Bases de Données (USSI0G)
Langage SQL

Thibault Bernard

Thibault.bernard@univ-reims.fr

Langage SL

Structered Query Langage

- Standard, langage relationnel incontournable
- 600 pages, cours donne un aperçu …

Historique

- 1970 Codd modèle relationnel
- 1974 SEQUEL SEQUEL/2 (System /R, IBM)
- 1981 Oracle (SQL)
- 1983 DB2 (IBM)
- 1986 SQL/86 normalisé
- 1989 SQL/89 (SQL/1)
- 1992 SQL/92 (SQL/2)
- 1999 SQL/3
- 2008 SQL/4
- 2011 SQL/5

SQL est:

- **LMD** Langage de manipulation de données
- **LDD** Langage de Définition de Données
- **LCD** Langage de Contrôle de Données

Une requête SQL peut être

- Interactive
- Incluse dans un programme d'application

Manipulation de données

Syntaxe Minimale

SELECT <liste d'attributs projetés> FROM <liste de relations>

- Syntaxe possiblement enrichie de très nombreuses clauses permettant d'exprimer
  - Projections
  - Restrictions
  - Jointures
  - Tris
  - …

Projection

- La projection s'exprime via la liste d'attributs :

  SELECT <liste attributs> FROM <liste relations>

- L'ensemble des attributs s'exprime par ***.**

- Contrairement à l'Algèbre relationnel, il n'y a pas suppression des doublons. On utilise alors le mot clé **DISTINCT**.

Restriction

- La restriction s'exprime derriere une clause **WHERE**

  SELECT <attributs> FROM <relations> WHERE <critères de restriction>

- Les mots clé suivant peuvent être utilisés dans un critère de restriction :
  - BETWEEN
  - IN
  - LIKE
  - NOT
  - NULL
  - Opérateurs de comparaison numérique
  - …

Tris

L'expression d'un tri s'effectue par le mot clé **ORDER BY :**

ORDER BY <attribut 1> [ASC / DESC] ,…, <attribut n> [ASC / DESC]

Exemple

Donner la liste des patients de reims triés sur leur nom dans l'ordre croissant et leur prenom dans l'ordre décroissant.

Présentation des colonnes

La présentation des colonnes s'exprime avec le mot clé **AS :**

SELECT <attributs> AS [nom apparaissant pour la colonne] FROM …

Exemple

Donner les numéros des affections et leur nomAff avec des noms de colonnes.

Jointures

- Produit cartésien
  - SELECT * FROM <relations>
  - Eviter les ambiguïtés en préfixant le nom des attributs utilisés par le nom de la relation concernée suivi d'un point : R.A
  - On peut aussi utiliser le mot clé AS dans la clause FROM pour utiliser des synonymes pour les relations.

- Jointure naturelle

  SELECT * FROM R,S WHERE R.A = S.A

Expression des jointures en SQL normalisé

- Jointure Naturelle

  SELECT * FROM table1 NATURAL JOIN table2 [USING (colonne1 [, colonne2 ...])]

- Jointure Interne

  SELECT * FROM Table1 INNER JOIN table2 ON <condition de jointure>

- Jointure Externe (permet de conserver les tuples d'une des deux relations)

  SELECT ... FROM <table gauche> LEFT | RIGHT | FULL OUTER JOIN <table droite 1> ON <condition de jointure> [LEFT | RIGHT | FULL OUTER JOIN <table droite 2> ON <condition de jointure 2>]

Calcul

- Attributs calculés :

  Donner le nombre d'affectations concernant le patient 133

- Des stats …

  AVG, COUNT, MAX, MIN, SUM

Pas d'imbrications, mais utilisable en sous requêtes ou en agrégats

Agrégats

- Un agrégat est un partitionnement horizontal d'une relation selon des valeurs d'un groupe d'attributs suivi d'un **regroupement** par une fonction de calcul.
- **Regroupement** : regrouper les données d'une table en sous-tables pour y faire des opérations par groupes.

