

# 03 – Introduction à SQL

**Infrastructure de données 1**

# Semaine passée

---

- **Pourquoi modéliser**

# Semaine passée

---

- **Pourquoi modéliser** Organiser l'information,  
besoins clients

# Semaine passée

---

- **Pourquoi modéliser** Organiser l'information, besoins clients
- **Types de modèles**

# Semaine passée

---

- **Pourquoi modéliser** Organiser l'information, besoins clients
- **Types de modèles** Conceptuel, Logique, Physique

# Semaine passée

---

- **Pourquoi modéliser** Organiser l'information, besoins clients
- **Types de modèles** Conceptuel, Logique, Physique
- **Contraintes**

# Semaine passée

---

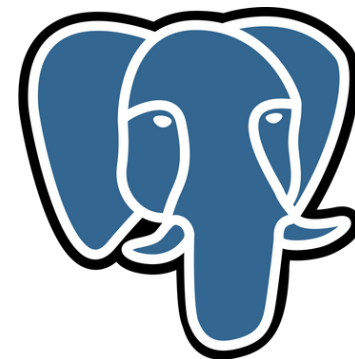
- **Pourquoi modéliser** Organiser l'information, besoins clients
- **Types de modèles** Conceptuel, Logique, Physique
- **Contraintes** Clé primaires/étrangère, Unicité, Nullabilité

# Systemes de gestion de base de données (SGBD)

---

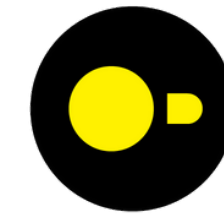
## OLTP (Online Transaction Processing)

Bases optimisées pour les transactions rapides et fréquentes



## OLAP (Online Analytical Processing)

Bases optimisées pour l'analyse de grands volumes de données.



