

SQL : Langage d'Interrogation de Données

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com



Plan

1 Introduction

2 Requêtes simples

- SELECT ... FROM ...
- LIMIT et OFFSET
- DISTINCT
- COUNT
- WHERE
- BETWEEN
- IN
- NOT
- LIKE
- IS
- IFNULL
- ORDER BY
- GROUP BY
- HAVING
- UNION

Plan

3

Jointure

- Jointure implicite
- Jointure explicite
 - JOIN ... ON
 - LEFT JOIN ... ON
 - RIGHT JOIN ... ON

4

Requêtes imbriquées

- WHERE ... IN ... SELECT
- WHERE ... ANY ... SELECT
- WHERE EXISTS ... SELECT
- FROM ... SELECT ... AS

5

Fonctions sur les chaînes de caractère

SQL

SQL

- Langage de définition de données
- Langage de manipulation de données
- Langage de contrôle de données
- **Langage d'interrogation de données**

SQL

SQL

- Langage de définition de données
- Langage de manipulation de données
- Langage de contrôle de données
- **Langage d'interrogation de données**

Langage d'interrogation de données

- Langage qui permet de lire de données stockées dans la base de données
- Utilisant des concepts connus en algèbre relationnel : projection, jointure, intersection, union, produit cartésien...

Considérons la base de données poeⁱ contenant les deux tables suivantes

personne				
num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
2	Leberre	Bernard	1500	Marseille
3	Benamar	Pierre	1800	Lyon
4	Hadad	Karim	2500	Paris
5	Wick	John	3000	Paris

num : la clé primaire

vehicule				
immatriculation	marque	modele	annee	nump
100	Peugeot	5008	2018	5
200	Renault	clio	2000	4
300	Ford	fiesta	2010	1
400	Peugeot	106	2002	3
500	Citroen	C4	2015	4
600	Ford	Kuga	2019	
700	Fiat	punto	2008	5

immatriculation : clé primaire

nump : clé étrangère

SQL

Script de la création de la base de données

```
create database poei;

use poei;

create table personne (
    num int(3) primary key,
    nom varchar(20),
    prenom varchar(20),
    salaire int(4),
    ville varchar(20)
);

create table vehicule (
    immatriculation int(3) primary key,
    marque varchar(20),
    modele varchar(20),
    annee int(4),
    numbp int(3),
    constraint fk_vehicule_personne foreign key (numbp) references
        personne(num)
);
```

SQL

Script d'insertion de données

```
insert into personne values
    (1, 'Cohen', 'Sophie', 2000, 'Marseille'),
    (2, 'Leberre', 'Bernard', 1500, 'Marseille'),
    (3, 'Benamar', 'Pierre', 1800, 'Lyon'),
    (4, 'Hadad', 'Karim', 2500, 'Paris'),
    (5, 'Wick', 'John', 3000, 'Paris');

insert into vehicule values
    (100, 'Peugeot', '5008', 2018, 5),
    (200, 'Renault', 'clio', 2000, 4),
    (300, 'Ford', 'fiesta', 2010, 1),
    (400, 'Peugeot', '106', 2002, 3),
    (500, 'Citroen', 'C4', 2015, 4),
    (600, 'Ford', 'Kuga', 2019, null),
    (700, 'Fiat', 'punto', 2008, 5);
```

Remarque

Une requête SQL de lecture est composée d'au moins deux clauses

SQL

Remarque

Une requête SQL de lecture est composée d'au moins deux clauses

Quelques clauses possibles

- `select` : les colonnes à sélectionner
- `from` : les tables concernées
- `where` : les conditions
- ...

SQL

Pour sélectionner toutes les données de la table personne

```
SELECT *
FROM personne;
```

SQL

Pour sélectionner toutes les données de la table personne

```
SELECT *
FROM personne;
```

Le résultat

num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
2	Leberre	Bernard	1500	Marseille
3	Benamar	Pierre	1800	Lyon
4	Hadad	Karim	2500	Paris
5	Wick	John	3000	Paris

SQL

Et si on veut sélectionner que les deux premières personnes

```
SELECT *
FROM personne
LIMIT 2;
```

SQL

Et si on veut sélectionner que les deux premières personnes

```
SELECT *
FROM personne
LIMIT 2;
```

Le résultat

num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
2	Leberre	Bernard	1500	Marseille

SQL

Et si on veut sélectionner les deux personnes suivantes

```
SELECT *
FROM personne
LIMIT 2
OFFSET 2;
```

