Interrogation de bases de données SQL

Table des matières

I Contexte

Durée : 2h

Environnement de travail : DB Fiddle

Pré-requis : Aucun

Si une base de données a pour but premier d'assurer la cohérence des données stockées, elle doit aussi permettre d'y accéder efficacement.

La clause SELECT permet de récupérer toutes les données d'une table, mais bien souvent vous n'avez besoin de récupérer qu'un fragment de ces données. En d'autres termes, vous avez besoin de poser des conditions sur les données à sélectionner : seulement les personnes qui ont plus de 30 ans, seulement les jeux-vidéos en stock, etc.

Pour des raisons de performances, il est important de savoir récupérer seulement les données dont vous avez besoin. SQL fournit un ensemble d'outils permettant de filtrer les données à renvoyer à partir d'une requête SELECT.

II Interrogation des données (SELECT FROM WHERE)

Objectif

- Savoir récupérer des données dans une table.

Mise en situation

Supposez que vous gériez une libraire, et qu'un client vous demande de consulter tous les ouvrages d'Isaac Asimov que vous avez en stock.

Or, vous gérez des dizaines de milliers d'ouvrages, et il serait très fastidieux de rechercher à la main tous les livres écrits par Asimov.

Heureusement, SQL fournit les outils pour consulter les données d'une table en fonction d'un ou plusieurs critères, comme l'égalité d'un mot ou la supériorité d'un nombre.

Sélection

Az Définition

Une sélection est un type de requête du langage de manipulation de données de SQL.

Elle permet de **consulter les données** présentes dans les tables d'une base de données.

SELECT — FROM — WHERE

Syntaxe

Une sélection se décompose comme :

```
SELECT <liste d attributs projetés>
FROM <liste des tables>
WHERE <condition de la restriction>;
```

- La partie SELECT indique le sous-ensemble des attributs qui doivent apparaître dans la réponse.
- La partie FROM décrit les tables qui sont utilisables dans la requête (c'est-à-dire l'ensemble des attributs que l'on peut utiliser).
- La partie WHERE exprime les conditions logiques que doivent respecter les attributs d'un enregistrement pour pouvoir être dans la réponse. Cette partie est optionnelle.

Exemple

Si on dispose de la table suivante contenant les informations de personnes :

```
CREATE TABLE personne (
nom VARCHAR(50),
prenom VARCHAR(50),
age DECIMAL
);
```

On peut sélectionner les nom et prénom des personnes majeures ainsi :

```sql
SELECT nom, prenom
FROM personne
WHERE age > 18;
```

Cette requête sélectionne les attributs `nom` et `prenom` des tuples de la relation `personne`, ayant un attribut `age` supérieur à 18.

Notation préfixée

Syntaxe

Afin de décrire un attribut d'une relation en particulier (dans le cas d'une requête portant sur plusieurs relations notamment), on utilise la notation `relation.attribut`.

Exemple

```sql
SELECT personne.nom, personne.prenom, vol.depart
FROM personne, vol
WHERE personne.vol=vol.numero;
```

SELECT *

Syntaxe

Pour projeter l'ensemble des attributs d'une relation, on peut utiliser le caractère * à la place de la liste des attributs à projeter.

Exemple

```sql
SELECT *
FROM avion;
```

Cette requête sélectionne tous les attributs de la relation `Avion`.

III Exercice : Appliquer la notion

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

On crée les tables associées et leurs enregistrements.

```sql
CREATE TABLE societe(
nom VARCHAR(50),
pays VARCHAR(50),
activite VARCHAR(50),
PRIMARY KEY (nom)
);


CREATE TABLE employe(
num INTEGER,
nom VARCHAR(50),
prenom VARCHAR(50),
age INTEGER,
salaire REAL,
societe VARCHAR(50),
PRIMARY KEY (num),
FOREIGN KEY (societe) REFERENCES societe(nom)
);


INSERT INTO societe (nom, pays, activite)
VALUES ('Pied Piper', 'USA', 'Éditeur Logiciel');

INSERT INTO societe (nom, pays, activite)
VALUES ('LeapMusic', 'France', 'Éditeur Logiciel');

INSERT INTO societe (nom, activite)
VALUES ('Aperture Science', 'Fabrication de rideaux de douche');

INSERT INTO societe(nom, pays)
VALUES ('Black Mesa', 'USA');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(1, 'Hendricks', 'Richard', 28, 420000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(2, 'Gilfoyle', 'Bertram', 29, 399666, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(3, 'Dunn', 'Donald "Jared"', 25, 120000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(4, 'Hall', 'Monica', 27, 420000, 'Pied Piper');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(5, 'Lake', 'Marty', 28, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(6, 'Gamesh', 'Jul', 18, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(7, 'Famous', 'PokeMe', 32, 820000, 'LeapMusic');
INSERT INTO employe  (num, nom, prenom, age, salaire, societe)
VALUES(8, 'Jones', 'Eddy', 44, 420000, 'LeapMusic');

```

```sql
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(9, Null, 'Caroline', 28, 23000, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(10, 'Johnson', 'Cave', 56, 4444444, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(11, 'Rattmann', 'Doug', 45, 1337, 'Aperture Science');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(12, Null, 'G-Man', 42, 72000, 'Black Mesa');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(13, 'Freeman', 'Gordon', 34, 420000, 'Black Mesa');

```

Question 1

[solution n°1 p. 37]

On veut consulter l'intégralité des données des sociétés.

