Cours de Base de Données
Cours n.6
Le langage SQL (partie II)

L2 Informatique - Portail Siences et Technologies

Elisabetta De Maria - http://www.i3s.unice.fr/∼edemaria/

DS4H et Laboratoire I3S, CNRS

2023-2024

Université Côte d'Azur

Requêtes imbriquées (1)

SELECT *liste d'attributs recherchés*

FROM *liste_des_relations*

WHERE ***bloc SFW dans la condition***

Intérêt: indiquer qu'un attribut doit prendre ses valeurs dans une liste de valeurs définies par un autre bloc *SWF*

- Itérations imbriquées
- autre façon de faire certaines formes de jointure
- Sous-requêtes indépendantes ou pas

UNIVER
CÔTE D'A

Requêtes imbriquées (2)

Principaux connecteurs: = != > < IN EXISTS ANY ALL

EXISTS *R* : retourne TRUE si *R* n'est pas vide, FALSE sinon

t **IN** *R* : retourne TRUE si t appartient à *R*, FALSE sinon

valeur *comp* **ANY** *R* : retourne TRUE si la comparaison avec **au moins un** des tuples de **R** renvoie TRUE

valeur *comp* **ALL** *R* : retourne TRUE si la comparaison avec **tous** les tuples de **R** renvoie TRUE

UNIVER
CÔTE D'A

Sous-requêtes indépendantes

Quelles sont les références produit dont le prix HT est supérieur au prix HT moyen des produits ?

```
SELECT référence
FROM  Produit
WHERE Produit.prix_ht  > (SELECT AVG(P.prix_ht)
                          FROM Produit P);
```

➡ Le bloc SFW imbriqué peut être évalué séparément du bloc principal

UNIVER
CÔTE D'A

Sous-requêtes indépendantes (1)

Quels sont les marques de produits vendus le 1/1/2007 ?

```
SELECT marque
FROM  Produit, Vente
WHERE Vente.ref-produit = Produit.référence
          AND date = 1/1/2007;
```

ou

```
SELECT Produit.marque
FROM  Produit
WHERE Produit.référence IN
                    (SELECT Vente.ref-produit
                     FROM Vente
                     WHERE date = 1/1/2007);
```

Jointure ou requête imbriquée

UNIVER
CÔTE D'A

Sous-requêtes indépendantes (2)

Quelles sont les marques de produits qui n'ont pas le prix HT le plus élevé ?

SELECT marque
FROM *Produit* < SELECT max (P.prix_ht) …
WHERE Produit.prix-ht < **ANY**
(**SELECT** *P*.prix-ht
FROM *Produit P;*)

Quelles sont les marques de produits qui ont le prix HT le plus élevé ?

SELECT marque > SELECT max (P.prix_ht) …
FROM *Produit*
WHERE *Produit*.prix_ht >= **ALL**
(**SELECT** *P*.prix_ht **FROM** *Produit P;*)

UNIVER
CÔTE D'A

Sous-requêtes corrélées (1)

Quels produits dont le prix HT est supérieur au prix HT moyen des produits de la même marque ?

```
SELECT Produit.référence
FROM  Produit
WHERE Produit.prix-ht > (SELECT AVG(P.prix-ht)
                         FROM Produit P
                         WHERE Produit.marque = P.marque);)
```

Le bloc *SFW* principal et le bloc *SFW* imbriqué doivent être évalués simultanément

UNIVER
CÔTE D'A

Sous-requêtes corrélées (2)

Quels sont les marques dont aucun produit n'a été vendu le 1/01/2007 ?

```
SELECT Produit.marque
FROM  Produit
WHERE NOT EXISTS
        (SELECT Vente.ref-produit
         FROM Vente, Produit P
          WHERE Vente.ref-produit = P.référence and
                P.marque = Produit.marque and
                date = 1/01/2007;)
```

UNIVER
CÔTE D'A

Opérateur *Division*

R1(a,b) ÷ R2(b)

Afficher les valeurs de R1.a pour lesquelles :
il n'existe pas de valeur de R2.b telle que :
(R1.a, R2.b) n'appartienne pas à R1

Département

numéro-département	...	responsable

Employé

numéro-employé	département	...	fonction

Donner le numéro du département, s'il existe, qui emploie tous les ingénieurs système. (cas particulier: un employé ∈ un seul dept.)

```
SELECT D.numéro-département
FROM Département D
WHERE not exists (SELECT * FROM Employe E
                  WHERE E.fonction = 'Ingénieur Système'
                  AND E.département != D.numéro-département);
```

