

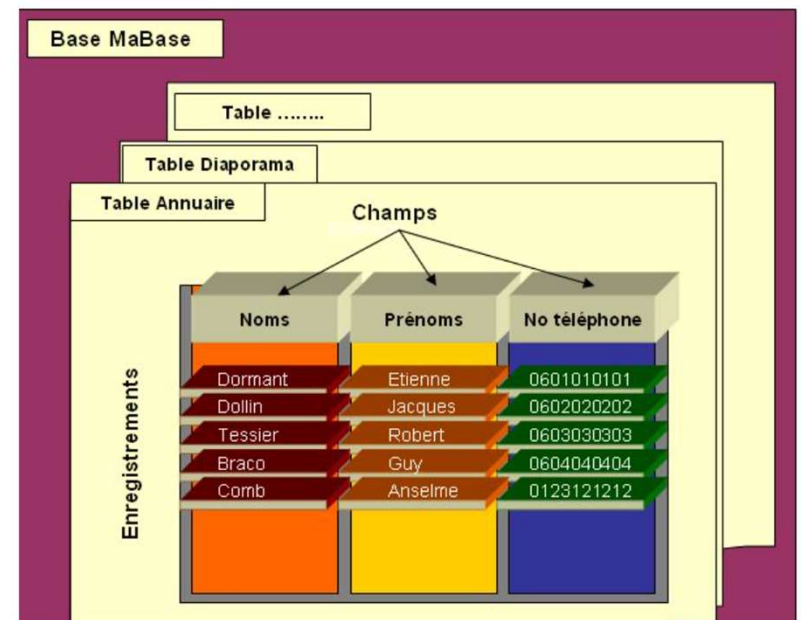
Module: Merise, UML et SGBD

CM3: SGBD et SQL

Systèmes de Gestion de Base de Données et Langage SQL

Base de données

- Une base de données est une collection organisée de données structurées qui sont stockées et accessibles électroniquement.
- Elle permet de stocker, gérer et manipuler des informations de manière efficace.
- Les bases de données facilitent la recherche, la récupération et la mise à jour des données, ce qui en fait un outil essentiel pour de nombreux domaines, tels que l'informatique, les affaires et les sciences.
- Une base de données est constituée de tables, qui sont des structures de données organisées en lignes et colonnes.



Langage SQL

- SQL (Structured Query Language, en français langage de requête structurée).
- C'est un langage informatique normalisé servant à gérer et exploiter des bases de données relationnelles.
- Il permet de définir, rechercher, ajouter, modifier ou supprimer des données dans les bases de données relationnelles.
- Il est créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégié SGBDR).

Langage SQL

- Les instructions SQL couvrent quatres domaines :
 - Langage de définition de données,
 - Langage de manipulation de données,
 - Langage de contrôle de données,
 - Langage de contrôle des transactions.

SELECT INSERT UPDATE DELETE MERGE	Data manipulation language (DML)
CREATE ALTER DROP RENAME TRUNCATE COMMENT	Data definition language (DDL)
GRANT REVOKE	Data control language (DCL)
COMMIT ROLLBACK SAVEPOINT	Transaction control

Langage de définition de données

- C'est le langage de manipulation des métadonnées : description de la structure, l'organisation et les caractéristiques de la base de données.
- Utilise les mots-clés CREATE, ALTER, DROP, RENAME, COMMENT ou TRUNCATE qui correspondent aux opérations d'ajouter, modifier, supprimer, renommer, commenter ou vider une métadonnée.
- Ces mots clés sont immédiatement suivis du type de métadonnée à manipuler - TABLE, VIEW, INDEX...

! Dans ce cours, on se focalise sur la définition des TABLES

Langage de manipulation de données

- C'est le langage de manipulation du contenu de la base de données
- Utilise les mots clés SELECT, UPDATE, INSERT ou DELETE qui correspondent respectivement aux opérations de recherche de contenu, modification, ajout et suppression.
- Divers autres mots-clés tels que FROM, JOIN et GROUP BY permettent d'indiquer les opérations d'algèbre relationnelle à effectuer en vue d'obtenir le contenu à manipuler.

```
SELECT nom, service FROM employe WHERE statut = 'stagiaire' ORDER BY nom;
```

Langage de contrôle des transactions et des données

- C'est le langage de programmation et un sous-ensemble de SQL pour contrôler l'accès aux données d'une base de données, et celui utilisé pour le contrôle transactionnel dans une base de données, c'est-à-dire les caractéristiques des transactions, la validation et l'annulation des modifications.
- Exemples des mots-clés LCD et LCT:
 - COMMIT, SAVEPOINT, ROLLBACK, SET TRANSACTION...
 - Les mots clés GRANT et REVOKE permettent d'autoriser des opérations à certaines personnes, d'ajouter ou de supprimer des autorisations.
 - Les mots clés COMMIT et ROLLBACK permettent de confirmer ou annuler l'exécution de transactions.

Langage de contrôle de données

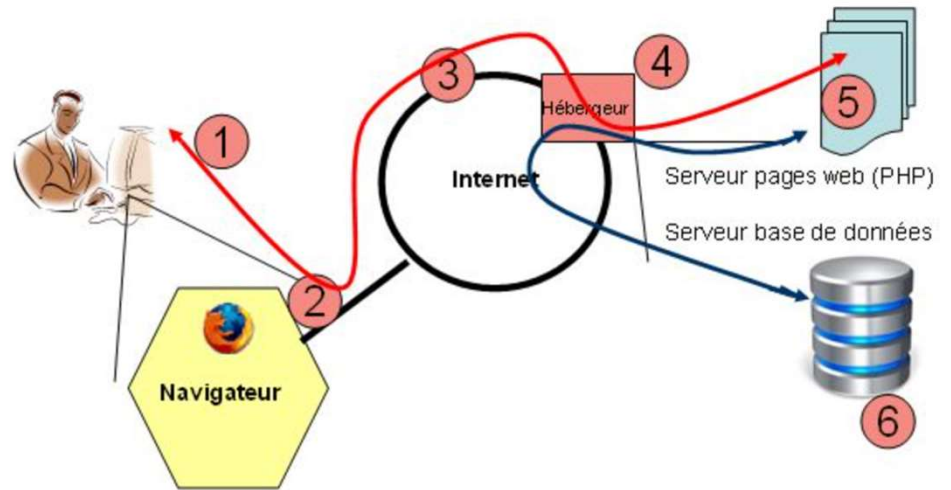
- On distingue typiquement six types de commandes SQL de contrôle de données :
 - GRANT : autorisation d'un utilisateur à effectuer une action ;
 - DENY : interdiction à un utilisateur d'effectuer une action ;
 - REVOKE : annulation d'une commande de contrôle de données précédente ;
 - COMMIT : validation d'une transaction en cours ;
 - ROLLBACK : annulation d'une transaction en cours ;
 - LOCK : verrouillage sur une structure de données.

