Chapitre 4 : Le Modèle physique des données

Introduction

La traduction d'un MLD conduit à un modèle physique de données (MPD) qui précise notamment le stockage de chaque données à travers sont type et sa taille.
L'implémentation du MLD dans le système de gestion de base de données (SGBD) se fait en utilisant le langage SQL.

SQL signifie Structured Query Language, c'est-à-dire langage de requête structuré. Il constitue un langage complet de gestion de bases de données relationnelles.
Il a été conçu par IBM dans les années 70. Il est devenu le langage standard des systèmes de gestion de bases de données relationnelles (SGBDR)

SQL permet principalement :

- la définition des éléments d'une base de données (création, modification,et suppression des tables dans une base de données relationnelle).→SQL est un langage de définition de données (**LDD)**
- la manipulation des données (insertion, suppression, modification, extraction des données dans une table)→ SQL est un langage de manipulation de données **(LMD)**
- la gestion des droits d'accès aux données (définir des permissions aux niveaux des utilisateurs)→ SQL est un langage de contrôle de données **(LCD)**

1. Création d'une base de données:

1.1. Création d'une table

Une table est un ensemble de lignes et de colonnes. La création consiste à définir le nom de ces colonnes, leur format (type), la valeur par défaut à la création de la ligne et les règles de gestion s'appliquant à la colonne (contraintes).

Syntaxe :

```
CREATE TABLE nom_table (nom_col1 TYPE1, nom_col2 TYPE2, ...)
```

Les types de données utilisés dans une table :

Ces types sont définit par le SGBD, Ils peuvent varier d'un SGBD à l'autre, surtout par leur nom et leur caractéristiques précises. Mais les types généraux restent toujours les mêmes.

Nous listerons dans ce qui suit quelques types de données définis par le SGBD Oracle.

- `NUMBER (n, [d])` : nombre de n chiffres [optionnellement *d* après la virgule]
- `CHAR(n)` : chaîne de caractères de longueur fixe, n représente la longueur maximale de la chaine.
- `VARCHAR2(n)` : chaînes de caractères de longueur variable.
- `DATE` : une donnée de type date, inclut un temps en heure, minutes, secondes

✍ Quand on crée une table, il faut définir les contraintes d'intégrité que devront respecter les données que l'on mettra dans la table.

1.2. Contraintes d'intégrité

Une contrainte d'intégrité est une assertion vérifiée par les données de la base à tout moment. L'objectif des contraintes est d'assurer la cohérence logique de la base de données.

On distingue :

- *Contrainte d'intégrité de domaine* : s'assurer de la validité des valeurs d'un attribut. C'est pourquoi la commande de création de table doit préciser, en plus du nom, le type de chaque colonne.
- *Contrainte d'intégrité de relation* : contrainte qui porte sur la clé primaire (not NULL et unique dans la table).
- *Contrainte d'intégrité de référence* : les données dans une table dépendent des valeurs d'attributs situées dans d'autres.

Syntaxe :

A la création d'une table, les contraintes d'intégrité se déclarent de la façon suivante :

```
CREATE TABLE nom_table (
nom_col_1  type_1 [CONSTRAINT nom_1_1]  contrainte_de_colonne_1_1
                  [CONSTRAINT nom_1_2]  contrainte_de_colonne_1_2 ,
nom_col_2  type_2 [CONSTRAINT nom_2_1] contrainte_de_colonne_2_1,
[CONSTRAINT nom_1] contrainte_de_table_1,
[CONSTRAINT nom_2] contrainte_de_table_2,
... ...
[CONSTRAINT nom_p] contrainte_de_table_p
)
```

✍ Une contrainte de colonne est une contrainte qui porte sur une seule colonne. Celle de table porte sur une ou plusieurs colonnes.

