Base de données : SQL – Partie 2

Cours et exercices

Langage de manipulation de données

INSERT, UPDATE, DETETE SELECT

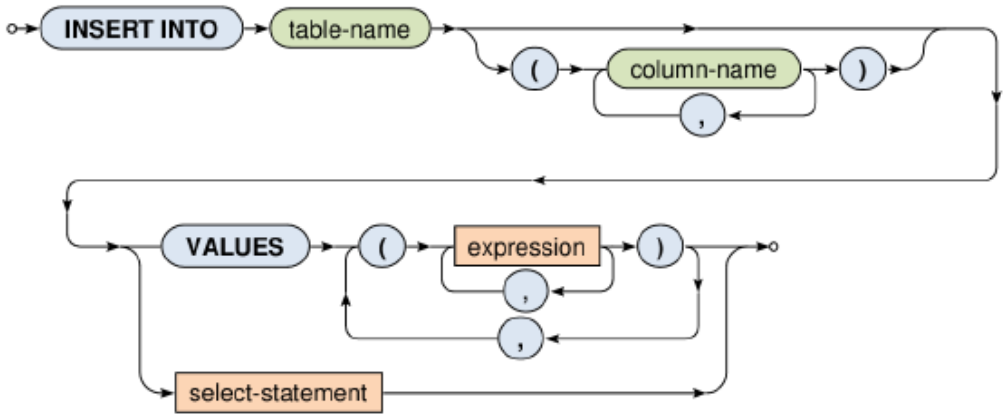
École d'ingénieurs du numérique

Langage de modification de données

- **INSERT INTO** – insertion de lignes
- **UPDATE** – modification de lignes
- **DELETE FROM** – suppression de lignes

INSERT INTO

- Insertion de nouvelles lignes dans une table
  - Avec une énumération explicite
  - Depuis le résultat d'un SELECT
- Les valeurs par défaut sont appliquées en cas de colonnes absentes
- Si pas de valeur par défaut, alors la colonne prendra la valeur NULL



INSERT INTO : Exemple

```
CREATE TABLE Product (
    id INTEGER PRIMARY KEY,
    name VARCHAR(128) UNIQUE,
    price DECIMAL(6,2) NOT NULL,
    produced DATE,
    available BOOLEAN DEFAULT TRUE,
    weight FLOAT,
    producer INTEGER
);
```

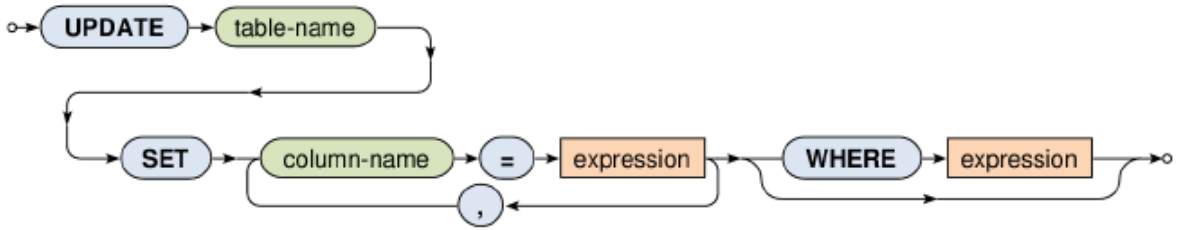
```
INSERT INTO Product VALUES (0, 'Chair1', 2000, '2015-05-06', TRUE, 3.5, 11);
```

```
INSERT INTO Product
   (id, name, price, produced, weight, producer)
   VALUES (1, 'Chair2', 1500, '2015-05-06', 4.5, 11);
```

La colonne **available** n'est pas mentionnée, la ligne insérée prendra la valeur par défaut (TRUE)

UPDATE WHERE

- Modification d'une ligne existante dans une table
  - Ne sont considérées que les lignes qui remplissent la condition
- Les nouvelles valeurs assignées peuvent être:
  - NULL, literal, une valeur donnée par une expression, résultat d'une sous-requête



