

Pour les exemples, on considère une base de données avec 3 tables dont les schémas relationnels sont :

- film (id, titre, annee, directeur, budget, recette)
- acteur (id, nom)
- casting (id_film, id_acteur)

Une **clé** d'une table est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

La **clé primaire** d'une table permet de garantir l'unicité des enregistrements. On choisit souvent comme clé primaire un entier avec auto-incrément.

Une **clé étrangère** est un attribut (ou ensemble d'attributs) faisant référence à la clé primaire d'une autre table.

Syntaxe générale de **SELECT**, dans cet ordre ([...] indiquant une commande optionnelle) :

```
SELECT [DISTINCT] expr1 [AS alias1], expr2, ...
FROM table1 [AS alias1], table2, ...
[WHERE condition]
[GROUP BY expr]
[HAVING condition]
[ORDER BY expr [DESC]]
[LIMIT entier]
[OFFSET entier]
```

- **SELECT** [**DISTINCT**] **expr1** [**AS** **alias1**], **expr2**, ...
Renvoie une table dont les colonnes correspondent à **expr1**, **expr2**... qui sont des expressions, pouvant contenir des attributs, calculs et fonctions.

Si un attribut **attr** est ambigu (car il est le même dans 2 tables **t1** et **t2**), il faut le préfixer par son nom de table, par ex. **t1.attr**.

* est un raccourci pour sélectionner toutes les colonnes.

AS renomme une colonne pour, par exemple, y faire référence ensuite.

Films avec leur profit :

```
SELECT titre, (recette - budget) AS p FROM film;
```

DISTINCT supprime les doublons.

Obtenir tous les acteurs (sans doublon) :

```
SELECT DISTINCT nom FROM acteur;
```

- **FROM** **table1** [**AS** **alias1**], **table2**, ...
Sélectionne les valeurs dans les tables mentionnées.

table1, **table2** est la table correspondant au produit cartésien de **table1** et **table2**.

table1 JOIN table2 ON colonne1 = colonne2 réalise la jointure de **table1** et **table2**, où la **colonne1** de **table1** est identifiée avec **colonne2** de **table2**. On peut mettre plusieurs **JOIN** à la suite.

Tous les directeurs et acteurs ayant travaillé ensemble :

```
SELECT directeur, nom FROM film
JOIN casting ON film.id = id_film
JOIN acteur ON id_acteur = acteur.id
```

- [**WHERE** **condition**]

Ne considère que les enregistrements vérifiant **condition**. **condition** peut contenir des attributs, calculs, **AND**, **OR**, **<**, **<=**, **!=**, **LIKE**, **IN**...

Tous les directeurs qui sont aussi acteurs :

```
SELECT DISTINCT directeur FROM film, acteur
WHERE directeur = nom
```

- [**GROUP BY** **expr**
[**HAVING** **condition**]]

Regroupe tous les enregistrements ayant la même valeur **expr** en un seul enregistrement. Seuls les groupes vérifiant **condition** sont renvoyés.

Les fonctions d'agréations (dans un **SELECT** ou **HAVING**) s'appliquent alors pour chaque groupe : **COUNT**(attribut) (nombre d'enregistrements non **NULL**), **COUNT**(*) (nombre d'enregistrements), **SUM**(attribut), **MAX**(attribut), **AVG**(attribut) (moyenne), ...

Nombre de films réalisés chaque année depuis 2000 :

```
SELECT annee, COUNT(*)
FROM film
WHERE annee >= 2000
GROUP BY annee;
```

Directeurs ayant rapporté au moins 1 milliard :

```
SELECT directeur FROM film
GROUP BY directeur
HAVING SUM(recette) >= 1000000000
```

- [**ORDER BY** **expr** [**DESC**]]

Trie les enregistrements selon **expr**, croissant par défaut (décroissant si **DESC** est utilisé).

Acteurs triés par le nombre de films joués :

```
SELECT nom, COUNT(*) AS nb_films
FROM acteur JOIN casting ON acteur.id = id_acteur
JOIN film ON film.id = id_film
GROUP BY nom
ORDER BY nb_films DESC
```

- [**LIMIT** **n**
[**OFFSET** **p**]]

Affiche seulement les **n** premiers enregistrements (en commençant à partir du (**p** + 1)ème). Souvent utilisé après un **ORDER BY**.