DuckDB



ClickHouse



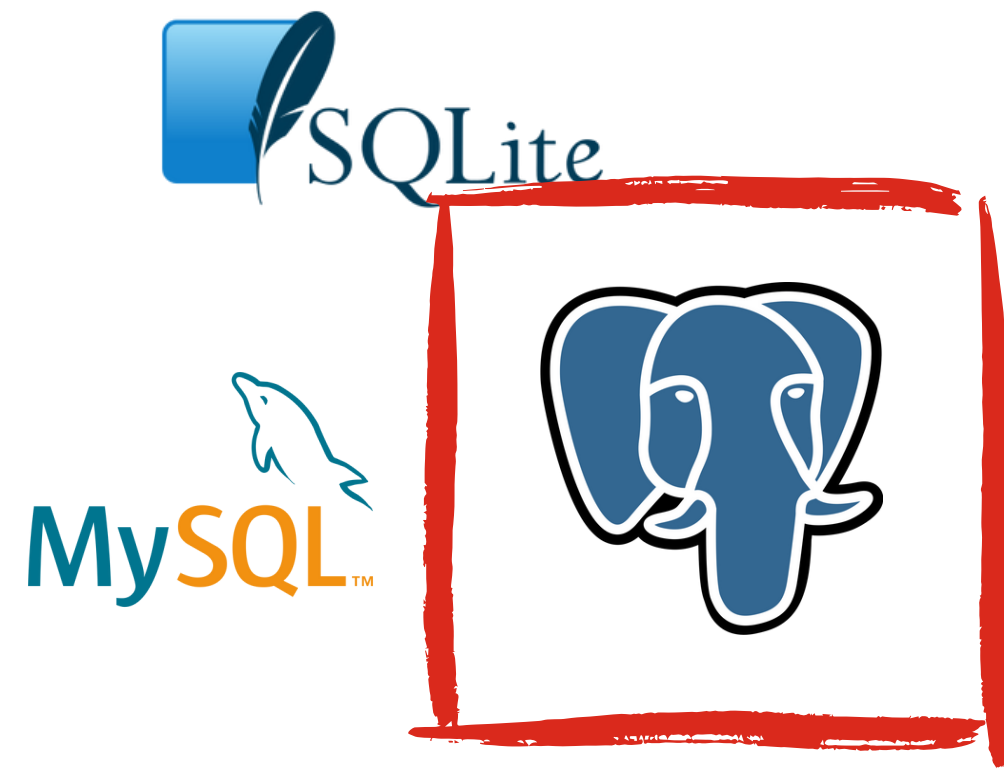


# Systemes de gestion de base de données (SGBD)

---

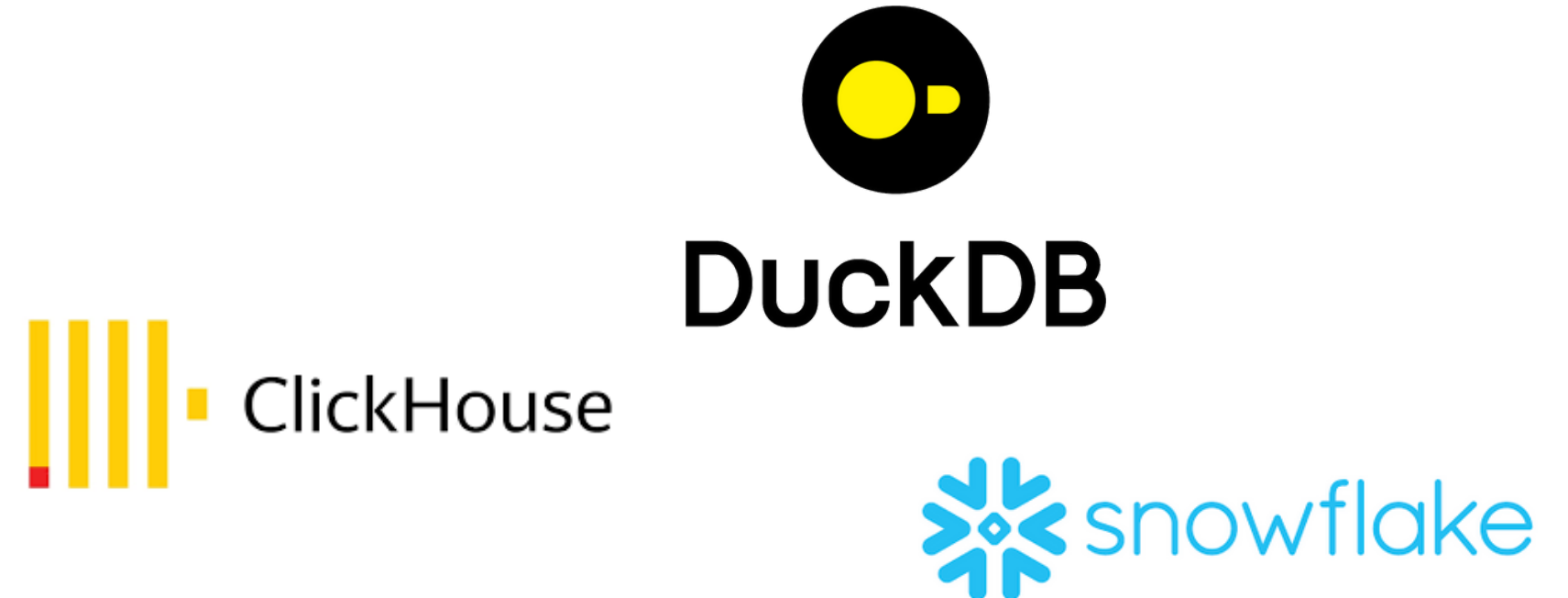
## OLTP (Online Transaction Processing)

Bases optimisées pour les transactions rapides et fréquentes



## OLAP (Online Analytical Processing)

Bases optimisées pour l'analyse de grands volumes de données.



---

# Installation

# Environnement de travail

---

- Systeme de gestion de base de données  
(**SGBD**)