Les transactions ACID

- Les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) sont des propriétés fondamentales des bases de données qui garantissent l'intégrité et la fiabilité des opérations.
- **Atomicité** signifie que toutes les modifications d'une transaction sont effectuées avec succès ou pas du tout.
- **Cohérence** assure que chaque transaction maintient la cohérence globale de la base de données.
- **Isolation** garantit que chaque transaction est exécutée de manière isolée, sans interférence avec d'autres transactions.
- **Durabilité** garantit que les modifications effectuées lors d'une transaction sont permanentes et ne peuvent pas être perdues, même en cas de panne du système.

Principe des échanges avec le serveur de bases de données

- L'internaute (1) se connecte (via 2, 3, 4 sur la page WEB hébergée (5)).
- Les instructions php contenues dans la page WEB nécessite des demandes de données contenues dans le serveur de base de données (6).
- La demande et la réception des données s'effectuent à l'aide d'une requête **SQL**. Les données récupérées sont utilisées pour mettre à jour la page WEB qui est envoyée à l'utilisateur.



Systèmes de Gestion de Base de Données

- Un SGBD (Système de Gestion de Base de Données) est un logiciel qui permet de gérer et d'organiser de manière efficace et structurée les données d'une base de données. Il fournit des fonctionnalités pour créer, stocker, modifier et récupérer les informations stockées dans la base de données.
- Un SGBD permet de garantir l'intégrité et la cohérence des données, d'assurer la sécurité des données, ainsi que d'optimiser les performances des opérations de manipulation de données. Il facilite également l'interaction avec la base de données en fournissant des langages de requête et des interfaces pour interagir avec les données.

Systèmes de Gestion de Base de Données

- Certains SGBDR (Systèmes de Gestion de Base de Données Relationnelles) populaires :
 - Oracle Database
 - Microsoft SQL Server
 - MySQL
 - PostgreSQL
 - IBM Db2
 - SQLite
 - MongoDB (SGBD orienté document)
 - MariaDB
 - SAP HANA
 - Amazon RDS (service de base de données managé)
- Ces SGBDR offrent une variété de fonctionnalités, de performances et de compatibilité avec différents systèmes d'exploitation, et sont utilisés dans diverses applications et environnements.

MySQL



- MySQL est un système de gestion de base de données relationnelle (SGBDR), mais il est développé par Oracle Corporation.
- MySQL est open source et largement utilisé, offrant une solution fiable et performante pour le stockage et la gestion des données.
- Il prend également en charge le langage SQL pour l'interaction avec la base de données, ainsi que des fonctionnalités telles que la réplication, la sécurité et la gestion des transactions.
- MySQL est souvent utilisé dans les applications web et est connu pour sa rapidité et sa flexibilité.

SQL Server



- SQL Server est un système de gestion de base de données relationnelle (SGBDR) développé par Microsoft.
- Il offre une plateforme complète pour stocker, gérer et manipuler de grandes quantités de données.
- SQL Server prend en charge le langage de requête SQL (Structured Query Language) pour interagir avec la base de données, ainsi que des fonctionnalités avancées telles que la sécurité, la réplication, la haute disponibilité et l'analyse de données.

Oracle Database



- Oracle Database est un système de gestion de base de données relationnelle développé par Oracle Corporation.
- Il est largement utilisé dans les entreprises et offre une gamme complète de fonctionnalités pour la gestion et l'analyse de grandes quantités de données.
- Oracle Database prend en charge le langage SQL ainsi que des fonctionnalités avancées telles que la haute disponibilité, la sécurité des données, la réplication et l'optimisation des performances.



PostgreSQL



- PostgreSQL est un système de gestion de base de données relationnelle open source.
- Il est reconnu pour sa fiabilité, sa stabilité et sa conformité aux normes SQL. PostgreSQL prend en charge les fonctionnalités avancées telles que les vues, les déclencheurs, les procédures stockées et les transactions ACID.
- Il offre également des extensions et une grande flexibilité pour personnaliser et étendre ses fonctionnalités de base.

MariaDB



- MariaDB est un système de gestion de base de données relationnelle open source, qui est une branche et une alternative compatible avec MySQL.
- MariaDB est conçu pour être une solution performante, évolutive et fiable.
- Il offre une compatibilité avec MySQL et prend en charge les fonctionnalités avancées telles que les transactions ACID, les vues, les déclencheurs et les procédures stockées.
- MariaDB est souvent utilisé dans les applications web et est apprécié pour sa facilité d'utilisation et sa compatibilité.

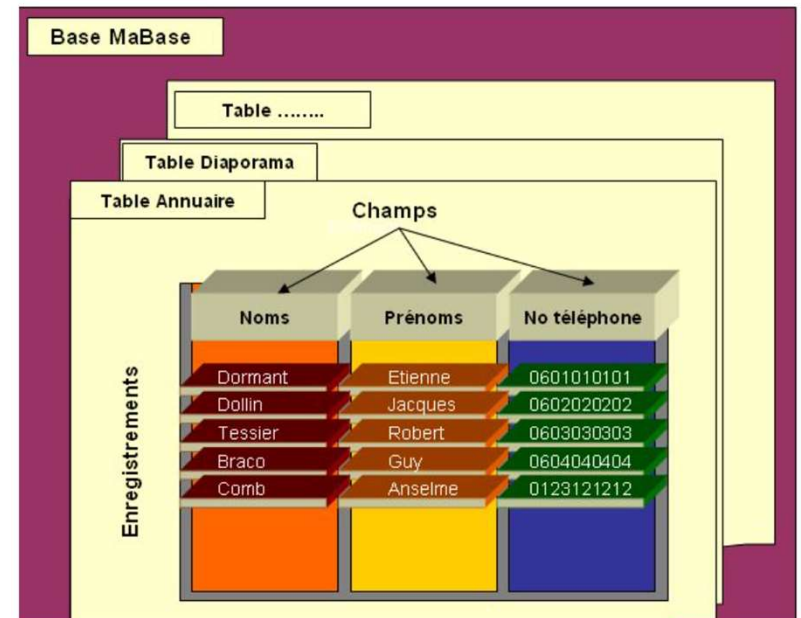
SQLite



- SQLite est un système de gestion de base de données relationnelle open source et sans serveur.
- **Contrairement aux autres SGBDR, SQLite stocke la base de données sous forme d'un fichier unique sur le système de fichiers plutôt que d'un serveur distinct.**
- **Il est léger, rapide et facile à intégrer dans des applications, en particulier pour les applications mobiles et les systèmes embarqués.**
- SQLite prend en charge les fonctionnalités de base SQL et est largement utilisé pour les petites et moyennes bases de données.

