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

Mise à jour d'informations

- Insertion

  INSERT INTO <relation> [att_1,…, att_n] VALUES (val_1,…,val_n)

  INSERT INTO <relation> (att_1,…, att_n) SELECT …
- Une relation citée dans le champ INTO du INSERT ne peut pas être citée dans le champ FROM du sous-select de ce même INSERT
- Mise à jour

  UPDATE <relation> SET att 1 = val 1 , … , att n = val n WHERE <condition>

  UPDATE <relation> SET (att 1, … , att n) = (SELECT …) WHERE <condition>
- Suppression

  DELETE FROM <relation> WHERE <condition>