01/04/2023
Version 4

BDD

JÉRÉMY PERROUAULT

SGBDR

GESTION BASE DE DONNÉES

SGBDR

- **S**ystèmes de **G**estion de **B**ases de **D**onnées **R**elationnelles
- Outil pour
  - Structurer
  - Stocker
  - Interroger
  - Garantir l'**intégrité** des données
- Processus actif
  - Accessible via un port de communication spécifique
  - Rôle de « serveur »
- Utilisation d'un client et du langage **SQL** pour interagir avec ce système
  - **S**tructured **Q**uery **L**anguage

SGBDR

DDL	DML	DCL	TCL
Data Definition Langage	Data Manipulation Langage	Data Control Langage	Transaction Control Langage
Définition de structure	Manipulations des données	Sécurisation des données	Gestion des transactions
CREATE ALTER DROP TRUNCATE COMMENT RENAME	SELECT INSERT UPDATE DELETE CALL LOCK TABLE	GRANT REVOKE	SET TRANSACTION COMMIT ROLLBACK SAVEPOINT

SGBDR

- Quelques serveurs
  - MySQL
  - MariaDB
  - Oracle
  - PostgreSQL
  - Microsoft SQL Server
  - SQLite
  - …

SGBDR

- Un **SGBDR** peut gérer plusieurs bases de données
- Une base de données peut contenir plusieurs tables
  - Un ensemble d'**entités**, ou d'**enregistrements**
- Une table possède plusieurs colonnes
  - Un **attribut**
- Chaque enregistrement est identifié grâce à une clé primaire
- On peut créer un lien entre enregistrements grâce à la clé étrangère
- Un tuple est une construction théorique d'un ensemble d'attributs
  - Le résultat d'une sélection

SGBDR

- Dans la base de données, voici la table **client**

- Les colonnes sont
  - `ID`, `NOM` et `PRENOM`

- Une clé primaire est un élément obligatoire
  - Ici, la colonne `ID` est la clé primaire de la table

ID	NOM	PRENOM
1	PERROUAULT	Jérémy
2	PERROUAULT	Alissa
3	CESBRON	Martin

SGBDR

ID	NOM	PRENOM
1	PERROUAULT	Jérémy
2	PERROUAULT	Alissa
3	CESBRON	Martin

ID_CLIENT	ID_PRODUIT
1	1
1	2
2	1

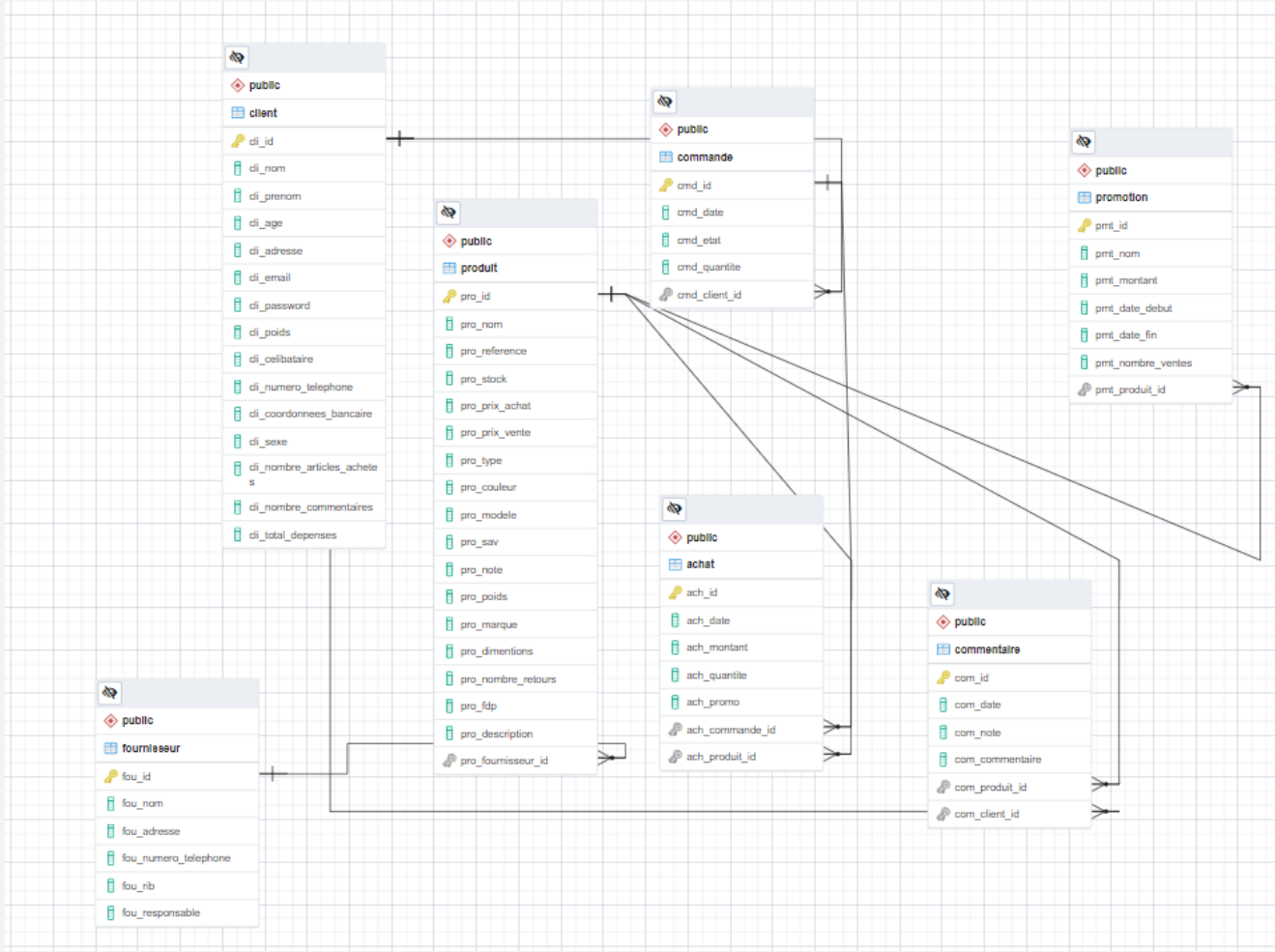
ID	LIBELLE	PRIX
1	GoPRO HERO 10	429.99
2	Parachute de France	6999.99

Dans cet exemple
Les ID sont des clés primaires
ID_CLIENT et ID_PRODUIT sont
- Des clés primaires
- Des clés étrangères

SGBDR

- Plusieurs tables
  - Plusieurs enregistrements
- Chaque table peut être liée
  - À une ou plusieurs autres tables
  - À elle-même

- Un enregistrement est identifié
  - De manière unique, clé primaire
- Un enregistrement se lie à un autre
  - Via une clé étrangère
    - Qui fait référence à une clé primaire



DML

INTERROGATION DES DONNÉES

INTERROGATION DES DONNÉES

- Pour communiquer avec un serveur **SGBD**
  - Un client **SGBD**
    - Un client en ligne de commande
    - Un client graphique
    - Un client « connecteur » dans un univers de développement (**JAVA**, **C#**, **C**, **PHP**, **PYTHON**, etc.)
  - Un langage
    - **SQL** (**S**tructured **Q**uery **L**anguage)
- Chaque requête (`Query`) est un ordre (commande) adressé au **SGBD**
  - On peut en donner plusieurs en une connexion, séparées par un « ; » (point-virgule)
  - Permet
    - D'extraire des informations
    - D'ajouter, modifier, supprimer
    - D'administrer la base de données (ajouter des tables, des bases de données, gérer les utilisateurs, etc.)