- Editeur **SQL**

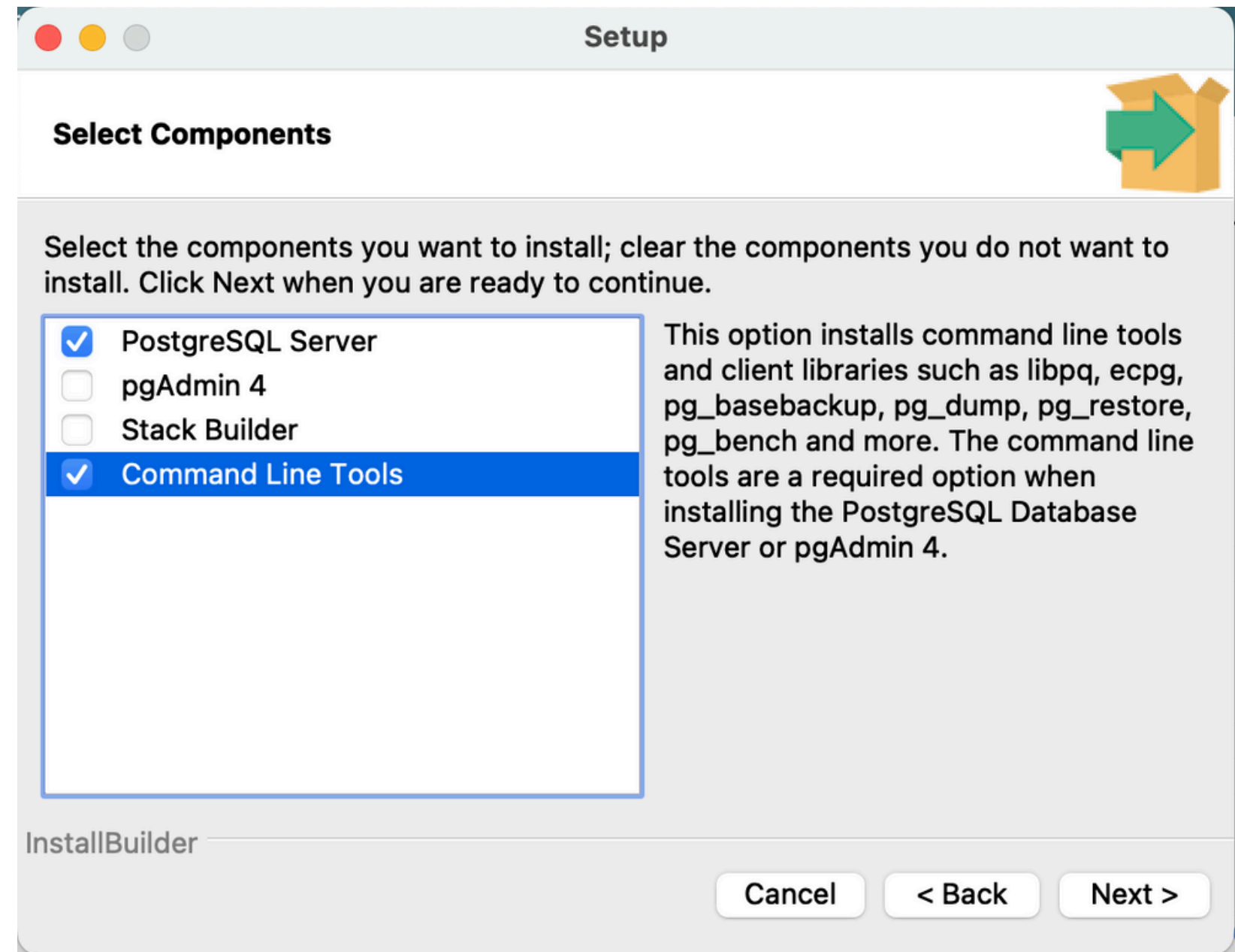


# Installation

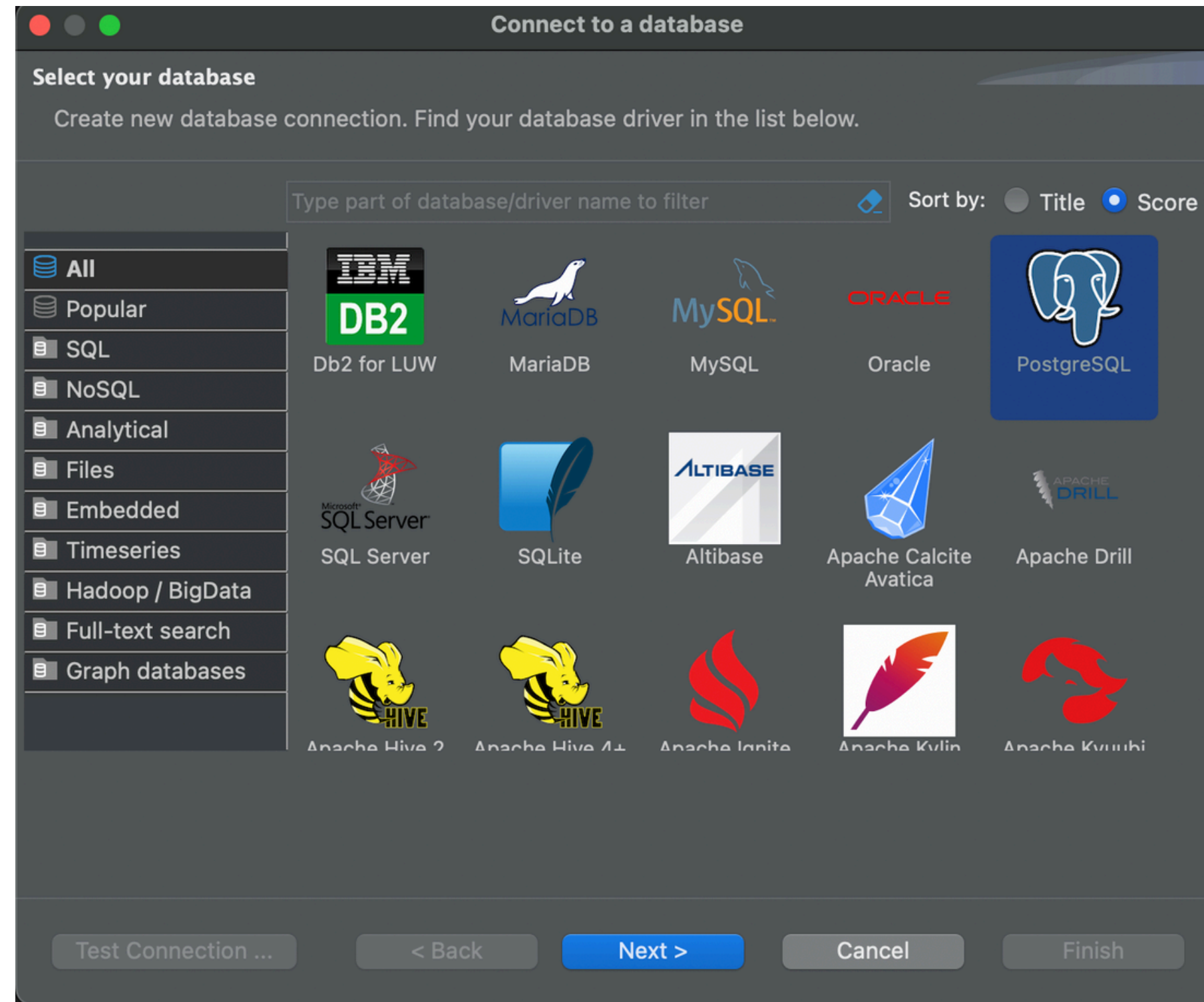
---



# Installation



# Connexion



# Connexion



Le mot de passe que vous  
avez choisi lors de  
l'installation PostgreSQL

Connect to a database

Connection Settings  
PostgreSQL connection settings

PostgreSQL

Main PostgreSQL Driver properties SSH SSL + Network configurations...

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:postgresql://localhost:5432/postgres

Host: localhost Port: 5432

Database: postgres ☒ Show all databases

Authentication

Authentication: Database Native

Username: postgres

Password:  ☒ Save password

Advanced

Session role: Local Client: <not present>

[Connection variables information](#) [Database documentation](#) Connection details (name, type, ...)