UPDATE WHERE : Exemple

```
CREATE TABLE Product (
    id INTEGER PRIMARY KEY,
    name VARCHAR(128) UNIQUE,
    price DECIMAL(6,2) NOT NULL,
    produced DATE,
    available BOOLEAN DEFAULT TRUE,
    weight FLOAT,
    producer INTEGER
);
```

- *Remplacer "Laptop" par "Notebook"*

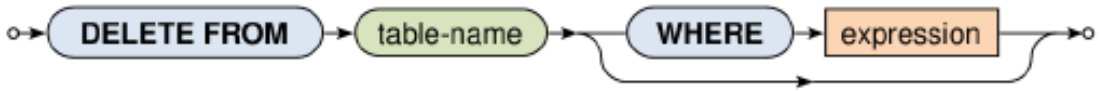
```
UPDATE Product
    SET name = 'Notebook'
    WHERE (name = 'Laptop');
```

- *Mettre à jour le prix avec une remise de 10% sur els produits produits avant le 1er Janvier 2015*

```
UPDATE Product
    SET Price = Price * 0.9
    WHERE (Produced < '2015-01-01');
```

DELETE FROM

- Suppression de lignes existantes d'une table
  - Ne sont considérées que les lignes qui remplissent la condition



DELETE FROM : Exemple

```
CREATE TABLE Product (
    id INTEGER PRIMARY KEY,
    name VARCHAR(128) UNIQUE,
    price DECIMAL(6,2) NOT NULL,
    produced DATE,
    available BOOLEAN DEFAULT TRUE,
    weight FLOAT,
    producer INTEGER
);
```

Supprimer tous les produits dont les prix sont supérieurs à 1000

```
DELETE FROM Product
WHERE (Price > 1000);
```

SELECT

- Permet d'interroger une BDD en :
  - sélectionnant certaines colonnes d'une table : **projection**
  - sélectionnant certaines lignes d'une table en fonction de leur contenu : **sélection**
  - combinant des informations venant de plusieurs tables : **jointure, union, intersection, différence et division**
  - combinant entre elles ces différentes opérations
- Une requête est une combinaison d'opérations portant sur des tables et dont le résultat est lui-même une table dont l'existence est éphémère (le temps de la requête)

SELECT : La projection

- Syntaxe :
  - **SELECT** colonne1 **FROM** tableA ➔ Récupère une colonne en particulier
  - **SELECT * FROM** tableA ou **SELECT** tableA.* **FROM** tableA ➔ Récupère toutes les colonnes
  - **SELECT DISTINCT** colonne1 **FROM** tableA ➔ Récupère une colonne en particulier sans doublons
    - **ALL** (par défaut) ➔ toutes les lignes sont présentes dans le résultat
  - **SELECT** colonne1 **AS** C1, colonne2 **AS** C2 **FROM** tableA **AS** TA ➔ Alias
  - **SELECT concat**(colonne1, ' ',colonne_2) AS colonne FROM T_CLIENT ➔ Concatène les colonnes

SELECT : Exemples de projection

Flights:

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Aircrafts:

Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

- **SELECT ALL** * FROM Aircrafts

Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

- **SELECT** Company FROM Aircrafts

Company
CSA
KLM
KLM

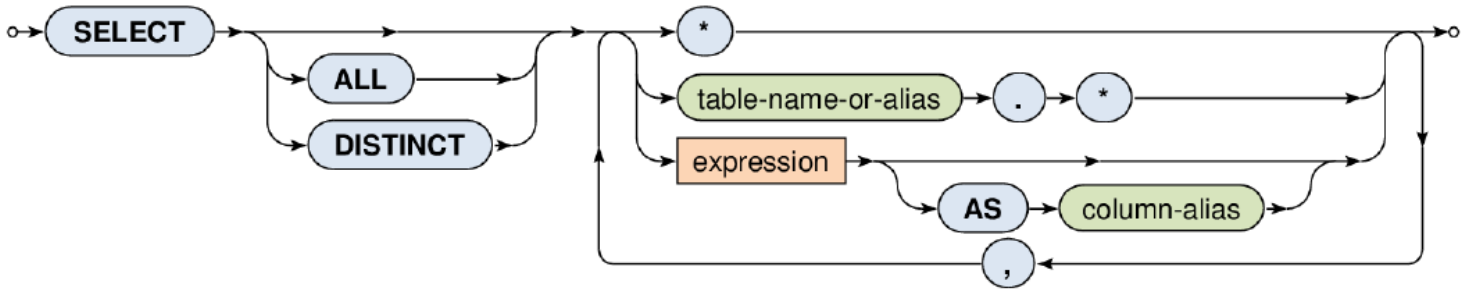
- **SELECT DISTINCT** Company FROM Aircrafts

Company
CSA
KLM

- **SELECT DISTINCT** Company **AS** Carrier FROM Aircrafts

Carrier
CSA
KLM

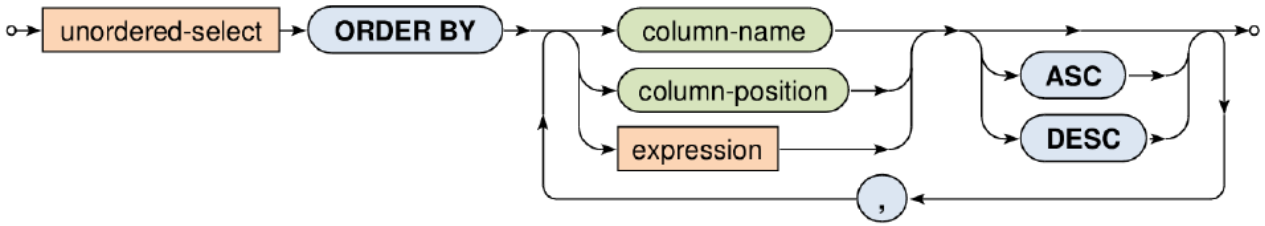
SELECT : La projection (exemple)



SELECT : ORDER BY

- Permet de trier des colonnes
  - Soit en précisant le nom littéral de la colonne
  - Soit en précisant son n° d'ordre dans l'énumération qui suit le mot clef SELECT
- ASC (par défaut) ou DESC
- **SELECT** colonne1, colonne2
  FROM tableA
  ORDER BY colonne2

SELECT colonne1, colonne2
FROM tableA
ORDER BY 2



SELECT : ORDER BY (exemple)

Flights:

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Aircrafts:

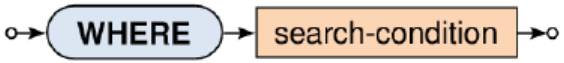
Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

- Retourner une liste ordonnée de toutes les destinations programmées

```
SELECT DISTINCT Destination
FROM Flights
ORDER BY Destination ASC
```

Destination
Amsterdam
London
Milano
New York
Rotterdam
Stuttgart
Toronto

SELECT : La sélection WHERE



- Représente la condition de sélection que doit satisfaire une ligne afin d'apparaitre dans un résultat
- Utilisation d'expressions qui peuvent être combinées grâce à **AND**, **OR et NOT**
- Exemples :
  - … **WHERE** (Capacity > 200) **AND** (Aircraft **LIKE** 'Airbus%') …
  - … **WHERE** (Company **IN** ('KLM', 'Emirates')) …
  - … **WHERE NOT** (Passengers **BETWEEN** 100 AND 200) …

SELECT : WHERE (exemples)

Flights:

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Aircrafts:

Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

- ```
  SELECT *
  FROM Flights
  WHERE Company='KLM'
  ```