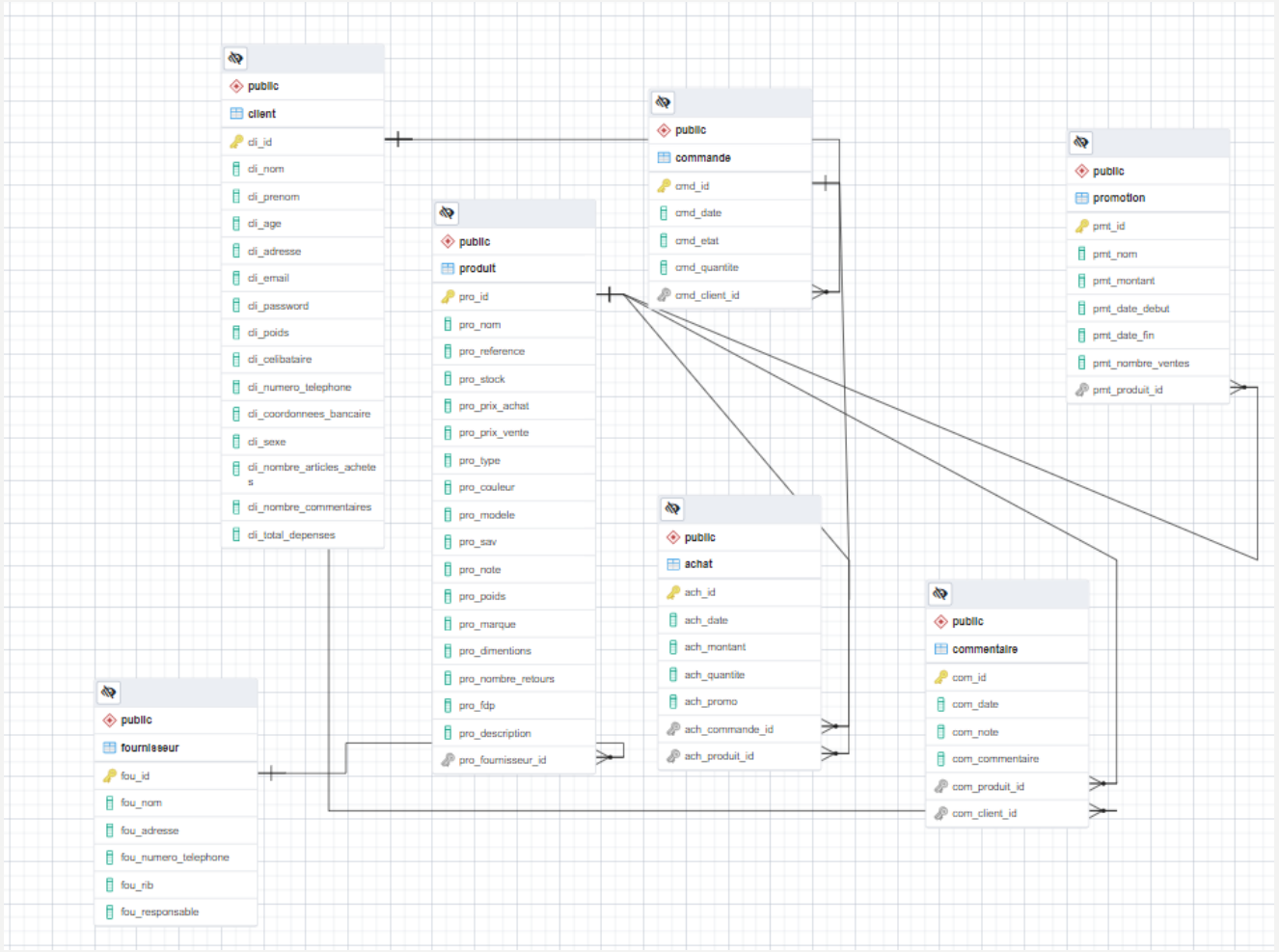
INTERROGATION DES DONNÉES

- Langage **DML** (**D**ata **M**anipulation **L**anguage)
  - Langage d'interrogation et de manipulation des données

- **CRUD** (**C**REATE, **R**EAD, **U**PDATE, **D**ELETE)
  - `INSERT INTO` (**C**)
    - Ajouter des données
  - `SELECT` (**R**)
    - Sélectionner des données
  - `UPDATE` (**U**)
    - Mettre à jour des données
  - `DELETE` (**D**)
    - Supprimer des données

INTERROGATION DES DONNÉES

- Importance de bien
- connaître la structure
- (ou partie) à interroger



INTERROGATION DES DONNÉES

- Le SELECT est la commande **SQL** de base
  - Permet d'extraire des données d'une ou plusieurs bases, une ou plusieurs tables
  - Avec éventuellement des calculs, des transformations, des regroupements, etc.

```
SELECT [DISTINCT ou ALL] *|colonne(s) [AS alias(es)]
FROM la_table|la_vue [alias]
[INNER|LEFT|RIGHT [OUTER] JOIN la_table_bis [alias] ON condition(s)]
[WHERE prédicat(s)]
[GROUP BY groupe(s)]
[HAVING condition(s)]
[ORDER BY colonne(s)]
[OFFSET start]
[LIMIT [start, ] nbr]
```

INTERROGATION DES DONNÉES

- `SELECT`
  - Spécifie les colonnes du résultat, et éventuellement leur nom aliasé
- `FROM`
  - Spécifie la ou les tables dans lesquelles rechercher / extraire les informations
- `INNER|LEFT|RIGHT [OUTER] JOIN`
  - Spécifique la ou les jointures
  - Une syntaxe avec `OUTER` existe, souvent optionnelle, il n'y a aucune différence
- `WHERE`
  - Filtre sur des informations (condition(s) ou groupe(s) de conditions à remplir)
- `HAVING`
  - Filtre sur des opérations de regroupement
- `ORDER BY`
  - Trie les résultats
- `LIMIT`
  - Limiter les résultats

INTERROGATION DES DONNÉES

- SELECT

```
SELECT [DISTINCT] colonne1, colonne2 FROM ma_table
```

```
SELECT * FROM ma_table;
SELECT DISTINCT col1, col2 FROM ma_table;
```

- SELECT et ALIAS

```
SELECT colonne1 AS COL1, colonne2 AS COL2 FROM ma_table t
```

EXERCICE

- Sélectionner tous les clients
- Sélectionner le nom et le prix de tous les produits
- Sélectionner toutes les commandes

INTERROGATION DES DONNÉES

- On peut effectuer des opérations / fonctions sur les sélections
  - `CONCAT` ou || (double pipe)
    - Concaténer des informations
      - `col1||' '||col2`
      - `CONCAT(col1, ' ', col2)`
  - `IFNULL` (**MySQL**) | `COALESCE` (**PostgreSQL**) | `ISNULL` (**MSSQL**) | `NVL` (**Oracle**)
    - Remplacer une valeur **NULL** par une autre valeur de remplacement
    - `IFNULL`(Expression testée, Expression de remplacement si **NULL**)
  - `NOW()`
    - Date & heure d'aujourd'hui
  - `DATEDIFF`
    - Calculer une différence entre deux dates
  - `DATEPART` | `DATE_PART` (**PostgreSQL**)
    - Extraire une partie de la date (jour, mois, année, etc.)
  - `UPPER` / `LOWER`

EXERCICE

- Sélectionner tous les nom & prénom des clients dans une seule colonne
  - L'afficher en majucules
- Sélectionner tous le mois et l'année des commandes (mois-année)
  - Si la date est **NULL**, afficher « 0 »

INTERROGATION DES DONNÉES

- SELECT et restriction WHERE

Signification	Opérateur
Egal à	=
Différent de	!= (ou <>)
Strictement supérieur à	>
Supérieur ou égal à	>=
Strictement inférieur à	<
Inférieur ou égal à	<=
Contient	LIKE '%val%'
Est / N'est pas	IS (ou <=>) / IS NOT { TRUE | FALSE | NULL }
Dans une liste	IN
Entre	BETWEEN

```
SELECT colonne1, colonne2
FROM ma_table
WHERE colonne1 = 'valeur'
```

```
SELECT colonne1, colonne2
FROM ma_table
WHERE
  (colonne1 = 'valeur' OR colonne1 = 'valeur 2')
  AND colonne2 = 'valeur 3'
```

Type de logique	Opérateur
ET	AND
OU	OR
NON	NOT

EXERCICE

- Sélectionner le client ID 1
- Sélectionner le produit dont le nom est égal à « Casque Tonfly »
- Sélectionner les produits contenant « a »
- Sélectionner les produits commençant par « C »
- Sélectionner le client ID 1 ET le client ID 2
  - Avec `AND`/`OR` et `IN`
- Sélectionner les commandes qui n'ont pas de date

INTERROGATION DES DONNÉES

- ORDER BY
  - Permet de ranger les informations par ordre croissant ou décroissant

```sql
SELECT colonne1, colonne2
FROM ma_table
ORDER BY
  colonne1 ASC,
  colonne2 DESC
```

```sql
SELECT colonne1, colonne2
FROM ma_table
ORDER BY
  1 ASC,
  2 DESC
```

EXERCICE

- Sélectionner tous les clients par ordre alphabétique (Prénom puis Nom)
- Sélectionner les produits dont le prix d'achat est entre 100 et 1000 euros, rangés par prix de vente décroissant

INTERROGATION DES DONNÉES

- LIMIT
  - Permet de sélectionner une rangée d'informations

```
SELECT colonne1, colonne2
FROM ma_table
LIMIT start, nbr
```

```
SELECT colonne1, colonne2
FROM ma_table
OFFSET start LIMIT nbr
```

  - Les 30 premiers

```
SELECT colonne1, colonne2
FROM ma_table
LIMIT 0, 30
```

```
SELECT colonne1, colonne2
FROM ma_table
OFFSET 0 LIMIT 30
```

  - Les 30 suivants

```
SELECT colonne1, colonne2
FROM ma_table
LIMIT 30, 30
```

```
SELECT colonne1, colonne2
FROM ma_table
OFFSET 30 LIMIT 30
```

EXERCICE

- Sélectionner les 2 produits les moins cher (prix de vente)
- Sélectionner les 2 suivants