Écrire une requête SQL permettant de réaliser cela.

Question 2

[solution n°2 p. 37]

On veut connaître l'ensemble des employés de ces entreprises.

Écrire une requête SQL permettant de réaliser cela.

IV Projection (SELECT)

Objectifs

- Savoir récupérer les valeurs d'attributs spécifiques.
- Savoir ajouter des résultats de fonction aux résultats d'une sélection.

Mise en situation

Souvent, les informations que vous voulez consulter dans une base de données sont spécifiques. Par exemple, si vous avez créé une table qui répertorie les animaux, vous aurez un attribut pour le nom commun, un pour le nom scientifique, un pour la taille moyenne, un pour le nombre de pattes, etc.

Cette table pourrait contenir des dizaines d'attributs : si vous voulez récupérer tous les animaux à six pattes, il serait pertinent de n'afficher que leur nom.

SQL fournit la syntaxe pour restreindre les résultats d'une requête de sélection aux valeurs des attributs qui vous intéressent, et permet même d'enrichir le résultat avec des valeurs extérieures à la table, comme la date du jour.

Projection

Az Définition

Une projection est un type de sélection où seulement une **partie des attributs des tables** choisies est retenue pour le résultat.

Clause SELECT

Syntaxe

Supposons que l'on ait la table suivante définie comme :

```sql
CREATE TABLE R(
p1 INTEGER,
p2 INTEGER,
--- d'autres attributs ...
pn INTEGER);
```

On peut projeter sur les trois premiers attributs de cette table en utilisant :

```sql
SELECT p1, p2, p3
FROM R;
```

Illustration d'un projection

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R

A	C
1	10
2	10
3	20
4	

```
Projection(R1,A,C)

SELECT A, C
FROM R1
```

Tableau 1 Exemple de projection (SQL et Algèbre)

Ici, on ne garde que les attributs A et C de la table R1.

Exemple

```sql
CREATE TABLE parent (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0)
);

INSERT INTO parent VALUES (1, 'Brasseur', 'Alexandre', 91);
INSERT INTO parent VALUES (2, 'Brasseur', 'Pierre', 67);
```

```sql
SELECT prenom, nom
FROM parent;
```

```
  prenom   |   nom
-----------+----------
 Alexandre | Brasseur
 Pierre    | Brasseur
```

Alias de table

Syntaxe

Il est possible de redéfinir le nom des relations au sein de la requête afin d'en simplifier la syntaxe.

```sql
SELECT t1.attribut1
FROM table1 t1
```

Exemple

```sql
SELECT p.prenom, p.nom
FROM parent p;
```

```
 prenom   |   nom
----------+----------
 Alexandre | Brasseur
 Pierre    | Brasseur
```

Alias d'attribut (AS)

Syntaxe

Il est possible de redéfinir le nom des propriétés de la relation résultat.

```sql
SELECT attribut1 AS a1, attribut2 AS a2
FROM table;
```

Exemple

```sql
SELECT p.prenom AS prenom_parent, p.nom AS nom_parent
FROM parent p;
```

```
 prenom_parent | nom_parent
---------------+------------
 Alexandre     | Brasseur
 Pierre        | Brasseur
```

Projection de constante

Complément

Il est possible de projeter directement des constantes (on utilisera généralement un alias d'attribut pour nommer la colonne).

```sql
SELECT constante AS nom;
```

Cette requête renverra une table avec une seule ligne et une seule colonne à la valeur de *constante*.

Exemple

```sql
SELECT p.prenom AS prenom_parent, p.nom AS nom_parent, 'parent' AS statut
FROM parent p;
```

```
 prenom_parent | nom_parent | statut
---------------+------------+--------
 Alexandre     | Brasseur   | parent
 Pierre        | Brasseur   | parent
```

V Exercice : Appliquer la notion

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

On crée les tables associées et leurs enregistrements.

```
CREATE TABLE societe(
nom VARCHAR(50),
pays VARCHAR(50),
activite VARCHAR(50),
PRIMARY KEY (nom)
);


CREATE TABLE employe(
num INTEGER,
nom VARCHAR(50),
prenom VARCHAR(50),
age INTEGER,
salaire REAL,
societe VARCHAR(50),
PRIMARY KEY (num),
FOREIGN KEY (societe) REFERENCES societe(nom)
);


INSERT INTO societe (nom, pays, activite)
VALUES ('Pied Piper', 'USA', 'Éditeur Logiciel');

INSERT INTO societe (nom, pays, activite)
VALUES ('LeapMusic', 'France', 'Éditeur Logiciel');

INSERT INTO societe (nom, activite)
VALUES ('Aperture Science', 'Fabrication de rideaux de douche');

INSERT INTO societe(nom, pays)
VALUES ('Black Mesa', 'USA');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(1, 'Hendricks', 'Richard', 28, 420000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(2, 'Gilfoyle', 'Bertram', 29, 399666, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(3, 'Dunn', 'Donald "Jared"', 25, 120000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(4, 'Hall', 'Monica', 27, 420000, 'Pied Piper');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(5, 'Lake', 'Marty', 28, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(6, 'Gamesh', 'Jul', 18, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(7, 'Famous', 'PokeMe', 32, 820000, 'LeapMusic');
INSERT INTO employe  (num, nom, prenom, age, salaire, societe)
VALUES(8, 'Jones', 'Eddy', 44, 420000, 'LeapMusic');

```

```sql
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(9, Null, 'Caroline', 28, 23000, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(10, 'Johnson', 'Cave', 56, 4444444, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(11, 'Rattmann', 'Doug', 45, 1337, 'Aperture Science');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(12, Null, 'G-Man', 42, 72000, 'Black Mesa');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(13, 'Freeman', 'Gordon', 34, 420000, 'Black Mesa');

```

Question 1

[solution n°3 p. 37]

On veut connaître les nom de tous employés des entreprises.

