SQL Module 2 :vers des interrogations plus élaborées

Table des matières

1. Interrogation de données :

Dans cette section on améliore l'interrogation de données à l'aide

- des operateurs ensemblistes de l'algèbre relationnel
- des select avec partition simple (goup by having)
- l'utilisation de sous requêtes (ou requêtes imbriquées- c'est la même chose)

1.1. Opérateurs ensemblistes

C'est très simple

Le mot clé **UNION** permet de faire l'union des résultats de deux requetes SELECT . C'est un opérateur ensembliste, il ne peut y avoir deux fois la même ligne dans le résultat

Si l'on désire conserver les doublons il faut utiliser **UNION ALL**

De même le mot clé INTERSECT permet de faire l'intersection de deux requêtes SELECT et le mot clé EXCEPT permet de faire leur différence

Pour ces trois opérateurs

- les deux SELECT de part et d'autre de l' opérateur doivent avoir le même nombre de colonnes
- Les colonnes doivent avoir deux à deux des types similaires
- les colonnes des deux SELECT doivent être dans le même ordre

On met ça en pratique ?

1.2. select avec partition simple: GROUP BY

UN SELECT peut contenir une clause **GROUP BY** permet de partitionner les lignes en regroupant dans une même classe d'équivalence (dite regroupement) toutes les lignes qui coïncident sur tous les attributs mentionnés dans le GROUP BY.

Il est alors possible d'agréger des résultats sur une classe d'équivalence

Une agrégation est une opération qui agrège en une seule valeur (un agrégat) les différentes valeurs d'une colonne d'un regroupement de lignes SUM, AVG, MIN, MAX,COUNT

Seuls les agrégats et les attributs mentionnés dans la clause GROUP BY peuvent être affichés c'est à dire apparaître entre SELECT et FROM.

Exemple de requête :

select directorid, sum(length) from movie group by directorid;

Table movie

	movieid [PK] character varying (10)	movietitle character varying (30)	releasedate date	genreid character varying (10)	directorid character varying (10)	length integer
1	M01	Superman vs Batman	2016-03-25	G01	D01	89
2	M02	Deadpool	2016-02-12	G02	D02	93
3	M03	Furious 7	2015-04-03	G03	D03	87
4	M04	PK	2014-12-19	G04	D04	101
5	M05	Gladiator	2000-05-05	G01	D05	112
6	M06	The Hangover	2009-06-05	G02	D06	94
7	M07	3 Idiots	2009-12-25	G04	D04	76
8	M08	Spectre	2015-11-06	G03	D07	112
9	M09	Batman Begins	2005-06-15	G01	D08	93
10	M10	The Dark Knight	2008-07-18	G05	D08	89

résultat de la requête select directorid, sum(length) from movie group by directorid; :

	directorid character varying (10)	sum bigint
1	D07	112
2	D03	87
3	D08	182
4	D02	93
5	D05	112
6	D01	89
7	D04	177
8	D06	94

La requête

select directorid, genreid, sum(length) from movie group by directorid;

est incorrecte, le regroupement est fait sur les identifiants des metteurs en scène, on ne peut pas afficher le genre de tous les films du metteur en scène d'identifiant D04, de même qu'on ne pourrait pas afficher leur longueur, on peut en revanche afficher la somme de leur longueur

A vous de jouur: allez faire la question 1 de ce questionnaire

Cette requête

select directorid, genreid, sum(length) from movie group by directorid,genreid;

est correcte, elle regroupe les films ayant le même metteur en scène et le même genre et additionne leur longueur et affiche les trois informations

Cette requete

select directorid sum(length) from movie group by directorid,genreid;

est syntaxiquemet correcte, mais le resultat sera inexploitable, si un metteur en scene a produit des films de deux genres differents, son nom sera afficher deux fois, mais il sera impossible de savoir à quels genre sont associés les longueurs affichées.

Remarque : il est possible d'utiliser des agregats sans group by a condition de n'utiliser que des agregats

exemple

select avg (length) from movie est correct et retourne la longueur moyenne d'un film de la base

A vous de jouer: allez faire la question 2 de ce questionnaire

1.3. AGREGAT COUNT

L'agregat COUNT permet de compter combien de lignes ont été agrégées

select directorid, count(*) from movie group by directorid

ou

select directorid , count(n'importe quel attribut) from movie group by directorid

permet de connaître le nombre de films dirigés par chaque metteur en scene

par exemple select directorid, count(genreid) from movie group by directorid ne compte pas le nombre de genre de films dirigés par le metteur en scène , mais le nombre de films dirigés par le metteur en scène.

En revanche la requete

select directorid, count(**distinct** genreid) from movie group by director id compte bien le nombre de genre de films dirigés

1.4. GROUP BY HAVING

La clause GROUP BY peut être compléter par HAVING suivi d'une condition

Pour une requête de la forme

SELECT ... FROM....WHERE Condition1 GROUP BY.... HAVING Condition2

Condition1 est evaluée avant le regroupement (donc individuellement sur chaque ligne) tandis que Condition2 est évaluée après les regroupements

select directorid, count(*) from movie where releasedate >='2005-01-01' group by directorid having count(*)>=2

va afficher le nombre de film dirigés par les metteurs en scène depuis le premier janvier 2005, a condition qu'il y en ait au moins deux

La requête

select directorid from movie where releasedate >='2005-01-01' group by directorid having count(*)>=2

est correcte est affiche les metteurs en scène qui ont dirigés au moins deux films depuis le premier janvier 2005

