June 4, 2015

SQL

Fondamentaux

SQL

- Relationnel
- SELECT
- INSERT
- UPDATE
- DELETE
- CREATE / DROP
- GRANT / REVOKE
- COMMIT / ROLLBACK

Règles de Codd

Information
Information représentée d'une seule manière, une valeur dans un champs de colonnes de tables.

Accès Garanti
Données accessibles sans ambiguïté.

Valeurs Nulles
Chaque champs doit pouvoir être Null (ou vide).

Catalogue Relationnel
Le système doit intégrer un catalogue en ligne, relationnel, accessible au moyen de leur langage d'interrogation régulier.

Sous-Langage
Langage avec une syntaxe linéaire, employable interactivement, complet.

Mise à jour des Vues
Toute vue pouvant théoriquement être mise à jour doit pouvoir l'être par le système.

Ins., MaJ, Supp. Haut Niveau
Ajout, Modification et Effacement possible par lot, sur plusieurs tables.

Indépendance Physique
Les changements physiques sur les données ne changent pas les applications.

Indépendance Logique
Les changements logiques des données (tables, tuples, attributs…) ne changent pas les applications.

Indépendance Intégrité
Contraintes d'intégrité indiquées séparément des programmes d'application. Modifiables séparément.

Indépendance Distribution
Distribution physique des données ne doit pas être visible ou contraignantes pour l'utilisateur.

Non-Subversion
Pas de contournement possible par une interface bas-niveau.

Vocabulaire

- Base de données / Banque de données
- Relation / Table
- Tuple
- Attribut
- Projection
- Vue
- Index
- Clé
- CRUD

SELECT

- SELECT <attribut> [AS <alias>] {, <attribut> [AS <alias>]}
  FROM <table> [<alias>] {, <table> [<alias>]}

```
SELECT *
FROM Personne
```

```
SELECT Nom AS Qui
FROM Personne
```

```
SELECT P.Nom
FROM Personne P
```

```
SELECT P.Nom AS Qui
FROM Personne P
```

SELECT – DISTINCT

- SELECT DISTINCT <attribut>

```
SELECT Nom
FROM Personne
```

```
SELECT DISTINCT Nom
FROM Personne
```

SELECT – WHERE

- WHERE <attribut> <operateur> <valeur>
- AND <attribut> <operateur> <valeur>
- OR <attribut> <operateur> <valeur>

```
SELECT * FROM Personne
WHERE Nom = 'Jensen'
```

```
SELECT * FROM Personne
WHERE Nom = 'Jensen'
AND Prenom = 'Mira'
```

```
SELECT * FROM Personne
WHERE Nom = 'Jensen'
OR Nom = 'Brown'
```

SELECT – WHERE – LIKE

- WHERE <attribut> LIKE <valeur>

```
SELECT * FROM Personne
WHERE Nom LIKE 'S%'
```

```
SELECT * FROM Personne
WHERE Nom LIKE '%s'
```

```
SELECT * FROM Personne
WHERE Nom LIKE '_a%'
```

```
SELECT * FROM Personne
WHERE Nom LIKE '__a%'
```

SELECT – WHERE – BETWEEN

- WHERE <attribut> BETWEEN <valeur1> AND <valeur2>

```
SELECT * FROM Personne
WHERE Naissance BETWEEN
'2000-01-01' AND '2015-
01-01'
```

```
SELECT * FROM Personne
WHERE Nom BETWEEN 'A'
AND 'D'
```

SELECT – WHERE – IS NULL

- WHERE <attribut> IS [NOT] NULL

```
SELECT * FROM Personne
WHERE Naissance IS NULL
```

```
SELECT * FROM Personne
WHERE Naissance IS NOT
NULL
```

SELECT – IN

- WHERE <attribut> IN (<valeur> {, <valeur>})

Toutes les personnes qui se prénomment Helen, Lee, Urielle, Joe, Wing ou Orli

```
SELECT *
FROM Personne
WHERE Prenom IN ( 'Helen', 'Lee', 'Urielle',
                  'Joe', 'Wing', 'Orli'
                )
```

