Base de Données

Session 2
Les Fonctions
Les Agrégations
Les Jointures

Un cours sur le langage SQL n’est vraiment utile que si on essaye de le mettre en pratique dans un contexte d’usage réel.

Les fonctions “Chaînes de caractères”

- CONCAT()

```
SELECT CONCAT(prenom, ' ', nom) AS affichage_nom, date_inscription
FROM utilisateur;
```

- LENGTH()

```
SELECT LENGH('example') FROM utilisateur LIMIT 1;  -- ⇒ 7
```

- REPLACE()

```
SELECT REPLACE('Hello tout le monde', 'Hello', 'Bonjour')
FROM utilisateur LIMIT 1;
```

Les fonctions “Chaînes de caractères”

- SUBSTRING()

```
SELECT SUBSTRING('poney', 3) FROM table;      -- ⇒ 'ney'
SELECT SUBSTRING('poney', 1, 2) FROM table;   -- ⇒ 'po'
```

- LEFT()

```
SELECT LEFT('abcdefghij', 2) FROM utilisateur LIMIT 1; -- ⇒ 'ab'
```

- RIGHT()

```
SELECT RIGHT('abcdefghij', 3) FROM utilisateur LIMIT 1; -- ⇒ 'hij'
```

Les fonctions “Chaînes de caractères”

- UPPER

```
SELECT UPPER('Exemple') FROM table LIMIT 1;    -- ⇒ 'EXEMPLE'
```

- LOWER

```
SELECT LOWER('BONJOUR') FROM table LIMIT 1; -- ⇒ 'bonjour'
```

- ...

Les fonctions “Mathématique”

- ROUND()

```
SELECT ROUND(nom_colonne) FROM table;
SELECT ROUND(nom_colonne, 2) FROM table;
```

- RAND()

Les fonctions “Date”

- YEAR()

```
SELECT YEAR('2014-02-01') FROM table ;          -- ⇒ 2014
```

- MONTH()

```
SELECT MONTH('2014-02-01') FROM table ;         -- ⇒ 2
```

- DAY()

```
SELECT DAY('2014-02-01') FROM table ;           -- ⇒ 1
```

- WEEK()

```
SELECT WEEK('2014-02-01') FROM table ;          -- ⇒ 4
```

- QUARTER()

```
SELECT QUARTER('2014-02-01') FROM table ;       -- ⇒ 1
```

Les fonctions “Date”

- NOW()

```
SELECT YEAR(NOW()) FROM table ;                -- ⇒ 2023
```

- HOUR()

```
SELECT HOUR(NOW()) FROM table ;                -- ⇒ 13
```

- DAYOFWEEK() // 1 = Dimanche

```
SELECT DAYOFWEEK('2018-09-15') FROM table ;    -- ⇒ 7 (samedi)
```

- DAYOFYEAR()

```
SELECT DAYOFYEAR('2018-09-15') FROM table ;    -- ⇒ 258
```

Les fonctions “Date”

- NOW()

```
SELECT YEAR(NOW()) FROM table ;                -- ⇒ 2023
```

- HOUR()

```
SELECT HOUR(NOW()) FROM table ;                -- ⇒ 13
```

- DAYOFWEEK() // 1 = Dimanche

```
SELECT DAYOFWEEK('2018-09-15') FROM table ;    -- ⇒ 7 (samedi)
```

- DAYOFYEAR()

```
SELECT DAYOFYEAR('2018-09-15') FROM table ;    -- ⇒ 258
```

Les fonctions d'Agrégation

- COUNT()

```
SELECT COUNT(*) FROM table;
SELECT COUNT(nom_colonne) FROM table;
SELECT COUNT(DISTINCT(nom_colonne)) FROM table;
```

- MIN()

```
SELECT MIN(nom_colonne) FROM table;
```

- MAX()

```
SELECT MAX(nom_colonne) FROM table;
```

Les fonctions d'Agrégation

- SUM()

```sql
SELECT SUM(nom_colonne) FROM table;
```

- AVG()

```sql
SELECT AVG(nom_colonne) FROM table;
```

Les Agrégations

- GROUP BY

```
SELECT colonne_1, fonction(colonne_2)
FROM table
GROUP BY colonne_1;
```

- HAVING()

```
SELECT colonne_1, fonction(colonne_2)
FROM table
GROUP BY colonne_1
HAVING fonction(colonne_2) operateur valeur;
```

Les Agrégations

id	client	tarif	date_achat
1	Pierre	102	2012-10-23
2	Simon	47	2012-10-27
3	Marie	18	2012-11-05
4	Marie	20	2012-11-14
5	Pierre	160	2012-12-03

```
SELECT client, SUM(tarif)
FROM table
GROUP BY client
HAVING SUM(tarif) > 40;
```

client	SUM(tarif)
Pierre	262
Simon	47

Les Jointures

Les jointures en SQL permettent d'associer plusieurs tables dans une même requête.