Les différentes contraintes que l'on peut déclarer sont les suivantes :

- `NOT NULL` ou `NULL` : Interdit (NOT NULL) ou autorise (NULL) l'insertion de valeur NULL pour cet attribut.
- `UNIQUE` : interdit qu'une colonne (ou la concaténation de plusieurs colonnes) contienne deux valeurs identiques.
- `PRIMARY KEY` : définit la clé primaire de la table. la contrainte PRIMARY KEY est totalement équivalente à la contrainte UNIQUE NOT NULL.
- `REFERENCES` table [(colonne)] [ON DELETE CASCADE]
- `FOREIGN KEY (colonne1, colonne2,...) REFERENCES tableref [(col1, col2,...)][ON DELETE CASCADE]`: indique que la concaténation de colonne1, colonne2,... est une clé étrangère qui fait référence à la concaténation des colonnes col1, col2,... de la table tableref. Si aucune colonne de tableref n'est indiquée, c'est la clé primaire de tableref qui est prise par défaut.

  L'option ON DELETE CASCADE indique que la suppression d'une ligne de tableref va entraîner automatiquement la suppression des lignes qui la référencent dans la table. Si cette option n'est pas indiquée, il est impossible de supprimer des lignes de tableref qui sont référencées par des lignes de la table.
- `CHECK (condition)` : donne une condition que les colonnes de chaque ligne devront vérifiée lors de l'insertion.

– `DEFAULT` valeur **:** permet de spécifier la valeur par défaut de l'attribut.

`Exemple` : Soit le schéma relationnel minimaliste suivant :

– acteur (num_act, nom, prénom)

– jouer(numero_act#, numero_film#)

– film(num_film, titre, annee)

```
Create table acteur (num_act number (8) constraint PK_act primary key, nom varchar2 (20), prenom varchar2 (20));

Create table film (num_film number (5) primary key, titre  varchar2 (30), annee date);

Create table jouer (numero_act number(8),numero_film number(5), titre varchar2 (30), annee date,primary key (numero_act, numero_film), constraint FK_numact foreign key (numero_act) references acteur (num_act), constraint FK_numfilm foreign key (numero_film) references film (num_film);
```

L'expression `constraint FK_numfilm foreign key (numero_film) references film (num_film)` peut être réduite à `references film (num_film)` à condition q'elle soit placée juste après la declaration du champs numero_film (cad `numero_film number(5) references film (num_film )`.

1.3. Suppression d'une table :

Supprimer une table revient à éliminer sa structure et toutes les données qu'elle contient. La syntaxe est la suivante :

```
DROP TABLE nom_table
```

1.4. Modification une table

a) Ajout ou modification de colonnes

```
ALTER TABLE Nom_de_la_table
ADD/MODIFY  Nom_de_la_colonne  Type_de_donnees
```

b) supprimer une colonne

```
ALTER TABLE Nom_de_la_table
DROP COLUMN Nom_de_la_colonne
```

c) Renommer une colonne

```
ALTER TABLE nom_table RENAME COLUMN ancien_nom TO nouveau_nom
```

d) Renommer une table

```
ALTER TABLE nom_table RENAME TO nouveau_nom
```
ou
```
Rename nom_table to nouveau_nom
```

Des contraintes d'intégrité peuvent être ajoutées ou supprimées par la commande ALTER TABLE.

Exemple :

departement (num,nom, lieu)

employe(matr,nom, prenom,num_dept#,poste)

```
Create table departement (num number(10),nom varchar2 (20),lieu varchar2(20) [constraint PK_dept] primary key(num));

Create table employe
(matr number,nom varchar2(30), prenom varchar2(30),num_dept number ,poste varchar2(30);
constraint PK_emp primary key (matr),
constraint nom_unique unique (nom),
constraint FK_dept foreign key (num_dept) references department(num),
constraint c_dept check (num_dept in(10,20,35,50)) );
```

→Ajout d'une colonne salaire

```
ALTER TABLE employe
       ADD salaire  number(10);
```

→suppresssion de la contrainte nom_unique et l'ajout d'une nouvelle contrainte verif_sal permettant de fixer le salaire minimal à 500

```
Alter table employe
      Drop constraint nom_unique
      Add constraint verif_sal check(salaire >=500)
```

✍ L'ajout de la contrainte verif_sal exige que chacune de valeurs des salaires déjà stockées, soit > 500, sinon l'ajout ne peut pas s'effectuer.