SQL

Et si on veut sélectionner les deux personnes suivantes

```
SELECT *
FROM personne
LIMIT 2
OFFSET 2;
```

Le résultat

num	nom	prenom	salaire	ville
3	Benamar	Pierre	1800	Lyon
4	Hadad	Karim	2500	Paris

SQL

On peut aussi simplifier l'écriture précédente

```
SELECT *
FROM personne
LIMIT 2, 2;
```

SQL

On peut aussi simplifier l'écriture précédente

```
SELECT *
FROM personne
LIMIT 2, 2;
```

Le résultat est le même

num	nom	prenom	salaire	ville
3	Benamar	Pierre	1800	Lyon
4	Hadad	Karim	2500	Paris

SQL

Pour filtrer les colonnes (ici la ville de chaque personne)

```
SELECT ville  
FROM personne;
```

Le résultat

ville
Marseille
Marseille
Lyon
Paris
Paris

SQL

Pour filtrer les colonnes (ici la ville de chaque personne)

```
SELECT ville  
FROM personne;
```

Le résultat

ville
Marseille
Marseille
Lyon
Paris
Paris

Comment on fait pour supprimer les doublons ?

SQL

Pour supprimer les doublons

```
SELECT distinct(ville)
FROM personne;
```

Le résultat

ville
Marseille
Lyon
Paris

Pour compter le nombre de ville dans la table personne

```
SELECT COUNT( distinct(ville))  
FROM personne;
```

Le résultat

+	-----+
	COUNT(distinct(ville))
+	-----+
	3
+	-----+

Pour compter le nombre de ville dans la table personne

```
SELECT COUNT( distinct(ville))  
FROM personne;
```

Le résultat

+	-----+
	COUNT(distinct(ville))
+	-----+
	3
+	-----+

Remarque

count compte les tuples (les lignes) et pas les colonnes.

Pour compter le nombre de ville dans la table personne

```
SELECT COUNT( distinct(ville))  
FROM personne;
```

Le résultat

+	-----+
	COUNT(distinct(ville))
+	-----+
	3
+	-----+

Remarque

count compte les tuples (les lignes) et pas les colonnes.

Question

Comment remplacer le nom de la colonne COUNT(distinct(ville)) par un autre personnalisé ?

SQL

Pour modifier le nom d'une colonne dans l'affichage du résultat

```
SELECT COUNT( distinct(ville)) as nombre_ville  
FROM personne;
```

Le résultat

nombre_ville
3

SQL

Pour sélectionner les personnes dont la ville = 'Marseille'

```
SELECT *
FROM personne
WHERE ville = 'Marseille';
```

Le résultat

num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
2	Leberre	Bernard	1500	Marseille

SQL

Il est possible de définir plusieurs conditions en utilisant les opérateurs logiques

- and
- or
- !

SQL

Exercice 1

Écrire une requête SQL qui permet de sélectionner les personnes qui habitent Marseille ou Lyon.

SQL

Exercice 1

Écrire une requête SQL qui permet de sélectionner les personnes qui habitent Marseille ou Lyon.

Exercice 2

Écrire une requête SQL qui permet de sélectionner les personnes dont le salaire est compris entre 2000 et 3000.

SQL

Pour l'exercice 2, on peut aussi utiliser **between et and**

```
SELECT *
FROM personne
WHERE salaire BETWEEN 2000 AND 3000;
```

Le résultat

num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
4	Hadad	Karim	2500	Paris
5	Wick	John	3000	Paris

SQL

Pour sélectionner les personnes ayant un salaire = 2000 ou 3000

```
SELECT *
FROM personne
WHERE salaire IN (2000, 3000);
```

Le résultat

num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
5	Wick	John	3000	Paris

Exercice 3

Écrire une requête SQL qui permet de sélectionner les personnes qui habitent Marseille et dont le salaire est soit inférieur à 2000 soit supérieur à 2500.