Écrire une requête SQL permettant de réaliser cela.

Question 2

[solution n°4 p. 38]

On veut connaître les activités de toutes les sociétés.

Écrire une requête SQL permettant de réaliser cela.

VI Restriction (WHERE)

Objectif

- Savoir filtrer les résultats d'une requête de sélection selon un ou plusieurs critères.

Mise en situation

SQL permet de récupérer tous les enregistrements d'une table qui répondent à une ou plusieurs conditions, comme l'égalité d'un attribut avec une constante.

Mais il y a des cas où ces conditions sont plus subtiles. Dans une table qui stocke des articles et leurs prix, il pourrait être utile de récupérer les articles dans une certaine tranche de prix, entre 5 et 10 € par exemple.

Dans cette même table, pour récupérer tous les écrans d'ordinateurs, on voudrait pouvoir faire une recherche approximative sur le nom des articles, pour récupérer tous ceux qui contiennent le mot *« écran »*.

SQL fournit des opérateurs permettant de réaliser ces comparaisons plus fines.

Restriction

Az Définition

Une restriction est un type de sélection où l'**on se restreint à des enregistrements** dont les attributs vérifient une ou plusieurs conditions.

Clause WHERE

Syntaxe

```sql
SELECT *
FROM R
WHERE <condition>
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R

A	B	C
1	Alpha	10
2	Bravo	10

```
Restriction(R1,C<20)

SELECT *
FROM R1
WHERE C<20
```

Tableau 2 Exemple de restriction (SQL et Algèbre)

Ici on se restreint à des enregistrements de la table R1 dont l'attribut « *C* » est inférieur strictement à 20.

Introduction

La clause WHERE d'une instruction de sélection est définie par une condition. Une telle condition s'exprime à l'aide d'opérateurs de comparaison et d'opérateurs logiques. Le résultat d'une expression de condition est toujours un booléen.

Condition

Az Définition

```
Condition Elémentaire ::= Attribut <Opérateur de comparaison> Constante
Condition ::= Condition <Opérateur logique> Condition | Condition Elémentaire
```

Opérateurs de comparaison

Syntaxe

Les opérateurs de comparaison sont, avec attribut A et constante C :

- `A = C`
- `A <> C` (différence)
- `A < C`
- `A > C`
- `A <= C`
- `A >= C`
- `P BETWEEN C1 AND C2`
- `P IN (C1, C2, ...)`
- `P LIKE 'chaîne'`
- `P IS NULL`

Opérateurs logiques

Syntaxe

Les opérateurs logiques sont :

- `OR`
- `AND`
- `NOT`

Opérateur LIKE

Remarque

L'opérateur `LIKE 'chaîne'` permet d'insérer des *jokers* dans l'opération de comparaison (alors que l'opérateur = teste une égalité stricte) :

- Le joker % désigne 0 ou plusieurs caractères quelconques.
- Le joker _ désigne 1 et 1 seul caractère.

On préférera l'opérateur = à l'opérateur `LIKE` lorsque la comparaison n'utilise pas de joker.

VII Exercice : Appliquer la notion

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

On crée les tables associées et leurs enregistrements.

```sql
CREATE TABLE societe(
nom VARCHAR(50),
pays VARCHAR(50),
activite VARCHAR(50),
PRIMARY KEY (nom)
);


CREATE TABLE employe(
num INTEGER,
nom VARCHAR(50),
prenom VARCHAR(50),
age INTEGER,
salaire REAL,
societe VARCHAR(50),
PRIMARY KEY (num),
FOREIGN KEY (societe) REFERENCES societe(nom)
);


INSERT INTO societe (nom, pays, activite)
VALUES ('Pied Piper', 'USA', 'Éditeur Logiciel');

INSERT INTO societe (nom, pays, activite)
VALUES ('LeapMusic', 'France', 'Éditeur Logiciel');

INSERT INTO societe (nom, activite)
VALUES ('Aperture Science', 'Fabrication de rideaux de douche');

INSERT INTO societe(nom, pays)
VALUES ('Black Mesa', 'USA');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(1, 'Hendricks', 'Richard', 28, 420000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(2, 'Gilfoyle', 'Bertram', 29, 399666, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(3, 'Dunn', 'Donald "Jared"', 25, 120000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(4, 'Hall', 'Monica', 27, 420000, 'Pied Piper');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(5, 'Lake', 'Marty', 28, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(6, 'Gamesh', 'Jul', 18, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(7, 'Famous', 'PokeMe', 32, 820000, 'LeapMusic');
INSERT INTO employe  (num, nom, prenom, age, salaire, societe)
VALUES(8, 'Jones', 'Eddy', 44, 420000, 'LeapMusic');

```