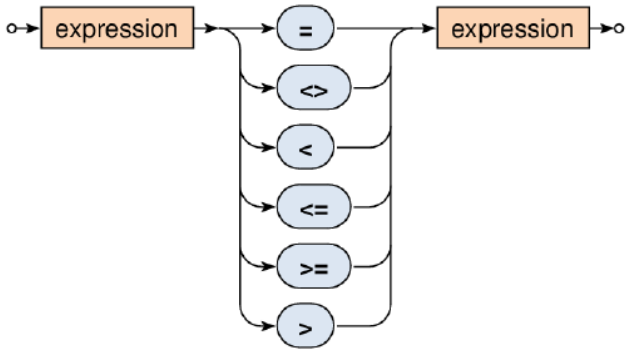
Flight	Company	Destination	Passengers
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

- ```
  SELECT Destination, Passengers
  FROM Flights
  WHERE Company='KLM' AND Passengers>100
  ```

Destination	Passengers
Amsterdam	130

SELECT : WHERE et Opérateurs (1)

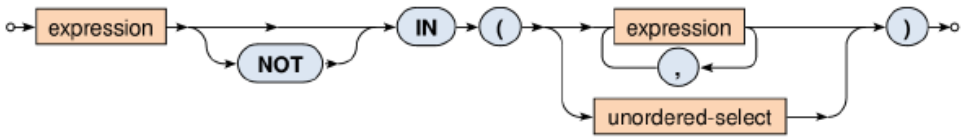
- Opérateurs de **comparaison**



- Opérateur d'intervalle **BETWEEN**
  - **value BETWEEN Min AND Max** équivalent à **(Min <= value) AND (value <= Max)**
  - **Ex :** …`WHERE Passengers BETWEEN 100 AND 200`

SELECT : WHERE et Opérateurs (2)

- Opérateur **IN** : vérifie qu'une valeur existe (ou pas) dans un ensemble de données
  - Ex : Company **IN** ('KLM', 'Emirates')
  - Ex : Company **NOT IN** ('KLM', 'Emirates')



SELECT : WHERE et Opérateurs (3)

- Opérateur **LIKE**
  - Permet de faire une comparaison partielle
  - Surtout employé avec les colonnes contenant des données de type alpha
  - utilise les jokers % et _
    - % remplace n'importe quelle chaîne de caractères, y compris la chaîne vide
    - _ remplace un et un seul caractère
- Exemples :
  - LIKE 'B%' : valeur qui commence par B
  - LIKE '%B' : valeur qui se termine par B
  - LIKE '_B%' : valeur qui contient le 'B' en seconde position
  - LIKE 'B__%' : valeur qui commence par B et qui a au moins 3 caractères de long
  - LIKE 'A%B' : valeur qui commence par A et se termine par B

SELECT : Fonctions d'agrégations

- **AVG()** calcule la moyenne d'un ensemble de valeurs.
  - Ex : le prix du panier moyen pour de chaque client
- **COUNT()** compte le nombre de lignes concernées.
  - Ex : combien d'achats ont été effectués par chaque client
- **MAX()** récupère la plus haute valeur
  - Ex : le produit le plus cher
- **MIN()** récupère la plus petite valeur.
  - Ex : la date du premier achat d'un client
- **SUM()** calcule la somme de plusieurs ligne
  - Ex : le total de tous les achats d'un client

SELECT : Fonctions d'agrégations (exemple)

```
SELECT
   COUNT(*) AS Flights,
   COUNT(DISTINCT Company) AS Companies,
   SUM(Passengers) AS PSum,
   AVG(Passengers) AS PAvg,
   MIN(Passengers) AS PMin,
   MAX(Passengers) AS PMax
FROM Flights
```

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Flights	Companies	PSum	PAvg	PMin	PMax
7	4	858	123	37	276

SELECT : GROUP BY

- Regroupe les lignes qui ont les mêmes valeurs dans un résumé
  - Ex : Trouver le nombre de client dans chaque pays

SELECT : HAVING

- Similaire au WHERE sauf qu'il permet de filtrer en utilisant des fonctions d'agrégation
- HAVING est très souvent utilisé en même temps que GROUP BY mais ce n'est pas obligatoire

