Module BASES DE DONNEES RELATIONNELLES
TD 2
SQL

Hervé Owsinski
Herve.Owsinski@uphf.fr

Semestre 3





2026-2027

Création de tables

Langage SQL DDL
Data Definition Langage

Exercice 1

Soit le schéma relationnel suivant :

Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)

1. Donner la commande qui permet de créer la table Commande.

Exercice 1 : soyons naïf ! Echec, mais pourquoi ?

Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)

```sql
CREATE TABLE Commande (
    id_commande INT AUTO_INCREMENT,
    id_client INT,
    date_commande DATE,
    statut VARCHAR(40),
    PRIMARY KEY (id_commande),
    FOREIGN KEY (id_client) REFERENCES Client(id_client)
);
```

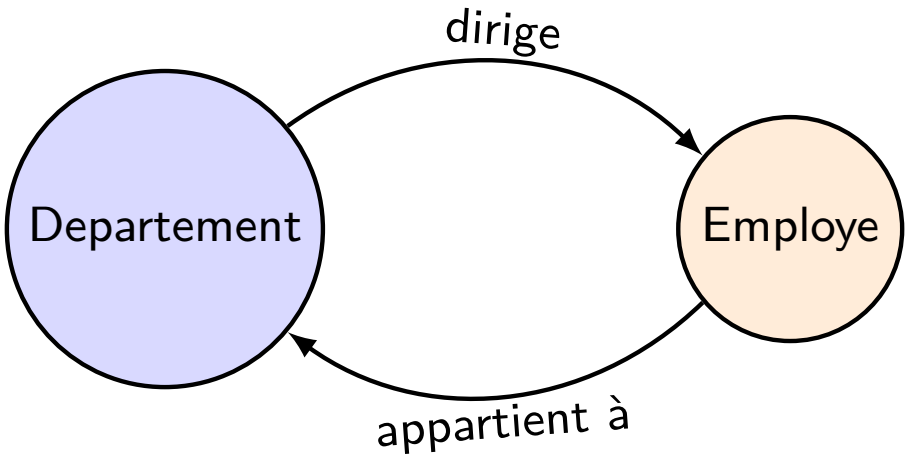
Exercice 1 : correction

Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)

```sql
CREATE TABLE Client(
    id_client INT AUTO_INCREMENT,
    nom VARCHAR(80),
    prenom VARCHAR(80),
    ville VARCHAR(40),
    date_inscription DATE,
    PRIMARY KEY (id_client)
);
CREATE TABLE Commande (
    id_commande INT AUTO_INCREMENT,
    id_client INT,
    date_commande DATE,
    statut VARCHAR(40),
    PRIMARY KEY (id_commande),
    FOREIGN KEY (id_client) REFERENCES Client(id_client)
);
```

Lien avec le TP 1 : références circulaires

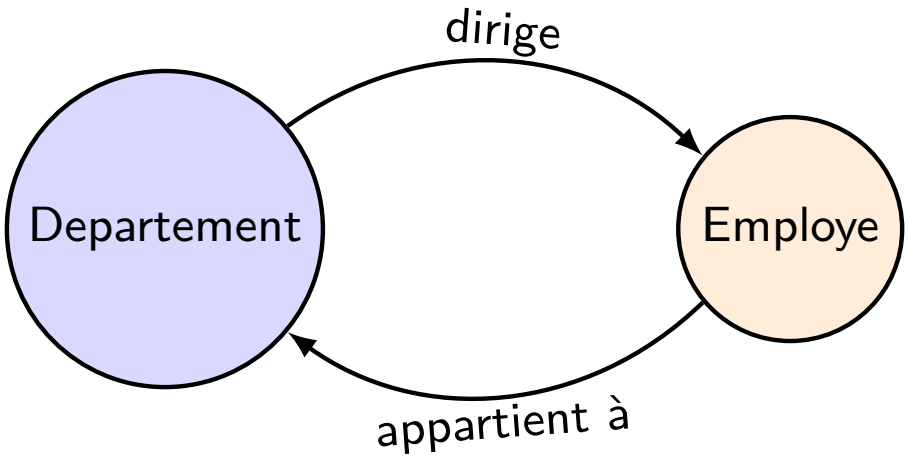
```
Employe (idE, nom, prenom, fonction, dateEmb, ville, #idDept)
Departement (idDept, nom, #directeur)
```



Dept a une clé étrangère vers **Employe** : il faut créer **Employe** d'abord.
Or **Employe** a une clé étrangère vers **Dept** : il faut créer **Dept** d'abord !!
Référence circulaire ! La solution consiste donc à ?