INTERROGATION DES DONNÉES

Jointures

ID	LIBELLE	PRIX
1	GoPRO HERO 5	429.99
2	GoPRO KARMA	699.99

ID	NOM	PRENOM
1	PERROUAULT	Jérémy
2	PERROUAULT	Alissa
3	CESBRON	Martin

C'est un produit cartésien des deux tables

ID	LIBELLE	PRIX	ID	NOM	PRENOM
1	GoPRO HERO 5	429.99	1	PERROUAULT	Jérémy
2	GoPRO KARMA	699.99	1	PERROUAULT	Jérémy
1	GoPRO HERO 5	429.99	2	PERROUAULT	Alissa
2	GoPRO KARMA	699.99	2	PERROUAULT	Alissa
1	GoPRO HERO 5	429.99	3	CESBRON	Martin
2	GoPRO KARMA	699.99	3	CESBRON	Martin

EXERCICE

- Sélectionner les achats du client 1
  - Avec les informations du client
- Sélectionner les produits ID 1 achetés
  - Avec les informations du produit

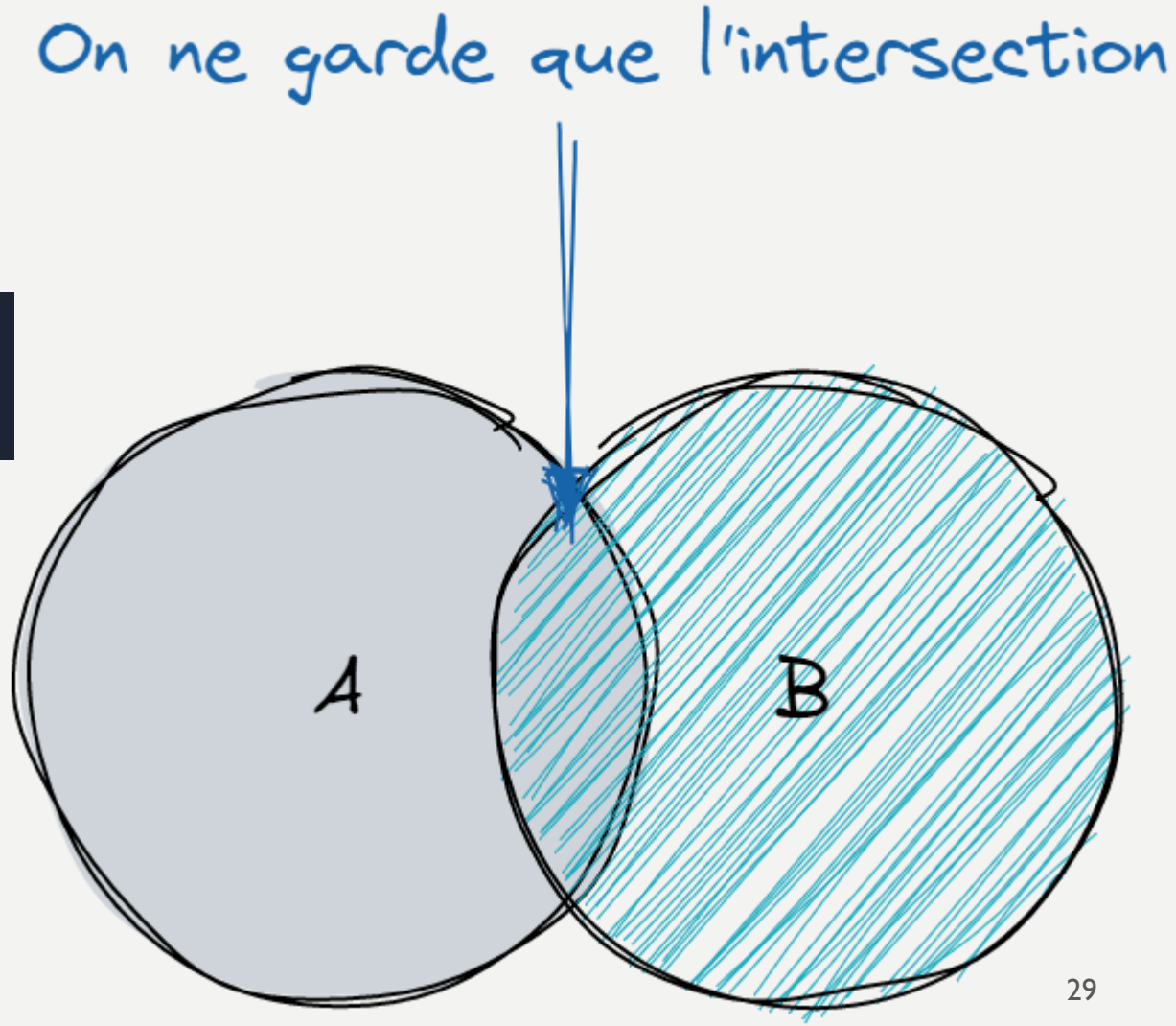
INTERROGATION DES DONNÉES

- Jointures
  - `INNER JOIN` (`JOIN`)
  - `LEFT JOIN` (`LEFT OUTER JOIN`)
  - `RIGHT JOIN` (`RIGHT OUTER JOIN`)
  - `FULL JOIN` (`FULL OUTER JOIN`)

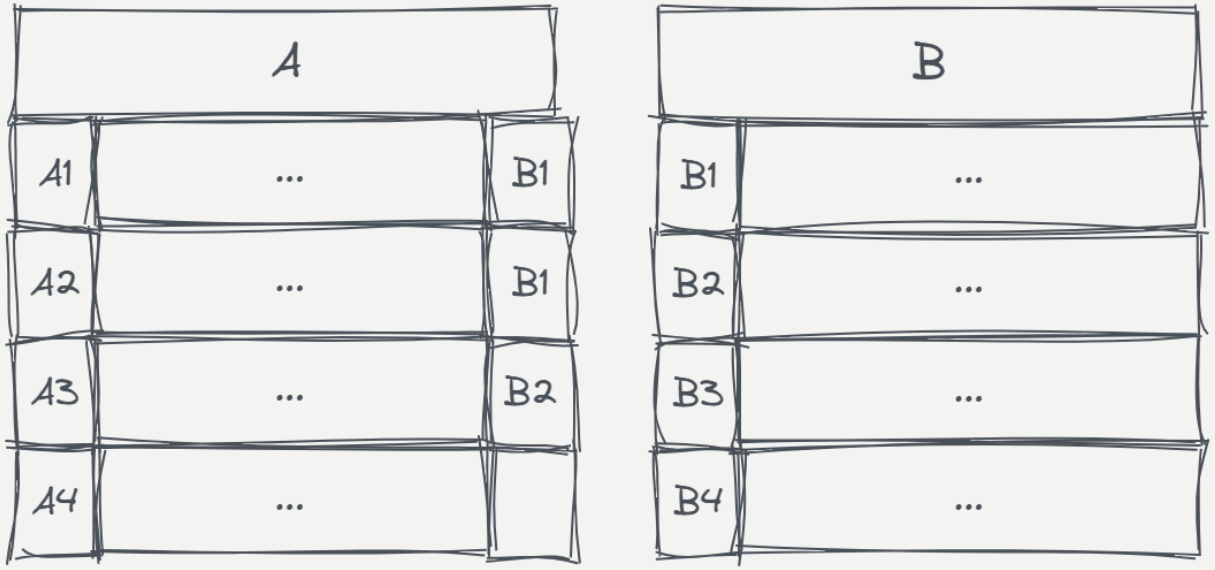
INTERROGATION DES DONNÉES

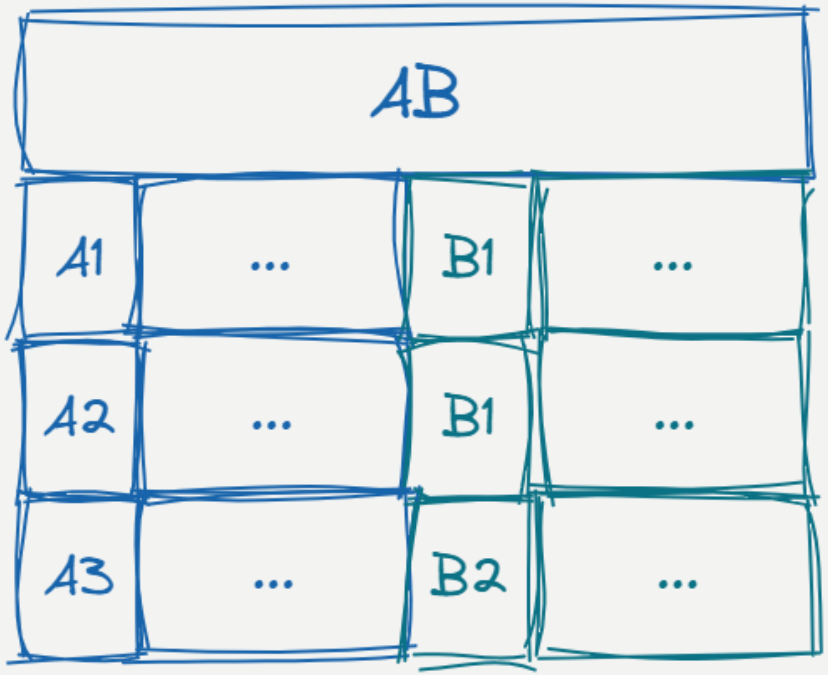
INNER JOIN (JOIN)

```sql
SELECT colonne1, colonne2
FROM table1 a
INNER JOIN table2 b ON b.col = a.col
```



