d'accès

L'instruction de suppression d'un index d'une relation est :

DROP INDEX <nom index> ;



En permettant l'exécution à tout moment des instructions CREATE INDEX et DROP INDEX le SGBD assure l'indépendance des programmes vis à vis des chemins d'accès (indépendance physique) : en effet, pour améliorer les performances d'une application, nous avons besoin de changer l'ordre de stockage des informations. *Ceci est possible à l'aide de la définition d'un index selon le nouvel ordre sans toucher aux programmes d'application.*

Catalogues



2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues




Du point de vue d'un SGBD relationnel, une base de données comporte des relations de la base et des vues. De plus pour chaque relation stockée, il peut exister un ou plusieurs chemins d'accès. La manière la plus naturelle pour un SGBD relationnel de conserver toutes ces informations est de les stocker dans des catalogues qui sont eux mêmes des relations stockées dans la base ;

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

le SGBD permet aux usagers de consulter, grâce au langage d'interrogation, les informations relatives à la description de la base.

Les principaux catalogues pour décrire une base de données relationnelle sont : 

- **Relation** : ce catalogue contient un n-uplet pour chaque relation de la base de données y compris lui même et les autres catalogues. On y trouve des informations sur le nom de la relation, son créateur, son degré, certaines statistiques sur son contenu etc....

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

le SGBD permet aux usagers de consulter, grâce au langage d'interrogation, les informations relatives à la description de la base.

Les principaux catalogues pour décrire une base de données relationnelle sont :



- **Relation**
- **Attribut** : pour chaque relation de la base décrite dans le catalogue Relation, le catalogue Attribut comporte autant de lignes qu'il y a d'attributs dans la relation. Chaque n-uplets décrit un attribut : son nom, son type, sa longueur, etc....

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

le SGBD permet aux usagers de consulter, grâce au langage d'interrogation, les informations relatives à la description de la base.

Les principaux catalogues pour décrire une base de données relationnelle sont :



- **Relation**
- **Attribut**
- **Chemin d'accès** : on y trouve la description des chemins d'accès définis sur les relations de la base. Pour chaque index, par exemple, on y trouve la relation et les constituants concernés et s'il s'agit d'un index primaire ou secondaire etc....

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

le SGBD permet aux usagers de consulter, grâce au langage d'interrogation, les informations relatives à la description de la base.

Les principaux catalogues pour décrire une base de données relationnelle sont :

- **Relation**
- **Attribut**
- **Chemin d'accès**



Les autres catalogues seront présentés au fur et à mesure que les notions qu'ils décrivent sont introduites.

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

Exemple d'une intension ainsi qu'une extension du schéma partiel des catalogues :

- **Relation « Relation »:**



<u>Nom-rel</u> (nom externe)	<u>Type-rel</u>	<u>Idf-rel</u> (nom interne)	<u>Tailletup</u> (nbre octets)	<u>Card</u> (nombre tuples)	<u>Degré</u> (nombre attributs)	<u>Datecréa</u> (date de création)	<u>Version</u>	<u>Adr</u> (adr 1 ^{er} att.)	...
Relation	S	1	100	50	09	25/03/02	1	@	
...									
Fourniss eur	B	10	110	1000	04	25/03/02	1	@	
...									

Chacun de ses tuples est un descripteur de relations qui peuvent être de trois types : base, système, ou vue.

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

Exemple d'une intension ainsi qu'une extension du schéma partiel des catalogues :

- **Relation « Attribut »:**



Nom Att. (nom externe)	<u>Idf-att</u>	<u>Idf- rel</u>	<u>Type-att</u>	longueur	Position (pos <u>att. dans la rel</u>)	...
...						
NF	11	02	<u>Integer</u>	06	01	
NOM	12	02	Char	10	02	
...						

Chacun de ses tuples décrit un attribut d'une relation système, de base ou d'une vue

2.2. Fonctions des SGBD relationnels

2.2. 3. Les catalogues

Exemple d'une intension ainsi qu'une extension du schéma partiel des catalogues :

- **Relation « Index »:**

Nom index	<u>Idf-rel</u>	Type (<u>prim.</u> , sec)	Liste- <u>arg-index</u>	...
...				
XF	02	P	NOM	
XV	02	S	VILLE	
...				



Chaque tuple décrit une relation index créée sur les relations de base ou système.