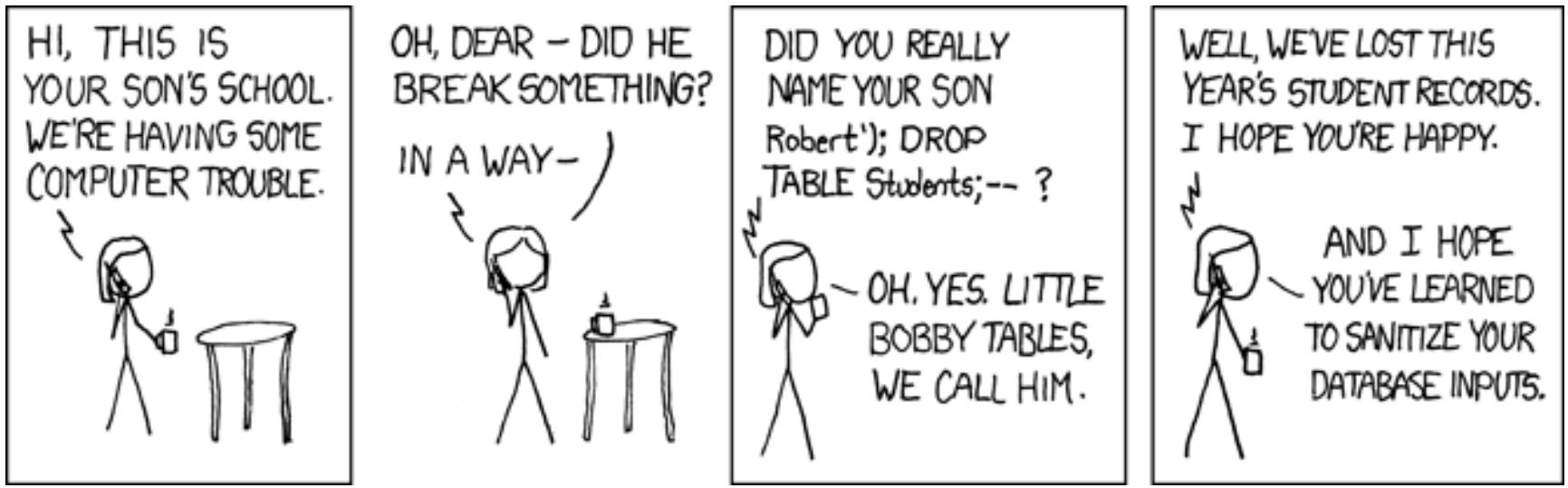
SQL 2 : Plusieurs tables
JOIN, UNION, INTERSECT...

Quentin Fortier

May 17, 2022



QCM ICNA

Question 12 Dans une base de données, on souhaite sélectionner les trois champs "id", "question" et "reponse" de tous les enregistrements d'une table nommée "QCM". Quelle(s) requête(s) SQL peut-on utiliser ?

A) SELECT * FROM QCM

B) SELECT in QCM ALL id,question,reponse

C) SELECT id,question,reponse FROM QCM

D) SELECT QCM WHERE id,question,reponse

Union

Si R_1 et R_2 sont des tables ayant le **même schéma relationnel**, $R_1 \cup R_2$ contient les enregistrements dans R_1 ou R_2:

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2		
A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 \cup R_2$		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

En SQL :

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

Exemple

Étant donné des tables `pokemon_rouge` et `pokemon_or`, on peut obtenir une table avec tous les pokémons de 1ère et 2ème génération :

```
SELECT * FROM pokemon_rouge UNION pokemon_or
```

Différence

Si R_1 et R_2 sont des tables ayant le **même schéma relationnel**, $R_1 - R_2$ contient les enregistrements dans R_1 mais pas dans R_2:

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2		
A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 - R_2$		
A	B	C
a_2	b_2	c_2

En SQL, on utilise `MINUS` (MySQL, Oracle) ou `EXCEPT` (PostgreSQL) :

```
SELECT * FROM R1 EXCEPT SELECT * FROM R2;
```

Intersection

Si R_1 et R_2 sont des tables ayant le **même schéma relationnel**, $R_1 \cap R_2$ contient les enregistrements à la fois dans R_1 et R_2:

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2		
A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 \cap R_2$		
A	B	C
a_1	b_1	c_1

En SQL :

```
SELECT * FROM R1 INTERSECT SELECT * FROM R2;
```

Produit cartésien

On peut réaliser le **produit cartésien** $R_1 \times R_2$ de deux tables :

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2	
D	E
d_1	e_1
d_2	e_2

$R_1 \times R_2$				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

En SQL :

```
SELECT * FROM R1, R2;
```

Produit cartésien

On peut réaliser le **produit cartésien** $R_1 \times R_2$ de deux tables :

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2	
D	E
d_1	e_1
d_2	e_2

$R_1 \times R_2$				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

En SQL :

```
SELECT * FROM R1, R2;
```

On peut aussi sélectionner seulement certaines colonnes de $R_1 \times R_2$ en écrivant, par exemple, `SELECT A, B FROM R1, R2;`

Clé étrangère

Considérons une base de donnée bibliotheque avec les tables :

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Clé étrangère

Considérons une base de donnée bibliotheque avec les tables :

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Dans la table emprunt, `id_emprunteur` et `titre_livre` sont des **clés étrangères**, ce qui signifie qu'elles font références à une clé primaire d'une autre table.

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les emprunteurs qui sont aussi auteurs?

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les emprunteurs qui sont aussi auteurs?

```
SELECT nom FROM emprunteur, livre
WHERE nom = auteur;
```

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre dont le titre est Le Banquet?