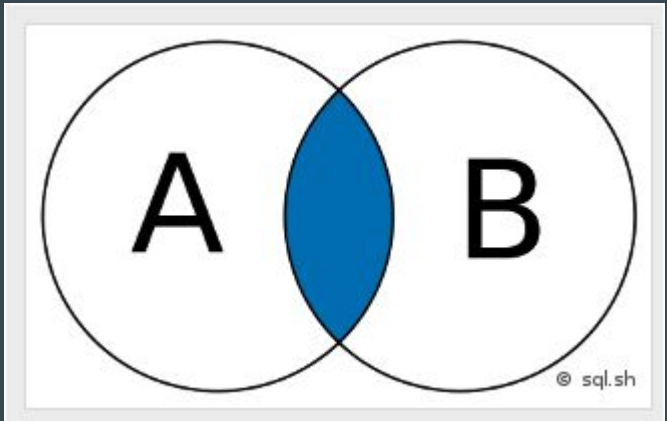
En général, les jointures consistent à associer des lignes de 2 tables en associant l'égalité des valeurs d'une colonne d'une première table par rapport à la valeur d'une colonne d'une seconde table.

Imaginons qu'une base de 2 données possède une table "utilisateur" et une autre table "adresse" qui contient les adresses de ces utilisateurs.

Avec une jointure, il est possible d'obtenir les données de l'utilisateur et de son adresse en une seule requête.

Les Types de Jointures :: INNER JOIN

- Jointure interne pour retourner les enregistrements
  Quand la condition est vraie dans les 2 tables.
  C'est l'une des jointures les plus communes.




```sql
SELECT *
FROM table_A AS A
INNER JOIN table_B AS B
  ON A.pk = B.fk;
```

Les Types de Jointures :: INNER JOIN

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

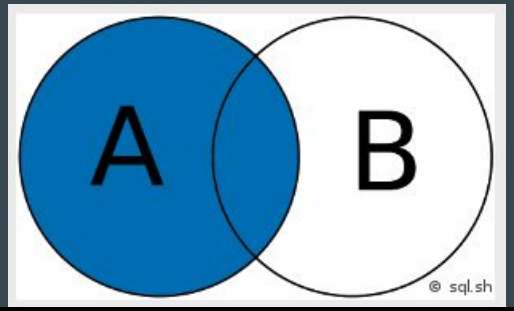
utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, date_achat,
 num_facture, prix_total
FROM utilisateur
INNER JOIN commande
   ON utilisateur.id = commande.utilisateur_id;
```

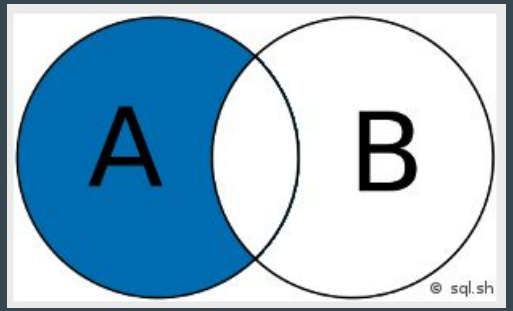
id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35

Les Types de Jointures :: LEFT JOIN (ou LEFT OUTER JOIN)

- Jointure externe pour retourner tous les enregistrements de la table de gauche. (LEFT = gauche)
  Même si la condition n'est pas vérifiée dans l'autre table.




```sql
SELECT *
FROM table_A AS A
LEFT JOIN table_B AS B
  ON A.pk = B.fk;
```




```sql
SELECT *
FROM table_A AS A
LEFT JOIN table_B AS B
  ON A.pk = B.fk
WHERE B.fk IS NULL;
```

Les Types de Jointures :: LEFT JOIN (ou LEFT OUTER JOIN)

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

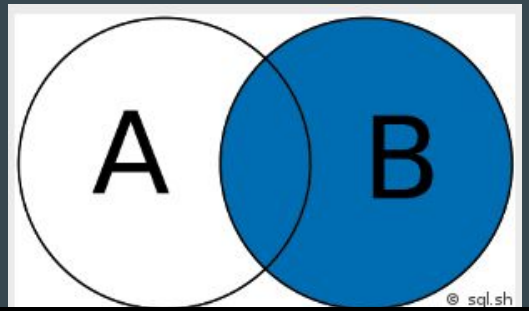
utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, date_achat,
 num_facture, prix_total
FROM utilisateur
LEFT JOIN commande
   ON utilisateur.id = commande.utilisateur_id;
```