UNIVER
CÔTE D'A

Opérateur *Division*

R1(a,b) ÷ R2(b) : Afficher les valeurs de R1.a pour lesquelles : il n'existe pas de valeur de R2.b telle que (R1.a,R2.b) n'appartienne pas à R1

Département-Employé

num-département	...	num-employé

Employé

num-employé	...	...	fonction

Donner le numéro du département, s'il existe, qui emploie tous les ingénieurs système.

```
SELECT distinct DE1.num-département
FROM Département-Employé DE1
WHERE not exists
  (SELECT * FROM Employe E
   WHERE E.fonction = 'Ingénieur Système' AND NOT EXISTS
     (SELECT * FROM Département-Employé  DE2
     WHERE DE1.num-département = DE2.num-département
              AND DE2.num-employé = E.num-employé));
```

Opérateur *Division*

```
SELECT distinct département
FROM EMPLOYE
WHERE fonction = 'Ingénieur Système';
GROUP BY département
HAVING count(numéro-employé) =
    (SELECT count(numéro-employé) FROM EMPLOYE E)
     WHERE E.fonction = 'Ingénieur Système');
```

```
SELECT *
FROM EtudiantUE
GROUP BY etudiant
HAVING count(uniteValeur) =
    (SELECT count(code) FROM UE);
```

Exercices

Donner le nom des employés qui ont le même salaire que Dupont

Employé

numéro-employé	nom	...	salaire

```
SELECT Y.nom
FROM  Employé  X , Employé  Y
WHERE
        X.nom='Dupont'
        and Y.salaire = X.salaire
        and X.nom != Y.nom;
```

Remplacer la jointure par une requête imbriquée

UNIVER
CÔTE D'A

Manipulation des structures de données

- Schéma logique / Base de données
  → CREATE/DROP SCHEMA/DATABASE ...

- Schéma de ***tables*** *(relations)* et de leur contenu
  → CREATE/ALTER/DROP TABLE ...

- Définition des ***contraintes*** qui assurent des contrôles sur l'intégrité des données

- Mais encore :
  – les index
  – les utilsateurs et les privilèges

UNIVER
CÔTE D'A

Création/Suppression d'un schéma

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *db_name*

DROP DATABASE [IF EXISTS] *db_name*

Pas de message d'erreur si la BD existe!

- Dans la norme SQL: création de schéma
- MySQL: *schema* = *database*
- Il faut avoir les droits de le faire (administrateur?)
- Création d'un compte s'il n'existe pas
- Association des droits

CREATE DATABASE *lahire06-07*

GRANT ALL PRIVILEGES
ON *lahire06-07.**
TO ' *lahire06-07* '@localhost
IDENTIFIED BY ' *lahire06-07* ';

DROP DATABASE *lahire06-07*

Attention !

UNIVER
CÔTE D'A

Types de la norme ANSI

INTEGER	Entiers relatifs	4 octets
SMALLINT	Entiers relatifs	2 octets
BIGINT	Entiers relatifs	8 octets
FLOAT	Flottants	4 octets
DOUBLE	Flottants	8 octets
REAL	Flottants	4 ou 8 octets
NUMERIC(N,D)/ DECIMAL(N,D)	Décimaux à précision fixe	N octets

CHAR(M)	Chaînes de longueur fixe	M octets
VARCHAR(M)	Chaînes de longueur variable	au plus M octets
BIT VARYING	Chaînes d'octets	longueur de la chaîne
DATE	date(hour, mois, an)	4 octets
TIME	heure(h, mn, sh)	4 octets
DATETIME	date et heure	8 octets
YEAR	année	2 octets

Création et gestion des relations

- Structures de données (relations, attributs, tuples)

relation	→	**table**
attribut	→	**column**
tuple	→	**row**

- Instructions sur les relations

CREATE TABLE	*créer une relation*
DROP TABLE	*supprimer une relation*
ALTER TABLE	*modifier la structure d'une relation*

Création de table/relation (1)

```
CREATE TABLE Produit (
référence CHAR(4),
    nom VARCHAR(20),
    marque VARCHAR(15),
    constructeur        CHAR(20)
    );
```

```
CREATE TABLE Client (
numéro INT(10),
    nom VARCHAR(20),
    adresse VARCHAR(30),
    téléphone CHAR(10)
    );
```

```
CREATE TABLE Vente (
numéro INT(10),
    ref-produit INT(10),
    ref-client INT(10),
    date DATE
    );
```