  Group By <Attribut1> , … , <Attributp>
  - Groupe en une seule ligne toutes les lignes pour lesquelles les attributs de regroupement ont la même valeur
  - Cette clause se place juste après la clause Where ou après la clause From si la clause Where n'existe pas.
- **Restriction** sur les regroupements
  - HAVING <prédicat> : restriction sur les tuples de la relation obtenue après le calcul sur le regroupement.
  - Se place après la clause GROUP BY

Synthèse

SELECT <liste d'attributs projetés>

FROM <liste de relations> JOIN … ON <critères de jointure>

[WHERE < restrictions >]

[GROUP BY <liste d'attributs à partitionner>

[HAVING <condition>]]

[ORDER BY <attribut> [ASC/DESC] [<attribut> [ASC/DESC]] …]

Sous requêtes

- Where -> conditions par comparaisons sur des valeurs
- Expressions de conditions sur des relations ?
    - Sous requêtes: décrire des requêtes complexes permettant d'effectuer des opérations dépendant d'autres requêtes.
    - Utilisable derrière WHERE et HAVING
    - Sous requête renvoie
        - Une valeur unique : on utilise alors des opérateurs de comparaisons classiques
        - Un attribut => opérateurs IN, EXISTS, opérateurs de comparaisons classiques + ALL ou ANY

Mot Clé EXISTS

L'expression SQL EXISTS (SELECT… FROM…)

Est évalué à Vrai si et seulement si le résultat de l'évaluation du SELECT … FROM est non vide (le résultat donne au moins un tuple).

La requête « appelante » n'est évaluée que si le résultat de la sous requête est évaluée à vrai.

Division

- [Algèbre relationnelle]:La division de la relation R de schéma $R(A_1,\ldots,A_n)$ par la relation S de schéma $S(A_{p+1},\ldots,A_n)$ est la relation T de schéma $T(A_1,\ldots,A_p)$ formés de tous les tuples qui concaténées à chaque tuples de S donnent toujours un tuple de R.
- Notation R / S
- Opérateur type qui permet de répondre aux questions du type: Ç donner les docteurs qui soignent tous les patients

Division

- SQL n’offre pas la possibilité d’exprimer directement le quantificateur ∀. On utilise une négation du quantificateur ∃.
- La question « Donner le nom des fournisseurs livrant tous les produits » devient « Donner le nom des fournisseurs pour lesquels il n’existe aucun produit non livré »
- Procéder par étapes :

Division

- Division $R(A,B) \ / \ S(B) = T(A)$

  $=\{a \in R(A) \ / \ \forall \ b \in S(B), \ (a,b) \in R(A,B)\}$

  $=\{a \in R(A) \ / \ \nexists \ b \in S(B), \ (a,b) \notin R(A,B)\}$

- La traduction mot à mot donne

```
SELECT A FROM R AS R1 WHERE NOT EXISTS
        (SELECT B FROM S WHERE NOT EXISTS
        (SELECT A, B FROM R WHERE R1.A = A AND S.B = B ) )
```

Opérations ensemblistes

- UNION
- INTERSECT
- EXCEPT

s’intercalent entre deux SELECT

Définition de données

Création d'une Base de Données

CREATE DATABASE ma_base;

Créer une base de donnée de nom *ma_base*

DROP DATABASE ma_base;

Supprimer la base de données de *nom ma_base*

Gestion Utilisateur

- **CREATE USER bob@localhost IDENTIFIED BY toto;**
  - bob : Nom d'utilisateur
  - localhost : Nom du serveur
  - toto : Mot de passe de l'utilisateur

- **GRANT permission_type ON database_name.table_name TO bob@localhost;**
  - **Permission_type peut avoir les valeurs :**
    - CREATE – création des bases de données/tableaux
    - SELECT – récupération des données
    - INSERT – ajout de nouvelles entrées dans les tableaux
    - UPDATE – modification des entrées existantes dans les tableaux
    - DELETE – suppression des entrées dans les tableaux
    - DROP – suppression des bases de données/tableaux en entiers