Base de données

- Une base de données est une collection organisée de données structurées qui sont stockées et accessibles électroniquement.
- Elle permet de stocker, gérer et manipuler des informations de manière efficace.
- Les bases de données facilitent la recherche, la récupération et la mise à jour des données, ce qui en fait un outil essentiel pour de nombreux domaines, tels que l'informatique, les affaires et les sciences.
- Une base de données est constituée de tables, qui sont des structures de données organisées en lignes et colonnes.



Table

- Une table est composée de lignes et de colonnes comme sur un tableau Excel :
 - Chaque ligne correspond à un enregistrement.
 - Un enregistrement est composé de plusieurs données, réparties dans plusieurs colonnes.
 - Chaque donnée correspond à un champ. Un enregistrement est donc composé de plusieurs champs.
 - Chaque colonne correspond à un attribut qui permet de classer un champ.

Table

- Exemples:
 - Considérons une base de données d'une boutique en ligne. Celle-ci peut être composée de deux types de données :
 - Les données sur les différentes commandes du magasin : Nom du produit et prix
 - Les données sur les clients du magasin : Prénom, nom, adresse.

Table "Commande"

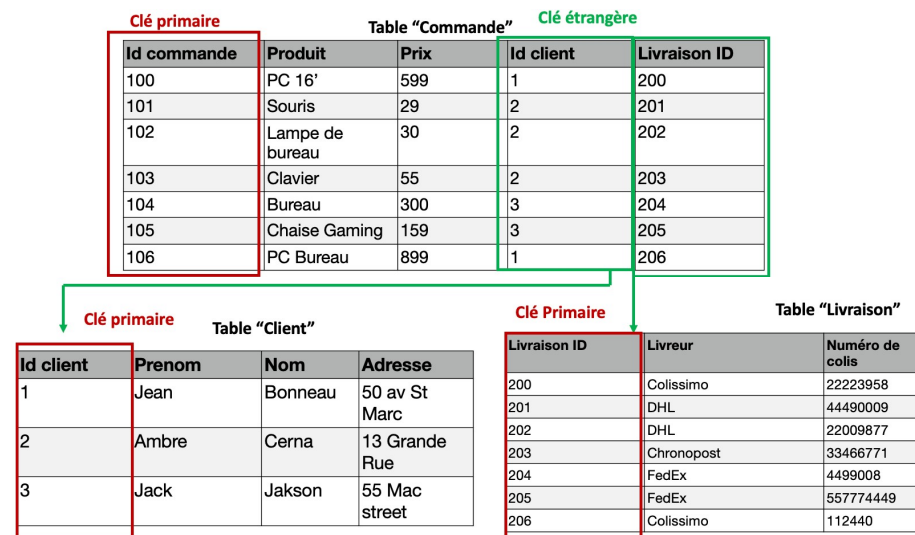
Produit	Prix
PC 16"	599
Souris	29
Lampe de bureau	30
Clavier	55

Table "Client"

Prenom	Nom	Adresse
Jean	Bonneau	50 av St Marc
Ambre	Cerna	13 Grande Rue

La clé primaire

- La clé primaire d'une table est un attribut ou un ensemble d'attributs qui **identifie de manière unique chaque enregistrement** dans une table d'une base de données.
- Elle garantit l'unicité des données et permet un accès rapide et efficace aux enregistrements.
- La clé primaire est essentielle pour maintenir l'intégrité et la cohérence des données dans une base de données relationnelle.



Pourquoi utiliser une clé étrangère?

Table "Commande"

Produit	Prix
PC 16'	599
Souris	29
Lampe de bureau	30
Clavier	55

Table "Client"

Prenom	Nom	Adresse
Jean	Bonneau	50 av St Marc
Ambre	Cerna	13 Grande Rue

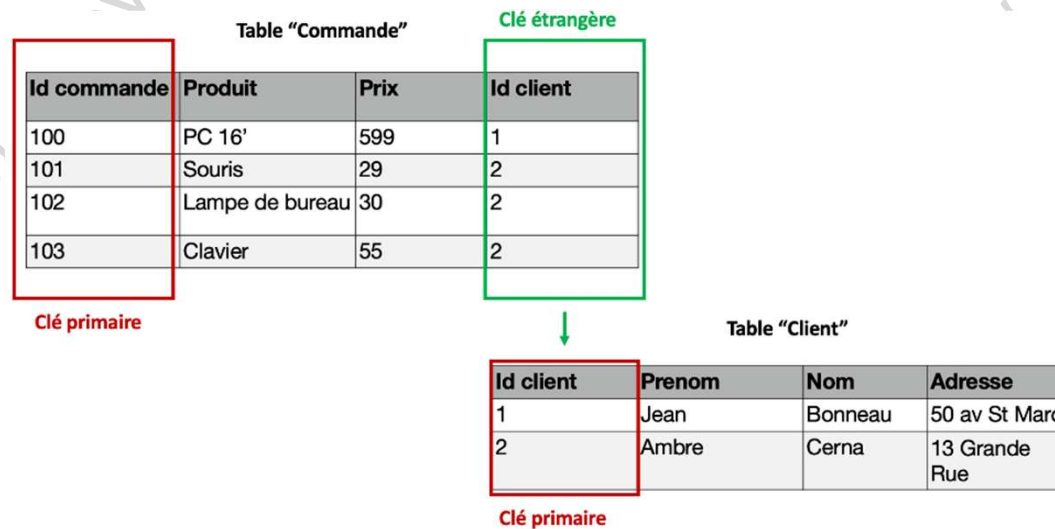
- Dans cet exemple, il faut trouver un moyen d'**associer chaque commande à un client**.
- Il faut mettre en relation les deux tables pour associer les clients Jean Bonneau et Ambre Cerna aux différentes commandes de la boutique.
- **C'est ici que la clé étrangère entre en jeu !**

La clé étrangère

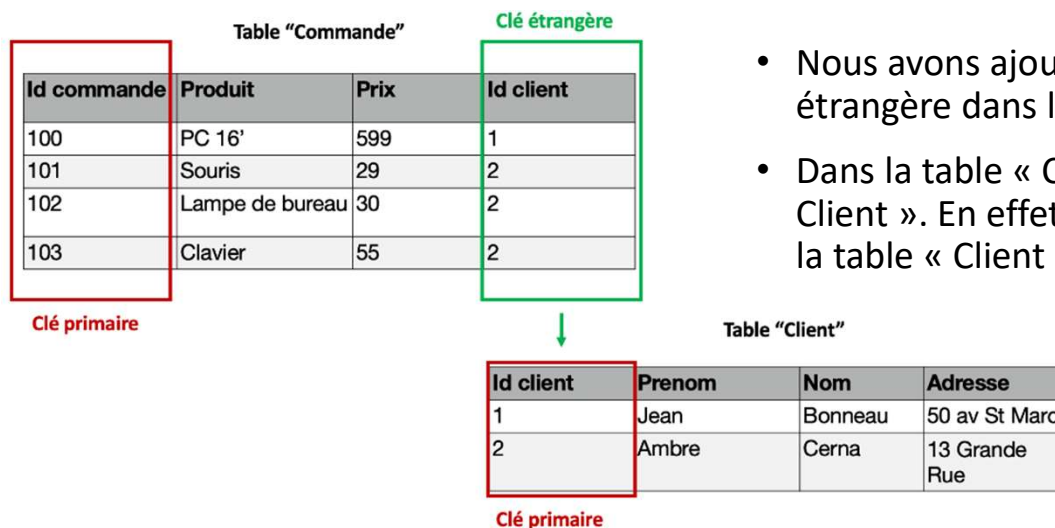
- La clé étrangère met en relation deux tables au sein d'une BDD relationnelle.
- Elle permet d'assurer l'intégrité référentielle des données. Autrement dit, seules les valeurs devant apparaître dans la base de données sont permises.
- **La clé étrangère fait référence à la clé primaire d'une autre table.**