```sql
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(9, Null, 'Caroline', 28, 23000, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(10, 'Johnson', 'Cave', 56, 4444444, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(11, 'Rattmann', 'Doug', 45, 1337, 'Aperture Science');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(12, Null, 'G-Man', 42, 72000, 'Black Mesa');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(13, 'Freeman', 'Gordon', 34, 420000, 'Black Mesa');

```

Question 1

[solution n°5 p. 38]

Écrire une requête SQL permettant de sélectionner les employés de *Pied Piper*.

Question 2

[solution n°6 p. 39]

Écrire une requête SQL permettant de sélectionner les sociétés exerçant en France.

VIII Jointure (JOIN ON)

Objectif

- Savoir combiner les informations issues de plusieurs tables.

Mise en situation

Une base de données relationnelle est rarement composée d'une seule table, principalement pour éviter la duplication d'informations.

On pourra par exemple imaginer la base de données d'une messagerie instantanée, qui gère dans une table les messages, et dans une autre table les utilisateurs. Ces deux tables sont liées par une clé étrangère.

Mais cette séparation pose un problème : comment récupérer le pseudonyme de l'auteur d'un message, puisque ce pseudonyme est stocké dans une autre table que le message lui-même ?

C'est l'objet des jointures, qui permettent de combiner les données issues de plusieurs tables.

Jointure

Az Définition

Une jointure est un type de sélection pour consulter les données de plusieurs tables qui se base sur les valeurs jointes de certains des attributs de ces tables.

Jointure par la clause ON

Syntaxe

```
1 SELECT *
2 FROM R1 INNER JOIN R2
3 ON <condition>
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox

```
Jointure(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 INNER JOIN R2
ON R1.C=R2.X
```

Tableau 3 Exemple de jointure (SQL et Algèbre)

Méthode

- Comme une jointure implique plusieurs tables, on utilise les alias de table pour simplifier l'écriture des requêtes.
- Il est également fréquent d'utiliser des alias d'attribut pour différencier des attributs qui ont le même nom dans les tables jointes.

Exemple

```sql
CREATE TABLE parent (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0)
);

INSERT INTO parent VALUES (1, 'Brasseur', 'Alexandre', 91);
INSERT INTO parent VALUES (2, 'Brasseur', 'Pierre', 67);
```

```sql
CREATE TABLE enfant (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0),
parent INTEGER NOT NULL,
FOREIGN KEY (parent) REFERENCES parent(id)
);

INSERT INTO enfant VALUES (3, 'Brasseur', 'Claude', 42, 2);
```

```sql
SELECT e.prenom, e.nom, p.prenom AS parent
FROM enfant e INNER JOIN parent p
ON e.parent=p.id;
```

```
 prenom |   nom    | parent
--------+----------+--------
 Claude | Brasseur | Pierre
```

Jointure par la clause WHERE

Syntaxe

On peut aussi écrire une jointure comme la composition d'un produit et d'une restriction :

```sql
SELECT *
FROM R1, R2, Ri
WHERE <condition>
```

Avec `condition` permettant de joindre des attributs des `Ri`

Exemple

```sql
SELECT e.prenom, e.nom, p.prenom AS parent
FROM enfant e, parent p
WHERE e.parent=p.id;
```

Complément

Le mot-clé `INNER` est optionnel car c'est le mode par défaut d'une jointure (on parle de jointure interne), on peut donc également écrire :

```
SELECT *
FROM R1 JOIN R2
ON <condition>
```

IX Exercice : Appliquer la notion

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

On crée les tables associées et leurs enregistrements.

```
CREATE TABLE societe(
nom VARCHAR(50),
pays VARCHAR(50),
activite VARCHAR(50),
PRIMARY KEY (nom)
);


CREATE TABLE employe(
num INTEGER,
nom VARCHAR(50),
prenom VARCHAR(50),
age INTEGER,
salaire REAL,
societe VARCHAR(50),
PRIMARY KEY (num),
FOREIGN KEY (societe) REFERENCES societe(nom)
);


INSERT INTO societe (nom, pays, activite)
VALUES ('Pied Piper', 'USA', 'Éditeur Logiciel');

INSERT INTO societe (nom, pays, activite)
VALUES ('LeapMusic', 'France', 'Éditeur Logiciel');

INSERT INTO societe (nom, activite)
VALUES ('Aperture Science', 'Fabrication de rideaux de douche');

INSERT INTO societe(nom, pays)
VALUES ('Black Mesa', 'USA');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(1, 'Hendricks', 'Richard', 28, 420000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(2, 'Gilfoyle', 'Bertram', 29, 399666, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(3, 'Dunn', 'Donald "Jared"', 25, 120000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(4, 'Hall', 'Monica', 27, 420000, 'Pied Piper');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(5, 'Lake', 'Marty', 28, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(6, 'Gamesh', 'Jul', 18, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(7, 'Famous', 'PokeMe', 32, 820000, 'LeapMusic');
INSERT INTO employe  (num, nom, prenom, age, salaire, societe)
VALUES(8, 'Jones', 'Eddy', 44, 420000, 'LeapMusic');

```

```sql
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(9, Null, 'Caroline', 28, 23000, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(10, 'Johnson', 'Cave', 56, 4444444, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(11, 'Rattmann', 'Doug', 45, 1337, 'Aperture Science');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(12, Null, 'G-Man', 42, 72000, 'Black Mesa');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(13, 'Freeman', 'Gordon', 34, 420000, 'Black Mesa');

```

Question 1

[solution n°7 p. 39]

Écrire une requête SQL permettant de sélectionner les noms des employés avec le pays de leur entreprise.

Indice :

On ne retournera pas les lignes correspondant à des employés qui n'ont pas de nom.

Question 2

[solution n°8 p. 40]

Écrire une requête SQL permettant de sélectionner l'âge des employés de *Black Mesa*.