SELECT : HAVING (exemple)

- Combien de vols chaque compagnie a planifié ?
  - Cependant, nous ne sommes pas intéressés par les vol vers Stuttgart et Munich
  - Et nous ne voulons pas les compagnies qui ont effectué un vol ou moins

SELECT : HAVING (exemple)

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
OK012		Milano	37
OK321		London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245		Amsterdam	130

Company	Flights
CSA	3
Air Canada	1
KLM	2

Company	Flights
CSA	3
KLM	2

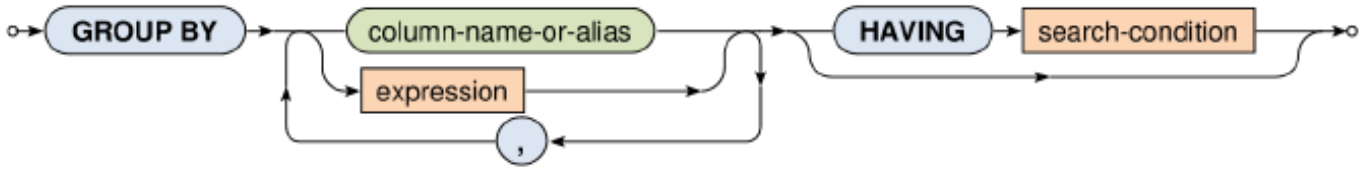
```
SELECT Company, COUNT(*) AS Flights
FROM Flights
WHERE (Destination NOT IN ('Stuttgart', 'Munich'))
GROUP BY Company HAVING (Flights > 1)
```

isep
École d'ingénieurs du numérique

SELECT : Ordre

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*

- D'abord…
  - Les clauses **FROM** and **WHERE** sont évaluées
  - Ceci réside dans une table intermédiaire
- Puis…
  - Les colonnes de cette table intermédiaire sont divisés en groupe suivant les colonnes identiques spécifiées dans le **GROUP BY**
- Enfin
  - Ces colonnes agrégées peuvent être filtrées grâce à la condition du **HAVING**



FROM

- **Définit les tables sur lesquelles faire la requêtes**
- Deux façons de faire :
  - **Notation WHERE**
    - Liste des tables séparée par une virgule
    - **Cartesian product** of their rows is assumed
    - La condition de jointure est spécifiée dans la clause WHERE

```
SELECT …
FROM Table1, Table2
WHERE Table1.xxx = Table2.yyy
  AND condition
```

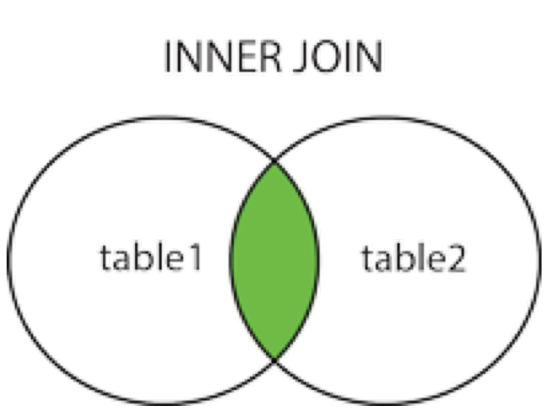
  - **Notation avec différents opérateurs JOIN**

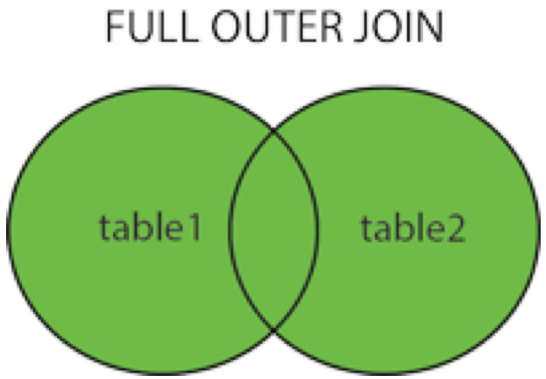
```
SELECT …
FROM Table1 JOIN Table2
WHERE condition
```