Driver name: PostgreSQL Driver Settings Driver license

Test Connection ... < Back Next > Cancel Finish

---

# Syntaxe SQL



# Syntaxe SQL

---

- **Langage déclaratif** : on décrit ce qu'on veut comme résultat, mais pas comment le faire
- **Data Definition Language (DDL)** : créer manipuler les *objets du modele de données*  
**CREATE, DROP, ALTER, RENAME, TRUNCATE**
- **Data Manipulation Language (DML)** : manipuler les *données*  
**INSERT, SELECT, UPDATE, DELETE**

# Objectifs

---

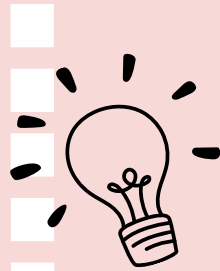
- 1 Créer une base de données
- 2 Lire les données
- 3 Mettre à jour les données
- 4 Supprimer des données
- 5 Filtrer les données
- 6 Agréger les données
- 7 Trier les données
- 8 Mettre en relation les données

---

**1** Créer

# Créer

- **CREATE** : créer un objet de la base de données (i.e. **DATABASE**, **TABLE**, **INDEX** etc.)
- **INSERT INTO** : ajouter une ou plusieurs lignes (**VALUES**) dans une table



Insérer des données dans une base de données directement à partir d'un fichier CSV :

**COPY** table\_name(column1, column2, ...)  
**FROM** chemin/du/fichier.csv [options]

```
# Créer une base de données  
CREATE DATABASE database_name;
```

```
# Créer une table  
CREATE TABLE table_name (  
    column1 [type],  
    column2 [type],  
    column3 [type],  
    ....  
);
```

```
INSERT INTO table_name  
(column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

# Type de données

---

- Numériques: **INT, DECIMAL, FLOAT, DOUBLE** etc.
- Chaîne de caractères: **CHAR, VARCHAR, TEXT**
- Booléenne: **BOOLEAN**
- Géométriques: **GEOMETRY, POINT, LINESTRING, POLYGON**
- Dates et heures: **DATE, TIME, DATETIME, TIMESTAMP, YEAR**
- ... et beaucoup d'autres !



PostgreSQL Datatypes

# Contraintes

---

- **NOT NULL** Garantit qu'une colonne ne peut pas contenir de valeurs NULL.
- **UNIQUE** Assure que toutes les valeurs d'une colonne (ou d'un ensemble de colonnes) sont uniques.
- **PRIMARY KEY** Identifie de manière unique chaque ligne d'une table (implique **NOT NULL** et **UNIQUE**).
- **FOREIGN KEY** Garantit l'intégrité référentielle entre les tables en reliant une colonne à la clé primaire d'une autre table.
- **CHECK** Valide les données selon une expression booléenne.
- **EXCLUSION** Empêche les conflits entre lignes en s'assurant que deux lignes ne se chevauchent pas sur des critères définis.



PostgreSQL DDL constraints

# Challenge

## Films

id	title	release_year
1	Toy Story	1995
2	A Bug's Life	1998
3	Toy Story 2	1999
4	Monsters, Inc.	2001
...	...	...

## Boxoffice

movie_id	rating	domestic_sales	international_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
4	8.1	289916256	272900000
2	7.2	162798565	200600000
...	...	...	...



Accès aux fichiers CSV

# Challenge

- 
- Créer une base de données **movies\_db**
- Créer une table **movies**, ajouter les contraintes d'intégrité, et charger les données
- Créer une table **boxoffice**, ajouter les contraintes d'intégrité, et charger les données

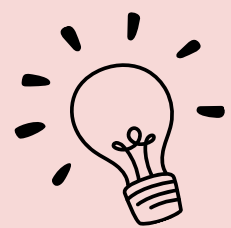


---

2 Lire

## 2 Lire

- **SELECT** : choisir une ou plusieurs colonnes spécifiques (*column\_name*) ou toutes les colonnes (\*).
- **FROM**: à partir d'une table (*table\_name*)



Utiliser des alias (**AS**) pour renommer les colonnes dans vos résultats. Par exemple:

```
SELECT movie_title AS 'Title'  
FROM movies;
```

```
SELECT column_name1,  
column_name2, ...  
FROM table_name;
```

```
SELECT *  
FROM table_name;
```

# Challenge

---

- - Lister toutes les données de **movies**
  - Lister tous les données de **boxoffice**
  - Lister *l'identifiant* et le *titre* de tous les films

---

## ③ Mettre à jour

## 3 Mettre à jour

- **UPDATE** : mettre à jour une table et établir (**SET**) une valeur d'une colonne
- **ALTER TABLE**: créer une colonne (**ADD**), supprimer une colonne (**DROP COLUMN**), rajouter des contraintes (**MODIFY**), renommer une colonne (**RENAME COLUMN**)