La clé étrangère

- Exemple:
 - Reprenons l'exemple de la boutique en ligne.
 - Voici comment mettre en relation la table "Commande" avec la table "Client" :



La clé étrangère



- Nous avons ajouté les clés primaires de chaque table et une clé étrangère dans la table « Commande »
- Dans la table « Commande », la clé étrangère est l'attribut « Id Client ». En effet, celle-ci référence la clé primaire « Id Client » de la table « Client ».

- Maintenant que la colonne « Id Client » se trouve dans les deux tables, il est facile de retrouver les commandes effectuées par Ambre Cerna. Il nous suffit de ne garder que les commandes dont l'identifiant est 2. Idem pour Jean Bonneau, son identifiant est 1.
- La clé étrangère « Id client » met donc en relation la table « Commande » et la table « Client ».

Clé étrangère et intégrité référentielle de la BDD

- La clé étrangère est une contrainte qui s'assure du respect de l'intégrité référentielle de la base de données. → Concrètement, une donnée qui compose la clé étrangère d'une table A doit faire référence à une donnée existante dans la clé primaire d'une table B.
- L'intégrité référentielle est un concept clé dans les bases de données relationnelles. Il s'agit de la cohérence des relations entre les tables d'une base de données. L'intégrité référentielle est généralement définie à l'aide de contraintes qui garantissent que les relations entre les clés primaires et étrangères sont maintenues de manière correcte.

Clé étrangère et intégrité référentielle de la BDD

- Plus précisément, l'intégrité référentielle garantit que **toutes les valeurs d'une clé étrangère (colonne référençant une autre table) existent dans la table référencée (table avec la clé primaire correspondante)**. Cela signifie qu'il ne peut y avoir aucune valeur dans une colonne de clé étrangère qui n'a pas de correspondance dans la table référencée.
- En utilisant des contraintes d'intégrité référentielle, telles que les clés étrangères et les actions de mise à jour ou de **suppression en cascade**, la base de données s'assure que les relations entre les tables restent cohérentes et maintient l'intégrité des données. Ainsi, l'intégrité référentielle garantit que les données sont correctement liées entre elles, évitant ainsi les incohérences et les données orphelines dans la base de données.

Clé étrangère et intégrité référentielle de la BDD

- Exemples:

- Dans la boutique en ligne, la contrainte est qu'une commande doit nécessairement être associée à un « Id Client » qui est déjà référencé dans la table « Client ». Sinon, la clé étrangère pointe vers du vide et la mise en relation est impossible !
- Pour faire cela, nous avons défini une clé étrangère dans la table « Commande » : « Id Client ». Ainsi, la table « Commande » n'accepte que des « Id Client » qui existe dans la table « Client ».
- De cette manière, on s'assure que la table « Commande » contient uniquement des informations sur des clients existants dans la table « Client ».
- En définissant une clé étrangère, nous avons donc respecté l'intégrité référentielle dans la base de données de la boutique en ligne !

			Clé étrangère
Id commande	Produit	Prix	Id client

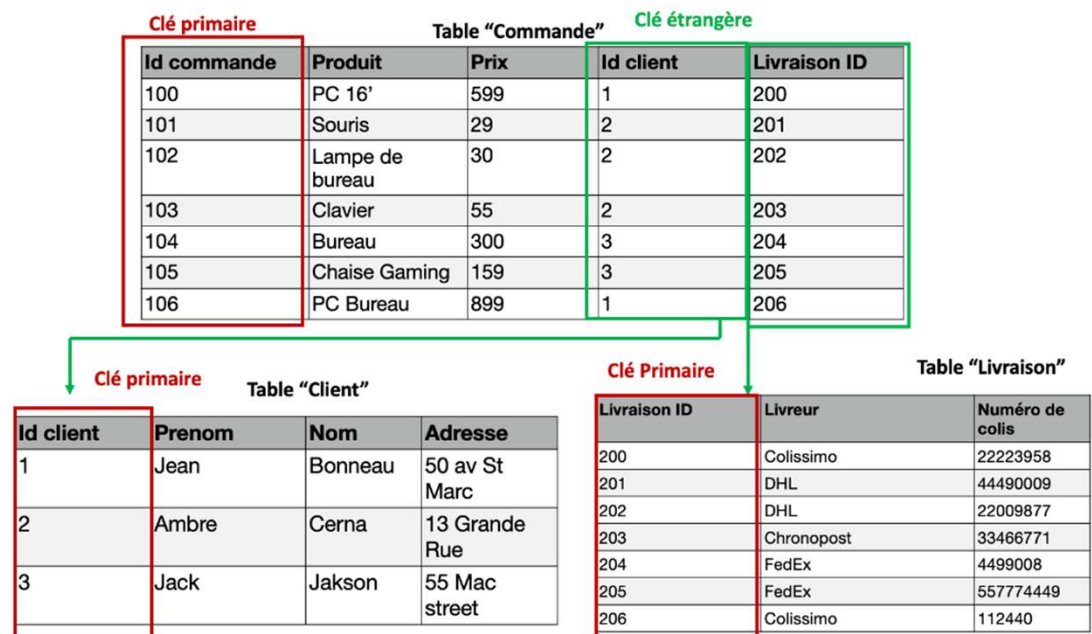
Clé primaire

Limitation et contraintes de la clé étrangère

- La clé étrangère doit suivre une série de contraintes :
 - La clé étrangère ne peut faire référence qu'à une colonne (ou des colonnes) au sein de la même base de données, sur le même serveur.
 - La colonne de la clé étrangère et celle qu'elle référence doivent être de même type (INT, VARCHAR etc...).
 - Une clé étrangère ne peut pas être appliquée dans des tables temporaires.
 - Fais donc bien attention à respecter ces contraintes quand tu manipules une clé étrangère !