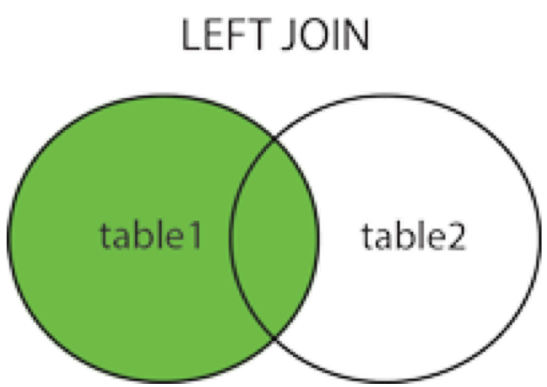
FROM : Jointures

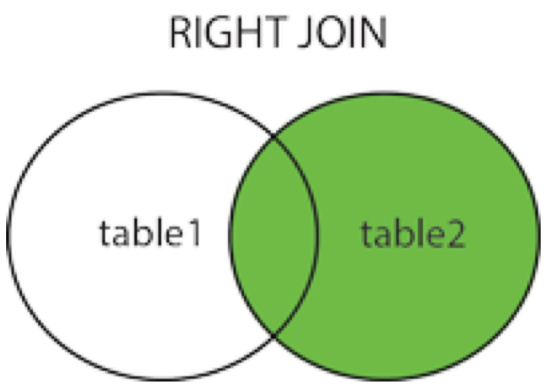
- **CROSS JOIN ou JOIN**
- **NATURAL JOIN**
- **INNER JOIN**
- **CROSS JOIN**
- **LEFT JOIN (ou LEFT OUTER JOIN)**
- **RIGHT JOIN (ou RIGHT OUTER JOIN)**
- **FULL JOIN (ou FULL OUTER JOIN)**
- **UNION JOIN**

FROM : Jointures



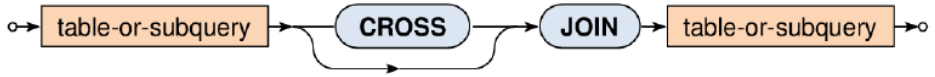






W3SCHOOL source

FROM: (Cross) join



- Jointure croisée permettant de faire le produit cartésien de 2 tables. i.e. joindre chaque ligne d'une table avec chaque ligne d'une seconde table. Attention, le nombre de résultats est en général très élevé.
  - `SELECT * FROM T1 CROSS JOIN T2`
  - `SELECT * FROM T1 JOIN T2`

Notation Where :

```
SELECT *
FROM T1, T2
```

A	T1.*
1	...
2	...
3	...

A	T2.*
1	...
4	...

T1.A	T1.*	T2.A	T2.*
1	...	1	...
1	...	4	...
2	...	1	...
2	...	4	...
3	...	1	...
3	...	4	...

FROM : 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
  - I.e. columns of the same name

Notation Where :

```
SELECT *
FROM T1, T2
WHERE T1.A = T2.A
```

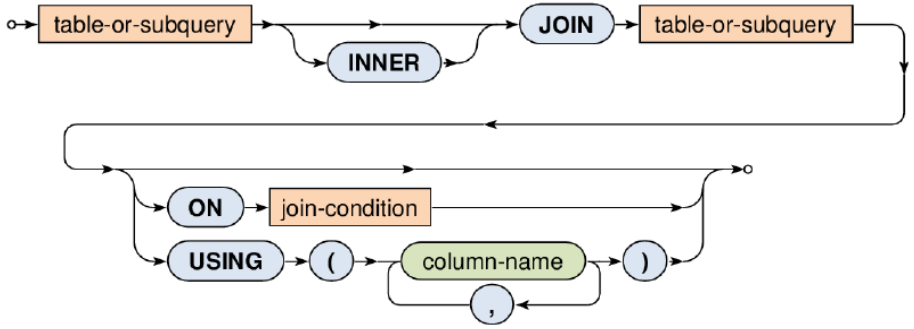
- `SELECT * FROM T1` **`NATURAL JOIN`** `T2`