X Opérateurs ensemblistes

Objectifs

- Comprendre l'analogie entre table et ensemble.
- Savoir réaliser des opérations ensemblistes sur des tables.

Mise en situation

Imaginez que vous gériez une base de données qui liste tous les langages de programmation. Vous décidez d'effectuer votre découpage en tables, selon le type du langage : impératif, comme C, déclaratif, comme SQL, événementiel, comme JavaScript, etc. Vous obtenez donc une table par type de langage.

Mais certains langages ont plusieurs types ! Par exemple, JavaScript est à la fois impératif et événementiel.

Comment faire pour récupérer tous les langages qui sont à la fois impératifs et événementiels ? Cette opération est très facile si on utilise les opérateurs ensemblistes.

Introduction

On peut voir les tables comme des ensembles sur lesquels on peut réaliser des opérations ensemblistes. Pour ce faire, on utilise des opérateurs ensemblistes.

Union

Syntaxe

```
SELECT * FROM R1
UNION
SELECT * FROM R2
```

Exemple

```
CREATE TABLE parent (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0)
);

INSERT INTO parent VALUES (1, 'Brasseur', 'Alexandre', 91);
INSERT INTO parent VALUES (2, 'Brasseur', 'Pierre', 67);

CREATE TABLE enfant (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0)
);

INSERT INTO enfant VALUES (2, 'Brasseur', 'Pierre', 67);
INSERT INTO enfant VALUES (3, 'Brasseur', 'Claude', 42);
```

```sql
SELECT *
FROM enfant
UNION
SELECT *
FROM parent;
```

```
 id |   nom    |  prenom   | age
----+----------+-----------+-----
  2 | Brasseur | Pierre    |  67
  3 | Brasseur | Claude    |  42
  1 | Brasseur | Alexandre |  91
```

Dédoublonnage Remarque

Par défaut `UNION` supprime les doublons, la clause `UNION ALL` permet de conserver les doublons.

Attention

Pour qu'un opérateur ensembliste soit valide il est **nécessaire** que les deux relations membres aient le **même schéma**.

Exemple

```sql
CREATE TABLE parent (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0)
);

INSERT INTO parent VALUES (1, 'Brasseur', 'Alexandre', 91);
INSERT INTO parent VALUES (2, 'Brasseur', 'Pierre', 67);

CREATE TABLE enfant (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0),
parent INTEGER NOT NULL,
FOREIGN KEY (parent) REFERENCES parent(id)
);

INSERT INTO enfant VALUES (2, 'Brasseur', 'Pierre', 67, 1);
INSERT INTO enfant VALUES (3, 'Brasseur', 'Claude', 42, 2);
```

```sql
/** Cette requête renvoie une erreur car les schémas ne sont pas identiques. */
SELECT *
FROM enfant
UNION
SELECT *
FROM parent;
```

```sql
/** Cette requête fonctionne car après la projection les schémas sont identiques. */
SELECT id, nom, prenom, age
FROM enfant
UNION
SELECT id, nom, prenom, age
FROM parent;
```

Intersection

Syntaxe

```sql
SELECT * FROM R1
INTERSECT
SELECT * FROM R2
```

Exemple

```sql
SELECT id, nom, prenom, age
FROM enfant
INTERSECT
SELECT id, nom, prenom, age
FROM parent;
```

```
 id |   nom    | prenom | age
----+----------+--------+-----
  2 | Brasseur | Pierre |  67
```

On sélectionne ici les personnes qui sont enfants et parents.

Différence

Syntaxe

```sql
SELECT * FROM R1
EXCEPT
SELECT * FROM R2
```

Exemple

```sql
SELECT id, nom, prenom, age
FROM enfant
EXCEPT
SELECT id, nom, prenom, age
FROM parent;
```

```
 id |   nom    | prenom | age
----+----------+--------+-----
  3 | Brasseur | Claude |  42
```

On sélectionne ici les enfants qui ne sont pas des parents.

Remarque

Les opérations `INTERSECT` et `EXCEPT` n'existent que dans la norme SQL2, et non dans la norme SQL1. Certains SGBD sont susceptibles de ne pas les implémenter.

XI Exercice : Appliquer la notion

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

On crée les tables associées et leurs enregistrements.

```
CREATE TABLE societe(
nom VARCHAR(50),
pays VARCHAR(50),
activite VARCHAR(50),
PRIMARY KEY (nom)
);


CREATE TABLE employe(
num INTEGER,
nom VARCHAR(50),
prenom VARCHAR(50),
age INTEGER,
salaire REAL,
societe VARCHAR(50),
PRIMARY KEY (num),
FOREIGN KEY (societe) REFERENCES societe(nom)
);


INSERT INTO societe (nom, pays, activite)
VALUES ('Pied Piper', 'USA', 'Éditeur Logiciel');

INSERT INTO societe (nom, pays, activite)
VALUES ('LeapMusic', 'France', 'Éditeur Logiciel');

INSERT INTO societe (nom, activite)
VALUES ('Aperture Science', 'Fabrication de rideaux de douche');

INSERT INTO societe(nom, pays)
VALUES ('Black Mesa', 'USA');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(1, 'Hendricks', 'Richard', 28, 420000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(2, 'Gilfoyle', 'Bertram', 29, 399666, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(3, 'Dunn', 'Donald "Jared"', 25, 120000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(4, 'Hall', 'Monica', 27, 420000, 'Pied Piper');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(5, 'Lake', 'Marty', 28, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(6, 'Gamesh', 'Jul', 18, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(7, 'Famous', 'PokeMe', 32, 820000, 'LeapMusic');
INSERT INTO employe  (num, nom, prenom, age, salaire, societe)
VALUES(8, 'Jones', 'Eddy', 44, 420000, 'LeapMusic');