- *Compléter la description*
- *Voir avec les types MySQL*

Création de table (valeurs nulles, défaut)

- NOT NULL : l'attribut correspondant doit *toujours* avoir une valeur

  nom **VARCHAR(20)** **NOT NULL**

- DEFAULT : définir la valeur d' un attribut par défaut

  adresse **VARCHAR (30)** **DEFAULT** 'Inconnue'

```
CREATE  TABLE  Produit (
référence CHAR(4) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    marque VARCHAR(15),
    constructeur        CHAR(20) DEFAULT 'Renault'
    );
```

UNIVER
CÔTE D'A

Création de table (clé primaire)

Identification des tuples

PRIMARY KEY (*att*)

PRIMARY KEY (*att1*, ..., *attn*)

→ mono attribut ou multi attributs

→ Attributs figurant dans une clé : déclaration NOT NULL

```
CREATE  TABLE  Produit (
référence CHAR(4) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    marque VARCHAR(15),
    constructeur        CHAR(20) DEFAULT 'Renault',
    PRIMARY KEY (référence)
    );
```

UNIVER
CÔTE D'A

Création de table (unicité)

Unicité des valeurs dans une colonne

UNIQUE (nom, prenom)

→ Principalement pour les clés secondaires

```
CREATE  TABLE  Produit (
référence CHAR(4) NOT NULL,
    nom VARCHAR(20) NOT NULL,
    marque VARCHAR(15),
    constructeur        CHAR(20),
    PRIMARY KEY (référence),
    UNIQUE (nom)
    );
```

NOT NULL non spécifié

UNIQUE ne s'applique pas aux valeurs nulles

UNIVER
CÔTE D'A

Création de table (clé étrangère)

FOREIGN KEY (att) REFERENCES ...
FOREIGN KEY (att1, ..., attn) REFERENCES ...

FOREIGN KEY(ref-produit) **REFERENCES** *Produit*
→ ref-produit référence la clé primaire de la table **Produit**

```
CREATE TABLE Vente (
numéro INT(10) NOT NULL,
    ref-produit INT(10) NOT NULL,
    ref-client INT(10) NOT NULL,
    date DATE,
    PRIMARY KEY (numéro),
    FOREIGN KEY (ref-produit) REFERENCES Produit,
    FOREIGN KEY (ref-client) REFERENCES Client
    );
```

UNIVER
CÔTE D'A

Gestion de l'intégrité référentielle

Vérification, si insertion, suppression, mise à jour …

Les valeurs prises par la clé étrangère correspondent-elles à la clé?

Action par défaut : **rejet** de l'opération

Autres actions si celle par défaut ne convient pas

- **ON UPDATE** *(en cas de mise à jour)*
- **ON DELETE** *(en cas de suppression)*

Objectif: faire respecter la contrainte

- **SET NULL** clé étrangère mise à NULL
- **CASCADE** application de la même opération
- **SET DEFAULT** valeur par défaut pour la clé étrangère

UNIVER
CÔTE D'A

Intégrité référentielle (exemple)

```
CREATE TABLE Vente (
numéro INT(10) NOT NULL,
    ref-produit INT(10) NOT NULL,
    ref-client INT(10) NOT NULL,
    date DATE,
    PRIMARY KEY (numéro),
    FOREIGN KEY (ref-produit) REFERENCES Produit
    ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY (ref-client) REFERENCES Client
    ON DELETE SET NULL ON UPDATE CASCADE
    );
```

- Si suppression dans Produit : ref-produit = NULL
- Si mise à jour dans Produit : ref-produit = mise à jour

UNIVER
CÔTE D'A

Mise à jour d'un schéma de BD

- Suppression d'une relation

  DROP TABLE *nom de relation ;*

- Modification d'une table

  ALTER TABLE *nom de_relation* ACTION *description*

  ADD, MODIFY, ALTER, DROP, RENAME

  description commande

Mais aussi :

ALTER TABLE Vente TYPE = *innodb*

UNIVER
CÔTE D'A

Modification d'une table (structure)

ALTER TABLE *Produit* MODIFY nom VARCHAR(25);

ALTER TABLE *Produit* ADD quantité-stock VARCHAR(20);

