Le langage S.Q.L. - Select

P.Mathieu

Université de Lille

LP DA2I Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

8 septembre 2019

Le langage SQL - Select

Fondements

Université de Lille

- Langage de définition et de manipulation de bases de données relationnelles.
- Signification : “Structured Query Language”
- Normalisé par l’organisme `ANSI`.
- Historique : SEQUEL (1977), SEQUEL/2 pour System/R, SQL pour Oracle (1981), SQL pour DB/2 (1983)

SQL fonctionne :

- En mode intéractif
- En batch, intégré dans des langages hôtes

Le langage SQL - Select

Basé sur l’Algèbre relationnelle

Université de Lille

- Projection
- Restriction
- Union
- Difference
- Produit cartésien

- Ces cinq opérations forment un ensemble cohérent et minimal.
- Aucune d’entre-elles ne peut s’écrire à l’aide des autres.
- A partir de ces cinq opérations élémentaires, d’autres opérations peuvent être définies.
- Toute requète s’exprime à l’aide d’une combinaison de ces opérations.

Le langage SQL - Select

Opérations construites

- Intersection :

$$R \cap S = R - (R - S) = S - (S - R)$$

- Quotient
  Si l'on note V l'ensemble des colonnes à obtenir (ici A,B)

$$R/S = \pi_V(R) - \pi_V((\pi_V(R) * S) - R)$$

- Jointure
  Opération fondamentale, utilisation raisonnée du produit cartésien

$$R \bowtie_Q S = \sigma_Q(R * S)$$

Le langage SQL - Select

SQL contient :

- Un langage de description : le DDL (Data Definition Language)
- Un langage de manipulation de données : le DML (Data Manipulation Language)
- un langage de gestion des protections : le DCL (Data Control Language)

DDL	DML	DCL
ALTER	DELETE	GRANT
CREATE	INSERT	REVOKE
COMMENT	SELECT	
DESCRIBE	UPDATE	
DROP		
RENAME		

La manipulation des données : le DML

Expression des projections

lister les numéros et noms des fournisseurs

```
SELECT fno, nom FROM fournisseurs ;
```

lister la table fournisseurs

```
SELECT * FROM fournisseurs ;
```

SQL n'élimine pas les doubles à moins que ça ne soit explicitement précisé par le mot clé `DISTINCT`.

lister les différentes désignations de produits

```
SELECT DISTINCT design FROM produits;
```

La manipulation des données : le DML

Expression des restriction : la clause WHERE

l'association projection-restriction est triviale :

lister les désignations de produits différentes dont le poids est supérieur à 15

```
SELECT DISTINCT design FROM produits
WHERE poids > 15 ;
```

Les qualifications peuvent utiliser `>, >=, <, <=, =, <>` et `AND, OR, NOT`.

lister les produits dont le poids est compris entre 15 et 40

```
SELECT * FROM produits
WHERE poids > 15 AND poids < 40 ;
```


La manipulation des données : le DML

Les autres comparateurs

lister les fournisseurs habitant Lille, Lyon ou Nice

```
SELECT * FROM fournisseurs
WHERE ville IN ('Lille','Lyon','Nice') ;
```

lister les produits dont le poids n'est pas compris entre 15 et 35

```
SELECT * FROM produits
WHERE poids NOT BETWEEN 15 AND 35 ;
```

lister les fournisseurs dont le nom ne commence pas par 'D'

```
SELECT * FROM fournisseurs
WHERE nom NOT LIKE 'D%'
```

Voir aussi `SIMILAR TO`

La manipulation des données : le DML

Quelques expressions régulières utiles

code postal	`^[0-9]{5}$`
login	`^[a-zA-Z][a-zA-Z0-9]{3,8}$`
date	`^20[0-9]{2}-(0|1)[0-9]-(0|1|2|3)[0-9]$`
email	`^[a-z]+\.[a-z]+@[a-z]+(\.com|\.fr|\.eu)$`
téléphone	`^[0-9]{2}-[0-9]{2}-[0-9]{2}-[0-9]{2}-[0-9]{`
Adresse ip	`^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{`
url	`^(http:|https:)//.*(\.fr|\.com)$`