SELECT – Sous-Requête

- WHERE <attribut> IN/ALL (SELECT …)
- WHERE <attribut> ANY/SOME (SELECT …)
- WHERE [NOT] EXISTS (SELECT …)

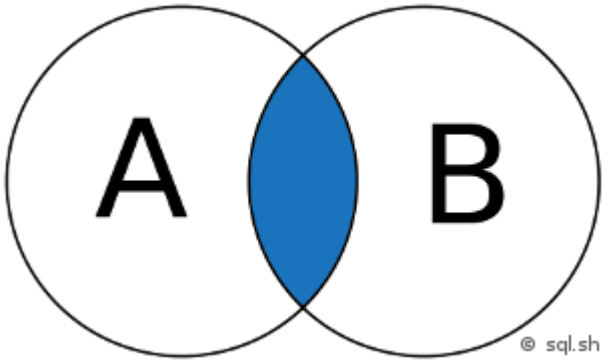
Toutes les personnes traitées en avril 2015

```
SELECT *
FROM Personne
WHERE Id IN ( SELECT Sujet
              FROM Etude
              WHERE ingestion BETWEEN '2015-04-01'
                                  AND '2015-04-30'
            )
```

SELECT – Jointure

- Jonction de 2 ensemble de données.
- Inner (EquiJoin) : Jointure des données présentes dans les 2 tables
- FROM <table> INNER JOIN <table> ON <attribut> = <attribut>

```
SELECT * FROM Personne
INNER JOIN Etude ON id = sujet
WHERE ingestion BETWEEN '2015-04-01'
                    AND '2015-04-30'
```

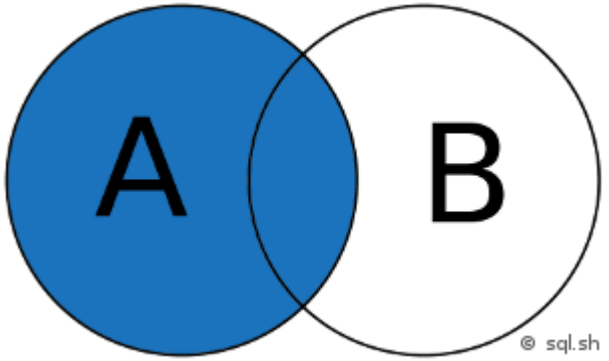


SELECT – Jointure Externe

- Left / Right / Full : Jointure des données d'une table avec les données éventuellement présentes dans une autre
- FROM <table> LEFT JOIN <table> ON <attribut> = <attribut>

Traitements des personnes nées de 1985 à 1988

```
SELECT * FROM Personne
LEFT JOIN Etude ON id = sujet
WHERE naissance BETWEEN '1985-01-01'
                    AND '1989-01-01'
```



SELECT – Jointure Croisée

- Cross : Jointure des données d'une table avec les données d'une autre
- FROM <table> CROSS JOIN <table>
- FROM <table>, <table>

```
SELECT *
FROM Personne
CROSS JOIN Etude
```

```
SELECT *
FROM Personne, Etude
```

SELECT – Jointure Naturelle

- Natural : Jointure de deux tables sur deux champs de même nom
- FROM <table> NATURAL JOIN <table>

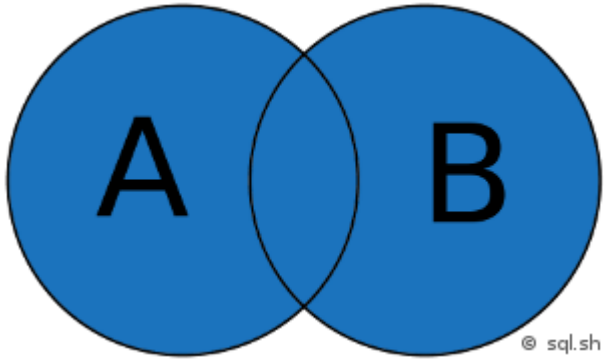
```
SELECT *
FROM Etude
NATURAL JOIN Resultats
```