```

```sql
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(9, Null, 'Caroline', 28, 23000, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(10, 'Johnson', 'Cave', 56, 4444444, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(11, 'Rattmann', 'Doug', 45, 1337, 'Aperture Science');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(12, Null, 'G-Man', 42, 72000, 'Black Mesa');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(13, 'Freeman', 'Gordon', 34, 420000, 'Black Mesa');

```

Question 1

[solution n°9 p. 40]

Écrire une requête SQL permettant de sélectionner les prénoms des employés de l'entreprise *LeapMusic*.

Question 2

[solution n°10 p. 40]

Écrire une requête SQL permettant de sélectionner les noms des employés de l'entreprise *Aperture Science*.

Question 3

[solution n°11 p. 41]

Écrire une requête SQL permettant de sélectionner les prénoms des employés de l'entreprise *Leap Music* et les noms des employés de l'entreprise *Aperture Science*.

XII Tri (ORDER BY) et dédoublonnage (SELECT DISTINCT)

Objectifs

- Savoir trier les résultats d'une requête SQL.
- Savoir enlever les doublons des résultats d'une requête SQL.

Mise en situation

Les enregistrements d'une table ne sont pas ordonnés, ce qui implique que l'ordre des résultats d'une requête de sélection n'est pas prévisible.

Or il peut être très utile de trier les résultats selon un critère, par exemple pour afficher les articles en stock, en allant des prix les plus bas aux pris les plus hauts.

Aussi, lorsque l'on ne récupère que quelques attributs d'une table, comme le nom d'une personne, il y des risques de doublons. Comment supprimer ces doublons de l'affichage ?

SQL fournit des instructions pour le tri et le dédoublonnage des résultats.

Tri des enregistrement — Fondamental

On peut trier les résultats de sélection en fonction des valeurs de certains attributs des enregistrements retournés.

ORDER BY — Syntaxe

```
SELECT liste d'attributs projetés
FROM liste de relations
WHERE condition
ORDER BY liste ordonnée d'attributs
```

Les tuples sont triés d'abord par le premier attribut spécifié dans la clause ORDER BY, puis en cas de doublons par le second, etc.

Exemple

```
SELECT *
FROM parent
ORDER BY nom, age;
```

Tri décroissant — Remarque

Le tri défini à l'aide de ORDER BY est un tri croissant. Pour effectuer un tri décroissant on fait suivre l'attribut du mot-clé DESC.

Exemple

```sql
SELECT *
FROM parent
ORDER BY nom, age DESC;
```

Dédoublonnage des enregistrement Attention

L'opérateur SELECT n'élimine pas les doublons (i.e. les tuples identiques dans la relation résultat) par défaut. Il faut pour cela utiliser l'opérateur `SELECT DISTINCT`.

Exemple

```sql
CREATE TABLE parent (
id INTEGER PRIMARY KEY,
nom VARCHAR(255),
prenom VARCHAR(255),
age INTEGER CHECK (age > 0)
);

INSERT INTO parent VALUES (1, 'Brasseur', 'Alexandre', 91);
INSERT INTO parent VALUES (2, 'Brasseur', 'Pierre', 67);
```

```sql
SELECT nom
FROM parent;
```

```
   nom
----------
 Brasseur
 Brasseur
(2 rows)
```

Cette requête sélectionne l'attribut `nom` de la relation `parent` et renvoie tous les enregistrements.

```sql
SELECT DISTINCT nom
FROM parent;
```

```
   nom
----------
 Brasseur
(1 row)
```

Cette requête sélectionne l'attribut `nom` de la relation `parent` et renvoie les enregistrements sans doublons (donc elle ne renvoie qu'un seul enregistrement si plusieurs parents ont le même nom).

XIII Exercice : Appliquer la notion

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

On crée les tables associées et leurs enregistrements.

```sql
CREATE TABLE societe(
nom VARCHAR(50),
pays VARCHAR(50),
activite VARCHAR(50),
PRIMARY KEY (nom)
);


CREATE TABLE employe(
num INTEGER,
nom VARCHAR(50),
prenom VARCHAR(50),
age INTEGER,
salaire REAL,
societe VARCHAR(50),
PRIMARY KEY (num),
FOREIGN KEY (societe) REFERENCES societe(nom)
);


INSERT INTO societe (nom, pays, activite)
VALUES ('Pied Piper', 'USA', 'Éditeur Logiciel');

INSERT INTO societe (nom, pays, activite)
VALUES ('LeapMusic', 'France', 'Éditeur Logiciel');

INSERT INTO societe (nom, activite)
VALUES ('Aperture Science', 'Fabrication de rideaux de douche');

INSERT INTO societe(nom, pays)
VALUES ('Black Mesa', 'USA');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(1, 'Hendricks', 'Richard', 28, 420000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(2, 'Gilfoyle', 'Bertram', 29, 399666, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(3, 'Dunn', 'Donald "Jared"', 25, 120000, 'Pied Piper');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(4, 'Hall', 'Monica', 27, 420000, 'Pied Piper');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(5, 'Lake', 'Marty', 28, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(6, 'Gamesh', 'Jul', 18, 420000, 'LeapMusic');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(7, 'Famous', 'PokeMe', 32, 820000, 'LeapMusic');
INSERT INTO employe  (num, nom, prenom, age, salaire, societe)
VALUES(8, 'Jones', 'Eddy', 44, 420000, 'LeapMusic');