ALTER TABLE *Client* ALTER adresse SET DEFAULT '*NICE*';

ALTER TABLE *Client* DROP adresse;

ALTER TABLE *Client*
ADD CONSTRAINT *CT_UN* UNIQUE (nom) ;

ALTER TABLE *Client*
ADD CONSTRAINT *CT_PR* PRIMARY KEY (numéro) ;

ALTER TABLE *Vente* ADD CONSTRAINT CT_ET
FOREIGN KEY (ref-client) REFERENCES *Client* ;

UNIVER
CÔTE D'A

Mises à jour des données

- **INSERT INTO**
  pour insérer des tuple

- **DELETE FROM**
  pour supprimer des tuples

- **UPDATE**
  pour mettre à jour des tuples

UNIVER
CÔTE D'A

Insertion de tuples (1)

Insertion avec désignation explicite des colonnes

```
INSERT INTO Produit (nom, marque, constructeur)
VALUES ('mercedes 300 SL', 'mercedes benz',
        'daimler');
```

Insertion dans l'ordre des colonnes

```
INSERT INTO Produit
VALUES (DEFAULT, 'mercedes 300 SL', 'mercedes
        benz', 'daimler');
```

UNIVER
CÔTE D'A

Insertion de tuples (2)

Insertion à partir d'une autre table

INSERT INTO *Produit* (nom, marque, constructeur)
SELECT P.nom, P.marque, p.constructeur
FROM *OldProduit P*
WHERE *P*.marque = '*Peugeot*'

➡ **Evolution du schéma relationnel**

Suppression de tuples

Suppression de tous les tuples

DELETE FROM *Produit*

ou

DELETE FROM *Produit* **WHERE** 1 > 0

Suppression conditionnelle

DELETE FROM *Produit*

WHERE marque = '*peugeot*'

```
DELETE FROM VENTE WHERE ref-produit IN
      (SELECT  référence
      FROM Produit
      WHERE Produit.marque = 'peugeot') ;
```

UNIVER
CÔTE D'A

Modification des tuples

```
UPDATE Produit
SET nom = 'mercedes 300 SLK'
WHERE numero =3;
```

Modification d'un tuple

```
UPDATE Produit
SET prix-ht = prix-ht * 0,9
WHERE référence NOT IN
      (SELECT ref-produit
       FROM VENTE
       WHERE quantité > 10);
```

Modification d'une collection de tuple

UNIVER
CÔTE D'A

Compléments: Définition de vues

Une vue est une table virtuelle dérivée de tables de base :

- On stocke seulement la définition de vue
- Son contenu est généré dynamiquement

```
CREATE VIEW ProduitPhare
AS
SELECT *
FROM Vente V, Produit P
WHERE
  V.ref-produit = P.référence
  AND V.quantité > 100 ;
```

```
CREATE VIEW VenteInfo
    (référence, date, marque)
AS
SELECT P.référence, V.date,
            P.marque
FROM  Vente V, Produit P
WHERE P.référence =V.ref-produit;
```

Evolution du schéma relationnel

Requêtes fréquentes

DROP VIEW VenteInfo

UNIVER CÔTE D'A

Compléments: Création de contraintes

CREATE ASSERTION
CHECK *pour décrire des contraintes générales*

```
CREATE ASSERTION QuantiteVendue
CHECK NOT EXISTS (
        SELECT V.quantité
        FROM Vente V, Produit P
        WHERE
          V.ref-produit = P.référence AND
          P.quantité-vendue < V.quantité);
```

On suppose que la table Produit contient la colonne quantité-vendue

UNIVER
CÔTE D'A

Compléments: Création d'index

CREATE INDEX

Un index offre un chemin d'accès aux lignes d'une table

CREATE INDEX index1 ON Client(nom,prenom);

un index est systématiquement défini sur la clé primaire de chaque table

pour chaque clause UNIQUE utilisée dans la création de la table, un index permet de vérifier rapidement, au moment d'une insertion, que la clé n'existe pas déjà

UNIVER
CÔTE D'A

Compléments: Gestion des privilèges

SHOW DATABASES ;
SHOW PRIVILEGES ;

GRANT *privilèges* ON ... TO ...

REVOKE *privilèges* ON ... FROM ...

ALL [PRIVILEGES]	Tous les droits sauf WITH GRANT OPTION
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
...	...

UNIVER
CÔTE D'A