UNION

- Regroupement de deux projections
- SELECT … UNION [ALL] SELECT …

Toutes les personnes nées en 2003 ou traitées en avril 2015

```
SELECT * FROM Personne
WHERE Naissance
BETWEEN '2003-01-01'
AND '2004-01-01'
UNION
SELECT * FROM Personne
WHERE Id IN ( SELECT Sujet FROM Etude
WHERE Ingestion BETWEEN '2015-04-01'
                    AND '2015-04-30' )
```

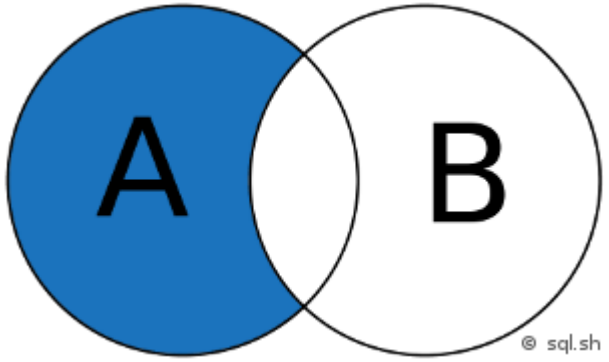


MINUS / EXCEPT

- Exclusion des résultats d'une projection dans une autre

Toutes les personnes nées en 2003 non traitées en avril 2015

```
SELECT * FROM Personne
WHERE Naissance
BETWEEN '2003-01-01'
AND '2004-01-01'
MINUS
SELECT * FROM Personne
WHERE Id IN ( SELECT Sujet FROM Etude
WHERE Ingestion BETWEEN '2015-04-01'
                    AND '2015-04-30' )
```

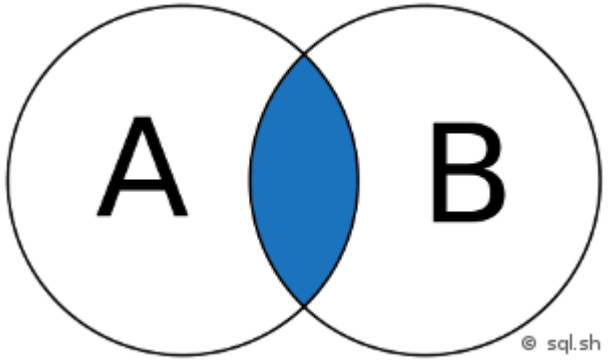


INTERSECT

- Exclusion des résultats d'une projection dans une autre

Toutes les personnes nées en 2003 et traitées en avril 2015

```
SELECT * FROM Personne
WHERE Naissance
BETWEEN '2003-01-01'
AND '2004-01-01'
INTERSECT
SELECT * FROM Personne
WHERE Id IN ( SELECT Sujet FROM Etude
WHERE Ingestion BETWEEN '2015-04-01'
                    AND '2015-04-30' )
```



SELECT – ORDER BY

- SELECT … ORDER BY <attribut> [DESC | ASC]
  {, <attribut> [DESC | ASC] }

```
SELECT Nom, Prenom
FROM Personne
ORDER BY Nom ASC
```

```
SELECT *
FROM Personne
ORDER BY Nom, Prenom
```

SELECT – LIMIT

- SELECT … LIMIT [<Debut> ,] <Nombre>
- SELECT … LIMIT <Nombre> [OFFSET <Debut>]

```
SELECT *
FROM Etude
LIMIT 105
```

```
SELECT *
FROM Etude
LIMIT 27 , 105
```

SELECT – GROUP BY

- SELECT … GROUP BY <attribut> {, <attribut> }

```
SELECT Nom, Prenom
FROM Personne
GROUP BY Nom
```

```
SELECT *
FROM Personne
GROUP BY Nom, Prenom
```

