Instructions SQL

Deuxième partie : Langage de manipulation des données

Correspondance avec l'algèbre relationnelle (modèle relationnel binaire)

Exemples utilisés à partir des tables créées ainsi :

```
CREATE TABLE personne
(
num             NUMERIC (5),
nom             CHARACTER (40) NOT NULL,
adresse         CHARACTER VARYING (200),
sexe            CHARACTER ,
age             NUMERIC(3),
UNIQUE (adresse)
PRIMARY KEY (num)
CHECK(age<150)
ON DELETE CASCADE
ON UPDATE CASCADE
…)

CREATE TABLE departement
(
numdep          NUMERIC (3) CONSTRAINT departement_PK PRIMARY KEY,
nomdep          CHARACTER(20) NOT NULL
UNIQUE (nomdep)
)
CREATE TABLE employe
(
num-emp         NUMERIC (5) CONSTRAINT employe_PK PRIMARY KEY,
numdep          NUMERIC (3),
nom             CHARACTER (40) NOT NULL,
adresse         CHARACTER VARYING (200),
sexe            CHARACTER NOT NULL,
telperso        tel,
telburo         tel UNIQUE,
indice-sal      NUMERIC (6) NOT NULL,
commission      NUMERIC (3,2),
FOREIGN KEY (numdep) REFERENCES departement
CHECK (sexe IN ('M', 'F')),
```

1. Projection SELECT FROM

Syntaxe

SELECT [ALL | DISTINCT] liste d'attributs FROM <nom table>

Correspond à la projection de la table <nomtable> sur la liste d'attributs.

ALL permet de visualiser toutes les lignes obtenues, même les doublons
DISTINCT permet de ne garder que les lignes différentes.
SELECT * FROM <nom table> visualise toute la table.

2. Sélection SELECT FROM WHERE

2.1. Sélection d'enregistrements complets :

Syntaxe
SELECT * FROM <nom table> WHERE [<conditions de sélection>]
<conditions de sélection> : :=<expression logique simple> | <expression logique avec requête>

1.1.1. Expression logique simple

SELECT * FROM personne WHERE num=135

1.1.2. Expression logique avec requête

SELECT * FROM employe WHERE Indice-sal > (SELECT AVG (Indice-sal) FROM employe)
Les principales fonctions numériques sont AVG (moyenne) SUM (somme) etc.

1.2. Sélection et projection

Syntaxe
SELECT [ALL | DISTINCT] liste d'attributs FROM <nom table> WHERE [<conditions de sélection>]
<conditions de sélection> : :=<expression logique simple> | <expression logique avec requête>

3. Union de tables UNION

Syntaxe :
<nomtable>|<requête> UNION [ALL] <nomtable>|<requête>

Table1 UNION Table 2 donne une table dont les colonnes sont les unions des colonnes de chaque table.
Si ALL n'est pas spécifié alors les doublons sont éliminés.
Exemple avec requête :
SELECT * FROM personne WHERE num > 125 UNION SELECT * FROM personne WHERE num< 25

4. Intersection de tables INTERSECT

Syntaxe :
<nomtable>|<requête> INTERSECT [ALL] <nomtable>|<requête>
mêmes aspects syntaxiques que l'UNION, la sémantique étant de créer une table dont les colonnes correspondent aux valeurs communes des deux tables.

5. Différence entre deux tables EXCEPT

Complémentaire de l'intersection de tables.

Syntaxe :
<nomtable>|<requête> INTERSECT [ALL] <nomtable>|<requête>
Exemple
SELECT * FROM
(SELECT nom FROM personne EXCEPT
SELECT nom FROM personne WHERE age > 25)
Sélectionne les noms des personnes de moins de 26 ans.

6. Equijointure avec SELECT WHERE

Il existe quatre type d'opérateurs JOIN dans SQL III mais on ne verra ici que la jointure la plus courante réalisée grâce à la traduction en SQL de la sémantique de la jointure relationnelle binaire (sélection de valeurs égales aux colonnes communes dans le produit cartésien.
Syntaxe :
SELECT liste de sélection FROM table1, table2
WHERE table1.attribut=table2.attribut
Le produit cartésien est ici représenté par la virgule entre les noms de table.

7. Quelques mots-clés intéressants

7.1. IN

Indique dans quel ensemble il faut rechercher les valeurs.
Permet des requêtes imbriquées.
Syntaxe
IN <nomtable> | (<requête>)
Exemple
SELECT nump-emp, indice-sal
FROM employe
WHERE num-emp IN
(SELECT num-emp, numdep FROM employe, departement
WHERE employe.numdep =departement.numdep
AND departement.nomdep = 'ADMINISTRATION')
Donne les numéros, les indices salariaux des employés du département admistratif.

1.2. GROUP BY

Produit une table ordonnée selon les valeurs des colonnes qui suivent le GROUP BY. Sont mis ensemble tous les enregistrements ayant les mêmes valeurs pour ces colonnes.
Syntaxe
GROUP BY <nom-colonne> [<clause de motif>] [<{nom -colonne> [<clause de motif>]}…]
Exemple
SELECT nom, sexe FROM personne GROUP BY sexe
Crée une table des noms de personnes triées selon le genre.

1.3. COUNT

Permet de compter le nombre d'enregistrements.
Exemple
SELECT sexe, COUNT(*) FROM personne GROUP BY sexe
Permet de compter le nombre de personnes de chaque genre.

1.4. HAVING

Permet de selectionner sur une condition (ou plusieurs) des enregistrements groupés (si il y a une clause GROUP BY) sinon c'est toute la table qui est sélectionnée.
Syntaxe
HAVING <conditions de recherche>
Exemple :

```
SELECT nomdep IN (
SELECT nomdep, numdep, num-emp, COUNT(*) FROM employe, departement
    WHERE employe.numpdep = departement.numdep)
HAVING COUNT (*) > 20
```

Donne les noms des départements ayant plus de 20 employés.
ALTER TABLE enployé ALTER COLUMN Indice-sal SET DEFAUT 100

8. Prédicats (éléments d'expressions logiques) et constantes logiques

Mots clés	commentaire	type
TRUE/FALSE/UNKNOWN	Vrai , faux , inconnu	constante booléenne
IS/IS NOT	S'utilise avec TRUE/FALSE/UNKNOWN Et avec NULL	Opérateur booleen ou avec NULL
NOT	Falsifie une variable booleenne ou une expression logique	Operateur logique
EQUAL	Egalité	Opérateur logique
AND/OR	Et et ou logiques	Opérateurs logiques
>,< ,=		Opérateurs arithmétiques de comparaison
BETWEEN (NOT BETWEEN)	Correspond à a1<x<a2 si x est numérique . Est vrai si x est dans l'intervalle considéré.	Opérateur logique
LIKE(NOT LIKE)	Teste l'egalité avec un motif alphabétique dans une chaîne de caractères	Opérateur logique
NULL	Teste si tous les enregistrements d'une table donnée ont bien l'état « null » pour les éléments référencés.	Predicat
EXISTS (NOT EXISTS)	Prédicat testant la valeur d'une sous-requête. Est à vrai si cette sous-requête produit au moins un enregistrement. Permet de simuler le quotient.	Predicat