```

```sql
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(9, Null, 'Caroline', 28, 23000, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(10, 'Johnson', 'Cave', 56, 4444444, 'Aperture Science');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(11, 'Rattmann', 'Doug', 45, 1337, 'Aperture Science');

INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(12, Null, 'G-Man', 42, 72000, 'Black Mesa');
INSERT INTO employe (num, nom, prenom, age, salaire, societe)
VALUES(13, 'Freeman', 'Gordon', 34, 420000, 'Black Mesa');

```

Question

[solution n°12 p. 41]

Écrire une requête SQL qui affiche les salaires de tous les employés de manière unique et dans un ordre décroissant.

XIV Essentiel

SQL possède de nombreuses options permettant d'interroger une base de données et de récupérer finement les informations souhaitées.

La clause `SELECT` permet de ne récupérer que quelques attributs, comme le nom et le prénom d'une personne.

La clause `WHERE` permet de sélectionner des enregistrements qui répondent à des conditions précises, par exemple en sélectionnant uniquement les personnes nées après 1970.

Les jointures permettent de réunir des informations présentes dans des tables différentes, par exemple pour récupérer le nom de l'auteur d'un livre à partir d'une simple clé étrangère désignant l'auteur.

Enfin, les clauses `UNIQUE` et `ORDER BY` permettent de mieux présenter les résultats, en les triant et en supprimant les doublons.

XV Quiz

Exercice 1 : Quiz - Culture

[solution n°13 p. 42]

Exercice

Quels sont les termes qui se rapportent aux requêtes de sélection de données ?

- **A** Projection
- **B** Création
- **C** Contraintes
- **D** Conditions
- **E** Jointure
- **F** Restriction

Exercice

Quels sont les mots-clés associés aux requêtes de sélection de données ?

- **A** `TABLE`
- **B** `CREATE`
- **C** `SELECT`
- **D** `WHERE`
- **E** `DROP`
- **F** `ORDER`
- **G** `ALTER`
- **H** `FROM`
- **I** `JOIN`

Exercice 4 : Quiz - Méthode

[solution n°14 p. 43]

Exercice

Cette requête est-elle syntaxiquement correcte ?

```sql
SELECT b.numero, b.superficie, b.perimetre, b.nb_etages
FROM batiment;
```

- A Oui
- B Non

Exercice

Cette requête est-elle syntaxiquement correcte ?

```sql
SELECT b.numero, b.superficie, b.perimetre, b.nb_etages
FROM batiment b
ORDER BY b.numero
WHERE b.nb_etages > 0;
```

- A Oui
- B Non

Exercice

Cette requête est-elle syntaxiquement correcte ?

```sql
SELECT DISTINCT b.nb_etages, a.rue, a.ville
FROM batiment b JOIN adresse a
ON b.id_adresse = a.id
WHERE b.nb_etages > 0
ORDER BY b.nb_etages DESC;
```

- A Oui
- B Non

Exercice 8 : Quiz - Code

[solution n°15 p. 44]

Exercice

Cette requête :

```sql
SELECT DISTINCT prix_paye
FROM paiement_loyer
WHERE code_loc='X'

INTERSECT

SELECT DISTINCT prix_paye
FROM paiement_loyer
WHERE code_loc='Y';
```

... contient (au moins) :

- **A** Une projection
- **B** Une restriction
- **C** Une jointure
- **D** Un tri
- **E** Un dédoublonnage
- **F** Une opération ensembliste

Exercice

Cette requête :

```sql
SELECT *
FROM location l JOIN appartement a
ON l.code_appt=a.code_appt
ORDER BY l.code_appt;
```

... contient :

- **A** Une projection
- **B** Une restriction
- **C** Une jointure
- **D** Un tri
- **E** Un dédoublonnage
- **F** Une opération ensembliste

Solutions des exercices

Solution n°1

[exercice p. 7]

```
1 SELECT *
2 FROM societe;
```

On obtient le résultat suivant :

nom	pays	activite
Pied Piper	USA	Éditeur Logiciel
LeapMusic	France	Éditeur Logiciel
Aperture Science	null	Fabrication de rideaux de douche
Black Mesa	USA	null

Solution n°2

[exercice p. 7]

```
1 SELECT *
2 FROM employe;
```

On obtient un résultat qui commence par :

num	nom	prenom	age	salaire	societe
1	Hendricks	Richard	28	420000	Pied Piper
2	Gilfoyle	Bertram	29	399666	Pied Piper
3	Dunn	Donald "Jared"	25	120000	Pied Piper

Solution n°3

[exercice p. 12]

```
1 SELECT nom
2 FROM employe;
```

On obtient :

nom
Hendricks
Gilfoyle
Dunn
Hall

nom
Lake
Gamesh
Famous
Jones
null
Johnson
Rattmann
null
Freeman

Solution n°4

[exercice p. 12]

```
1 SELECT activite
2 FROM societe;
```

On obtient :

activite
Éditeur Logiciel
Éditeur Logiciel
Fabrication de rideaux de douche
null

Solution n°5

[exercice p. 17]

```
1 SELECT *
2 FROM employe
3 WHERE societe='Pied Piper';
```