→Désactivation de la contrainte verif_sal

```
Alter table employe
      MODIFY CONSTRAINT verif_sal DISABLE
```

→Réactivation de la contrainte verif_sal

```
Alter table employe
      MODIFY CONSTRAINT verif_sal ENABLE
```

2. Manipulation des données

2.1. Insertion :

La commande INSERT permet d'insérer une ligne dans une table en spécifiant les valeurs à insérer. La syntaxe est la suivante :

```
INSERT INTO nom_table [(nom_col_1, nom_col_2, ...)] VALUES (val_1, val_2, ...)
```

La liste des noms de colonne est optionnelle. Si elle est omise, la liste des colonnes sera par défaut la liste de l'ensemble des colonnes de la table dans l'ordre de la création de la table. Si une liste de colonnes est spécifiée, les colonnes ne figurant pas dans la liste auront la valeur NULL.

Exemple :

```
(a) insert into departement values (10,'Finance','Tunis');
(b) insert into departement (lieu, nom_dept) values ('Production','Paris');
```

2.2. Modification :

La commande UPDATE permet de modifier les valeurs d'une ou plusieurs colonnes, dans une ou plusieurs lignes existantes d'une table.

La syntaxe est la suivante :

```
UPDATE nom_table
SET   col1 = exp1,
      col2 = exp2,...
WHERE  condition
```

Les valeurs des colonnes col1, col2, ..., sont modifiées dans toutes les lignes qui satisfont le prédicat 'predicat'. En l'absence d'une clause WHERE, toutes les lignes sont mises à jour.

Exemple :

a) Faire passer BEN AHMED dans le département 10 :

```
UPDATE employe
SET num_dept = 10
WHERE nom = 'BEN AHMED'
```

(b) Augmenter de 10 % les commerciaux :

```
UPDATE employe
SET sal = sal * 1.1
WHERE poste = 'COMMERCIAL'
```

2.3. Suppression :

La commande DELETE permet de supprimer des lignes d'une table.

La syntaxe est la suivante :

```
DELETE FROM nom_table
WHERE predicat
```

Toutes les lignes pour lesquelles predicat est évalué à vrai sont supprimées. En l'absence de clause WHERE, toutes les lignes de la table sont supprimées.

Exemple :

```
DELETE FROM departement
WHERE num_dept = 10
```

2.4. Recherche des données

L'ordre permettant de retrouver des informations stockées dans la base est SELECT.

Une requête SELECT possède six clauses différentes, dont seules les deux premières sont obligatoires. Elles sont données ci-dessous, dans l'ordre dans lequel elles doivent apparaître, quand elles sont utilisées :

```
SELECT ...
FROM ...
WHERE ...
GROUP BY ...
HAVING ...
ORDER BY ...
```

– *Clause SELECT*

Cette clause permet de spécifier les attributs que l'on désire voir apparaître dans le résultat de la requête

```
SELECT [DISTINCT] col1, col2,
FROM nom_table
where condition
```

```
SELECT *
FROM nom_table
where condition
```

Ou

```
SELECT exp1 [[AS] nom1 ], exp2 [[AS] nom2 ], .....
FROM nom_table
where condition
```

- Le mot clé facultatif DISTINCT permet d'éliminer les duplications : si, dans le résultat, plusieurs lignes sont identiques, une seule sera conservée.
- Le caractère *étoile* (`*`) signifie que toutes les colonnes de la table (indiquée par la clause from) sont sélectionnées.
- Le mot clé AS (optionnel) indique le titre de la colonne dans l'affichage du résultat de la sélection. Ces noms ne peuvent être utilisés dans les autres clauses (where par exemple).

