Cours 3 : Le langage SQL (partie 2) sous-requêtes et structure

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Principaux axes du cours

- Connecteurs In, Exists, Any, All
- Sous-requêtes indépendantes
- Sous-requêtes corrélées
- Opérateur division
- Modification des tables et des tuples
- Gestion des privilèges
- Compléments les vues et les index

Sous-requêtes

Syntaxe d'une requête contenant une sous-requête R

SELECT … FROM … WHERE Bloc contenant (R)

- Une sous-requête (aussi appelé requête imbriquée) est une requête SQL encapsulée à l'intérieur d'une autre requête

Exemple :

SELECT * FROM Etudiant WHERE AgeE = (R) avec R : SELECT AVG(AgeJ) FROM JTennis

SELECT * FROM Etudiant WHERE AgeE = (SELECT AVG(AgeJ) FROM JTennis)

- Donne les étudiants dont l'âge correspond à l'âge moyen des joueurs de tennis

Connecteurs (1)

Principaux connecteurs : =, !=, <, >

- Comparaison avec une seule valeur

Exemple : SELECT * FROM table WHERE Attribut1 = 'valeur'

Les connecteurs : IN, EXISTS, ANY, ALL

- Comparaison avec plusieurs valeurs
- Ces valeurs sont issues du résultat d'une requête R = ('valeur1', 'valeur2', …)

Exemple : **IN**

SELECT * FROM table WHERE Attribut1 IN (R)

- La condition WHERE est vérifiée si Attribut1 se trouve dans R

Connecteurs (2)

Exemple : **EXISTS**

SELECT * FROM table WHERE EXISTS (R)

- La condition WHERE est vérifiée si R n'est pas vide

Exemple : **ANY**

SELECT * FROM table WHERE Attribut1 > ANY (R)

- La condition WHERE est vérifiée si Attribut1 est supérieur à un élément de R

Exemple : **ALL**

SELECT * FROM table WHERE Attribut1 > ALL (R)

- La condition WHERE est vérifiée si Attribut1 est supérieur à tout élément de R

Sous-requêtes indépendantes (1)

Sous-requêtes indépendantes = blocs indépendants

SELECT … FROM … WHERE Bloc contenant (R)

Bloc principal

- Dans une sous-requête **indépendante**, le bloc contenant R peut être évalué séparément du bloc principal
- La sous-requête R ne contient pas d'attribut provenant du bloc principal

Exemple :

```
SELECT * FROM Etudiant WHERE AgeE = (SELECT AVG(AgeJ) FROM JTennis);
```

Sous-requêtes indépendantes (2)

Les sous-requêtes indépendantes peuvent jouer le rôle d'une jointure

Exemple : Le nom des étudiants qui habitent Rabat

- En jointure :

```
SELECT Etudiant.nom FROM Etudiant, Adressse WHERE Etudiant.Adresse = Adresse.Num
and Adresse.Ville = 'Rabat';
```

- En sous-requêtes indépendantes :

```
SELECT Etudiant.nom FROM Etudiant WHERE Etudiant.Adresse IN (SELECT Adresse.Num
FROM Adresse WHERE Ville = 'Rabat';
```

Sous-requêtes indépendantes (3)

Les sous-requêtes indépendantes peuvent jouer le rôle d'une jointure

Exemple : Le nom des produits vendus le 01/01/2018

- En jointure :

```
SELECT Produit.nom FROM Produit, Vente WHERE Vente.RefP = Produit.Reference and Vente.date = DATE(01/01/2018);
```

- En sous-requêtes indépendantes :

```
SELECT Produit.nom FROM Produit WHERE Produit.Reference IN (SELECT Vente.RefP FROM Vente WHERE date = DATE(01/01/2018);
```

Sous-requêtes indépendantes (4)

Duplication d'une table pour préserver l'indépendance

- L'idée est de dupliquer la table en lui donnant un nouveau nom dans la sous-requête

Exemple : Les produits dont le prix est supérieur au prix moyen des produits

```
SELECT * FROM Produit, WHERE Produit.Prix > (SELECT AVG(P.Prix) FROM Produit P);
```

Sous-requêtes indépendantes (5)

Duplication d'une table pour préserver l'indépendance

Exemple : Les produits dont le prix est le plus élevé

```
SELECT * FROM Produit, WHERE Produit.Prix >= ALL (SELECT P.Prix FROM Produit P);
```

Exemple : Les produits dont le prix n'est pas le plus élevé

```
SELECT * FROM Produit, WHERE Produit.Prix < ANY (SELECT P.Prix FROM Produit P);
```

Sous-requêtes corrélées (1)

Sous-requêtes corrélées = blocs dépendants

SELECT … FROM … WHERE Bloc contenant (R)

Bloc principal

- Dans une sous-requête **corrélées**, le bloc contenant R ne peut pas être évalué séparément du bloc principal
- La sous-requête R contient des attributs provenant du bloc principal

