

SQL

Faire **vivre** la base de données : **créer**,
lire, **remplacer**, **supprimer** des
informations



Compétence demandée :
Connaître les utilisations avancées en
SQL

SQL : Structured Query Language

Pour vivre de l'information (base de données, REST API, SOAP), il faut des opérations basiques : les opérations
CRUD

Opérations **CRUD** pour les lignes

FRANCAIS	ANGLAIS	SQL
Créer	C REATE	I NSERT
Lire	R EAD	S ELECT
Remplacer	U PPDATE	U PPDATE
Supprimer	D ELETE	D ELETE

Opérations **CRUD** pour les TABLES

FRANCAIS	ANGLAIS	SQL
Créer	C REATE	CREATE TABLE
Lire	R EAD	-
Remplacer	U PPDATE	ALTER TABLE
Supprimer	D ELETE	DROP TABLE

Opérations **CRUD** pour les **BASES DE DONNÉES**

FRANCAIS	ANGLAIS	SQL
Créer	CREATE	CREATE DATABASE
Lire	READ	USE DATABASE
Remplacer	UPDATE	-
Supprimer	DELETE	DROP DATABASE

1. Recherche approximative,
union et intersection
2. Structure de contrôle
3. Préparation des requêtes

1. **Recherche approximative,
union et intersection**
2. Préparation des requêtes
3. Structure de contrôle

proprietaires

id	nom	prenom	age	ville	poids
1					
2					
3					
4					
5					
6					
7					
8					

restaurants

id	nom	adresse	note	ville	proprietaire_id
1					
2					
3					
4					
5					
6					
7					
8					

proprietaires

id	nom	prenom	age	ville	poids
1					
2					
3					
4					
5					
6					
7					
8					

restaurants

id	nom	adresse	note	ville	proprietaire_id
1					
2					
3					
4					
5					
6					
7					
8					

- a. Rechercher une ville en fonction de ses lettres
- b. Concaténer verticalement les villes des 2 tables
- c. Sélectionner les villes en commun

- a. Rechercher une ville en fonction de ses lettres (**LIKE**)
- b. Concaténer verticalement les villes des 2 tables (**UNION**, **UNION ALL**)
- c. Sélectionner les villes en commun (**IN**)

LIKE

```
SELECT ville FROM proprietaires  
WHERE ville LIKE "%ris"
```

Dans une requête avec LIKE, le caractère « % » a une signification et veut dire « **une ou plusieurs lettres** »


```
SELECT ville FROM propriétaires  
WHERE ville LIKE "%ris"
```

ris

```
SELECT ville FROM propriétaires  
WHERE ville LIKE "Par%"
```

Par

```
SELECT ville FROM propriétaires  
WHERE ville LIKE "%ou%"
```

ou

```
SELECT ville FROM propriétaires  
WHERE ville NOT LIKE "%ou%"
```

ou

ou



```
SELECT ville FROM proprietaires  
WHERE ville LIKE "%ris"
```

Qui se termine ...

```
SELECT ville FROM proprietaires  
WHERE ville LIKE "Par%"
```

Qui commence ...

```
SELECT ville FROM proprietaires  
WHERE ville LIKE "%ou%"
```

Qui contient ...

```
SELECT ville FROM proprietaires  
WHERE ville NOT LIKE "%ou%"
```

Qui NE contient PAS ...

- a. Rechercher une ville en fonction de ses lettres (**LIKE**)
- b. Concaténer verticalement les villes des 2 tables (**UNION**, **UNION ALL**)
- c. Sélectionner les villes en commun (**IN**)

A quoi sert UNION et UNION ALL ?
Quelle est leur différence ?



UNION & UNION ALL

proprietaires

id	nom	prenom	age	ville	poids
1					
2					
3				Paris	
4					
5					
6					
7					
8					

restaurants

id	nom	adresse	note	ville	proprietaire_id
1					
2					
3				Paris	
4					
5					
6					
7					
8					

Concaténer verticalement les villes des 2 tables

Concaténer verticalement les villes des 2 tables

[illegible]


```
SELECT ville FROM proprietaires  
UNION  
SELECT ville FROM restaurants
```

```
SELECT ville FROM proprietaires  
UNION  
SELECT ville FROM restaurants
```

UNION enlève automatiquement les doublons !
Si vous souhaitez garder les doublons, vous
pouvez utiliser UNION ALL

```
SELECT ville FROM proprietaires  
UNION  
SELECT ville FROM restaurants
```



UNION enlève automatiquement les doublons !
Si vous souhaitez garder les doublons, vous
pouvez utiliser UNION ALL

- a. Rechercher une ville en fonction de ses lettres (**LIKE**)
- b. Concaténer verticalement les villes des 2 tables (**UNION**, **UNION ALL**)
- c. Sélectionner les villes en commun (**IN**)

X  NATIS

IN

proprietaires

id	nom	prenom	age	ville	poids
1					
2					
3					
4					
5					
6					
7					
8					

restaurants

id	nom	adresse	note	ville	proprietaire_id
1					
2					
3					
4					
5					
6					
7					
8					

Sélectionner les villes en commun

```
SELECT ville FROM proprietaires  
WHERE ville IN (SELECT ville FROM restaurants)
```



```
SELECT ville FROM propriétaires  
WHERE ville IN (SELECT ville FROM restaurants)
```

C'est une requête imbriquée !