Attention, elles ne sont pas "parfaites"

La manipulation des données : le DML

Tester les valeurs manquantes

lister les fournisseurs dont l'adresse n'est pas renseignée

```
SELECT * FROM fournisseurs
WHERE adresse IS NULL;
```

lister les fournisseurs dont l'adresse est renseignée

```
SELECT * FROM fournisseurs
WHERE adresse IS NOT NULL ;
```

La manipulation des données : le DML

Les sémantiques de l'égalité

- `ville = 'Lille'` égalité stricte
- `ville IN ('Lille',Lens')` égalité avec une valeur d'un ens
- `ville LIKE 'Li%'` égalité approximative
- `ville SIMILAR TO '(fr|uk)$'` égalité avec expr rationnelle
- `Ville IS NULL` test de méta-valeur

La manipulation des données : le DML

Tri et présentation des résultats

Le tri se fait par la clause `order by` et les mots clés `asc` et `desc`.

afficher les produits rouges ou verts ou bleus triés sur le nom croissant et la couleur décroissante

```
SELECT design, couleur FROM produit
WHERE couleur IN ('rouge','vert','bleu')
ORDER BY design ASC, couleur DESC ;
```

Les colonnes résultat peuvent être renommées par le prédicat `AS`

afficher les produits et leurs couleurs avec des noms de colonnes compréhensibles, triés comme précédemment

```
SELECT design AS 'désignation du produit',
       couleur AS 'couleur du produit'
FROM produits;
```


La manipulation des données : le DML

Expression des jointures

lister le produit cartésien fournisseur produits*

```
SELECT * FROM fournisseurs, produits ;
```

effectuer l'equi-jointure entre fournisseurs et commandes

```
SELECT fournisseur.nom, commandes.pno, commandes.qute
FROM fournisseurs, commandes
WHERE fournisseurs.fno = commandes.fno ;
```


La manipulation des données : le DML

ambiguïté de noms de colonnes

lister les noms des fournisseurs avec les numéros de produits commandés ainsi que la quantité commandée

```
SELECT fournisseur.nom, commandes.pno, commandes.qute
FROM fournisseurs, commandes
WHERE fournisseurs.fno = commandes.fno ;
```

utilisation des synonymes de noms de tables

```
SELECT f.nom, C.pno, C.qute
FROM fournisseurs AS f, commandes AS C
WHERE f.fno = C.fno ;
```

La manipulation des données : le DML

Un peu de sport ...

Plusieurs jointures peuvent être exécutées en cascade.

lister les couples (nom de fourniss, nom de produit) en cmd

```
SELECT f.nom AS fournisseur, p.design AS produit
FROM fournisseurs AS f, produits AS p, commandes AS C
WHERE f.fno = C.fno
AND   p.pno = C.pno ;
```

jointure d'une table sur elle même.

Lister les couples de références de fournisseurs situés dans la même ville.

```
SELECT f1.fno, f2.fno
FROM fournisseur AS f1, fournisseur AS f2
WHERE f1.ville = f2.ville;
```


La manipulation des données : le DML

Eviter les doublons et symétriques

Num	Nom
10	Dupont
20	Durand

Num	Nom	Num	Nom
10	Dupont	10	Dupont
10	Dupont	20	Durand
20	Durand	10	Dupont
20	Durand	20	Durand

Lister les couples de références de fournisseurs situés dans la même ville sans symétriques.

```
SELECT f1.fno, f2.fno
FROM fournisseur AS f1, fournisseur AS f2
WHERE f1.ville = f2.ville  AND f1.fno < f2.fno;
```


La manipulation des données : le DML

Les apports de SQL 2 pour la jointure

afficher la jointure naturelle entre Commandes et Fournisseurs

- Expression des jointures dans la clause `FROM`

```
SELECT *
FROM fournisseur f INNER JOIN commande C
     ON f.fno=C.fno ;
```