Exemple :

```
SELECT * FROM Professeur WHERE Age > (SELECT AVG(P.Age) FROM Professeur P
WHERE Professeur.Ville = P.Ville);
```

Sous-requêtes corrélées (2)

Exemple : Les professeurs dont l'âge est supérieur à la moyenne d'âge de leurs villes

```
SELECT * FROM Professeur WHERE Age > (SELECT AVG(P.Age) FROM Professeur P
WHERE Professeur.Ville = P.Ville);
```

Exemple : Les produits dont le prix est supérieur à la moyenne des prix de leurs marques

```
SELECT * FROM Produit WHERE Prix > (SELECT AVG(P.Prix) FROM Produit P
WHERE Produit.Marque = P.Marque);
```

Sous-requêtes corrélées (3)

Sous-requêtes corrélées et jointure

Exemple : Les villes dont aucun professeur n'est responsable du module base de données SL2IBD

```
SELECT Ville FROM Professeur WHERE NOT EXISTS (SELECT * FROM Matiere,
Professeur P WHERE Matiere.Responsable = P.Numero and Matiere.Code = 'SL2IBD'
and Professeur.Ville = P.Ville);
```

Sous-requêtes corrélées (4)

Sous-requêtes corrélées et jointure

Exemple : Les marques dont aucun produit n'a été vendu le 01/01/2018

```
SELECT Marque FROM Produit WHERE NOT EXISTS (SELECT * FROM Vente,
Produit P WHERE Vente.RefP = P.Reference and Vente.date = Date (01/01/2018) and
Produit.Marque = P.Marque);
```

Opérateur division (1)

Définition : C'est une relation composée des tuples tels que le produit cartésien avec le diviseur soit un sous-ensemble de la relation dividende

R = A / B

Table : A

attr1	attr2
a	1
a	2
a	3
b	1
c	2

Table : B

attr2
1
2

Table : R

attr1
a

Opérateur division (2)

La division : c'est les valeurs de A.attr1 pour lesquelles :
Il n'existe pas de valeur B.attr2 tel que (A.attr1, B.attr2) n'appartienne pas à A

R = A / B

Table : A

attr1	attr2
a	1
a	2
a	3
b	1
c	2

Table : B

attr2
1
2

Table : R

attr1
a

Opérateur division (3)

La division : c'est les valeurs de A.attr1 pour lesquelles :
Il n'existe pas de valeur B.attr2 tel que (A.attr1, B.attr2) n'appartienne pas à A

R = A / B

Requête SQL :

```
SELECT DISTINCT A.attr1 FROM A AS A1WHERE
NOT EXISTS (SELECT * FROM B AS B0 WHERE NOT EXISTS
(SELECT * FROM A AS A2 WHERE A1.attr1 = A2.attr1 and A2.attr2 = B0.attr2))
```

Opérations sur les tables (1)

Principales opérations sur les tables

- Création

CREATE TABLE

- Modification

ALTER TABLE

- Suppression

DROP TABLE

Opérations sur les tables (2)

Création des tables Produit, Client et Vente

- Table Produit

```
CREATE TABLE Produit ( ref CHAR(4), nom VARCHAR(20), marque VARCHAR(20) );
```

- Table Client

```
CREATE TABLE Client ( num INT(10), nom VARCHAR(20), adresse VARCHAR(30) );
```

- Table Vente

```
CREATE TABLE Vente ( num INT(10), refP CHAR(4), refC INT(10), date DATE );
```

Opérations sur les tables (3)

Spécification des attributs non nuls et uniques et de la clé primaire

- Table Produit

```
CREATE TABLE Produit ( ref CHAR(4) NOT NULL,
                       nom VARCHAR(20) NOT NULL,
                       marque VARCHAR(20),
                       PRIMARY KEY (ref),
                       UNIQUE (nom)
                       );
```

Opérations sur les tables (4)

Spécification des clés étrangères

- Table Vente

```
CREATE TABLE Vente ( num INT(10) NOT NULL,
                     refP CHAR(4) NOT NULL,
                     refC INT(10) NOT NULL,
                     date DATE,
                     PRIMARY KEY (num),
                     FOREIGN KEY (refP) REFERENCES Produit (ref),
                     FOREIGN KEY (refC) REFERENCES Client (num),
                     );
```

Opérations sur les tables (5)

Gestion de l'intégrité référentielle

- Table Vente

```
CREATE TABLE Vente ( num INT(10) NOT NULL,
                     refP CHAR(4) NOT NULL, refC INT(10) NOT NULL,
                     date DATE, PRIMARY KEY (num),
                     FOREIGN KEY (refP) REFERENCES Produit (ref),
                     ON DELETE SET NULL ON UPDATE CASCADE
                     FOREIGN KEY (refC) REFERENCES Client (num),
                     ON DELETE SET NULL ON UPDATE CASCADE
                     );
```