On obtient les résultats :

num	nom	prenom	age	salaire	societe
1	Hendricks	Richard	28	420000	Pied Piper
2	Gilfoyle	Bertram	29	399666	Pied Piper
3	Dunn	Donald "Jared"	25	120000	Pied Piper
4	Hall	Monica	27	420000	Pied Piper

Solution n°6

[exercice p. 17]

```sql
SELECT *
FROM societe
WHERE pays='France';
```

On obtient les résultats :

nom	pays	activite
LeapMusic	France	Éditeur Logiciel

Solution n°7

[exercice p. 22]

```sql
SELECT e.nom AS nom_employe, s.pays AS pays_societe
FROM employe e INNER JOIN societe s
ON e.societe=s.nom
WHERE e.nom IS NOT NULL;
```

On obtient le résultat suivant :

Nom_Employe	Pays_Societe
Hendricks	USA
Gilfoyle	USA
Dunn	USA
Aperture Science	USA
Lake	France
Gamesh	France
Famous	France
Jones	France
Johnson	
Rattmann	
Freeman	USA

Solution n°8

[exercice p. 22]

```
1 SELECT e.age AS nom_employe, s.nom AS nom_societe
2 FROM employe e INNER JOIN societe s
3 ON e.societe=s.nom
4 WHERE s.nom = 'Black Mesa';
```

On obtient le résultat suivant :

Age_Employe	Nom_Societe
42	Black Mesa
34	Black Mesa

Solution n°9

[exercice p. 28]

```
1 SELECT employe.prenom
2 FROM employe JOIN societe
3 ON  employe.societe=societe.nom
4 WHERE societe.nom='LeapMusic';
```

On obtient :

prenom
Marty
Jul
PokeMe
Eddy

Solution n°10

[exercice p. 28]

```
1 SELECT employe.nom
2 FROM employe JOIN societe
3 ON employe.societe=societe.nom
4 WHERE societe.nom='Aperture Science';
```

On obtient :

nom
Johnson
Rattmann

Solution n°11

[exercice p. 28]

```sql
SELECT employe.prenom
FROM employe JOIN societe
ON employe.societe=societe.nom
WHERE societe.nom='LeapMusic'

UNION

SELECT employe.nom
FROM employe JOIN societe
ON employe.societe=societe.nom
WHERE societe.nom='Aperture Science';
```

On obtient :

prenom
Eddy
Johnson
Jul
Marty
PokeMe
Rattmann

Solution n°12

[exercice p. 32]

```sql
SELECT DISTINCT salaire
FROM employe
ORDER BY salaire DESC;
```

On obtient :

salaire
4444444
820000
420000
399666
120000
72000
23000
1337

Solution n°13

[exercice p. 34]

Exercice

Quels sont les termes qui se rapportent aux requêtes de sélection de données ?

- **A** Projection
- **B** Création
- **C** Contraintes
- **D** Conditions
- **E** Jointure
- **F** Restriction

Exercice

Quels sont les mots-clés associés aux requêtes de sélection de données ?

- **A** `TABLE`
- **B** `CREATE`
- **C** `SELECT`
- **D** `WHERE`
- **E** `DROP`
- **F** `ORDER`
- **G** `ALTER`
- **H** `FROM`
- **I** `JOIN`

Solution n°14

[exercice p. 35]

Exercice

Cette requête est-elle syntaxiquement correcte ?

```sql
SELECT b.numero, b.superficie, b.perimetre, b.nb_etages
FROM batiment;
```

A Oui

B Non

Pour pouvoir utiliser l'alias de table b il faut le déclarer :

```sql
SELECT b.numero, b.superficie, b.perimetre, b.nb_etages
FROM batiment b;
```

Exercice

Cette requête est-elle syntaxiquement correcte ?

```sql
SELECT b.numero, b.superficie, b.perimetre, b.nb_etages
FROM batiment b
ORDER BY b.numero
WHERE b.nb_etages > 0;
```

A Oui

B Non

Les clauses ORDER BY et WHERE ne sont pas dans le bon ordre : il faut les inverser.

```sql
SELECT b.numero, b.superficie, b.perimetre, b.nb_etages
FROM batiment b
WHERE b.nb_etages > 0
ORDER BY b.numero;
```

Exercice

Cette requête est-elle syntaxiquement correcte ?

```sql
SELECT DISTINCT b.nb_etages, a.rue, a.ville
FROM batiment b JOIN adresse a
ON b.id_adresse = a.id
WHERE b.nb_etages > 0
ORDER BY b.nb_etages DESC;
```

A Oui

B Non

Solution n°15

[exercice p. 35]

Exercice

Cette requête :

```sql
SELECT DISTINCT prix_paye
FROM paiement_loyer
WHERE code_loc='X'

INTERSECT

SELECT DISTINCT prix_paye
FROM paiement_loyer
WHERE code_loc='Y';
```

... contient (au moins) :

- **A** Une projection
- **B** Une restriction
- **C** Une jointure
- **D** Un tri
- **E** Un dédoublonnage
- **F** Une opération ensembliste

Exercice

Cette requête :

```sql
SELECT *
FROM location l JOIN appartement a
ON l.code_appt=a.code_appt
ORDER BY l.code_appt;
```

... contient :

- **A** Une projection
- **B** Une restriction
- **C** Une jointure
- **D** Un tri
- **E** Un dédoublonnage

F Une opération ensembliste

Index