Lien avec le TP 1 : références circulaires

```
Employe (idE, nom, prenom, fonction, dateEmb, ville, #idDept)
Departement (idDept, nom, #directeur)
```



Dept a une clé étrangère vers **Employe** : il faut créer **Employe** d'abord.
Or **Employe** a une clé étrangère vers **Dept** : il faut créer **Dept** d'abord !!
Référence circulaire ! La solution consiste donc à ?

1. CREATE TABLE **Dept** mais sans clé étrangère directeur ;
2. CREATE TABLE **Employe** complétement ;
3. Modifier avec ALTER TABLE **Dept** pour placer la clé étrangère.

Les requêtes d'interrogation

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

1. Afficher l'ensemble des clients.
2. Afficher uniquement le libellé et le prix de chaque produit.
3. Afficher la liste des villes où habitent les clients, sans doublon.
4. Afficher les produits dont le prix est supérieur à 50 €.
5. Afficher les produits de la catégorie 'Informatique' dont le prix est compris entre 20 € et 100 €.

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

1. Afficher l'ensemble des clients.

```sql
SELECT *
FROM Client;
```

```sql
SELECT id_client, nom, prenom
FROM Client;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

2. Afficher uniquement le libellé et le prix de chaque produit.

```sql
SELECT libelle, prix
FROM Produit ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

2. Afficher uniquement le libellé et le prix de chaque produit.

```sql
SELECT libelle, prix
FROM Produit ;
```

3. Afficher la liste des villes où habitent les clients, sans doublon.

```sql
SELECT DISTINCT ville
FROM Client ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

4. Afficher les produits dont le prix est supérieur à 50 €.

```sql
SELECT id_produit, libelle
FROM Produit
WHERE prix > 50 ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

4. Afficher les produits dont le prix est supérieur à 50 €.

```sql
SELECT id_produit, libelle
FROM Produit
WHERE prix > 50 ;
```

5. Afficher les produits de la catégorie 'Informatique' dont le prix est compris entre 20 € et 100 €.

```sql
SELECT id_produit, libelle
FROM Produit
WHERE categorie = 'Informatique' AND prix >= 20 AND prix <= 100 ;
```

```sql
WHERE categorie = 'Informatique' AND prix BETWEEN 20 AND 100 ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

6. Afficher les clients qui habitent à Lille ou à Paris.
7. Afficher les clients dont le nom commence par la lettre 'M'.
8. Afficher les produits du plus cher au moins cher.
9. Afficher les produits classés par catégorie (ordre alphabétique), puis par prix décroissant dans chaque catégorie.
10. Afficher les trois produits les plus chers

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

6. Afficher les clients qui habitent à Lille ou à Paris.

```sql
SELECT id_client, nom, prenom
FROM Client
WHERE ville = 'Lille' OR ville = 'Paris' ;
```

```sql
SELECT id_client, nom, prenom
FROM Client
WHERE ville = 'Lille'
UNION
SELECT id_client, nom, prenom
FROM Client
WHERE ville = 'Paris' ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

7. Afficher les clients dont le nom commence par la lettre 'M'.

```sql
SELECT id_client, nom, prenom
FROM Client
WHERE nom LIKE 'M%' ;
```

```sql
SELECT id_client, nom, prenom
FROM Client
WHERE nom >= 'M' AND nom < 'N' ;
```

```sql
SELECT id_client, nom, prenom
FROM Client
WHERE nom >= 'M'
INTERSECT
SELECT id_client, nom, prenom
FROM Client
WHERE  nom < 'N' ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

8. Afficher les produits du plus cher au moins cher.

```sql
SELECT id_produit, libelle
FROM Produit
ORDER BY prix DESC;  -- par prix décroissant
```

```sql
SELECT id_produit, nom
FROM Produit
ORDER BY prix DESC, libelle ASC ;  -- et nom croissant si égalité
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

9. Afficher les produits classés par catégorie (ordre alphabétique), puis par prix décroissant dans chaque catégorie.

```sql
SELECT id_produit, libelle
FROM Produit
ORDER BY categorie ASC, prix DESC;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

10. Afficher les trois produits les plus chers.