```
UPDATE table_name  
SET column_name = <value>;
```

```
UPDATE table_name  
SET column_name_1 = <value>,  
column_name_2 = <value>;
```

```
ALTER TABLE table_name  
ADD new_column [type];
```

# Challenge

---

- - Créer une colonne *category*
  - Mettre à jour la catégorie avec la valeur “Animation”

---

# **4 Supprimer**

## 4 Supprimer

- **DELETE** : supprimer les lignes d'une table
- **DROP** : supprimer un objet de la base de données (**DATABASE**, **TABLE**, **INDEX** etc.)

```
# Supprimer toutes les lignes  
DELETE FROM table_name;
```

```
# Supprimer la base de données  
DROP DATABASE database_name;
```

```
# Supprimer une table  
DROP TABLE table_name;
```



# Challenge

---

- Effacer les données de la table **boxoffice**
- Supprimer la table **boxoffice**

---

# 5 Filterer

## 5 Filtrer (WHERE)

- **WHERE** : filtre les enregistrements dans une table en fonction de conditions spécifiques (si plusieurs, enchaîner avec **AND** ou **OR**)
- Utilisé avec : **SELECT, UPDATE, DELETE, INSERT**
- Comparaisons: égalité (**=**), inégalité (**!=**), supérieure/inférieure (**<, >, <=, >=**), chaîne de caractères (**LIKE**), **BETWEEN, IN, IS NULL** etc.

```
SELECT *  
FROM table_name  
WHERE column_name > value;
```

```
UPDATE table_name  
SET column_name = value  
WHERE column_name = value;
```

```
SELECT *  
FROM table_name  
WHERE column_name LIKE 'value'  
AND column_name2 > value;
```

## 5 Filtrer (LIMIT)

- **LIMIT N**: restreindre le nombre de lignes retournées par une requête
- **OFFSET M**: ignorer un certain nombre de lignes avant de commencer à retourner les résultats

```
SELECT *  
FROM table_name  
LIMIT 10 OFFSET 8;
```

# Challenge

---

- - Lister les films sortis après 2000
  - Lister tous les films Toy Story
  - Effacer les films qui sont sortis avant 2000

---

# 6 Agréger

## 6 Agréger

- **GROUP BY** : regrouper les lignes d'un jeu de résultats en fonction d'une ou plusieurs colonnes
- Fonctions agrégation: **SUM, MAX, MIN, AVG, COUNT** etc.

```
SELECT column1,  
function_name(column2), ...  
FROM table_name  
GROUP BY column1, column2, ...;
```

# EXEMPLE

---

- - Créer une colonne *epoque*
  - Assigner une époque (90's, 00's, 10's) en fonction de l'année de sortie de chaque film
  - Compter le nombre de films sortis pour chaque époque



---

# **7** Trier

## 7 Trier

- **ORDER BY** : tri les résultats d'une requête en fonction d'une ou plusieurs colonnes (**ASC**, **DESC**)