A	T1.*
1	…
2	…
3	…

A	T2.*
1	…
4	…

A	T1.*	T2.*
1	…	…

FROM: Inner join



- Jointure interne pour retourner les lignes quand la condition qui suit le mot clé **ON** est vraie.
  - C'est l'une des jointures les plus communes.
  - `SELECT * FROM T1 INNER JOIN T2 ON (T1.A <= T2.A)`
  - `SELECT * FROM T1 JOIN T2 ON (T1.A <= T2.A)`

A	T1.*
1	…
2	…
3	…

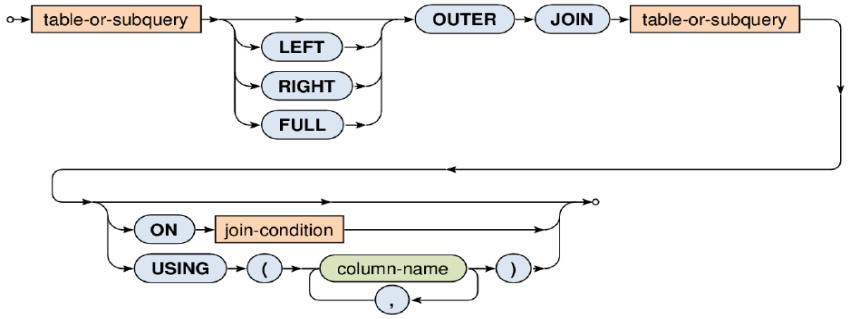
A	T2.*
1	…
4	…

T1.A	T1.*	T2.A	T2.*
1	…	1	…
1	…	4	…
2	…	4	…
3	…	4	…

Where notation:

```
SELECT *
FROM T1, T2
WHERE T1.A <= T2.A
```

FROM: Outer join



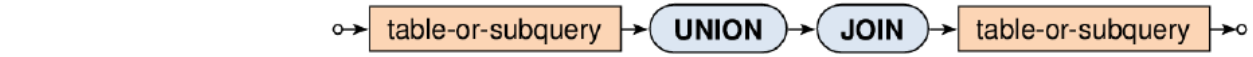
- Jointure externe pour retourner tous les lignes de la table de gauche (**LEFT OUTER JOIN**) ou droite (**RIGHT OUTER JOIN**) même si la condition n'est pas vraie dans l'autre table
  - **FULL** (par défaut): quand la condition est vraie dans au moins une des 2 tables
  - `SELECT * FROM T1 LEFT OUTER JOIN T2 ON (T1.A = T2.A)`

A	T1.*
1	…
2	…
3	…

A	T2.*
1	…
4	…

T1.A	T1.*	T2.A	T2.*
1	…	1	…
2	…	NULL	NULL
3	…	NULL	NULL

FROM: Union join



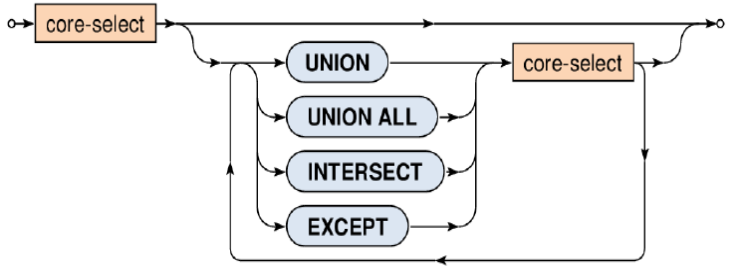
- Les lignes des deux tables sont intégrées au sein d'une même tables, pas de combinaison de lignes
  - `SELECT * FROM T1` **`UNION JOIN`** `T2`