Jointure sur Etude?

```
SELECT *
FROM Personne JOIN Etude ON Id = Sujet
GROUP BY Nom, Prenom
```

SELECT – Fonctions

- SELECT <Fonction>({[,] <paramètre>})

```
SELECT SUM(Resultat)
FROM Etude
```

```
SELECT AVG(Parent)
FROM Personne
```

Moyenne des résultats par Personne

```
SELECT Nom, Prenom, AVG(Resultat)
FROM Personne JOIN Etude ON Id = Sujet
GROUP BY Nom, Prenom
```

SELECT – Fonctions

AVG	COUNT	ASCII	UPPER	CONCAT
SUM	ISNULL	BIN	LOWER	REPLACE
MIN	IFNULL	CHAR	LEFT	LOCATE
MAX	REVERSE	TRIM	RIGHT	SUBSTRING

Premier mot des explications de résultats

```
SELECT Resultat,
SUBSTRING(Explication, 1,
LOCATE(' ', Explication))
FROM Resultats
```

Statut des personnes "Prénom Nom Statut"

```
SELECT CONCAT(Prenom, ' ', Nom, ' ',
SUBSTRING_INDEX(Explication, ' ',
1)) AS Statut FROM Personne
JOIN Etude E ON Id = Sujet
NATURAL JOIN Resultats R
WHERE E.Resultat = (SELECT
MIN(E1.Resultat) FROM Etude E1
WHERE E1.Sujet = Id) GROUP BY
Prenom, Nom
```

Moyenne des résultats des gens morts

```
SELECT Nom, Prenom,
AVG(Resultat) AS Moyenne
FROM Personne
JOIN Etude ON Id = Sujet
WHERE ID IN ( SELECT E1.Sujet
FROM Etude E1 WHERE E1.Resultat
IN ( 1 , 2 ) )
GROUP BY Prenom, Nom
```

SELECT – Exercices

Personnes nées après 1950

```
SELECT *
FROM Personne
WHERE Naissance > '1950-12-31'
```

Personnes dont le nom commence par un D

```
SELECT *
FROM Personne P1
WHERE 'D' IN ( SELECT
SUBSTRING(P2.Nom, 1, 1)
FROM Personne P2
WHERE P2.Id = P1.Id)
```

Personnes dont le nom commence par un D et se fini par un y

```
SELECT *
FROM Personne P1
WHERE 'D' IN ( SELECT
SUBSTRING(P2.Nom, 1, 1)
FROM Personne P2
WHERE P2.Id = P1.Id)
AND 'y' IN ( SELECT
SUBSTRING(P2.Nom, -1)
FROM Personne P2
WHERE P2.Id = P1.Id) )
```

Moyenne des résultats par Personne qui sont supérieures à 5

```
SELECT Nom, Prenom, AVG(Resultat)
FROM Personne P1 JOIN Etude E1 ON
Id = Sujet
WHERE 5 < (
SELECT AVG(E2.Resultat)
FROM  Personne P2  JOIN Etude E2
ON P2.Id = E2.Sujet
WHERE P2.ID = P1.Id )
GROUP BY Nom, Prenom
```

SELECT – HAVING

- SELECT … HAVING <fonction>({<paramètre>}) <operateur> <valeur>

Personnes nées après 1950

```
SELECT *
FROM Personne
HAVING YEAR(Naissance) > 1950
```

Personnes dont le nom commence par un D

```
SELECT *
FROM Personne
HAVING SUBSTRING(Nom, 1, 1) = 'D'
```

Personnes dont le nom commence par un D et se fini par un y

```
SELECT *
FROM Personne
HAVING SUBSTRING(Nom, 1, 1) = 'D'
AND SUBSTRING(Nom, -1) = 'y'
```

SELECT – HAVING / GROUP

- SELECT … GROUP BY … HAVING <fonction>({<paramètre>}) <operateur> <valeur>

Moyenne des résultats par Personne qui sont supérieures à 5

```
SELECT Nom, Prenom, AVG(Resultat)
FROM Personne JOIN Etude ON Id = Sujet
GROUP BY Nom, Prenom
HAVING AVG(Resultat) >= 5
```