```
SELECT column1, column2, ...  
FROM table_name  
ORDER BY column1 [ASC | DESC], column2  
[ASC | DESC], ...;
```

# EXEMPLE

---

- - Lister les films par ordre décroissant en fonction de l'année de sortie
  - Lister le nombre de films par époque par ordre décroissant

---

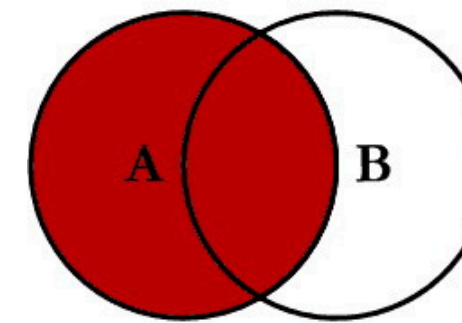
8

# Mettre en relation

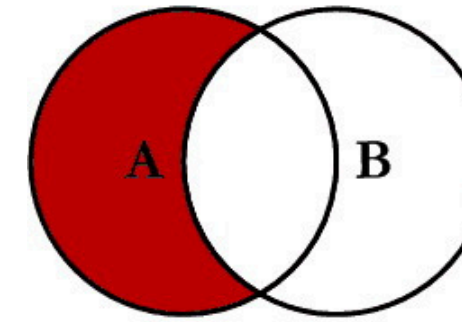
# 8 Mettre en relation les données

- **JOIN** : combiner les lignes provenant de deux ou plusieurs tables avec une condition de correspondance
- Types de jointures : **LEFT**, **RIGHT**, **FULL**, **INNER** etc.

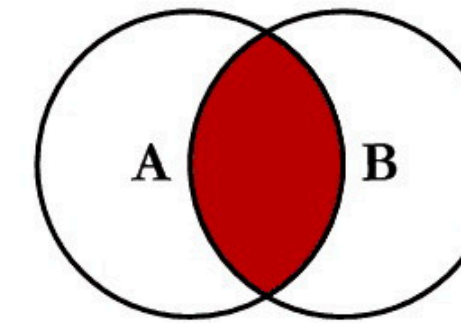
## SQL JOINS



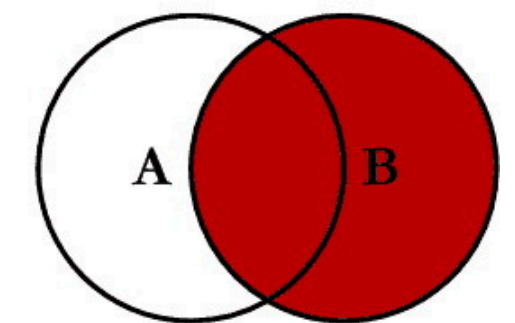
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



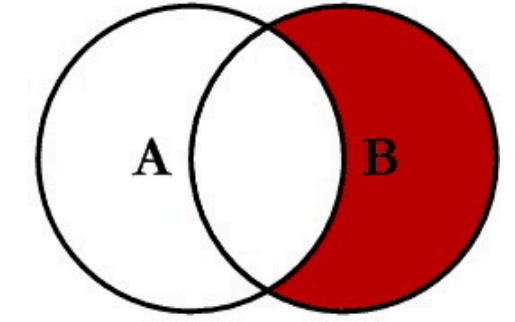
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



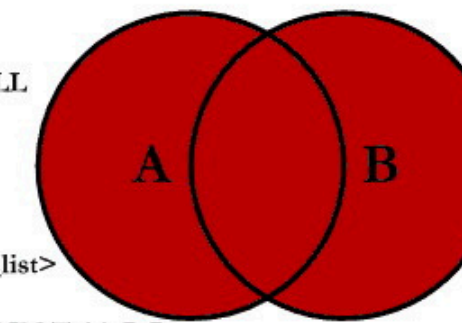
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



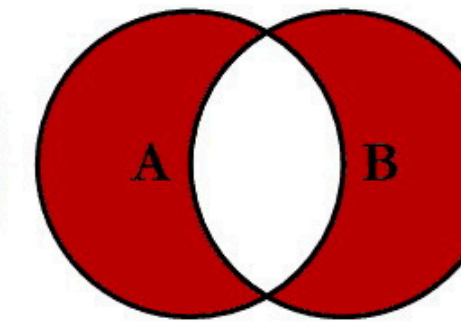
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

# EXEMPLE

---

- 
- Récréer la table **boxoffice**, avec contraintes et données respectives
- Lister les films ainsi que leur entrées internationales respectives
- Lister le film qui a connu le plus d'entrées domestiques

**Bravo !** ☒

---

- |                             |                                  |
|-----------------------------|----------------------------------|
| 1 Créer une base de données | 5 Filtrer les données            |
| 2 Lire les données          | 6 Agréger les données            |
| 3 Mettre à jour les données | 7 Trier les données              |
| 4 Supprimer des données     | 8 Mettre en relation les données |

# Projet

---

## Nouveaux besoins

- **Séance** avec *Chef·fes de projet*

## Diagramme UML

- **Intégrer** nouveaux besoins dans la logique de la base de données
- **Adapter** le schéma UML avec les bons types ainsi que les contraintes que vous jugez pertinentes
- **Créer** la base de données