```sql
SELECT id_produit, libelle
FROM Produit
ORDER prix DESC  -- trie par prix décroissant
LIMIT 3 ;        /* ne conserve que les 3 premieres
                    lignes récupérées
                 */
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

11. Afficher, pour chaque commande, son numéro, sa date, ainsi que le nom et le prénom du client qui l'a passée

```sql
SELECT id_commande, date_commande, nom, prenom
FROM Commande Co
JOIN Client   Cl ON Co.id_client = Cl.id_client;
```

```sql
SELECT id_commande, date_commande, nom, prenom
FROM Commande Co, Client Cl
WHERE Co.id_client = Cl.id_client;
```

Exemple

Employe (idE, nom, prenom, fonction, dateEmb, ville, #idDept)
Departement (idDept, nom, #directeur)

Considérons les deux contenus suivants pour nos deux tables :

idE	nom	prenom	fonction	dateEmb	ville	idDept
1	Muller	Gerard	ingenieur	1995-02-01	Valenciennes	1
2	Rochat	Juliette	ingenieur	1992-03-04	Lille	1
3	Legrand	Didier	technicien	1995-02-01	Paris	2
4	Lepetit	Jean	technicien	1992-03-04	Maubeuge	1
5	Dupont	Alice	ingenieur	1996-06-12	Lille	2
6	Bernard	Thomas	commercial	1994-09-23	Paris	3

idDept	nom	directeur
1	Informatique	2
2	Securite	3
3	Commercial	6

Exemple de produit cartésien

Employe (idE, nom, prenom, fonction, dateEmb, ville, #idDept)
Departement (idDept, nom, #directeur)

Résultat du produit cartésien *Employe* × *Dept* ?

Produit cartésien : on génère toutes les associations possibles

```
SELECT *
FROM Employe, Service;    -- Employe x Service
```

idE	nom	prenom	fonction	dateEmb	ville	idDept	idDept	nom	directeur
1	Muller	Gerard	ingenieur	1995-02-01	Valenciennes	1	1	Informatique	2
1	Muller	Gerard	ingenieur	1995-02-01	Valenciennes	1	2	Securite	3
1	Muller	Gerard	ingenieur	1995-02-01	Valenciennes	1	3	Commercial	6
1	Muller	Gerard	ingenieur	1995-02-01	Valenciennes	1	4	Developpement	5
2	Rochat	Juliette	ingenieur	1992-03-04	Lille	1	1	Informatique	2
2	Rochat	Juliette	ingenieur	1992-03-04	Lille	1	2	Securite	3
2	Rochat	Juliette	ingenieur	1992-03-04	Lille	1	3	Commercial	6
2	Rochat	Juliette	ingenieur	1992-03-04	Lille	1	4	Developpement	5
3	Legrand	Didier	technicien	1995-02-01	Paris	2	1	Informatique	2
3	Legrand	Didier	technicien	1995-02-01	Paris	2	2	Securite	3
3	Legrand	Didier	technicien	1995-02-01	Paris	2	3	Commercial	6
3	Legrand	Didier	technicien	1995-02-01	Paris	2	4	Developpement	5
4	Lepetit	Jean	technicien	1992-03-04	Maubeuge	1	1	Informatique	2
4	Lepetit	Jean	technicien	1992-03-04	Maubeuge	1	2	Securite	3
4	Lepetit	Jean	technicien	1992-03-04	Maubeuge	1	3	Commercial	6
4	Lepetit	Jean	technicien	1992-03-04	Maubeuge	1	4	Developpement	5
...					...				...

Parfois les deux idDept correspondent mais pas toujours.

Exemple de jointure "à l'ancienne"

```
SELECT *
FROM Employe, Dept
WHERE Employe.idDept = Dept.idDept;
```

Jointure : produit cartésien *Employe* $\times$ *Dept* et restriction sur les lignes choisies :

```
+-----+---------+----------+------------+------------+--------------+--------+--------+---------------+-----------+
| idE | nom     | prenom   | fonction   | dateEmb    | ville        | idDept | idDept | nom           | directeur |
+-----+---------+----------+------------+------------+--------------+--------+--------+---------------+-----------+
|   1 | Muller  | Gerard   | ingenieur  | 1995-02-01 | Valenciennes |      1 |      1 | Informatique  |         2 |
|   2 | Rochat  | Juliette | ingenieur  | 1992-03-04 | Lille        |      1 |      1 | Informatique  |         2 |
|   3 | Legrand | Didier   | technicien | 1995-02-01 | Paris        |      2 |      2 | Securite      |         3 |
|   4 | Lepetit | Jean     | technicien | 1992-03-04 | Maubeuge     |      1 |      1 | Informatique  |         2 |
+-----+---------+----------+------------+------------+--------------+--------+--------+---------------+-----------+
4 rows in set (0,001 sec)
```