INTERROGATION DES DONNÉES

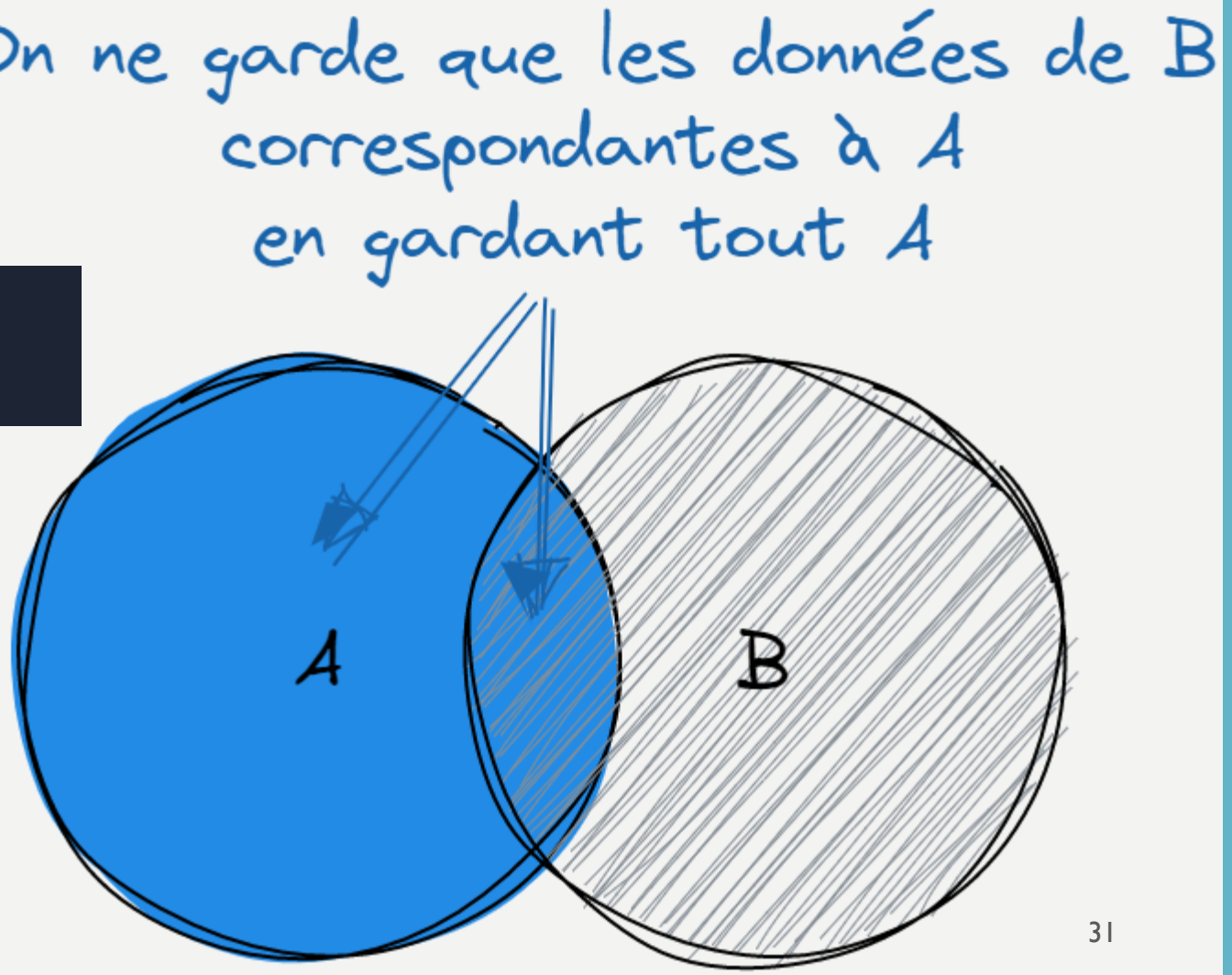




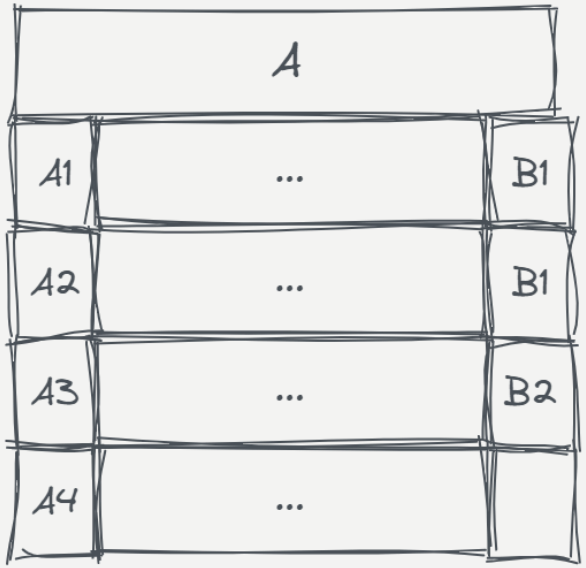
INTERROGATION DES DONNÉES

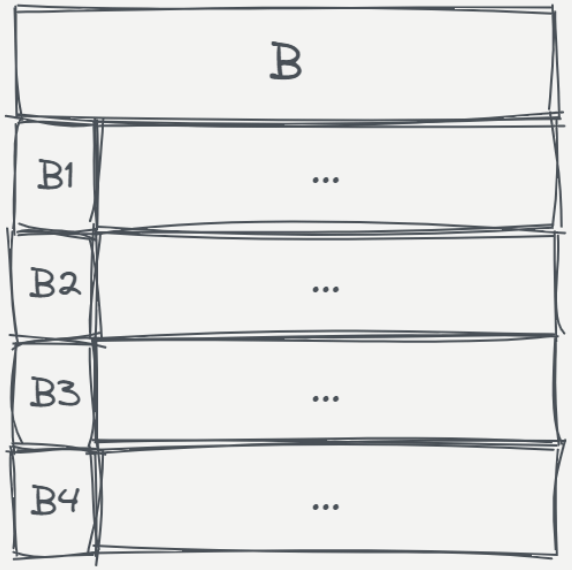
LEFT JOIN (LEFT OUTER JOIN)

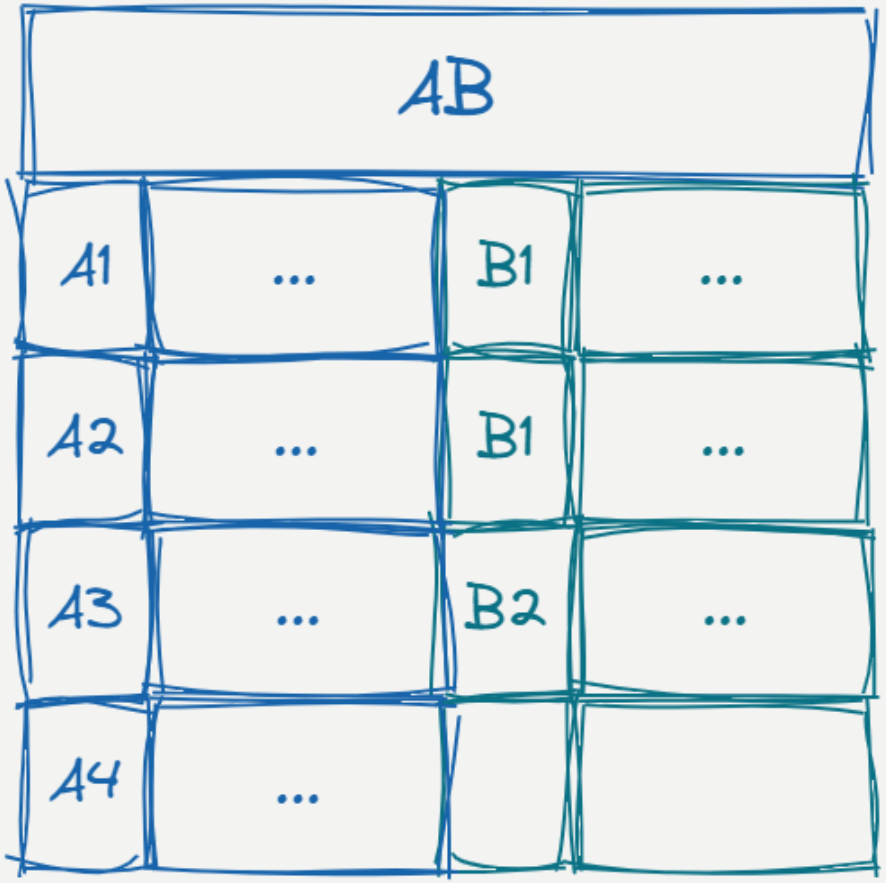
```
SELECT colonne1, colonne2
FROM table1 a
LEFT JOIN table2 b ON b.col = a.col
```