Exploitation de la clé étrangère avec une jointure

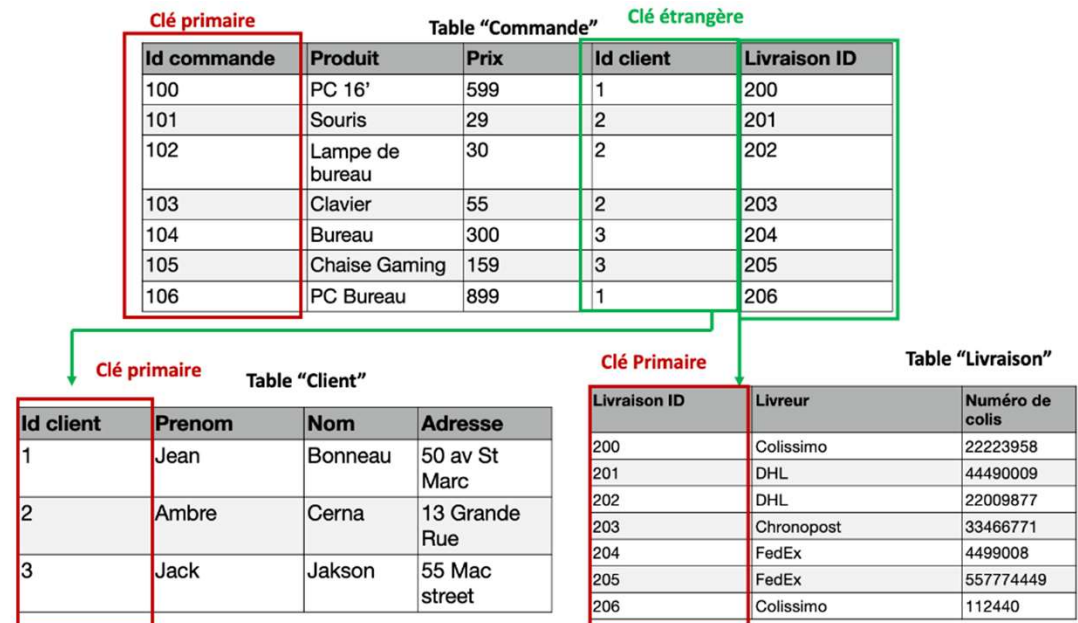
- Comment exploiter cet outil de référencement grâce à une technique très connue : les jointures ?
- Une jointure SQL permet de fusionner tout ou partie de plusieurs tables afin d'extraire les informations que tu souhaites analyser. Pour y arriver, il faut donc utiliser la ou les clés étrangères à ta disposition.



Exploitation de la clé étrangère avec une jointure

- Exemple:

- Sur cette base de données on peut effectuer des analyses ciblées pour mieux connaître notre clientèle.
- Par exemple, on veut connaître la liste de toutes les commandes effectuées par Jean Bonneau avec ses informations de livraison.
- Pour cela on va réaliser une jointure sur les trois tables. Cette jointure va utiliser les clés étrangères « Livraison ID » et « Id Client » qui mettent en relation les tables.



Id commande	Produit	Prix	Livreur	Numéro de colis
100	PC 16'	599	Colissimo	22223958
106	PC Bureau	899	Colissimo	112440

Grâce à la **clé étrangère**, nous avons réussi à associer plusieurs **tables** et à filtrer les données pour n'avoir que les commandes associées à un client !

Types de données

- Les principaux types de données en SQL sont :
 - **CHARACTER** (ou **CHAR**) : chaîne de caractères de longueur fixe.
 - **VARCHAR** (ou **CHARACTER VARYING**) : chaîne de caractères de longueur maximale fixée.
 - **TEXT** : suite longue de caractères (sans limite de taille).
 - **BOOLEAN** (ou **LOGICAL**) : vrai/faux
 - **DATE** : date du calendrier grégorien.
 - **NUMERIC**, **INTEGER** (ou **INT**), **DECIMAL**, **FLOAT**, **REAL** : des nombres réels avec des tailles et précisions variables.

!! Il est important de noter que les caractéristiques exactes de ces types de données peuvent varier selon le système de gestion de base de données (SGBD) utilisé, donc il est préférable de consulter la documentation spécifique du SGBD pour des détails précis.

Types de données

- Les principaux types de données SQL numériques:
 - **NUMERIC** : Utilisé pour stocker des nombres décimaux avec une précision fixe. La précision totale et la scale (nombre de chiffres après la virgule) doivent être spécifiées.
 - **INTEGER** (ou **INT**) : Utilisé pour stocker des nombres entiers sans décimales, avec une taille de stockage fixe.
 - **DECIMAL** : Un autre nom pour le type de données NUMERIC. Utilisé pour représenter des nombres décimaux avec une précision fixe, en spécifiant à la fois la précision totale et la scale.
 - **FLOAT** : Un type de données à virgule flottante qui permet de stocker des nombres réels avec une précision variable. Il peut stocker une plus large gamme de valeurs que le type DECIMAL, mais il peut également être moins précis.
 - **REAL** : Un autre nom pour le type de données FLOAT. Utilisé pour stocker des nombres réels avec une précision variable, généralement de 32 bits.

Opérateurs SQL

D'opérateurs logiques

- AND
- OR
- NOT

De comparateurs de chaîne :

- IN
- BETWEEN
- LIKE

D'opérateurs arithmétiques :

- +
- -
- *
- /
- %
- &
- |
- ^
- ~

Et de comparateurs arithmétiques :

- =
- !=
- >
- <
- >=
- <=
- >
- <

Do Not Copy

L'instruction SELECT

Projection

STUDENT				
<u>studno</u>	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

STUDENT				
<u>studno</u>	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s4	bloggs	ca	goble	1
s5	jones	cs	zobel	1
s6	peters	ca	kahn	3

```
select *
from student;
```

Selection

STAFF	
<u>lecturer</u>	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

tutor
bush
kahn
goble
goble
zobel
kahn

```
select tutor
from student;
```

La clause WHERE

STUDENT				
<u>studno</u>	name	hons	tutor	year
s1	jones	ca	bush	2
s2	brown	cis	kahn	2
s3	smith	cs	goble	2
s5	jones	cs	zobel	1

STAFF	
<u>lecturer</u>	roomno
kahn	IT206
bush	2.26
goble	2.82
zobel	2.34
watson	IT212
woods	IT204
capon	A14
lindsey	2.10
barringer	2.125

Table 1

Join

Table 2

<u>studno</u>	name	hons	tutor	year	<u>lecturer</u>	roomno
s1	jones	ca	bush	2	bush	2.26
s2	brown	cis	kahn	2	kahn	IT206
s3	smith	cs	goble	2	goble	2.82
s5	jones	cs	zobel	1	zobel	2.34

```
select * from student, staff
where tutor = lecturer ;
```

Projection

Table 1

<u>tutor</u>
bush
zobel

```
select tutor from student
where name = 'jones' ;
```