- Jointures externes
  - la table directrice aura tous ses enregistrements dans la table résultante
  - les colonnes manquantes sont complétées à `NULL`.
  - Opérateurs de jointure `LEFT JOIN`, `RIGHT JOIN`, `FULL JOIN`, `CROSS JOIN`

La manipulation des données : le DML

afficher tous les produits de moins de 20Kg avec les quantités en cours de commande si possible

```
SELECT p.pno, design, qute , poids
FROM produits AS p LEFT JOIN commandes AS C
     ON C.pno = p.pno
WHERE poids < 20 ;
```

result	p.pno	design	qute	poids
	101	fauteuil		7
	102	fauteuil	1	9
	102	fauteuil	8	9
	106	caisson		12
	107	caisson	12	12
	107	caisson	5	12

afficher les produits qui ne sont pas commandés

```
SELECT p.pno
FROM produits AS p LEFT JOIN commandes AS C
     ON C.pno = p.pno
WHERE C.pno IS NULL ;
```



```
SELECT * FROM t1 CROSS JOIN t2;

SELECT * FROM t1 INNER JOIN t2 ON t1.num = t2.num;
SELECT * FROM t1 INNER JOIN t2 USING (num);
SELECT * FROM t1 NATURAL INNER JOIN t2;

SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num;
SELECT * FROM t1 LEFT JOIN t2 USING (num);

SELECT * FROM t1 RIGHT JOIN t2 ON t1.num = t2.num;

SELECT * FROM t1 FULL JOIN t2 ON t1.num = t2.num;
```


SQL permet d'effectuer des calculs arithmétiques sur chaque tuple à l'affichage des résultats

Prix des produits avec une TVA à 20,6%

```
SELECT pno, design, prix*1.206 AS prixTTC
FROM produits ;
```

D'autre opérations sur chaque valeur sélectionnée peuvent être effectuées notamment sur les dates (`YEAR`, `MONTH`, `DAY`) et les chaînes de caractères (`SUBSTRING`, `UPPER`, `LOWER`, `CHARACTER_LENGTH`)

Opérations de calcul

volume financier commandé pour les commandes de moins de 10 articles

```
SELECT DISTINCT cno, qute*prix AS volume, qute
FROM produits INNER JOIN commandes
     ON produits.pno = commandes.pno
WHERE qute <10 ;
```

result	cno	volume	qute
	1003	7000	2
	1005	1500	1
	1007	1500	1
	1013	5000	5
	1017	7500	3
	1023	12000	8

Opérations de calcul

Les fonctions statistiques

Calculs de regroupement sur un ensemble de tuples de la table

AVG	moyenne
SUM	somme
MAX	maximum
MIN	minimum
COUNT	nombre d'éléments

calculer la somme des commandes

```
SELECT SUM(qute) FROM commandes ;
```


Opérations de calcul

compter le nombre de livraisons du produit numéro 102.

```
SELECT COUNT(*) AS nbre
FROM commandes
WHERE pno=102 ;
```

result	Nbre
	2

compter les noms de fournisseurs différents

```
SELECT COUNT(DISTINCT nom)
FROM fournisseurs ;
```

récupérer toutes les statistiques sur les quantités en commande

```
SELECT COUNT(*),SUM(qute),MAX(qute),MIN(qute),AVG(qute
FROM commandes ;
```


Opérations de calcul

Sémantique du COUNT

- `COUNT(*)` toutes les lignes
- `COUNT(fno)` les valeurs non nulles
- `COUNT(distinct fno)` les valeurs distinctes non nulles

Opérations de calcul

Les regroupements : la clause `GROUP BY`

Université de Lille

Effectuer les opérations statistiques sur des groupes de données ayant une caractéristique commune

afficher la somme des quantités commandées à chaque fournisseur

```
SELECT fno, SUM(qute)
FROM commandes
GROUP BY fno
```