INTERROGATION DES DONNÉES





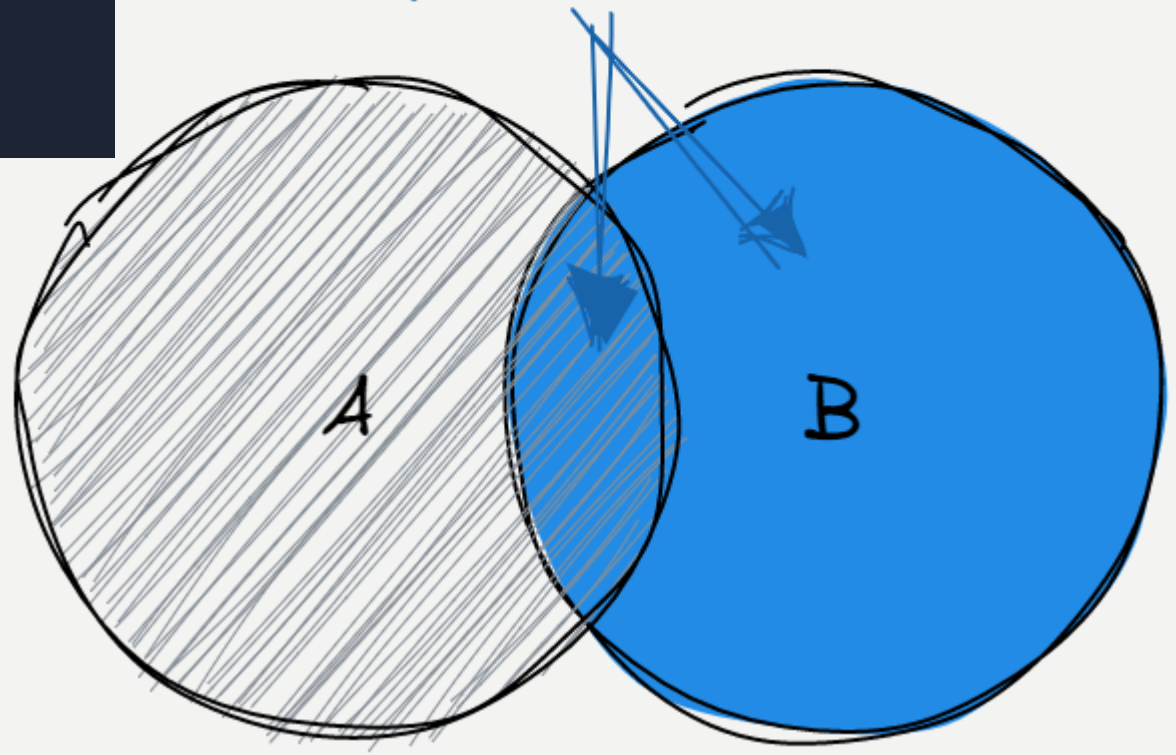


INTERROGATION DES DONNÉES

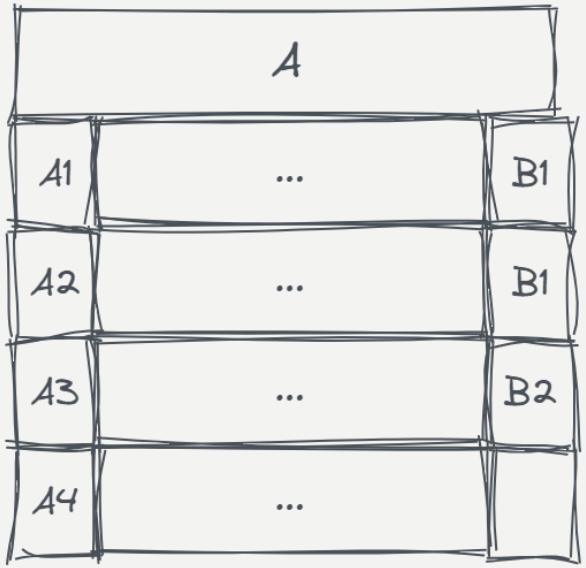
RIGHT JOIN (RIGHT OUTER JOIN)

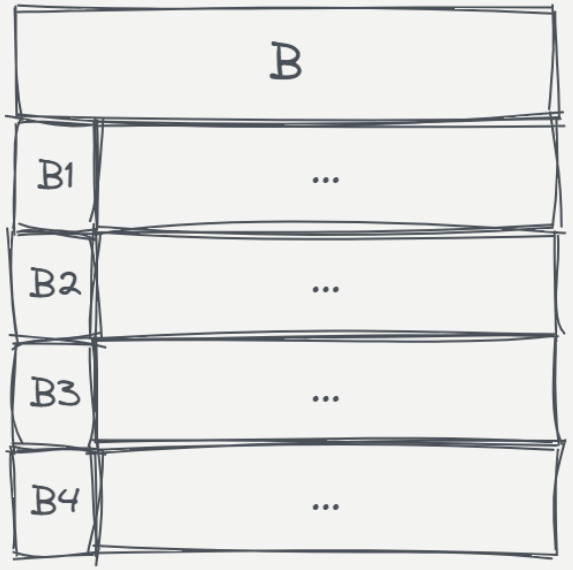
On ne garde que les données de A correspondantes à B en gardant tout B

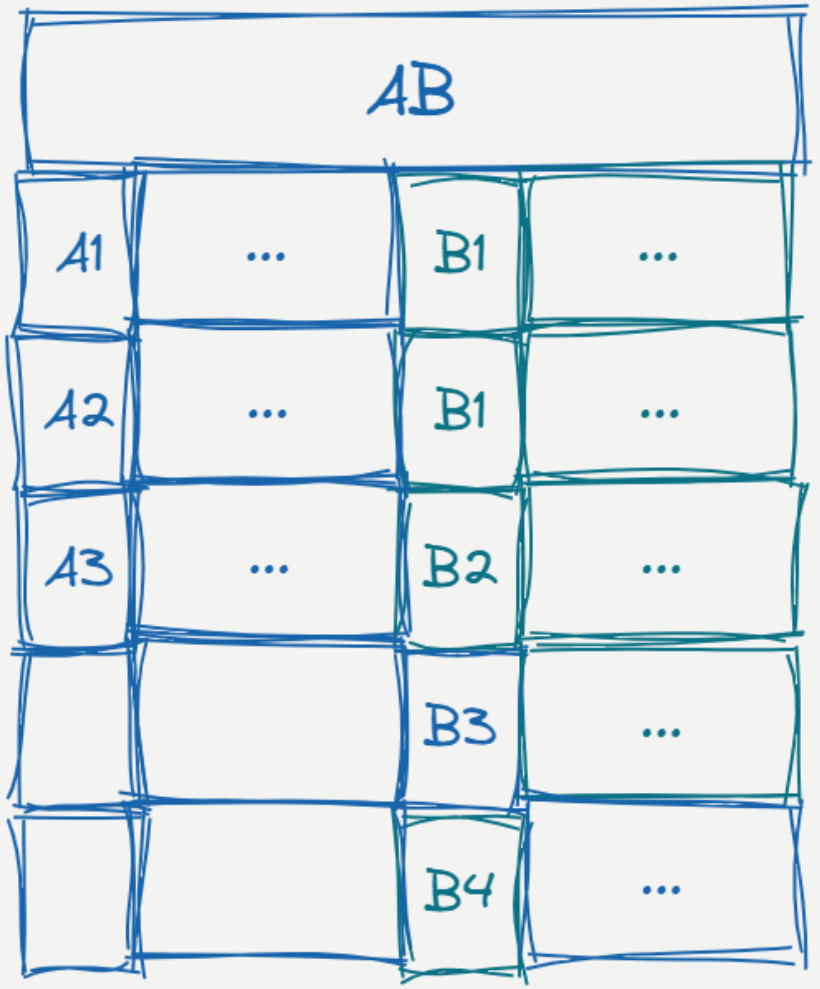
```
SELECT colonne1, colonne2
FROM table1 a
RIGHT JOIN table2 b ON b.col = a.col
```