Deuxième film à plus gros budget :

```
SELECT titre FROM film
ORDER BY budget DESC
LIMIT 1 OFFSET 2;
```

- **Sous-requêtes** : il est possible d'utiliser un **SELECT** renvoyant une seule valeur à l'intérieur d'un autre **SELECT**, dans une condition ou un calcul. *Tous les acteurs du film à plus gros budget :*

```

SELECT nom FROM acteur
JOIN casting ON id_acteur = acteur.id
JOIN film ON id_film = film.id
WHERE titre = (SELECT titre FROM film
ORDER BY budget DESC LIMIT 1)

```

• Opérateurs ensemblistes :

Étant donné deux requêtes de la forme **SELECT ...** renvoyant deux résultats **table1** et **table2** de même schéma relationnel, il est possible d'obtenir leur union, intersection et différence avec **UNION**, **INTERSECT**, **MINUS**. Ces opérateurs sont nettement moins utilisés que les jointures et produits cartésiens.

Exemple :

```

SELECT * FROM table1
MINUS SELECT * FROM table2

```

Modèle entité-association

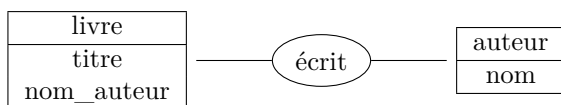
- Une **entité** est un ensemble d'objets similaires que l'on souhaite stocker.

Exemple : Livre, auteur...

- Une **association** (ou : **relation**) est une relation entre plusieurs entités. Une association est binaire si elle met en relation deux entités.

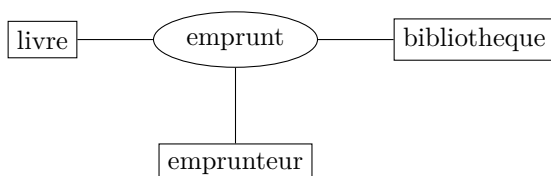
Exemple : Un auteur écrit un livre.

- Représentation sous forme de diagramme :

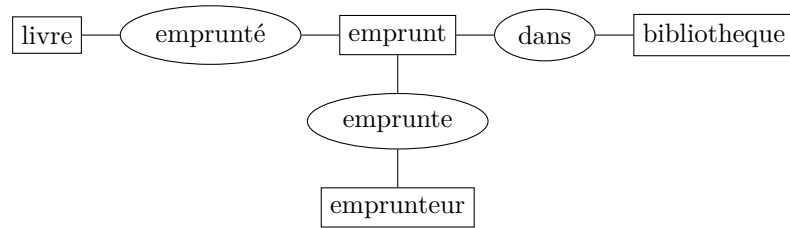


- Une relation n -aire peut être transformée en relation binaire en introduisant une nouvelle entité pour la relation.

Exemple :



Une relation ternaire

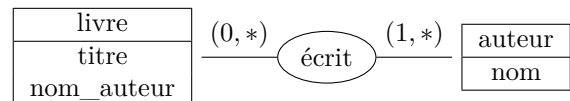


3 relations binaires

- On peut spécifier le lien entre une entité et une association avec un couple (n, p) indiquant le nombre minimum et maximum de fois que l'entité peut apparaître dans l'association ($p = *$ s'il n'y a pas de maximum).

Exemples :

- Un livre a été écrit par au moins une personne, sans limite supérieure. D'où la cardinalité $(1, *)$ pour le lien entre l'entité livre et l'association « écrit ».
- Une personne peut avoir écrit un nombre quelconque de livre. D'où la cardinalité $(0, *)$.
- Si on suppose qu'une personne peut emprunter au plus 5 livres, alors le lien entre l'entité personne et l'association « emprunt » est de cardinalité $(0, 5)$.



- Types possibles d'association entre deux entités :
 - $1 - 1$ (one-to-one) : La borne supérieure vaut 1 pour les 2 entités.
Exemple : L'association « dirige » est de type $1 - 1$ pour directeur_bibliotheque et bibliotheque.
 - $1 - *$ (one-to-many) : La borne supérieure vaut 1 pour une entité et $*$ pour l'autre.
Exemple : Chaque mail est écrit par un unique auteur, mais chaque auteur a pu écrire plusieurs mails.
 - $* - *$ (many-to-many) : La borne supérieure vaut $*$ des deux côtés.
Exemple : L'association « est de type » entre la table des pokémons et des types est de type $* - *$ (à chaque pokémon peut correspondre plusieurs types et plusieurs pokémons peuvent avoir le même type).

- Pour concevoir une base de donnée :

- Utiliser une table par entité.
- Pour chaque association entre a et b :
 - * Si association $1 - 1$: Fusionner les tables a et b .
 - * Si association $1 - *$: Ajouter un attribut (clé étrangère) dans b faisant référence à la clé primaire de a .
 - * Si association $* - *$: Ajouter une table ayant 2 clés étrangères pour faire référence à a et b .