– *Clause FROM*

La clause `FROM` spécifie la (les) table(s) sur lesquelles porte la requête ;

– *Clause WHERE*

Elle est facultative, énonce une condition que doivent respecter les données sélectionnés.

Exemples :

a. Afficher les noms de tous les départements

```
select nom from departement
```

b. Afficher les différents noms des départements

```
select distinct nom from departement
```

c. Afficher la liste de tous les départements (numero,nom et lieu)

```
Select * from departement
```

d. On veut la matricule d'employé, le nom ainsi que le numéro de département des employés qui appartiennent au département n°30 et dont leur fonction est ouvrier.

```
select matr,nom ,num_dept
from employe
where num_dept = 30 and poste = "Ouvrier" ;
```

e. Quels sont les noms des employés qui fonctionnent en tant que désigneur ou programmeurs.

```
select nom
from employe
where poste = "Ouvrier" or poste="Désigneur";
```

f. Afficher pour chaque employé son nom ainsi que la somme de son salaire et sa commission.

```
Select nom, salaire+comm as sal_comm
from employe
```

✍ Il existe différentes catégories d'opérateurs pour exprimer une expression logique :

- Connecteurs logiques : AND,OR,NOR (negation de OR)
- Opérateurs arithmétiques : +,-,*,/
- Comparateurs arithmétiques : <,>,<=,>=,=,<> ou !=
- Autres comparateurs :
  - *between ... and* : Il permet de sélectionner une valeur se trouvant dans un intervalle
  - *Not between ... and* : Il permet de sélectionner les valeurs qui ne font pas partie de l'intervalle.
  - *In* : Ce comparateur sert à sélectionner une valeur parmi une liste de valeur.

- *Like :* Il permet de comparer teste l'égalité de deux chaînes en tenant compte des caractères jokers dans *la 2ème chaîne*
  - "_" remplace 1 caractère exactement
  - "%" remplace une chaîne de caractères de longueur quelconque, y compris de longueur nulle

g. Quels sont les employés qui ont un salaire compris entre 800 et 1000.

```
Select nom, prenom
From employe
Where salaire between 800 and 1000
```

h. Quels sont les employés dont leurs noms commençant par un "HA" ?

```
Select nom, prenom
From employe
Where nom like 'HA%'
```

i. Quels sont les employés (prénom) qui travaillent dans les départements situé à Tunis?

```
Select prenom
From employe , department
Where departement.num=employe.num_dept
                    And lieu="Tunis"
```

j. Quels sont les employés (nom, prénom) qui travaillent dans le Département "Recherche"?

```
Select                    e.nom,                    d.prenom
From        employe         e,        department         d
Where d.num=e.num_dept
  And d.nom="Recherche"
```

– *GROUP BY exp1, exp2,...*

Permet de grouper en une seule ligne toutes les lignes pour lesquelles exp1, exp2,... ont la même valeur.

exemple :

```
select num_dept, nom
```

```
from employe
group by num_dept
```

- *HAVING condtion*

  Sert à préciser quels groupes doivent être sélectionnés.

  Elle se place après la clause GROUP BY.

  La condition suit la même syntaxe que celui de la clause WHERE. Cependant, il ne peut porter que sur des caractéristiques de groupe : fonction de groupe ou expression figurant dans la clause GROUP BY.

Exemple :

```
select num_dept, COUNT(*)
from employe
where poste = "secretaire"
GROUP BY num_dept  HAVING COUNT(*) > 1
```

Traitements des résultats: Quelques fonctions sur les colonnes

- Opérateurs sur attributs numériques
  - SUM: somme des valeurs d'une colonne pour un groupe de ligne
  - AVG: moyenne
- Opérateurs sur tous types d'attributs
  - MIN: minimum
  - MAX: maximum
  - COUNT: permet de compter le nombre de lignes incluses dans une table colonne