INTERROGATION DES DONNÉES





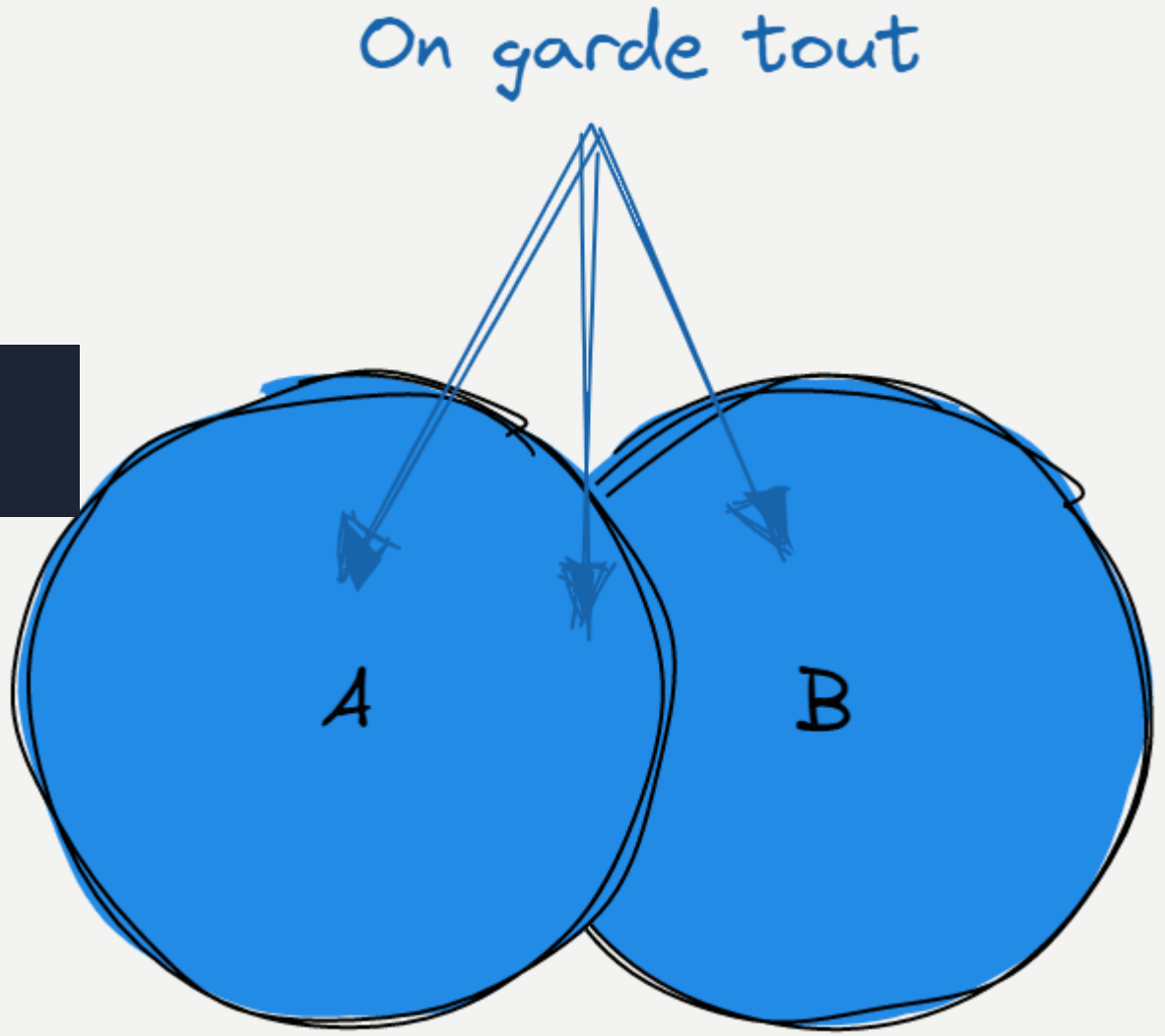


INTERROGATION DES DONNÉES

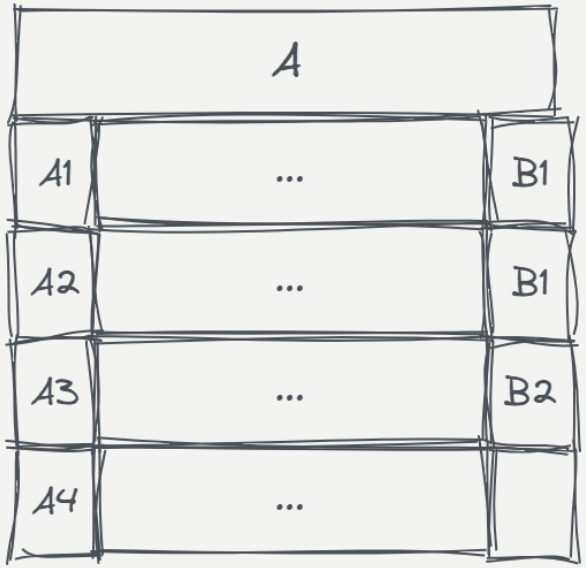
FULL JOIN (FULL OUTER JOIN)

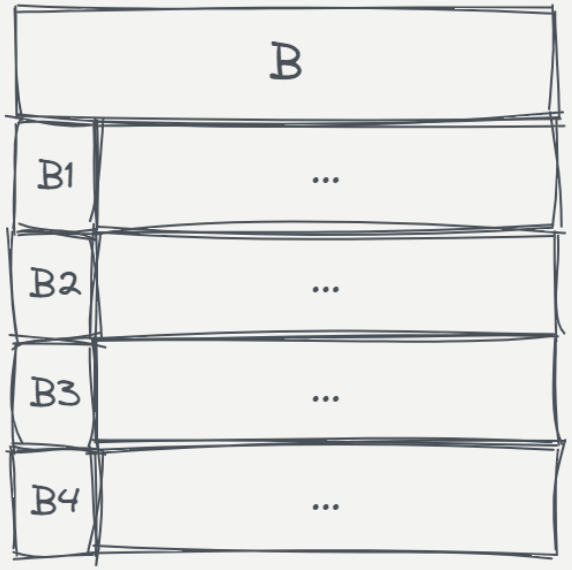
Ne fonctionne pas sous MySQL

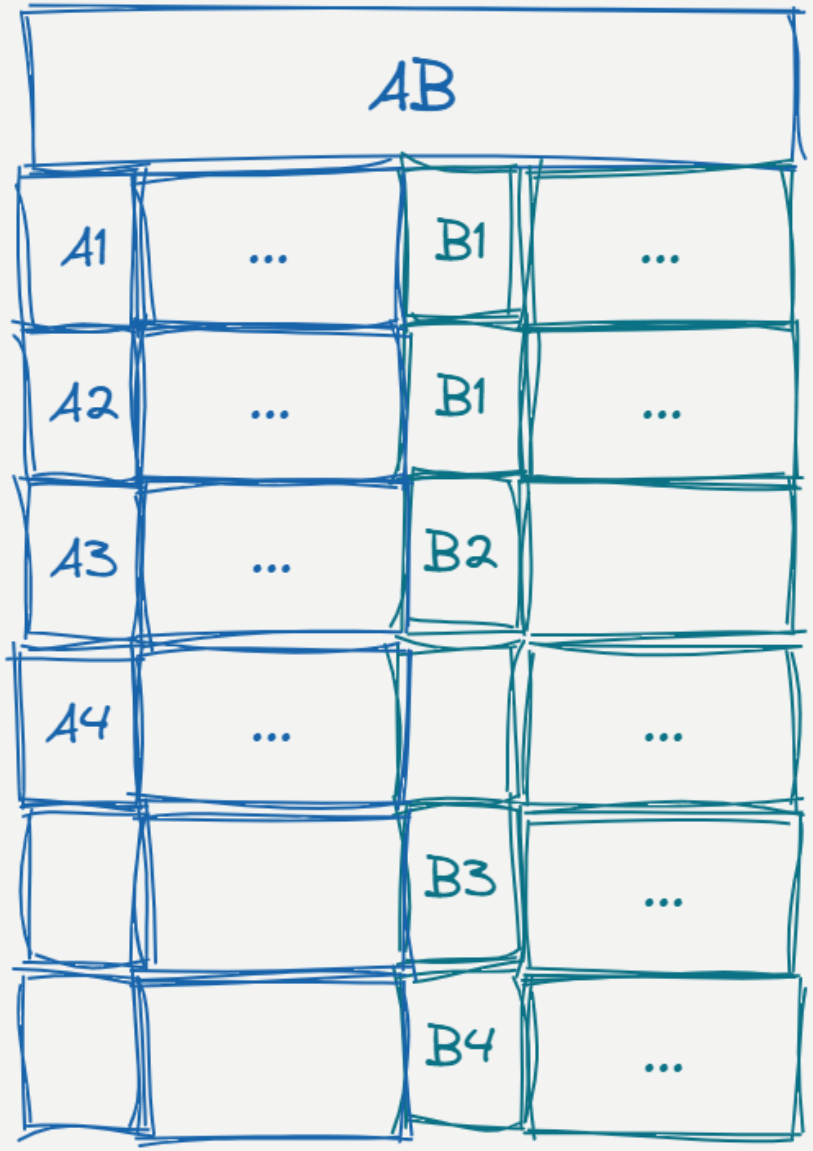
```
SELECT colonne1, colonne2
FROM table1 a
FULL JOIN table2 b ON b.col = a.col
```