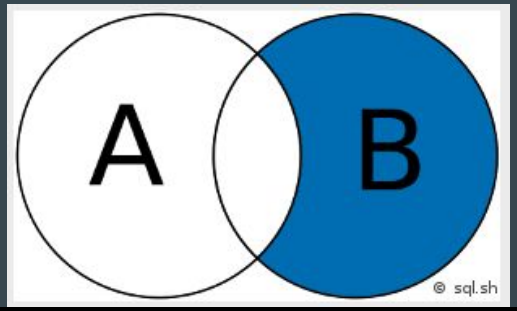
id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35
3	Marine	Prevost	NULL	NULL	NULL
4	Luc	Rolland	NULL	NULL	NULL

Les Types de Jointures :: RIGHT JOIN (ou RIGHT OUTER JOIN)

- Jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite)
  Même si la condition n'est pas vérifiée dans l'autre table.




```sql
SELECT *
FROM table_A AS A
RIGHT JOIN table_B AS B
  ON A.pk = B.fk;
```




```sql
SELECT *
FROM table_A AS A
RIGHT JOIN table_B AS B
  ON A.pk = B.fk
WHERE A.pk IS NULL;
```

Les Types de Jointures :: RIGHT JOIN (ou RIGHT OUTER JOIN)

Table utilisateur :

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1

Table commande :

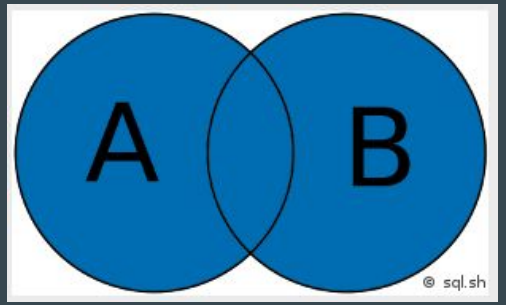
utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, date_achat,
 num_facture, prix_total
FROM utilisateur
LEFT JOIN commande
   ON utilisateur.id = commande.utilisateur_id;
```

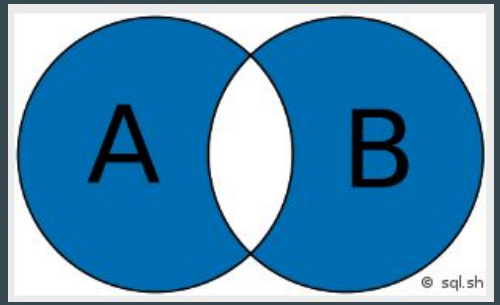
id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
NULL	NULL	NULL	5	2013-03-02	A00107

Les Types de Jointures :: FULL JOIN (ou FULL OUTER JOIN)

- Jointure externe pour retourner les résultats
  Quand la condition est vraie dans au moins une des 2 tables.




```sql
SELECT *
FROM table_A AS A
FULL JOIN table_B AS B
  ON A.pk = B.fp;
```




```sql
SELECT *
FROM table_A AS A
FULL JOIN table_B AS B
  ON A.pk = B.fk
WHERE A.pk IS NULL OR B.fk IS NULL;
```

Les Types de Jointures :: FULL JOIN (ou FULL OUTER JOIN)

Table utilisateur :

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT id, prenom, nom, date_achat,
 num_facture, prix_total
FROM utilisateur
FULL JOIN commande
   ON utilisateur.id = commande.utilisateur_id;
```

id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
4	Luc	Rolland	NULL	NULL	NULL
NULL	NULL	NULL	5	2013-03-02	A00107

Les Types de Jointures

- CROSS JOIN : jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque ligne d'une table avec chaque ligne d'une seconde table.
  Attention, le nombre de résultats est en général très élevé.

- SELF JOIN : permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table.

- NATURAL JOIN : jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL

- UNION JOIN : jointure d'union

DM

1. Obtenir le nom des 10 villes les plus peuplées en 2012, ainsi que le nom du département associé.

2. Obtenir la liste du nom de chaque département, associé à son code et du nombre de communes au sein de ces départements, en triant afin d'obtenir en priorité les départements qui possèdent le plus de communes.

3. Obtenir la liste des 10 plus grands départements, en termes de superficie.

4. Obtenir la liste des villes qui ont un nom existant plusieurs fois, et trier afin d'obtenir en premier celles dont le nom est le plus souvent utilisé par plusieurs communes.

5. Obtenir en une seule requête SQL la liste des villes dont la superficie est supérieure à la superficie moyenne.

6. Obtenir la liste des départements qui possèdent plus de 2 millions d'habitants