SQL

Pour l'exercice 3, on peut aussi utiliser **not** et **between**

```
SELECT *
FROM personne
WHERE ville = 'Marseille'
AND salaire NOT BETWEEN 2000 AND 2500;
```

Le résultat

num	nom	prenom	salaire	ville
2	Leberre	Bernard	1500	Marseille

SQL

Pour sélectionner les personnes dont le nom de la ville contient le caractère a

```
SELECT *
FROM personne
WHERE ville like '%a%';
```

Le résultat

num	nom	prenom	salaire	ville
1	Cohen	Sophie	2000	Marseille
2	Leberre	Bernard	1500	Marseille
4	Hadad	Karim	2500	Paris
5	Wick	John	3000	Paris

SQL

Remarque

- % : désigne 0, 1 ou plusieurs classes
- _ : désigne un seul caractère

SQL

Pour sélectionner les véhicules dont le numéro du propriétaire est non-nul

```
SELECT *
FROM  vehicule
WHERE  nump IS NOT NULL;
```

Le résultat

immatriculation	marque	modele	annee	nump
100	Peugeot	5008	2018	5
200	Renault	clio	2000	4
300	Ford	fiesta	2010	1
400	Peugeot	106	2002	3
500	Citroen	C4	2015	4
700	Fiat	punto	2008	5

SQL

Pour remplacer les valeurs nulles par une autre valeur, on peut utiliser la fonction `ifnull()`

```
SELECT marque, IFNULL(numprop, 'pas de propriétaire') AS propriétaire  
FROM vehicule;
```

Le résultat

marque	propriétaire
Peugeot	5
Renault	4
Ford	1
Peugeot	3
Citroen	4
Ford	pas de propriétaire
Fiat	5

SQL

Pour ordonner le résultat selon le numéro du propriétaire

```
SELECT *
FROM  vehicule
WHERE  nump IS NOT NULL
ORDER BY  nump;
```

Le résultat

immatriculation	marque	modele	annee	nump
300	Ford	fiesta	2010	1
400	Peugeot	106	2002	3
200	Renault	clio	2000	4
500	Citroen	C4	2015	4
100	Peugeot	5008	2018	5
700	Fiat	punto	2008	5

SQL

Pour compter le nombre de véhicule pour chaque personne, il faut les regrouper et utiliser une fonction d'agrégation

```
SELECT nump, COUNT(*) AS nombre_vehicule  
FROM vehicule  
WHERE nump is not null  
GROUP BY nump;
```

Le résultat

nump	nombre_vehicule
1	1
3	1
4	2
5	2

SQL

Pour filtrer les résultats de la fonction d'agrégation

```
SELECT nump, COUNT( *) as nombre_vehicule  
FROM vehicule  
WHERE nump is not null  
group by nump  
HAVING nombre_vehicule > 1;
```

Le résultat

nump	nombre_vehicule
4	2
5	2

SQL

Fonctions d'agrégation

- max
- min
- count
- sum
- avg

SQL

Fonctions d'agrégation

- max
- min
- count
- sum
- avg

Remarques

- Imbriquer les fonctions d'agrégation n'est pas possible.
- `distinct` n'est pas une fonction d'agrégation.
- Pas d'espace entre la fonction d'agrégation et la parenthèse ouvrante
(par exemple : `max(...)`)

SQL

Exercice 4

Écrire une requête SQL qui permet de compter la somme des salaires par ville

SQL

Exercice 4

Écrire une requête SQL qui permet de compter la somme des salaires par ville

Exercice 5

Écrire une requête SQL qui permet de sélectionner les numéros de personne qui ont un véhicule de marque Renault ou Citroen

SQL

Pour répondre à la question de l'exercice 5, on peut utiliser `union`

```
SELECT nump
FROM vehicule
WHERE marque = 'Renault'
UNION
SELECT nump
FROM vehicule
WHERE marque = 'Citroen' ;
```

Le résultat

nump
4

SQL

Opérations sur les ensembles

- union
- intersect
- except
- minus

SQL

Opérations sur les ensembles

- union
- intersect
- except
- minus

**Les trois dernières opérations ne fonctionnent pas avec MySQL.
Il est cependant possible de les remplacer par les requêtes imbriquées.**

Remarques

- Pour les propriétaires des véhicules, on affichait chaque fois le numéro.
- Pour l'utilisateur, il serait mieux de connaître les nom et prénom que le numéro.
- Les nom et prénom sont dans la table personne.

SQL

Remarques

- Pour les propriétaires des véhicules, on affichait chaque fois le numéro.
- Pour l'utilisateur, il serait mieux de connaître les nom et prénom que le numéro.
- Les nom et prénom sont dans la table personne.