Opérations sur les tables (6)

Gestion de l'intégrité référentielle

- ON UPDATE (mise à jour), ON DELETE (suppression)

```
FOREIGN KEY (refP) REFERENCES Produit (ref),
ON DELETE SET NULL ON UPDATE CASCADE
```

- En cas de mise à jour de la table Produit -> Mise à jour de refP
- En cas de suppression de la table Produit -> refP = NULL

Attention : Dans cette exemple, refP et refC peuvent être NULL

Opérations sur les tables (7)

Modification de la structure

- ALTER TABLE Produit MODIFY nom VARCHAR(40);
- ALTER TABLE Produit ADD stock VARCHAR(20);
- ALTER TABLE Client ALTER adresse SET DEFAULT 'Rabat';
- ALTER TABLE Client DROP adresse;
- ALTER TABLE Client ADD CONSTRAINT CT1 UNIQUE(nom);
- ALTER TABLE Client ADD CONSTRAINT CT2 PRIMARY KEY(num);
- ALTER TABLE Vente ADD CONSTRAINT CT3 FOREIGN KEY(refC) REFERENCES Client (num);

Opérations sur les tuples (1)

Principales opérations sur les tables

- Insertion

INSERT INTO

- Modification

UPDATE

- Suppression

DELETE FROM

Opérations sur les tuples (2)

Insertion de données

- Avec désignation des colonnes

```
INSERT INTO Produit ( ref, nom, marque ) VALUES ( 'MB23', 'AMG C63', 'Mercedes' );
```

- Selon l'ordre par défaut des colonnes

```
INSERT INTO Produit VALUES (DEFAULT, 'MB23', 'AMG C63', 'Mercedes' );
```

- A partir d'une autre table

```
INSERT INTO Produit ( ref, nom, marque ) SELECT P.ref, P.nom, P.marque FROM
OldProduit P WHERE P.marque = 'BMW';
```

Opérations sur les tuples (3)

Modification de données

- Modification d'un tuple

```
UPDATE Produit SET nom = '300 SLS' WHERE ref = 'MB75' ;
```

- Modification d'une collection de tuples

```
UPDATE Produit SET prix = prix * 0,9 WHERE ref NOT IN
(SELECT refP FROM Vente WHERE quantite > 10) ;
```

Opérations sur les tuples (4)

Suppression de données

- Suppression de tous les tuples

```
DELETE FROM Produit;
```

- Suppression avec condition

```
DELETE FROM Produit WHERE ref IN
(SELECT refP FROM Vente WHERE date = DATE (01/01/2018) ;
```

Gestion des privilèges (1)

Ajout et suppression de droits

- Ajouter des droits

GRANT privileges ON database.table TO user

- Enlever des droits

REVOKE privileges ON database.table FROM user

ALL [PRIVILEGES]	Tous les droits
ALTER	Autorise l'utilisation de ALTER TABLE
CREATE	Autorise l'utilisation de CREATE TABLE
DELETE	Autorise l'utilisation de DELETE
DROP	Autorise l'utilisation de DROP TABLE
…	…

Gestion des privilèges (2)

Création d'un nouvel utilisateur et affectation des droits

- Création d'un utilisateur

```
CREATE USER 'saad'@'localhost' IDENTIFIED BY 'password';
```

- Lui donner tous les droits sur toutes les tables de db1

```
GRANT ALL ON db1.* TO 'saad'@'localhost';
```

Les vues

Une vue est une table virtuelle dérivée d'une ou plusieurs tables de bases

- Création d'une vue

```
CREATE VIEW ProduitStar (ref, prix, nom, marque) AS SELECT FROM Vente V, Produit P
WHERE V.refP =P.ref and V.quantite > 100;
```

- Plusieurs représentations d'une même table
- Affichage des données selon les besoins de l'utilisateur
- Restreindre l'accès aux données en fonction de l'utilisateur

Les index (1)

Un index permet au SGBD de retrouver rapidement les données, il existe 3 types d'index : PRIMARY KEY, UNIQUE, ou INDEX

- Une clé primaire est strictement unique, les NULL ne sont pas autorisés
- Un index de type UNIQUE est comparable à une clé primaire, avec les valeurs NULL autorisées et potentiellement en plusieurs occurrences
- Un index de type INDEX signifie que l'on souhaite indexer une colonne susceptible de contenir des doublons

Les index (2)

La syntaxe pour créer un index est la suivante

```
CREATE INDEX index1 ON Client(nom, prenom)
```

- Lors des requêtes, l'index est en permanence comparé, il est donc utile de sélectionner des index peu volumineux (exemple : TINY INT)
- Les index sont à choisir parmi les champs concernés par une clause WHERE, ORDER BY, GROUP BY, MIN(), MAX(), ainsi que les champs qui permettent de relier des tables entre elles