Les opérateurs AND & OR

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

```
SELECT * FROM Customers  
WHERE Country='Germany'  
AND City='Berlin';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

```
SELECT * FROM Customers  
WHERE City='Berlin'  
OR City='München';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

La clause ORDER BY

SELECT * FROM *Table* ORDER BY *NomChamps* [ASC | DESC];

```
SELECT * FROM Customers  
ORDER BY Country;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

Les expressions arithmétiques

Opérateurs arithmétiques			
Symbole	Signification	Exemple	Résultat
-	Soustraction	10 - 5	5
+	Addition	12+12	24
*	Multiplication	4*7	28
/	Division	125/25	5
^	Exposant	12^2	144
%	Pourcentage	25%	0,25

```
SELECT last_name, salary, salary + 300
FROM employees;
```

	LAST_NAME	SALARY	SALARY+300
1	King	24000	24300
2	Kochhar	17000	17300
3	De Haan	17000	17300
4	Hunold	9000	9300
5	Ernst	6000	6300
6	Lorentz	4200	4500
7	Mourgos	5800	6100
8	Rajs	3500	3800
9	Davies	3100	3400
10	Matos	2600	2900

L'opérateur de concatenation

```
SELECT last_name || job_id AS "Employees"
FROM employees;
```

```
SELECT CONCAT(last_name, job_id) AS "Employees"
FROM employees;
```

	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP

```
SELECT last_name || ' is a ' || job_id
       AS "Employee Details"
FROM employees;
```

	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP

L'opérateur BETWEEN

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

L'opérateur IN

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

L'opérateur LIKE

- Effectuer des recherches de valeurs de chaîne de caractères
- % représente zéro ou plusieurs caractères
- _ représente un seul caractère

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE '_o%';
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

L'opération Alias

```
SELECT CustomerName AS Customer  
FROM Customers;
```

Customer

Alfreds Futterkiste
Ana Trujillo Emparedados y helados
Antonio Moreno Taquería
Around the Horn
Berglunds snabbköp

```
SELECT CustomerName, CONCAT(Address,', ',City,', ',PostalCode,', ',Country) AS Address  
FROM Customers;
```

CustomerName	Address
Alfreds Futterkiste	Obere Str. 57, Berlin, 12209, Germany
Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222, México D.F., 05021, Mexico
Antonio Moreno Taquería	Mataderos 2312, México D.F., 05023, Mexico
Around the Horn	120 Hanover Sq., London, WA1 1DP, UK
Berglunds snabbköp	Berguvsvägen 8, Luleå, S-958 22, Sweden

La clause DISTINCT

```
SELECT City FROM Customers;
```

City
Berlin
México D.F.
México D.F.
London
Luleå

```
SELECT DISTINCT City FROM Customers;
```

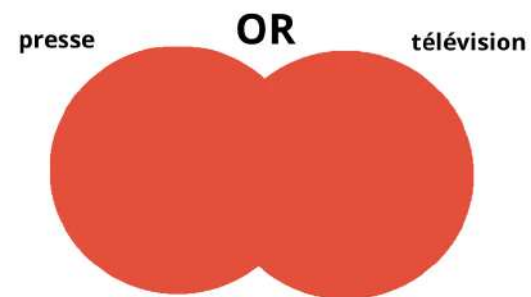
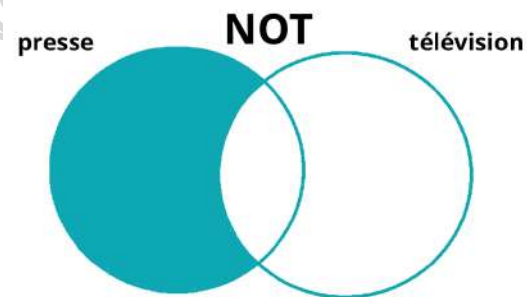
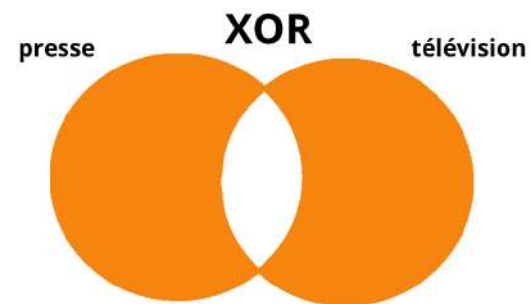
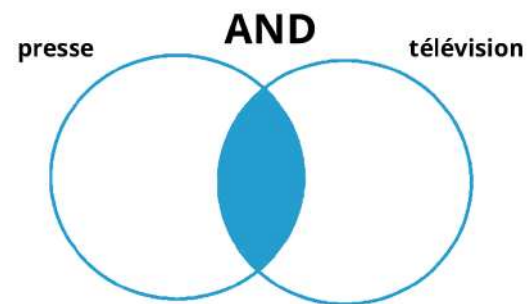
City
Berlin
México D.F.
London
Luleå

La condition NULL