INTERROGATION DES DONNÉES







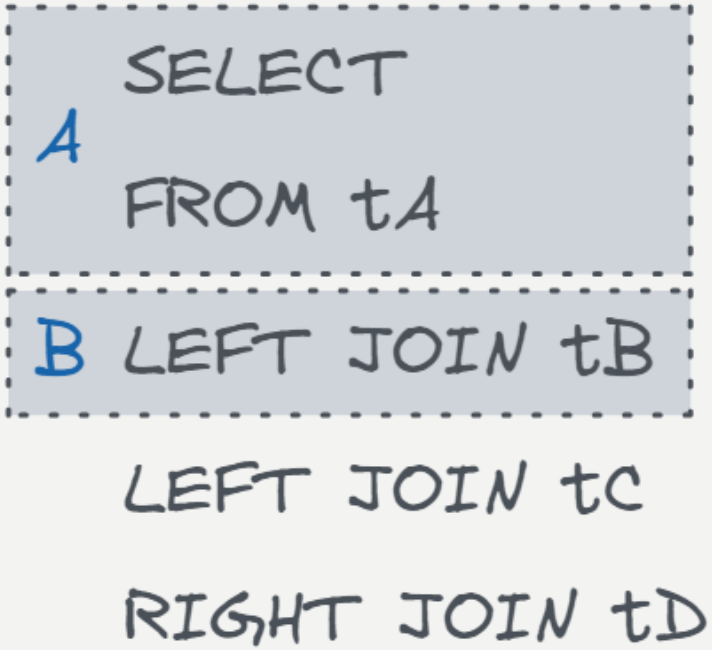
INTERROGATION DES DONNÉES

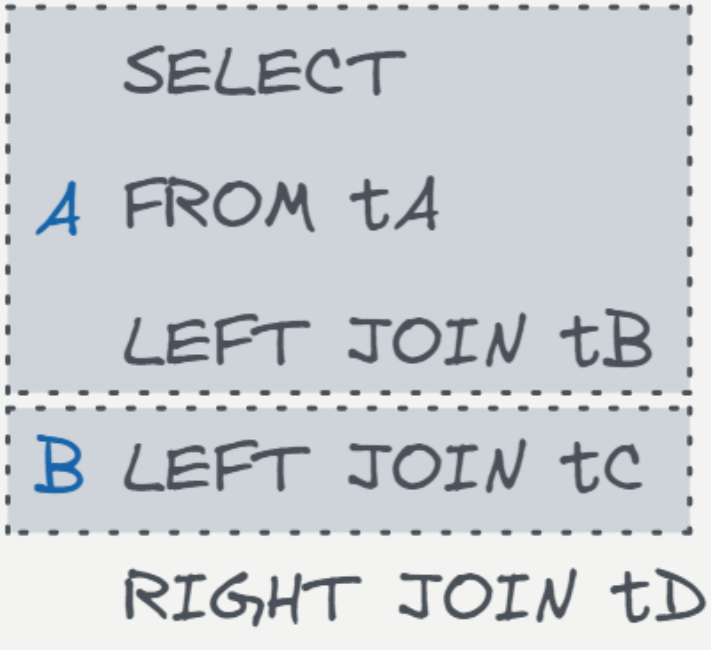
- Lorsqu'on enchaine plusieurs jointures
  - La table est combinée à la première jointure
  - Le résultat de la première jointure est combinée à la seconde
  - Le résultat de la seconde jointure est combinée à la troisième
  - etc.. Comme s'il n'y avait au final, qu'une seule jointure (chaque jointure devient un groupe, un sous-ensemble)

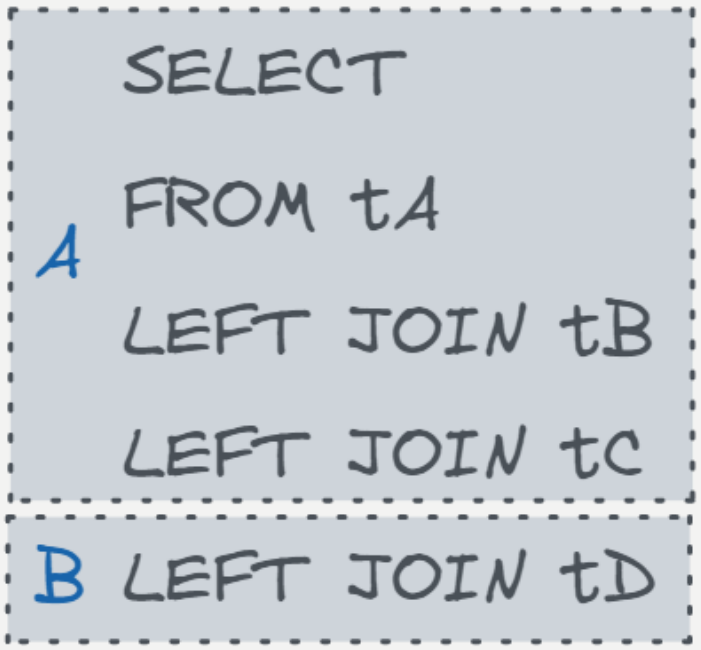
```
SELECT a.colonne1, a.colonne2
FROM table1 a
LEFT JOIN table2 b ON b.col = a.col
LEFT JOIN table3 c ON c.col = a.col
LEFT JOIN table4 d ON d.col = b.col

WHERE a.colonne2 = 'valeur'
```

INTERROGATION DES DONNÉES

Le résultat de requête
se calcule progressivement







EXERCICE

- Sélectionner les achats
  - Avec les informations du client
  - Avec les informations du produit
- Sélectionner tous les clients, et leurs achats
  - Avec les informations du produit, si disponible
- Sélectionner uniquement les clients ayant effectuées des achats
  - Sans les informations du produit

EXERCICE

- Sélectionner les 2 derniers achats
  - Avec les informations client, produit, fournisseur et éventuellement sa note

INTERROGATION DES DONNÉES

- Fonctions d'agrégation
  - `AVG()`
    ```
    SELECT AVG(colonne1) FROM ma_table
    ```
    - Moyenne d'une colonne
  - `SUM()`
    ```
    SELECT SUM(colonne1) FROM ma_table
    ```
    - Somme d'une colonne
  - `MIN()`
    ```
    SELECT MIN(colonne1) AS minimum FROM ma_table
    ```
    - Minimum d'une colonne
  - `MAX()`
    ```
    SELECT MAX(colonne1) AS maxCol FROM ma_table
    ```
    - Maximum d'une colonne
  - `COUNT()`
    ```
    SELECT COUNT(colonne1) FROM ma_table
    ```
    - Compter le nombre (selon une colonne)

INTERROGATION DES DONNÉES

- Fonctions
  - `GROUP BY`
    - Permet de regrouper par colonne
  - `HAVING`
    - Remplace la clause `WHERE` dans le cas de restriction sur opérations résultant de regroupements

EXERCICE

- Sélectionner tous les clients (nom, prénom) et leur CA
  - Ranger les informations par CA décroissant
- Sélectionner uniquement les clients dont le CA est > 1100
- Sélectionner les clients dont le CA est > 0, en ne prenant en compte que les produits qui ont un commentaire (réalisé ou pas par le client)
- Sélectionner les clients dont le CA est > 0, en ne prenant en compte que les produits qui ont une note > 4

INTERROGATION DES DONNÉES

- Possible de créer des requêtes imbriquées « sous-requêtes »
  - Comme une table (FROM, JOIN, …)

```
SELECT tb.colonne3 FROM (SELECT colonne3, colonne4 FROM ma_table) tb
```

  - Comme un champ – Ne doit retourner qu'un seul champ & valeur

```
SELECT
  colonne1,
  (
    SELECT MIN(colonne2)
    FROM ma_table2
  ) AS col
FROM ma_table t
```

  - Dans une clause WHERE – Ne doit retourner qu'un seul champ

```
SELECT colonne1
FROM ma_table t
WHERE colonne2 IN (SELECT colonne3 FROM ma_table2)
```

EXERCICE

- Sélectionner tous les clients et leur CA
  - Uniquement ceux dont le CA est compris entre 500 et 1500 euros
- Compter le nombre de clients dont le CA est supérieur à 1100 euros
- Sélectionner tous les clients et le nombre de produits uniques achetés

EXERCICE

- Sélectionner les clients
  - Le prix minimum d'un produit acheté, et son libellé
  - Le prix maximum d'un produit acheté, et son libellé
  - Son panier moyen
  - Sa première date d'achat
  - Sa dernière date d'achat