Opérations de calcul

Université de Lille

lister le nombre de commandes par fournisseur

```
SELECT fno,COUNT(*) AS Nbre
FROM commandes
GROUP BY fno ;
```

result	fno	Nbre
	10	1
	13	1
	14	1
	15	2
	17	3
	19	2

Opérations de calcul

Remarque : si dans un `SELECT` ou un `ORDER` certaines colonnes sélectionnées utilisent des fonctions de groupe et d'autres non, alors toutes les colonnes sélectionnées issues directement des tables doivent être spécifiées dans le `GROUP BY`

```
SELECT cno,COUNT(*) AS Nbre
FROM commandes
GROUP BY fno ;
```

INCORRECT

```
SELECT fno, nom, COUNT(*) AS Nbre
FROM commandes
GROUP BY fno ;
```

INCORRECT

```
SELECT COUNT(*) AS Nbre
FROM commandes
GROUP BY fno ;
```

CORRECT

Opérations de calcul

Tests sur les valeurs regroupées : la clause `HAVING`

- Les tests sur les colonnes simples se font dans la clause `WHERE`
- Les tests sur les fonctions de regroupement se font dans la clause `HAVING`.

lister les fournisseurs qui ont moins de 3 commandes.

```
SELECT fno, COUNT(*)
FROM commandes
GROUP BY fno
HAVING COUNT(*)<3 ;
```

Opérations de calcul

Remarque : Les tests sont indépendants des valeurs affichées. On peut effectuer un test sur un calcul dans les clauses `WHERE` ou `HAVING` sans que ce calcul soit affiché.

référence des fournisseurs qui fournissent plus d'un produit.

```
SELECT fno
FROM commandes
GROUP BY fno
HAVING COUNT(*) > 1 ;
```


Opérations de calcul

Forme générale du `SELECT` classique

```
SELECT C.fno, nom, MAX(qute)
FROM commandes AS C , fournisseurs AS f
WHERE cno > 6 AND C.fno = f.fno
GROUP BY C.fno, nom
HAVING AVG(qute) > 50
ORDER BY nom DESC
LIMIT 2
OFFSET 5;
```


Opérations de calcul

Forme générale du `SELECT` classique

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    [ * | expression [ [ AS ] output_name ] [, ...] ]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY grouping_element [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] SELECT ]
    [ ORDER BY expression [ ASC | DESC | USING OPERATOR ] [ NULLS { FIRST
    [ LIMIT { COUNT | ALL } ]
    [ OFFSET START [ ROW | ROWS ] ]
```

Les sous-requêtes

Sous-requête en place d'une constante

SQL donne la possibilité d'utiliser des sous-requêtes afin de décrire des requêtes complexes permettant d'effectuer des opérations dépendant d'autres requêtes.

La sous-requête renvoie une valeur unique

fournisseurs qui habitent la même ville que le fournisseur 10.

```
SELECT nom
FROM fournisseurs
WHERE ville = (SELECT ville
               FROM fournisseurs
               WHERE fno=10) ;
```


Les sous-requêtes

Cette fois-ci dans la clause `Having`

lister les fournisseurs dont la moyenne des quantités en commande est supérieure la moyenne des quantités de la table

```
SELECT fno
FROM commandes
GROUP BY fno
HAVING AVG(qute) >   (SELECT AVG(qute)
                      FROM commandes) ;
```


Les sous-requêtes

Sous-requête de calcul d'une liste

La sous-renquête renvoie une colonne unique
Opérateurs `IN`, `>ALL`, `>ANY`

Lister les références des fournisseurs livrant au moins un produit en quantité supérieure à chacun des produits livrés par le fno 19.