```
SELECT last_name, manager_id  
FROM employees  
WHERE manager_id IS NULL ;
```

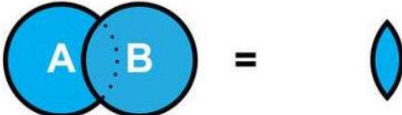
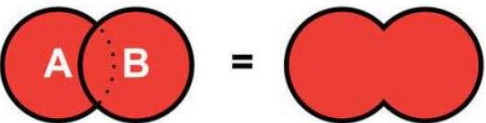
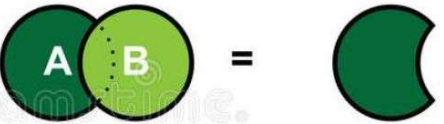
	LAST_NAME	MANAGER_ID
1	King	(null)

Les opérateurs booléens



Les opérateurs booléens

Les opérateurs booléens

Opérateurs Français Anglais		Effet	Résultats	Taille des résultats	Relie
ET	AND	 Croiser	A et B	Diminue	Les concepts
OU	OR	 Associer	A, B, A et B	Augmente	Les synonymes, termes équivalents, traductions
SAUF	NOT	 Exclure	A sans B	Diminue	Les concepts

Exemple d'équation de recherche

$(A \text{ OR } a) \text{ AND } (B \text{ OR } B \text{ OR } b) \text{ NOT } (C)$

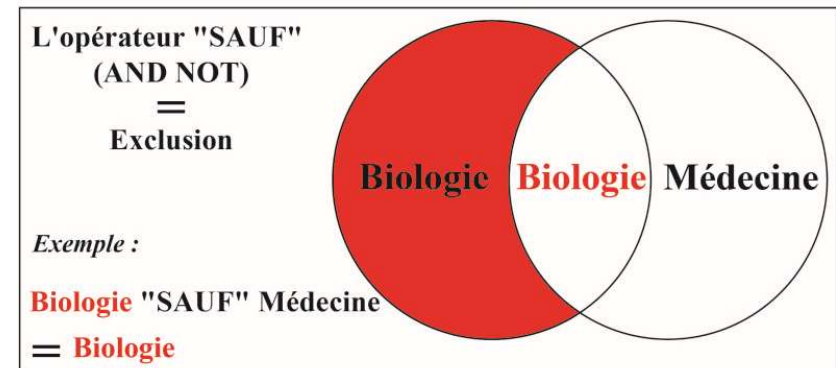
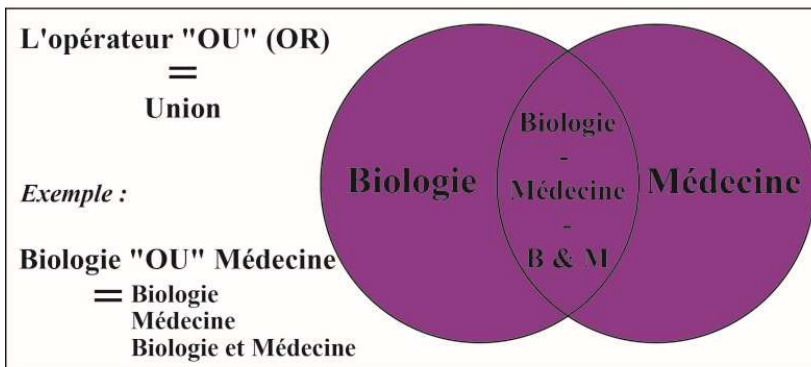
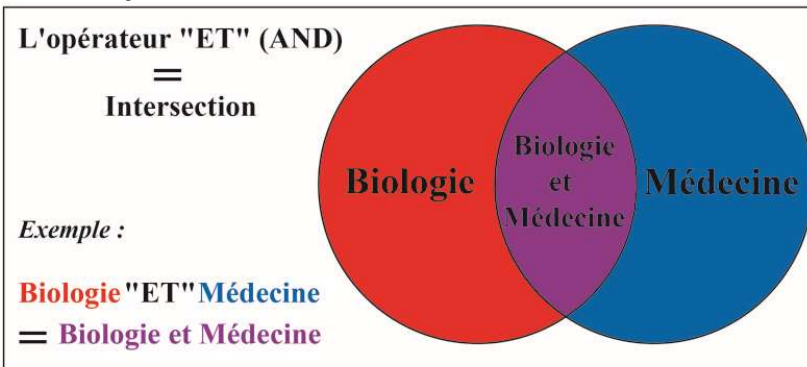
NB : Le langage peut être adapté en fonction de la base de données

© Cellule de Développement Pédagogique – Secteur des Sciences de la Santé - UCLouvain



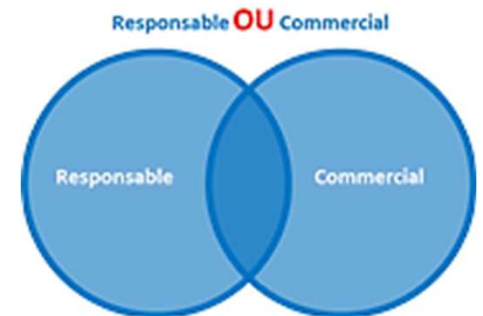
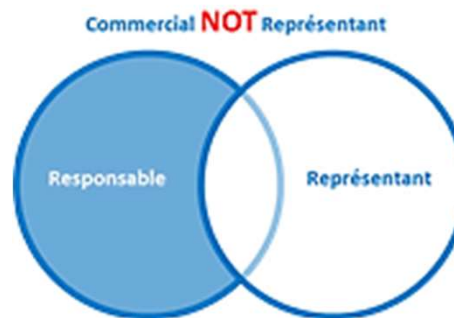
Les opérateurs booléens

- Exemples:



Les opérateurs booléens

- Exemples:



L'opérateur NOT

```
SELECT last_name, job_id
FROM employees
WHERE job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

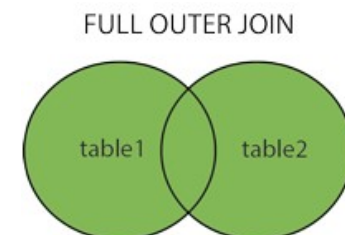
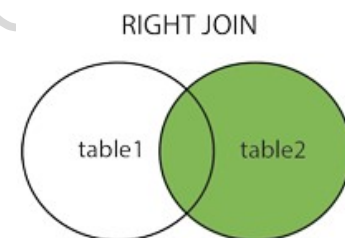
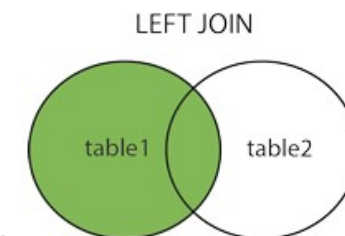
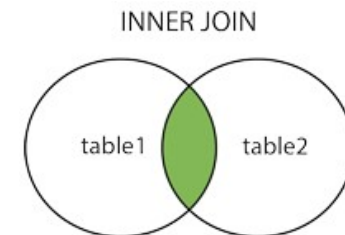
Interrogation de plusieurs tables

- Jointures
 - INNER JOIN
 - OUTER LEFT JOIN
 - OUTER RIGHT JOIN
 - FULL JOIN
- Opérateurs d'ensemble
 - UNION
 - INTERSECT
 - MINUS
- Sous-requêtes
 - une instruction SELECT à l'intérieur d'une autre instruction SELECT

Dans ce cours on va se focaliser sur INNER JOIN

La clause JOIN

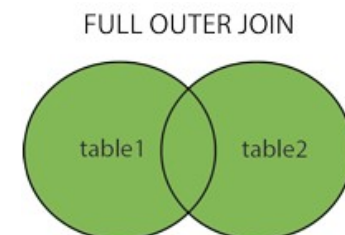
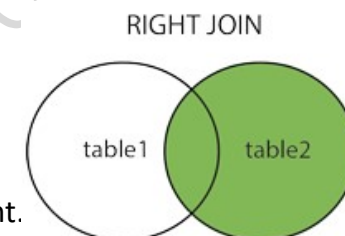
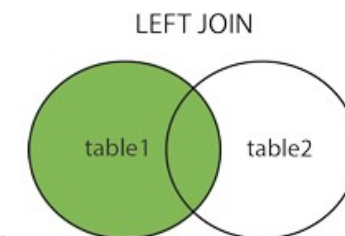
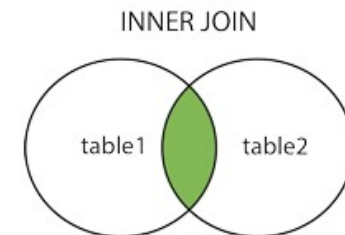
- La clause JOIN en SQL est utilisée pour combiner les lignes de deux tables ou plus, en se basant sur un champ commun entre elles.
- Il existe 4 types d'opérations JOIN :
 - **INNER JOIN** retourne tous les enregistrements lorsqu'il y a au moins une correspondance dans les deux tables.
 - **LEFT JOIN** retourne tous les enregistrements de la table de gauche (première table) et les valeurs correspondantes des enregistrements de la table de droite (deuxième table).
 - Le résultat est NULL du côté droit lorsqu'il n'y a aucune valeur correspondante pour les enregistrements de la deuxième table.
 - **RIGHT JOIN** retourne tous les enregistrements de la table de droite (deuxième table) et les valeurs correspondantes des enregistrements de la table de gauche (première table).
 - **FULL JOIN** retourne toutes les lignes lorsqu'il y a une correspondance dans UNE des tables.