- **REVOKE permission_type ON database_name.table_name FROM bob@localhost;**

- **DROP USER bob@localhost;**

Typage des données

- Type Numérique
  - Entier
    - TINYINT, 1 octet
    - SMALLINT, 2 octets
    - INT, 4 octets
    - BIGINT, 8 octets
    - UNSIGNED ...
  - Réel
    - NUMERIC/DECIMAL (x,y) : x nombre de chiffres, dont y sont positionnés derrière la virgule (en réalité, c'est une chaine de caractères).
    - FLOAT, DOUBLE

Typage des données

- Type Date
  - DATE : représentation AAAA:MM:JJ
  - TIME : représentation HH:MM:SS
  - DATETIME : représentation AAAA:MM:JJ:HH:MM:SS
  - YEAR : représentation AAAA
- Type Texte
  - CHAR (n) : chaîne de taille fixe, n *:* taille de chaîne
  - VARCHAR (n) : chaîne de taille variable, n max. de la taille
  - TEXT : texte de grande taille

Création de tables

- **CREATE TABLE nom_table (col1 type1, col2 type2, …, PRIMARY KEY (coli));**
  - Créer une table *nom_table* avec les attributs *col1* de type *type1*, …, de clé primaire *coli.*
  - Precision :
    - NOT NULL
    - DEFAULT
  - On peut ajouter une clé étrangère : **FOREIGN KEY (colk) REFERENCES tableA (colj)** (qui fera donc référence à la colonne *colj* de la table *tableA*.
- **DROP TABLE nom_table;**

  Supprime la table de la base

Modification de tables

- **ALTER TABLE nom_table instruction ;**
  - Modifie la table nom_table
  - Instruction peut prendre les valeurs :
    - **ADD nom_col type** : ajoute une colonne de nom *nom_col* de type *type*
    - **DROP nom_col** : supprime la colonne *nom_col*
    - **MODIFY nom_col type** : modifie le type de la colonne *nom_col* en type *type*
    - **CHANGE col_ancien col_nouveau type** : change le nom de la colonne *col_ancien* de type *type* en *col_nouveau*

Ajout de contraintes

- **ALTER TABLE table ADD PRIMARY KEY(col1);**
  Ajoute la clé primaire *col1* à la table *table*

- **ALTER TABLE table ADD INDEX(col);**
  Ajoute un index sur la colonne *col* de la table *table.* Nécessaire pour définir une clé étrangère.

- **ALTER TABLE table ADD CONSTRAINT nomC FOREIGN KEY (col1) REFERENCES table2(coli);**
  Ajoute une contrainte de clé étrangère *nomC* à la table *table* sur la colonne *col1* en référence à la colonne *coli* de la table *table2.*

- **ALTER TABLE table DROP PRIMARY KEY;**
  Supprime la clé primaire de la table *table*

- **ALTER TABLE table DROP FOREIGN KEY col;**
  Supprime la contrainte de clé étrangère sur la colonne *col* de la table *table*

Mise à jour d'informations dans une table

- Insertion

  INSERT INTO Table col_1,…, col_n VALUES (val_1,…,val_n);

- Mise à jour

  UPDATE Table SET col_1 = val_1 , … , col_n = val_n WHERE condition;

- Suppression

  DELETE FROM relation WHERE condition;