Solution

Les jointures

SQL

Deux types de jointure

- Implicite : sans le mot-clé `join`
- Explicite : avec le mot-clé `join`

SQL

Pour sélectionner toutes les données de la table personne

```
SELECT nom, prenom, ville, marque, modele  
FROM personne, vehicule  
WHERE personne.num = vehicule.nump;
```

Le résultat

nom	prenom	ville	marque	modele
Cohen	Sophie	Marseille	Ford	fiesta
Benamar	Pierre	Lyon	Peugeot	106
Hadad	Karim	Paris	Renault	clio
Hadad	Karim	Paris	Citroen	C4
Wick	John	Paris	Peugeot	5008
Wick	John	Paris	Fiat	punto

SQL

Remarques

- La jointure se fait, généralement, sur une colonne commune entre deux tables (clé primaire dans une première table qui est étrangère dans une deuxième).
- En cas d'ambiguïté, c'est-à-dire, si deux colonnes portent le même nom, il faut les préfixer par le nom de leurs tables respectives.
- Il est aussi possible de définir et d'utiliser des alias.
- Les personnes n'ont pas de véhicules et les véhicules sont propriétaires n'apparaissent pas dans le résultat.

SQL

On peut aussi utiliser les alias

```
SELECT nom, prenom, ville, marque, modele  
FROM personne p, vehicule v  
WHERE p.num = v.nump;
```

Le résultat est le même

nom	prenom	ville	marque	modele
Cohen	Sophie	Marseille	Ford	fiesta
Benamar	Pierre	Lyon	Peugeot	106
Hadad	Karim	Paris	Renault	clio
Hadad	Karim	Paris	Citroen	C4
Wick	John	Paris	Peugeot	5008
Wick	John	Paris	Fiat	punto

SQL

En l'absence d'ambiguïté, on peut aussi écrire

```
SELECT nom, prenom, ville, marque, modele  
FROM personne, vehicule  
WHERE num = nump;
```

Le résultat est le même

nom	prenom	ville	marque	modele
Cohen	Sophie	Marseille	Ford	fiesta
Benamar	Pierre	Lyon	Peugeot	106
Hadad	Karim	Paris	Renault	clio
Hadad	Karim	Paris	Citroen	C4
Wick	John	Paris	Peugeot	5008
Wick	John	Paris	Fiat	punto

SQL

Plusieurs syntaxes

- JOIN ... ON : exactement comme la jointure implicite
- LEFT JOIN ... ON : jointure gauche
- RIGHT JOIN ... ON : jointure droite
- FULL JOIN ... ON : jointures droite + gauche (ne fonctionne pas avec **MySQL**)
- ...

SQL

On peut utiliser le mot-clé `join` et indiquer les colonnes sur lesquelles on fait la jointure

```
SELECT nom, prenom, ville, marque, modele  
FROM personne  
JOIN vehicule ON num = nump;
```

Le résultat est le même

nom	prenom	ville	marque	modele
Cohen	Sophie	Marseille	Ford	fiesta
Benamar	Pierre	Lyon	Peugeot	106
Hadad	Karim	Paris	Renault	clio
Hadad	Karim	Paris	Citroen	C4
Wick	John	Paris	Peugeot	5008
Wick	John	Paris	Fiat	punto

SQL

Pour afficher aussi les personnes qui n'ont pas de voiture, on peut écrire

```
SELECT nom, prenom, ville, marque, modele  
FROM personne  
LEFT JOIN vehicule ON num = nump;
```

Le résultat

nom	prenom	ville	marque	modele
Cohen	Sophie	Marseille	Ford	fiesta
Leberre	Bernard	Marseille	NULL	NULL
Benamar	Pierre	Lyon	Peugeot	106
Hadad	Karim	Paris	Renault	clio
Hadad	Karim	Paris	Citroen	C4
Wick	John	Paris	Peugeot	5008
Wick	John	Paris	Fiat	punto

SQL

Pour afficher les véhicules qui n'ont pas de propriétaire, on peut écrire

```
SELECT nom, prenom, ville, marque, modele  
FROM personne  
RIGHT JOIN vehicule ON num = nump;
```

Le résultat

nom	prenom	ville	marque	modele
Cohen	Sophie	Marseille	Ford	fiesta
Benamar	Pierre	Lyon	Peugeot	106
Hadad	Karim	Paris	Renault	clio
Hadad	Karim	Paris	Citroen	C4
Wick	John	Paris	Peugeot	5008
Wick	John	Paris	Fiat	punto
NULL	NULL	NULL	Ford	Kuga