La clause JOIN

Exemples :

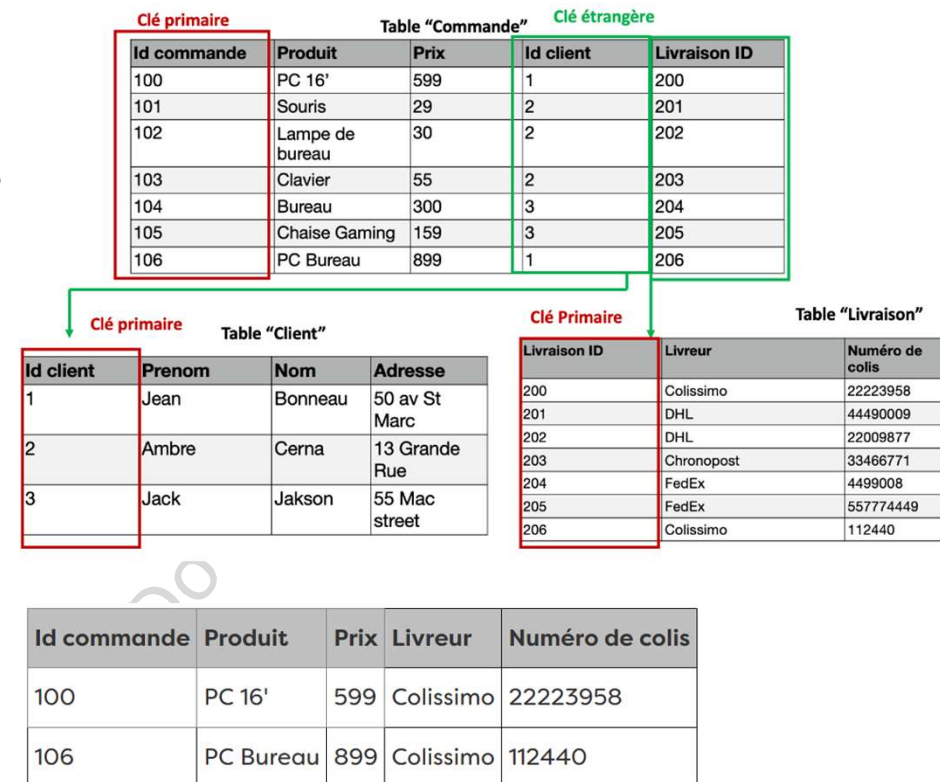
- INNER JOIN
 - Sélectionnez tous les départements qui ont au moins un employé qui y est affecté.
- LEFT JOIN
 - Sélectionnez tous les départements, y compris ceux qui n'ont aucun employé qui y est affecté.
 - (Départements (gauche) et Employés (droite))
- RIGHT JOIN
 - Sélectionnez tous les employés, y compris ceux qui ne sont pas affectés à un département.
- FULL JOIN
 - Sélectionnez tous les départements et employés, même si le département n'a aucun employé qui lui est affecté et si l'employé n'est affecté à aucun département.



La clause JOIN

- Exemple:
 - on veut connaître la liste de toutes les commandes effectuées par Jean Bonneau avec ses informations de livraison.
 - Pour cela on va réaliser une jointure sur les trois tables. Cette jointure va utiliser les clés étrangères « Livraison ID » et « Id Client » qui mettent en relation les tables.

```
SELECT IdCommande, produit, prix, livreur, numero_de_colis
FROM (Commande INNER JOIN Client ON Commande.IdClient = Client.IdClient)
INNER JOIN Livraison ON Commande.LivraisonId = Livraison.LivraisonId
WHERE IdClient = 1
```



La clause JOIN

Le type de jointure le plus courant est :
SQL INNER JOIN
(jointure simple).

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10365	3	3	1996-11-27	2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	1996-09-18
10365	Antonio Moreno Taquería	1996-11-27

La clause JOIN

- Table *loan*
(*Prêt individuel*)
- Table *borrower*
(*Emprunteur de prêt individuel pour véhicule*)

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

INNER JOIN

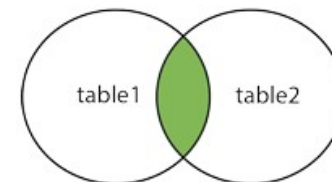
<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>	<i>loan-number</i>
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230

SELECT * FROM loan

INNER JOIN borrower

ON loan.loan-number = borrower.loan-number;

INNER JOIN



<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

LEFT OUTER JOIN

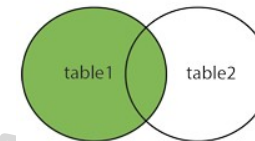
<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>	<i>loan-number</i>
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-260	Perryridge	1700	<i>null</i>	<i>null</i>

SELECT * FROM loan

LEFT OUTER JOIN borrower

ON loan.loan-number = borrower.loan-number;

LEFT JOIN



<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

RIGHT OUTER JOIN

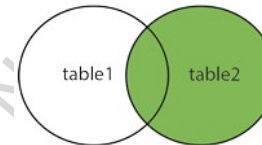
<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>	<i>loan-number</i>
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-155	null	null	Hayes	L-155

SELECT * FROM loan

RIGHT OUTER JOIN borrower

ON loan.loan-number = borrower.loan-number;

RIGHT JOIN



<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155

FULL JOIN

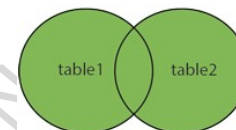
SELECT * FROM loan

FULL OUTER JOIN borrower

ON loan.loan-number = borrower.loan-number;

<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>	<i>customer-name</i>	<i>loan-number</i>
L-170	Downtown	3000	Jones	L-170
L-230	Redwood	4000	Smith	L-230
L-260	Perryridge	1700	<i>null</i>	L-260
L-155	<i>null</i>	<i>null</i>	Hayes	L-155

FULL OUTER JOIN



<i>loan-number</i>	<i>branch-name</i>	<i>amount</i>
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

<i>customer-name</i>	<i>loan-number</i>
Jones	L-170
Smith	L-230
Hayes	L-155