INTERROGATION DES DONNÉES

- Avec des opérateurs ensemblistes (algèbre relationnelle)
  - `UNION`
  - `INTERSECT` (pas implémenté sur tous les **SGBD**)
  - `EXCEPT` | `MINUS` (pas implémenté sur tous les **SGBD**)
  - S'utilisent entre deux clauses `SELECT`

```
SELECT col1, col2 FROM table1
UNION
SELECT col3, col4 FROM table2
```

INTERROGATION DES DONNÉES

- `UNION [ALL]`
  - Permet d'assembler deux résultats
  - Par défaut, les doublons sont éliminés
    - Possible des les garder avec la clause **UNION ALL**

```
SELECT col1, col2 FROM table1
UNION [ALL]
SELECT col3, col4 FROM table2
```

INTERROGATION DES DONNÉES

- INTERSECT
  - Permet de ne garder que les tuples identiques entre deux résultats

```
SELECT col1, col2 FROM table1
INTERSECT
SELECT col3, col4 FROM table2
```

- N'existant pas sur tous les **SGBD**, possible de le remplacer par une sous-requête

```
SELECT col1, col2 FROM table1
WHERE EXISTS (SELECT col3, col4 FROM table2 WHERE col1 = col3 AND col2 = col4)
```

INTERROGATION DES DONNÉES

- EXCEPT ou MINUS (**ORACLE**)
  - Permet de ne garder que les tuples de la première clause n'existant pas dans la deuxième clause

```
SELECT col1, col2 FROM table1
EXCEPT
SELECT col3, col4 FROM table2
```

- N'existant pas sur tous les **SGBD**, possible de le remplacer par une sous-requête

```
SELECT col1, col2 FROM table1
WHERE NOT EXISTS (SELECT col3, col4 FROM table2 WHERE col1 = col3 AND col2 = col4)
```

DML

MANIPULATION DES DONNÉES

MANIPULATION DES DONNÉES

- INSERT INTO

```sql
INSERT INTO ma_table (colonne1, colonne2) VALUES ('valeur 1', 'valeur 2');
```

MANIPULATION DES DONNÉES

- UPDATE

```
UPDATE ma_table
SET
  colonne1 = 'valeur 1',
  colonne2 = 'valeur 2'
WHERE colonne = 'valeur'
```

MANIPULATION DES DONNÉES

- DELETE

```
DELETE FROM ma_table
WHERE colonne = 'valeur'
```

DDL

STRUCTURE DES DONNÉES

STRUCTURE DES DONNÉES

- Langage **DDL** (**D**ata **D**efinition **L**anguage – **D**ata **D**escription **L**anguage)
- Langage de structuration des données

STRUCTURE DES DONNÉES

- CREATE
  - Création d'un élément de structure
- ALTER
  - Modification d'un élément de structure
- DROP
  - Suppression d'un élément de structure

STRUCTURE DES DONNÉES

- `CREATE DATABASE`
  - Créer une base de données
- `CREATE TABLE`
  - Créer une table
- `ALTER TABLE`
  - Modifier une table
    - Ajouter, modifier, supprimer une colonne
    - Ajouter, supprimer un index
- `DROP TABLE`
  - Supprimer une table
- `TRUNCATE TABLE` (pas implémentée sur tous les **SGBD**)
  - Vider une table

STRUCTURE DES DONNÉES

- CREATE DATABASE

```
CREATE DATABASE nom_db COLLATE utf8_general_ci
```

- SHOW DATABASES

```
SHOW databases
```

```
SELECT datname FROM pg_database
```

- USE DATABASE

```
USE nom_db
```

- SHOW TABLES

```
SHOW tables
```

```
SELECT *
FROM pg_catalog.pg_tables
WHERE schemaname != 'pg_catalog' AND
  schemaname != 'information_schema';
```

STRUCTURE DES DONNÉES

- CREATE TABLE (**MySQL**)

```
CREATE TABLE [nom_db.]matable (
  `nom colonne avec espace ou mot-clé` TYPE OPTIONS,
  id INT NOT NULL AUTO_INCREMENT,
  nom VARCHAR(100) NOT NULL,
  prenom VARCHAR(150) NOT NULL,
  age INT NOT NULL,
  id_parent INT NULL
) [ENGINE = InnoDb];
```

STRUCTURE DES DONNÉES

- CREATE TABLE (**PostgreSQL**)

```sql
CREATE TABLE matable (
  "nom colonne avec espace ou mot-clé" TYPE OPTIONS,
  id SERIAL,
  nom VARCHAR(100) NOT NULL,
  prenom VARCHAR(150) NOT NULL,
  age INT NOT NULL,
  id_parent INT NULL
);
```

STRUCTURE DES DONNÉES

- CREATE TABLE (**MSSQL**)

```
CREATE TABLE matable (
  [nom colonne avec espace ou mot-clé] TYPE OPTIONS,
  id INT NOT NULL IDENTITY,
  nom VARCHAR(100) NOT NULL,
  prenom VARCHAR(150) NOT NULL,
  age INT NOT NULL,
  id_parent INT NULL
);
```

STRUCTURE DES DONNÉES

- ALTER TABLE
  - Ajouter une colonne

```
ALTER TABLE nom_table ADD nom_colonne TYPE OPTIONS AFTER une_colonne;
ALTER TABLE matable ADD CA FLOAT NOT NULL AFTER age;
```

  - Supprimer une colonne

```
ALTER TABLE nom_table DROP nom_colonne;
ALTER TABLE matable DROP CA;
```

  - Modifier une colonne

```
ALTER TABLE nom_table CHANGE nom_colonne nouveau_nom_colonne TYPE OPTIONS;
ALTER TABLE matable CHANGE age age INT(3) NOT NULL;
```

STRUCTURE DES DONNÉES

- DROP TABLE

```
DROP TABLE matable
```

STRUCTURE DES DONNÉES

- Les **index** permettent de stocker dans un arbre les différentes valeurs
  - Les valeurs sont rangées triées
  - Recherche dichotomique

- Obligatoirement utilisés pour
  - Les clés primaires
  - Les clés étrangères

- Plusieurs types d'index
  - `INDEX` Autorise les doublons
  - `UNIQUE` N'autorise pas les doublons
  - `SPACIAL` Objets Géométriques
  - `FULLTEXT` Objets de texte

STRUCTURE DES DONNÉES

- Créer un **index**

```
CREATE UNIQUE INDEX nom_index ON nom_table (colonne1, colonne2)
```

- Supprimer un **index**

```
ALTER TABLE nom_table DROP INDEX nom_index
```

- Voir les **index** d'une table

```
SHOW INDEX FROM nom_table
```

STRUCTURE DES DONNÉES

- Les **contraintes** sont des **index**

- Plusieurs types
  - `PRIMARY KEY` Contrainte de clé primaire
  - `FOREIGN KEY` Contrainte de clé étrangère
  - `CHECK` Définir des règles de validation (valeurs booléennes)

STRUCTURE DES DONNÉES

- Créer une **contrainte** de clé primaire

```
ALTER TABLE nom_table
  ADD
    CONSTRAINT Nom_Index
    PRIMARY KEY (colonne1, colonne2)
```

```
ALTER TABLE matable
  ADD
    CONSTRAINT PK_MATABLE
    PRIMARY KEY (id)
```

STRUCTURE DES DONNÉES

- Créer une contrainte de clé étrangère

```
ALTER TABLE nom_table
  ADD
    CONSTRAINT Nom_Index
    FOREIGN KEY (colonne_table)
    REFERENCES nom_table_reference(colonne_table_reference)
```

```
ALTER TABLE matable
  ADD
    CONSTRAINT FK_ParentEnfants
    FOREIGN KEY (id_parent)
    REFERENCES matable2(id)
```

STRUCTURE DES DONNÉES

- Les contraintes de clés étrangères sont par défaut strict
  - La suppression d'une donnée référencée n'est pas autorisée
  - La modification d'un ID référencé n'est pas autorisé
  - ON ACTION
    - CASCADE
    - RESTRICT
    - SET NULL

```
ALTER TABLE matable
  ADD
    CONSTRAINT FK_ParentEnfants
    FOREIGN KEY (id_parent)
    REFERENCES matable(id)
    ON DELETE CASCADE
    ON UPDATE RESTRICT
```

STRUCTURE DES DONNÉES

- Créer les **contraintes** à la création de la table

```
CREATE TABLE matable (
  id INT NOT NULL AUTO_INCREMENT,
  nom VARCHAR(100) NOT NULL,
  prenom VARCHAR(100) NOT NULL,
  age INT(3) NOT NULL,
  id_parent INT NOT NULL,
  PRIMARY KEY (id),
  INDEX (id_parent),
  UNIQUE (nom)
) ENGINE = InnoDB;
```

```
CREATE TABLE matable (
  id SERIAL,
  nom VARCHAR(100) NOT NULL,
  prenom VARCHAR(100) NOT NULL,
  age INT(3) NOT NULL,
  id_parent INT NOT NULL,
  PRIMARY KEY (id),
  INDEX (id_parent),
  UNIQUE (nom)
);
```