MLD -> Bases de données

- **Produit** (IdProduit, NomProduit, PoidsProduit, CouleurProduit, #IdMarque)
- **Marque** (IdMarque, NomMarque, LogoMarque)
- **Magasin** (IdMagasin, NomMagasin, AdresseMagasin, CPMagasin, VilleMagasin)
- **Fournisseur** (IdFournisseur, NomFournisseur, AdresseFournisseur, CPFournisseur, VilleFournisseur)
- **Usine** (IdUsine, NomUsine, AdresseUsine, CPUsine, VilleUsine)
- **Fabrique** (# IdUsine, # IdProduit)
- **Commande** (IdCommande, Quantité, Date, IdProduit #, IdMagasin #, IDFournisseur #)

```
CREATE DATABASE Exemple;

CREATE TABLE Exemple.Marque (IdMarque int NOT NULL, NomMarque varchar(20) NOT NULL, LogoMarque text NOT NULL, PRIMARY KEY (IdMarque));

CREATE TABLE Exemple.Magasin ( IdMagasin INT NOT NULL , NomMagasin VARCHAR(50) NOT NULL , AdresseMagasin VARCHAR(200) NOT NULL , CPMagasin VARCHAR(6) NOT NULL , VilleMagasin VARCHAR(50) NOT NULL, PRIMARY KEY (IdMagasin));

CREATE TABLE Exemple.Fournisseur ( IdFournisseur INT NOT NULL , NomFournisseur VARCHAR(50) NOT NULL , AdresseFournisseur VARCHAR(200) NOT NULL , CPFournisseur VARCHAR(6) NOT NULL , VilleFournisseur VARCHAR(50) NOT NULL, PRIMARY KEY (IdFournisseur));

CREATE TABLE Exemple.Usine ( IdUsine INT NOT NULL , NomUsine VARCHAR(50) NOT NULL , AdresseUsine VARCHAR(200) NOT NULL , CPUsine VARCHAR(6) NOT NULL , VilleUsine VARCHAR(50) NOT NULL, PRIMARY KEY (IdUsine));

CREATE TABLE Exemple.Produit ( IdProduit INT NOT NULL , NomProduit VARCHAR(50) NOT NULL , PoidsProduit REAL NOT NULL , CouleurProduit VARCHAR(20) NOT NULL , IdMarque INT NULL, PRIMARY KEY (IdProduit));

ALTER TABLE Exemple.Produit ADD INDEX(IdMarque);

ALTER TABLE Exemple.Produit ADD CONSTRAINT Produitfk1 FOREIGN KEY (IdMarque) REFERENCES Exemple.Marque(IdMarque);

CREATE TABLE Exemple.Fabrique ( IdProduit INT NOT NULL , IdUsine INT NOT NULL , PRIMARY KEY (IdProduit,IdUsine));

ALTER TABLE Exemple.Fabrique ADD CONSTRAINT Fabriquefk1 FOREIGN KEY (IdProduit) REFERENCES Exemple.Produit(IdProduit);

ALTER TABLE Exemple.Fabrique ADD CONSTRAINT Fabriquefk2 FOREIGN KEY (IdUsine) REFERENCES Exemple.Usine(IdUsine);

CREATE TABLE Exemple.Commande ( IdCommande INT NOT NULL , Quantite INT NOT NULL, Date DATE NOT NULL, IdProduit INT NOT NULL , IdMagasin INT NOT NULL , IdFournisseur INT NOT NULL , PRIMARY KEY (IdCommande));

ALTER TABLE Exemple.Commande ADD INDEX(IdProduit);

ALTER TABLE Exemple.Commande ADD INDEX(IdFournisseur);ALTER TABLE Exemple.Commande ADD INDEX(IdMagasin);

ALTER TABLE Exemple.Commande ADD CONSTRAINT Commandefk1 FOREIGN KEY (IdProduit) REFERENCES Exemple.Produit(IdProduit);

ALTER TABLE Exemple.Commande ADD CONSTRAINT Commandefk2 FOREIGN KEY (IdMagasin) REFERENCES Exemple.Magasin(IdMagasin);

ALTER TABLE Exemple.Commande ADD CONSTRAINT Commandefk3 FOREIGN KEY (IdFournisseur) REFERENCES Exemple.Fournisseur(IdFournisseur);
```

MPD associé à la base créée