A	T1.*
1	…
2	…
3	…

A	T2.*
1	…
4	…

T1.A	T1.*	T2.A	T2.*
1	…	NULL	NULL
2	…	NULL	NULL
3	…	NULL	NULL
NULL	NULL	1	…
NULL	NULL	4	…

Opérateurs ensemblistes



- **UNION** : Union de deux tables sans duplication
- **UNION ALL** : Union de deux tables avec duplication
- **INTERSECT** : Intersection de deux tables
- **EXCEPT** : Différence entre deux tables

- **Les deux opérandes doivent être compatibles**
  - Même nombre de colonnes
  - Ces colonnes doivent être du même type

Opérateurs ensemblistes : Exemples

Flights:

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Aircrafts:

Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

```
SELECT Company FROM Flights
UNION
SELECT Company FROM
Aircrafts
```

Company
CSA
Lufthansa
Air Canada
KLM

```
SELECT Company FROM Flights
INTERSECT
SELECT Company FROM
Aircrafts
```

Company
CSA
KLM

```
SELECT Company FROM Flights
EXCEPT
SELECT Company FROM
Aircrafts
```

Company
Lufthansa
Air Canada

Requêtes imbriquées

- Une requête à l'intérieur d'une autre, elle peut être utilisée :
  - Après un opérateur
    - **ANY**, **SOME**, **ALL**
    - **IN**
    - **EXISTS**
  - Lors de la definition d'une table dans la clause FROM
  - Presque dans toute expression si des valeurs scalaires sont produites

Requêtes imbriquées : Exemple 1

- Trouver tous les vols programmés qui ont un nombre de passagers supérieur à la moyenne

Flights:

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Aircrafts:

Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

```
SELECT *
FROM Flights
WHERE (Passengers > (SELECT AVG(Passengers) FROM Flights))
```

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
OK321	CSA	London	156
KL1245	KLM	Amsterdam	130

Requêtes imbriquées : Exemple 2

- Renvoyer le nombre d'aéronefs appropriés pour chaque vol.
  - Seuls les avions d'une compagnie donnée et d'une capacité suffisante peuvent être utilisés
  - Notez comment les valeurs de la requête externe sont liées à la requête interne

Flights:

Flight	Company	Destination	Passengers
OK251	CSA	New York	276
LH438	Lufthansa	Stuttgart	68
OK012	CSA	Milano	37
OK321	CSA	London	156
AC906	Air Canada	Toronto	116
KL7621	KLM	Rotterdam	75
KL1245	KLM	Amsterdam	130

Aircrafts:

Aircraft	Company	Capacity
Boeing 717	CSA	106
Airbus A380	KLM	555
Airbus A350	KLM	253

Flight	Company	Destination	Passengers	Aircrafts
OK251	CSA	New York	276	0
LH438	Lufthansa	Stuttgart	68	0
OK012	CSA	Milano	37	1
OK321	CSA	London	156	0
AC906	Air Canada	Toronto	116	0
KL7621	KLM	Rotterdam	75	2
KL1245	KLM	Amsterdam	130	2

```
SELECT
  Flights.*,
  (
    SELECT COUNT(*)
    FROM Aircrafts AS A
    WHERE
      (A.Company = F.Company) AND
      (A.Capacity >= F.Passengers)
  ) AS Aircrafts
FROM Flights AS F
```

Système de stockage : MyISAM vs InnoDB

MyISAM

+

- Système par défaut MySQL
- Requêtes SELECT ou INSERT + rapides

-

- Pas de transactions
- Pas de clés étrangères
- Plus difficile de récupérer après un crash

InnoDB

+

- Gère les transactions
- Gère les clés étrangères et les contraintes d'intégrité
- Support de ACID pour s'assurer que tous les enregistrements sont réussi ou échoué.

-

- Moteur de stockage plus imposant. Il demande plus de ressources et est plus lent