Requêtes en langage SQL

I Requêtes "simples" (SELECT... FROM.. WHERE...)

1) SELECT * FROM T_1 : obtenir le contenu de T_1
2) SELECT col1, col2, ... : ne conserver que certaines colonnes.
3) DISTINCT : supprimer les doublons

Manipuler plusieurs tables

4) SELECT ... From T_1, T_2, T_3, ... : Produits cartesien
5) T_1 as t : renommer une table

Filtrer certaines lignes

6) SELECT ... FROM ... WHERE condition : ne conserve que les lignes vérifiant condition :

7) Syntaxe des conditions : col1 </>/<>/= col2/constante, col LIKE regex, col IN (cst1, ..., cstk), NOT condition, condition1 AND/OR condition2, col IS NULL

8) les jointure : T_1 NATURAL JOIN T_2, T_1 JOIN T_2 on condition

II Calcul relationnel et algèbre relationnelle

Considérons un ensemble de relations $\mathcal{R}$

A) Le calcul relationnel (independant du domaine)

Considérons un ensemble dénombrable de variable x et de constantes H

terme $t = x \in X \mid h \in H$

atome $R(t_1, \dots, t_k)$ $R \in \mathcal{R}$ d'arité k et $t_1, \dots, t_k$ termes

formule $\varphi = R(t_1, \dots, t_k)$ atome $\mid \exists x\, \varphi$ avec $x \in X \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \forall x\, \varphi$ avec $x \in X \mid \varphi_1 \vee \varphi_2$

Une requête q du calcul relationnel independant du domaine est $q = \{x_1, \dots, x_k \mid \varphi\}$ où $x_1, \dots, x_k$ est l'ensemble des variables libres de φ

Dev 1 : l'inclusion de 2 requêtes conjonctives est NP-complet

B) Algèbre relationnelle (nommée)

A partir des relations de $\mathcal{R}$, on en construit d'autres :

- selection : $\sigma_{condition}(R)$ — condition sur les colonnes de R
- projection : $\Pi_{col_1, \dots, col_k}(R)$ — $col_1, \dots, col_k$ sont des colonnes de R
- union : $R_1 \cup R_2$ — les colonnes de R_1 ont le même nom que celles de R_2
- différence : $R_1 - R_2$
- produit cartesien : $R_1 \times R_2$ — R_1 et R_2 n'ont pas de colonne ayant le même nom

- Ojointure : $R_1 \bowtie_{condition} R_2$
- jointure naturelle : $R_1 \bowtie R_2$
- intersection : $R_1 \cap R_2$
- division : $R_1 \div R_2$

Théorème (codd) : le calcul relationel independant du domaine et l'algèbre relationelle ont la même expressivité

Dev 2 : Preuve

III Etendre les requêtes SQL

A) Sous requêtes

- Utiliser le résultat d'une requête comme une table : SELECT... FROM (SELECT...) AS t ...
- Utiliser le résultat d'une requête pour une condition : colonne = ANY(SELECT...)
  EXISTS (SELECT...)

Remarque : On peut faire référence aux colonnes de la grande requête dans la sous-requête.

B) Associer plusieurs requête

- requête1 UNION requête2 : union ensembliste
- UNION ALL : union multiensembliste
- INTERSECT / INTERSECT ALL : intersection

I Requêtes inexprimable en calcul relationel

A) Aggrégats

+ Opérateur d'aggrégation : GROUP BY $col_1, \dots, col_k$: aggrège les lignes dont les valeurs des colonnes col_1, col_k sont égale
- Count / Avg / Min / Max / Sum : Pour chaque groupe, renvoit le nombre / la moyenne / le minimum / le maximum / la somme des éléments du groupe :
- Remarques : - si il n'y a pas de group by, le groupe considéré est l'ensemble des elements,
- on peut ajouter distinct dans les operateurs précedents pour considerer qu'une seule fois chaque valeur différente.

+ HAVING condition : ne conserve que les groupes respectant la condition.
- Remarque : L'idée est d'utiliser les opérateurs sur les groupes dans les conditions du HAVING.

B) Ordonner les résultats

- ORDER BY col : renvoit les lignes par ordre croissant de valeur col.
- ORDER BY col DESC : l'ordre décroissant

- LIMIT / TOP k : renvoit les k premières lignes.