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre dont le titre est Le Banquet?

```
SELECT nom FROM emprunteur, emprunt
WHERE id = id_emprunteur
AND titre_livre = 'Le Banquet';
```

JOIN

La jointure $R_1 \bowtie_{A=D} R_2$ de deux tables R_1 et R_2 revient à combiner les enregistrements de R_1 et R_2 en identifiant les colonnes A et D:

JOIN

La jointure $R_1 \bowtie_{A=D} R_2$ de deux tables R_1 et R_2 revient à combiner les enregistrements de R_1 et R_2 en identifiant les colonnes A et D:

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

R_2	
D	E
a_1	e_1
a_2	e_2

$R_1 \bowtie_{A=D} R_2$			
A	B	C	E
a_1	b_1	c_1	e_1
a_2	b_2	c_2	e_2

En SQL :

```
SELECT ... FROM R1 JOIN R2 ON A = D;
```

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre Le Banquet?

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre Le Banquet?

On peut aussi utiliser une jointure :

```
SELECT nom FROM emprunteur
JOIN emprunt ON id = id_emprunteur
WHERE titre_livre = 'Le Banquet';
```

Exemple

1. livre (<u>titre</u> : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (<u>id</u> : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les titres des livres empruntés par M. Machin?

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les titres des livres empruntés par M. Machin?

```
SELECT titre_livre FROM emprunteur
JOIN emprunt ON id = id_emprunteur
WHERE nom = 'Machin';
```

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les noms des personnes ayant emprunté un livre écrit par Stephen King?

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les noms des personnes ayant emprunté un livre écrit par Stephen King?

```
SELECT nom FROM emprunteur
JOIN emprunt ON id = id_emprunteur
JOIN livre ON titre_livre = titre
WHERE auteur = 'Stephen King';
```

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les plus gros livres empruntés avec leur nombre de pages?

Exemple

1. livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (id : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Comment obtenir les plus gros livres empruntés avec leur nombre de pages?

```
SELECT titre, pages FROM livre
JOIN emprunt ON titre_livre = titre
ORDER BY pages DESC;
```

Ambiguïté

1. livre (**id** : INT, titre : CHAR(50), auteur : CHAR(50), pages : INT)
2. emprunteur (**id** : INT, nom : CHAR(50))
3. emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Problème : comment savoir, dans livre $\times$ emprunteur, à quelle table **id** fait référence?

Ambiguïté

1. livre (**id** INT, titre CHAR(50), auteur CHAR(50), pages INT)
2. emprunteur (**id** INT, nom CHAR(50))
3. emprunt (id_emprunteur INT, titre_livre CHAR(50))

Tentative :

```
SELECT id FROM livre, emprunteur;
```

Résultat :
ERROR 1052 (23000): Column 'id' in field list is ambiguous

Ambiguïté

1. livre (**id** INT, titre CHAR(50), auteur CHAR(50), pages INT)
2. emprunteur (**id** INT, nom CHAR(50))
3. emprunt (id_emprunteur INT, titre_livre CHAR(50))

Solution :

```
SELECT livre.id FROM livre, emprunteur;
```

Ambiguïté

1. livre (**id** INT, titre CHAR(50), auteur CHAR(50), pages INT)
2. emprunteur (**id** INT, nom CHAR(50))
3. emprunt (id_emprunteur INT, titre_livre CHAR(50))

Problème 2 : afficher tous les couples de livres ayant le même nombre de pages.

Ambiguïté

1. livre (**id** INT, titre CHAR(50), auteur CHAR(50), pages INT)
2. emprunteur (**id** INT, nom CHAR(50))
3. emprunt (id_emprunteur INT, titre_livre CHAR(50))

Problème 2 : afficher tous les couples de livres ayant le même nombre de pages.

```
SELECT titre, titre FROM livre, livre WHERE pages = pages;
```

Ne marche pas du tout!

Ambiguïté

1. livre (**id** INT, titre CHAR(50), auteur CHAR(50), pages INT)
2. emprunteur (**id** INT, nom CHAR(50))
3. emprunt (id_emprunteur INT, titre_livre CHAR(50))

Solution : renommer les tables (temporairement).

Ambiguïté

1. livre (**id** INT, titre CHAR(50), auteur CHAR(50), pages INT)
2. emprunteur (**id** INT, nom CHAR(50))
3. emprunt (id_emprunteur INT, titre_livre CHAR(50))

Solution : renommer les tables (temporairement).

```
SELECT liv1.titre, liv2.titre
FROM livre AS liv1, livre AS liv2
WHERE liv1.pages = liv2.pages;
```

Extrait Mines-Pont 2017

On modélise ici un réseau routier par un ensemble de *croisements* et de *voies* reliant ces croisements. Les voies partent d'un croisement et arrivent à un autre croisement. Ainsi, pour modéliser une route à double sens, on utilise deux voies circulant en sens opposés.
La base de données du réseau routier est constituée des relations suivantes :

- Croisement(<u>id</u>, longitude, latitude)
- Voie(<u>id</u>, longueur, id_croisement_debut, id_croisement_fin)

Dans la suite on considère c l'identifiant (`id`) d'un croisement donné.

❏ **Q26** – Écrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c.

❏ **Q27** – Écrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c.

❏ **Q28** – Que renvoie la requête SQL suivante ?

```
1  SELECT V2.id_croisement_fin
2  FROM   Voie as V1
3  JOIN   Voie as V2
4  ON     V1.id_croisement_fin = V2.id_croisement_debut
5  WHERE  V1.id_croisement_debut = c
```