Trouver une intersection est le seul cas autorisé
pour l'utilisation de IN !

Tout autre cas est considéré comme mauvaise
pratique

```
SELECT ville FROM proprietaires  
WHERE ville IN (SELECT ville FROM restaurants)
```

C'est une requête imbriquée !

Trouver une intersection est le seul cas autorisé
pour l'utilisation de IN !

Tout autre cas est considéré comme mauvaise
pratique



- a. Rechercher une ville en fonction de ses lettres (**LIKE**)
- b. Concaténer verticalement les villes des 2 tables (**UNION**, **UNION ALL**)
- c. Sélectionner les villes en commun (**IN**)

1. Recherche approximative,
union et intersection
2. Préparation des requêtes
3. Structure de contrôle

1. Recherche approximative,
union et intersection
2. **Préparation des requêtes**
3. Structure de contrôle

- a. Les vues (`CREATE VIEW`)
- b. Les requête préparée (`PREPARE` et `EXECUTE`)
- c. Les procédures stockées (`CREATE PROCEDURE` et `CALL`)

CREATE VIEW

proprietaires

id	nom	prenom	age	ville	poids
1					
2					
3					
4					
5					
6					
7					
8					

restaurants

id	nom	adresse	note	ville	proprietaire_id
1					
2					
3					
4					
5					
6					
7					
8					

On crée une vue, c'est-à-dire une table virtuelle

Créer une table virtuelle ou une vue, c'est créer une dont les données sont recalculées à chaque requête

Créer une table virtuelle ou une vue, c'est créer une dont les données sont recalculées à chaque requête



```
CREATE VIEW v_dashboard AS  
SELECT proprietaires.nom,  
proprietaires.prenom, proprietaires.age,  
proprietaires.ville, restaurants.adresse  
FROM restaurants LEFT JOIN proprietaires  
WHERE restaurants.ville = "Paris"
```

```
SELECT * FROM v_dashboard;
```

Cette vue est utilisée comme une table par la suite, même si elle n'existe pas en soi !

A quoi sert PREPARE et EXECUTE ?



- a. Les vues (`CREATE VIEW`)
- b. Les requête préparée (`PREPARE` et `EXECUTE`)
- c. Les procédures stockées (`CREATE PROCEDURE`)

PREPARE et EXECUTE

Préparer une requête est **fixer son code** pour le réutiliser plus tard en **séparant le code des valeurs**.

Aussi, une requête préparée n'est pas sensible aux injections SQL et est donc plus sécurisée.

PREPARE hello FROM

"SELECT * FROM restaurants WHERE ville = ?"

```
PREPARE hello FROM  
"SELECT * FROM restaurants WHERE ville = ?"
```

```
SET @a = 'Paris';  
EXECUTE hello USING @a;
```

```
PREPARE hello FROM  
"SELECT * FROM restaurants WHERE ville = ?"
```

```
SET @a = 'Paris';  
EXECUTE hello USING @a;
```

Les valeurs doivent être mises dans des variables afin d'être utilisées dans les requêtes préparées

A quoi sert CREATE PROCEDURE et CALL ?



- a. Les vues (`CREATE VIEW`)
- b. Les requête préparée (`PREPARE` et `EXECUTE`)
- c. Les procédures stockées (`CREATE PROCEDURE`)

CREATE PROCEDURE et CALL

```
DELIMITER $$;  
CREATE PROCEDURE proc_hello (IN a CHAR(255))  
BEGIN  
    SELECT * FROM exo1_users WHERE email = a;  
END$$;  
DELIMITER;
```

```
DELIMITER $$;  
CREATE PROCEDURE proc_hello (IN a CHAR(255))  
BEGIN  
→ SELECT * FROM exo1_users WHERE email = a;  
END$$;  
DELIMITER;
```

restaurants *ville*

```
CALL proc_hello('Paris');
```

Une procédure est similaire à une fonction en PHP ou en Javascript.

Elle peut consigner plusieurs instructions (comme des INSERT, des UPDATE et des DELETE également) pour qu'elles ne forment qu'un seul bloc d'instructions.

1. Recherche approximative, union et intersection
2. Préparation des requêtes
3. **Structure de contrôle**

STRUCTURE DE CONTROLE



SELECT nom, IF(ville = 'Paris', 'local', 'en dehors')
FROM restaurants

The image shows a SQL query with two hand-drawn red arrows pointing down to the 'ville' column and the 'IF' function, and a red bracket spanning the entire query text.


```
SELECT nom, IF(ville = 'Paris', 'local', 'en dehors')  
FROM restaurants
```

```
SELECT nom, IF(ville = 'Paris', 'local', IF(ville =  
'Strasbourg', 'bureau Alsace', 'en dehors'))  
FROM restaurants
```

```
SELECT nom, IF(ville = 'Paris', 'local', IF(ville =  
'Strasbourg', 'bureau Alsace', 'en dehors'))  
FROM restaurants
```

```
SELECT nom, CASE ville  
WHEN 'Paris' THEN 'local'  
WHEN 'Strasbourg' THEN 'bureau Alsace'  
ELSE 'en dehors'  
END  
FROM restaurants
```

Comment faire une boucle ?
Comment casser une boucle en SQL ?