```
SELECT DISTINCT fno
FROM commandes
WHERE qute > ALL
      ( SELECT qute
        FROM commandes
        WHERE fno=19 ) ;
```


Les sous-requêtes

Le test d'existance : `EXISTS`

Lister les références des fournisseurs livrant au moins un produit en quantité supérieure à chacun des produits livrés par le fno 19.

(Si mon produit est supérieur à tout ce qui est livré par le 19
Il n'existe pas de produit du 19 en quantité supérieure)

```
SELECT DISTINCT c1.fno
FROM commandes c1
WHERE NOT EXISTS
          ( SELECT *
            FROM commandes c2
            WHERE c2.fno = 19
            AND c2.qute >= c1.qute) ;
```

C'est une requête corrélative.

Les sous-requêtes

Traitement d'une sous-requête par sa forme négative

Pour obtenir les noms des fournisseurs livrant tous les produits, il est plus facile de rechercher les fournisseurs pour lesquels il n'existe aucun produit non livré.

lister les noms des fournisseurs livrant tous les produits.

```
SELECT fnom
FROM fournisseurs f
WHERE NOT EXISTS (SELECT *
                  FROM produits p
                  WHERE NOT EXISTS (SELECT *
                                    FROM commandes C
                                    WHERE f.fno=C.fno
                                    AND C.pno=p.pno) )
```

C'est encore une requête corrélée !

Les sous-requêtes

lister les fournisseurs d'au moins un des produits fournis aussi par un fournisseur d'un produit rouge.

```
SELECT DISTINCT fno
FROM commandes
WHERE pno IN (SELECT DISTINCT pno
              FROM commandes
              WHERE fno IN (SELECT DISTINCT fno
                            FROM commandes
                            WHERE pno IN (SELECT pno
                                          FROM produit
                                          WHERE couleu
                                               'rouge'
```


Les sous-requêtes

Resumé

- `... WHERE fno > 125`
  Sélection classique, par comparaison avec une constante.
- `... WHERE fno > (SELECT fno from ...)`
  champ est comparé au résultat de la sous-requête.
- `... WHERE fno IN (SELECT fno from ...)`
  Ici la sous-requête renvoie plusieurs valeurs
- `... WHERE fno > ALL (SELECT fno from ...)`
  valeur supérieure à **toutes** les valeurs de la sous-requête.
- `... WHERE fno > ANY (SELECT fno FROM ...)`
  valeur supérieure à **au moins** une valeur de la sous-requête.
- `... WHERE EXISTS (SELECT .... WHERE fno = ...)`
  la sous-requête doit donner un succès (sous-requête "corrélée")

Design Pattern BDD

Design Pattern : Distinct

Quand une clé est sélectionnée, jamais de Distinct dans le select

```
SELECT nom,prenom,pno,libelle,qute
FROM fournisseurd f, produits p,commandes C
WHERE f.fno=C.fno AND C.pno=p.pno
```

Quand un group by est effectué, jamais de Distinct dans le select

```
SELECT fno, COUNT(*)
FROM commandes
GROUP BY fno
```


Design Pattern BDD

Design Pattern : Quotient

Traitement d'une sous-requête par sa forme négative

Pour obtenir les noms des fournisseurs livrant tous les produits, il est plus facile de rechercher les fournisseurs pour lesquels il n'existe aucun produit non livré.

lister les noms des fournisseurs livrant tous les produits.

```
SELECT fnom
FROM fournisseurs f
WHERE NOT EXISTS (SELECT *
                  FROM produits p
                  WHERE NOT EXISTS (SELECT *
                                    FROM commandes C
                                    WHERE f.fno=C.fno
                                    AND C.pno=p.pno) )
```


Design Pattern BDD

Design Pattern : ALL/ANY

quand on teste des valeurs numériques dans un Where
< ANY est équivalent à < select max()
> ANY est équivalent à > select min()
> ALL est équivalent à > select max()
< ALL est équivalent à < select min()
De même =ANY est équivalent à IN

ANY et ALL ont donc peu d'intérêt dans la clause Where
Par contre ils sont nécessaires dans la clause having car max(count(*)) est interdit !

Design Pattern BDD

Design Pattern : Le plus grand d'une liste

selectionner le produit le plus présent en commandes

```
SELECT pno
FROM commandes
GROUP BY pno
HAVING COUNT(*) >=ALL (SELECT COUNT(*)
                       FROM commandes GROUP BY pno)
```

Design Pattern BDD

Design Pattern : Jointure vs Sous-requête

Toute requête de jointure entre deux tables en renvoyant que les données d'une seule des deux tables peut-être réécrite à l'aide d'une sous-requête.