SELECT – CASE

- CASE [<attribut>]
  WHEN <condition> THEN <valeur>
  { WHEN <condition> THEN <valeur> }
  [WHEN <condition> THEN <valeur>]
  END

Afficher la tranche d'âge (regroupées sur 25 ans) d'une personne

```
SELECT Nom, Prenom,
CASE
WHEN YEAR(Naissance) < 1940 THEN '75-100 ans'
WHEN YEAR(Naissance) < 1965 THEN '50-75 ans'
WHEN YEAR(Naissance) < 1990 THEN '25-50 ans'
WHEN YEAR(Naissance) < 2016 THEN '0-25 ans'
ELSE 'age inconnu'
END AS 'Tranche d\'âge'
FROM Personne
```

CREATE - Vue

- CREATE [OR REPLACE] VIEW <vue> [(<attribut> {, <attribut>})]
  AS SELECT ...

```
CREATE VIEW VuePersonneMortes AS
SELECT *
FROM Personne
WHERE Id IN (SELECT Sujet
             FROM Etude
             WHERE Resultat IN (1, 2)
            )
```

DROP – Vue

- DROP VIEW [IF EXISTS] <vue> {, <vue>}

```
DROP VIEW VuePersonneMortes
```

CREATE – Base de Données

- Pas dans la norme SQL
- CREATE DATABASE [IF NOT EXISTS] <base>

```
CREATE DATABASE DBEssai
```

CREATE - Table

- CREATE TABLE [IF NOT EXISTS] <table> (<colonne> <type> <contraintes> {, <colonne> <type> <contraintes> } {, <contraintes> }

```
CREATE TABLE EssaiBasique (
    ID INT NOT NULL PRIMARY KEY,
    Nom VARCHAR(64) NOT NULL,
    Prenom VARCHAR(64) NOT NULL,
    Naissance DATE NULL DEFAULT ('1970-01-01'),
    NumSecu VARCHAR(15) UNIQUE
);
```

CREATE - Table

- CREATE TABLE [IF NOT EXISTS] <table> LIKE <table>

```
CREATE TABLE EssaiPersonne
LIKE CoursSQL.Personne
```

CREATE - Table

- CREATE TABLE [IF NOT EXISTS] <table> SELECT …

```
CREATE TABLE EssaiPersonne2
SELECT *
FROM CoursSQL.Personne
```

CREATE - Contraintes

- INDEX [<nom>] (<attribut> {, <attribut>})
- [CONSTRAINT <nom>] UNIQUE [INDEX] [<nom>] (<attribut> {, <attribut>})
- [CONSTRAINT <nom>] PRIMARY KEY (<attribut> {, <attribut>})
- [CONSTRAINT <nom>] FOREIGN KEY (<attribut> {, <attribut>}) REFERENCES <table> (<attribut> {, <attribut>})

ALTER TABLE – ADD

- ALTER TABLE <table>
- ADD
- [COLUMN] <attribut> <type> [AFTER <attribut> | FIRST]
- [COLUMN] (<attribut> <type> {, <attribut> <type> })
- INDEX [<nom>] (<attribut> {, <attribut>})
- [CONSTRAINT <nom>] UNIQUE [INDEX] [<nom>] (<attribut> {, <attribut>})
- [CONSTRAINT <nom>] PRIMARY KEY (<attribut> {, <attribut>})
- [CONSTRAINT <nom>] FOREIGN KEY (<attribut> {, <attribut>}) REFERENCES <table> (<attribut> {, <attribut>})

ALTER TABLE – DROP

- DROP [COLUMN] <colonne>
- DROP PRIMARY KEY
- DROP INDEX <nom>
- DROP FOREIGN KEY <nom>