Exemple de jointure ”moderne”

```
SELECT *
FROM Employe
JOIN Dept ON Employe.idDept = Employe.idDept;
```

La formulation avec JOIN donne le même résultat mais a un avantage : elle allège le WHERE :

- les restrictions de ligne liées à la jointure sont derrière le ON du JOIN ;
- les autres restrictions sont placées dans le WHERE.

```
+-----+---------+----------+------------+------------+--------------+--------+--------+---------------+-----------+
| idE | nom     | prenom   | fonction   | dateEmb    | ville        | idDept | idDept | nom           | directeur |
+-----+---------+----------+------------+------------+--------------+--------+--------+---------------+-----------+
|   1 | Muller  | Gerard   | ingenieur  | 1995-02-01 | Valenciennes |      1 |      1 | Informatique  |         2 |
|   2 | Rochat  | Juliette | ingenieur  | 1992-03-04 | Lille        |      1 |      1 | Informatique  |         2 |
|   3 | Legrand | Didier   | technicien | 1995-02-01 | Paris        |      2 |      2 | Securite      |         3 |
|   4 | Lepetit | Jean     | technicien | 1992-03-04 | Maubeuge     |      1 |      1 | Informatique  |         2 |
+-----+---------+----------+------------+------------+--------------+--------+--------+---------------+-----------+
4 rows in set (0,001 sec)
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

12. Afficher le détail de chaque ligne de commande : numéro de commande, nom du client, libellé du produit, quantité et total de la ligne (quantite * prix). Trier le résultat par numéro de commande

```sql
SELECT Co.id_commande, Cl.nom, libelle,
       (L.quantite * P.prix) AS total
FROM Ligne_commande L
JOIN Commande Co     ON L.id_commande = Co.id_commande
JOIN Client Cl       ON Co.id_client  = Cl.id_client
JOIN Produit P       ON P.id_produit  = L.id_produit
ORDER BY Co.id_commande ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

13. Afficher tous les clients avec leurs commandes éventuelles. Les clients sans commande doivent apparaître dans le résultat.

```sql
SELECT Cl.id_client, Cl.nom, Cl.prenom, Co.id_commande
FROM Client Cl
LEFT JOIN Commande Co ON Co.id_client = Cl.id_client ;
```

JOIN : lorque cela concorde entre Client et Commande (on obtient donc les clients qui ont passé au moins une commande).

LEFT JOIN :

- On obtient les clients qui ont passé une commande et
- les clients qui n'ont pas passé de commande. Dans ce cas, les colonnes correspondant à Commande contiendront NULL.

```sql
FROM Client Cl LEFT JOIN Commande Co  ON Co.id_client = Cl.id_client ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

14. Afficher uniquement les clients qui n'ont jamais passé de commande

```sql
SELECT Cl.id_client, Cl.nom, Cl.prenom
FROM Client Cl
LEFT JOIN Commande Co ON Co.id_client = Cl.id_client
WHERE Co.id_commande IS NULL;  /* on ne garde que les lignes
                                  sans numéro de commande */
```

```sql
SELECT id_client, nom, prenom              -- tous les clients
FROM Client
EXCEPT                                     -- sauf
SELECT Cl.id_client, Cl.nom, Cl.prenom     -- ceux avec commande
FROM Client Cl
JOIN Commande Co ON Co.id_client = Cl.id_client ;
```

```
Client (id_client, nom, prenom, ville, date_inscription)
Produit (id_produit, libelle, categorie, prix)
Commande (id_commande, #id_client, date_commande, statut)
Ligne_commande (#id_commande, #id_produit, quantite)
```

15. Afficher en une seule requête le nombre de produits, le prix moyen, le prix minimum et le prix maximum.

```sql
SELECT COUNT(*) AS nombre_produits,
       AVG(prix) AS prix_moyen,
       MIN(prix) AS prix_minimum,
       MAX(prix) AS prix_maximum
FROM Produit;
```

Ici les fonctions d'agrégation agissent sur toutes les lignes : il n'y a qu'un seul prix_moyen, moyenne des prix sur toute les lignes de Produit.