```
SELECT cno,qute
FROM fournisseurs f,
     commandes C
WHERE f.fno=C.fno
AND f.nom LIKE 'T%'
```

```
SELECT cno,qute
FROM commandes
WHERE fno IN (SELECT fno
              FROM fournisseurs
              WHERE nom LIKE 'T%')
```


Design Pattern BDD

Mais elle peut être corrélative !

pour peu que deux colonnes des deux tables soient comparées.

Les fournisseurs qui ne sont pas seuls dans la même ville

```
SELECT f1.*
FROM fournisseur f1, fournisseur f2
WHERE f1.fno<>f2.fno
AND f1.ville=f2.ville
```

```
SELECT *
FROM fournisseur f1
WHERE EXISTS (SELECT fno FROM fournisseur f2
                  WHERE f1.ville = f2.ville AND f1.fno
```

```
SELECT * FROM fournisseur
WHERE ville IN (SELECT ville FROM fournisseurs
                GROUP BY ville HAVING COUNT(*)>1)
```


Design Pattern BDD

Les opérateurs ensemblistes

s'intercalent entre deux `SELECT`.

- `UNION` : fournit la réunion des tuples des 2 select
- `INTERSECT` : fournit l'intersection des tuples des 2 select
- `EXCEPT` : fournit les tuples du premier select qui ne sont pas dans le second.

Remarques :

- `INTERSECT` et `EXCEPT` ne font pas partie de la norme SQL
- `UNION` supprime par défaut les doublons de l'opération. Si les doublons sont souhaités il faut utiliser la forme `UNION ALL`.

lister les numéros de produits dont le poids est supérieur à 20 ainsi que les produits commandés par le fournisseur 15

```
SELECT pno FROM produits WHERE poids>20
```

Conclusion

Tout arbre relationnel peut être codé avec SQL (puisque les opérateurs de base le sont) mais pas systématiquement en une seule requête.

C'est par exemple le cas pour l'opérateur Quotient qui nécessite pour être implémenté, la création de deux requêtes.

Exercice

On considère une base de données permettant de gérer des pièces de théâtre. De cette base nous avons extrait 3 relations :

`spectacle(num-spec, titre-spec, salle, monteur)`
`jouer(nom-acteur, num-spec)`
`représentations(date,heure,num-spec,tarif)`

Exercice

Ecrire les requêtes classiques suivantes en `SQL`

1. Lister la table des spectacles.
2. Donner la liste des numéros et titres de spectacles.
3. Donner la liste des titres de spectacles joués salle Chopin.
4. Donner les numéros de spectacle du 12/10/2018 .
5. Donner la liste des salles
6. Lister les données sur les spectacles se jouant salle Chopin ou qui sont montés par Dupont.
7. Lister les données des spectacles de numéro entre 20 et 50
8. lister les données sur les spectacles montés par un monteur dont le nom commence par D.
9. Donner les numéros de spectacles coûtant moins de 200€ parmi

Exercice

Ecrire en SQL les requêtes évoluées suivantes

1. Afficher le nombre de spectacles par salle.
2. Afficher le nombre de spectacles gérés par un même monteur dans une même salle, triés par salle et monteurs décroissants.
3. Afficher la moyenne des tarifs par salle pour les numéros de spectacles inférieurs à 50
4. Afficher les monteurs qui ne s'occupent que d'un seul spectacle.
5. Afficher les acteurs avec le nombre de spectacles auxquels ils participent et la moyenne des tarifs des représentations correspondantes mais uniquement pour les spectacles se jouant l'après-midi à des tarifs moyens inférieurs à 300€. Trier le résultat par tarif moyen décroissant et nom d'acteur décroissant.
6. Afficher les noms d'acteurs qui participent à des spectacles coûtant moins chers que la moyenne des spectacles.