1.5. Ordre des clauses dans le select

SELECT FROM WHERE GROUP BY HAVING ORDER BY

Seules les deux premières clauses SELECT et FROM sont obligatoires (dans la norme)

Hors norme mais acceptés select current_date; select 123;

L'ordre d'écriture des clauses n'est pas l'ordre de leur évaluation

A l'execution de la requete

SELECT E1 FROM E2 WHERE E3 GROUP BY E5 HAVING E6 ORDER BY E4

tout se passe comme ci l'evaluation était faite dans l'ordre

1. E2 (les jointures ou produits cartesiens)
2. E3 (la selection sur les tuples)
3. E5 (la création des regroupements)
4. E6 (la selection sur les regroupements)
5. E4 (le tri)
6. E1 (la projection)

En fait c'est pas tout à fait vrai , d'ou le tout se passe comme si.....En particulier, pour optimiser le temps de réponse, E2 et E3 peuvent ne pas etre exécutés l'un après l'autre mais entrelaçés

Rappel : **une requête SQL ne dit jamais comment obtenir le résultat, mais décrit le résultat souhaité.**

1.6. Sous Requêtes

Une sous-requête (ou requête imbriquée) est une requête SQL qui apparait dans une requête SQL (requête maître) .

Où et comment dépendent du type de résultat attendu de la sous requête.

Le résultat d'une requête est toujours une table composée de lignes et de colonnes, mais il y a des cas particuliers. Du plus particulier au plus général :

1. Une sous-requête qui ne produit **qu'une seule valeur** peut être utilisée partout où l'on pourrait utiliser une constante :

- Dans le select entre le select et le from :(exemple d'utilisation d'une constante entre le select et from : select t.A-3 from t; est une requête correcte si l'attribut A de la table t est numerique)

exemple d'utilisation d'un select entre le select et le from

select **(**select count (distinct genreid)from genre**)** - **(**select count (distinct genreid) from movie**)**

cette requête retourne le nombre de genre de film pour lesquels il n'y a aucun film dans movie

  - Dans le select entre le select et le from

exemple

select (select count (distinct genreid)from genre) - (select count (distinct genreid) from movie)

cette requête retourne le nombre de genre de film pour lesquels il n'y a aucun film dans movie

- Dans le where

exemple

select directorname from director where directorid =(select directorid from movie where movietitle ='The Dark Knight')

retourne le nom du metteur en scene de 'The Dark Knight'

- Dans le having

exemple

select directorid, count(*) from movie where genreid='G01' group by directorid having count(*) = (select count(movieid) from movie where genreid='G01')

retourne s'il existe le nom du metteur en scène qui a dirigé tous les films de genre G01 répertoriés dans la table movie

2. Une sous requête dont le résultat est composé d'une seule colonne peut etre considérée comme une liste et utilisée dans une condition (dans un where ou un having donc) associée à

- IN / NOT IN

exemple
SELECT genreid FROM genre WHERE genreid NOT IN (SELECT genreid FROM movie where directorid='D01');
Les genres de films que le metteur en scene d'identifiant D01 n'a jamais dirigé

- ALL

exemple

SELECT directorid FROM movie GROUP BY directorid HAVING count(*) >= ALL (SELECT count(*) FROM movie GROUP BY directorid)

Les metteurs en scènes qui ont dirigés le plus de films répertoriés dans movie
Ici la condition doit etre vraie pour toutes les valeurs de la liste

- ANY/SOME

exemple
SELECT directorid FROM movie GROUP BY directorid HAVING count(*) < ANY (SELECT count(*) FROM movie GROUP BY directorid)

Les metteurs en scène qui n'ont pas le record du nombre de films dirigés (selon la table movie)
Ici, la condition doit être vraie pour au moins une valeur de la liste

3. Dans tous les cas, une requête peut être utilisée comme sous requête

- dans un FROM: de la manière dont on utiliserait une table…mais nécessite un parenthésage et un alias

exemple
SELECT MAX(somme) AS MAX_somme FROM **(**SELECT SUM(length) AS somme FROM movie GROUP BY genreid**) L**
Le genre de film dont on a le plus de kilomètres.....

- dans une condition d'un WHERE ou d'un HAVING en utilisant EXISTS

exemple
select directorname from director D where exists
(select * from movie where D.directorid=movie.directorid and movie.genreid='G01' or movie.genreid='G05')
nom des metteurs en scène ayant dirigés des films de genre 1 ou de genre 5
Cet exemple constitue une sous requête **corrélée** car la sous-requête utilise un attribut(D.directorid)
La sous requête corrélée est exécutée de multiples fois

On vérifie qu'on a compris ?

1.7. Gestion des doublons

SQL construit des multi-ensembles (et non des ensembles comme l'algèbre relationnelle)

De même qu'une ligne peut apparaître plusieurs fois dans une table, lorsque une requête SQL construit une nouvelle table elle n'élimine pas automatiquement les doublons.

Pour éliminer les doublons utilisez le omot clé DISTINCT:

SELECT DISTINCT nom FROM marque;

Attention : l'usage de distinct a un coût important: il faut stocker toute la table et la trier

Exception : UNION, INTERSECT et EXCEPT sont des opérations ensemblistes qui éliminent les doublons

L'utilisation de UNION ALL, INTERSECT ALL et EXCEPT ALL permet de travailler sur des multi-ensembles

R EXCEPT ALL S :

élimine autant d'occurrences d'un tuple t dans R que celui-ci à d'occurrences dans S