ALTER TABLE – Modification

- MODIFY [COLUMN] <attribut> <type> [AFTER <attribut> | FIRST]
- CHANGE [COLUMN] <ancien attribut> <nouvel attribut> <type> [AFTER <attribut> | FIRST]
- ALTER [COLUMN] <attribut> SET DEFAULT <valeur>
- ALTER [COLUMN] <attribut> DROP DEFAULT

GRANT

- Ajouter des privilèges à un utilisateur
- GRANT <privilège> TO <utilisateur> {, <utilisateur>} [WITH GRANT OPTION]

ALL PRIVILEGES **SELECT** **INSERT**

REFERENCES **UPDATE** **DELETE**

```
GRANT INSERT (Nom, Prenom)
ON Essai
TO mysql
```

```
GRANT SELECT
ON Essai
TO PUBLIC
```

REVOKE

- Suppression des droits utilisateurs
- REVOKE [GRANT OPTION FOR] <privilèges> FROM <utilisateur> {, <utilisateur>} [RESTRICT | CASCADE]

```
REVOKE INSERT (Nom,
Prenom)
ON Essai
FROM mysql
```

```
REVOKE SELECT
ON Essai
FROM PUBLIC
```

INSERT

- INSERT INTO <table> [(<attribut> {, <attribut>})]
  VALUES (<attribut> {, <attribut>})
  {, (<attribut> {, <attribut>}) } ;

```
INSERT INTO Personne
VALUES (NULL, 'Joe', 'Black', NULL, 0)
```

UPDATE

- UPDATE <table>
  SET <attribut> = <valeur> {, <attribut> = <valeur>}
  [WHERE …]

```
UPDATE Personne
SET Naissance = '0-0-0 00:00:00'
WHERE Prenom = 'Joe'
AND Nom = 'Black'
```

INSERT – ON DUPLICATE

- INSERT … ON DUPLICATE KEY UPDATE <attribut> = <valeur> {, <attribut> = <valeur> }

Ajouter un Joe Black en position 101, né le 01/01/0000 à minuit

```
INSERT INTO Personne
VALUES (101, 'Joe', 'Black', '0001-01-01
00:00:00', 0 )
ON DUPLICATE KEY UPDATE
Naissance = '0001-01-01 00:00:00'
```

INSERT – ON DUPLICATE

- INSERT … ON DUPLICATE KEY UPDATE <attribut> = <valeur> {, <attribut> = <valeur> } WHERE <condition>

```
INSERT INTO Personne
VALUES (101, 'Joe', 'Black', NULL, 0),
(102, 'Maurice', 'Black', NULL, 0),
(103, 'Joe', 'White', NULL, 0)
ON DUPLICATE KEY UPDATE
Naissance = '0001-01-01 00:00:00‘
WHERE Prenom = 'Joe'
```

INSERT – SELECT

- INSERT INTO <table> [(<attribut> {, <attribut>})]
  SELECT … ;

Ajouter une sœur jumelle Meg à tous les Palmer

```
INSERT INTO Personne
SELECT NULL, 'Meg', Nom, Naissance, Parent
FROM Personne
WHERE Prenom = 'Palmer'
```

DELETE

- DELETE FROM <table> [WHERE …]

```
DELETE FROM Personne
WHERE Nom = 'Joe'
AND Prenom = 'Black'
```

TRUNCATE

- TRUNCATE TABLE <table>
- ⇔ DELETE FROM <table>

```
TRUNCATE TABLE Resultats
```

DROP – Table

- DROP TABLE [IF EXISTS] <table> {, <table> }

```
DROP DATABASE EssaiBasique, EssaiUnique
```

DROP – Base de Données

- DROP DATABASE [IF EXISTS] <base>

```
DROP DATABASE DBEssai
```

Transactions

- BEGIN [WORK]
- START TRANSACTION
- COMMIT [WORK] [AND CHAIN]
- ROLLBACK [WORK] [AND CHAIN]

- SET autocommit=0;

Bibliographie

- Site SQL.sh (http://sql.sh/cours)
- Documentation MySQL (http://dev.mysql.com/doc)
- W3Schools (http://www.w3schools.com/sql)