

Bases de données

Dr H. HACHICHI

BDD

- *Définition informelle:* On peut considérer une BDD comme une grande quantité de données centralisées ou non , servant pour les besoins d'une ou plusieurs applications , interrogeables et modifiables par un groupe d'utilisateurs .
- *Définition formelle:* Une base de données est un ensemble d'informations sur un sujet qui est : exhaustif, non redondant, structuré, persistant.

SGBD

- Un système de gestion de base de données est un logiciel qui permet de : décrire, modifier, interroger, administrer, les données d'une base de données.
- Est un logiciel de haut niveau qui permet de manipuler les informations stockées dans une BDD.

SGBD

Exemple de SGBD:

- Interbase,
- Microsoft SQL Server,
- Microsoft Access,
- Oracle,
- Sybase,
- mySQL,
- PostgreSQL
-

SGBD

Fonctions d'un SGBD :

- Décrire les données qui seront stockées
- Manipuler ces données (ajouter, modifier, supprimer des informations)
- Consulter les données et traiter les informations obtenues (sélectionner, trier, calculer,...)
- Définir des contraintes d'intégrité sur les données (contraintes de domaines, d'existence,...)
- Définir des protections d'accès (mots de passe, autorisations,...)
- Résoudre les problèmes d'accès multiples aux données (blocages, interblocages)
- Prévoir des procédures de reprise en cas d'incident (sauvegardes, journaux,...)

SGBD

Objectifs des systèmes de gestion de bases de données

- *Indépendance par rapport :*
 - aux traitements : Pour faciliter la maintenance, un SGBD doit favoriser l'indépendance des traitements
 - à l'implantation physique des données (codage, support d'enregistrement, ordre dans lequel les données sont enregistrées,...)
 - à l'implantation logique des données (existence d'index, décomposition en "fichiers logiques",...)

SGBD

- *Manipulations des données par des non informaticiens:* Il faut pouvoir accéder aux données sans savoir programmer ce qui signifie des langages "quasi naturels".
- *Efficacité des accès aux données:* Ces SGBD's doivent permettre d'obtenir des réponses aux interrogations en un temps "raisonnable". Ils doivent donc être optimisés et, entre autres, il faut un mécanisme permettant de minimiser le nombre d'accès aux disques. Tout ceci, bien sur, de façon complètement transparente pour l'utilisateur.
- *Administration centralisée des données:* Des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.

SGBD

- *Non redondance des données* Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.
- *Cohérence des données* Les données sont soumises à un certain nombre de contraintes d'intégrité qui définissent un état cohérent.

Cycle de vie d'une base de données

- **Conception** On appelle conception d'une base de données la phase d'analyse qui aboutit à déterminer le futur contenu de la base. Lorsqu'une entreprise décide, pour son informatisation, d'adopter une approche base de données, le premier problème à résoudre, peut-être le plus difficile, est de déterminer les informations qu'il conviendra de mettre dans la base de données.

Cycle de vie d'une base de données

- **Implémentation** Ceci sera fait au moyen d'un langage symbolique, spécifique du SGBD choisi, que l'on appelle langage de description de données (LDD). Une fois que le SGBD aura pris connaissance de cette description, il sera possible aux utilisateurs d'entrer les données, c'est-à-dire de constituer la première version, initiale, de la base de données.

Cycle de vie d'une base de données

- **Manipulation** Une fois l'implantation terminée, peut commencer l'utilisation de la base de données. Celle-ci se fait au moyen d'un langage, dit langage de manipulation de données (LMD), qui permet d'exprimer aussi bien les requêtes d'interrogation (pour obtenir des informations contenues dans la base) que des requêtes de mise à jour (pour ajouter de nouvelles informations, supprimer des informations périmées, modifier le contenu des informations).

Principaux modèles conceptuels

- 1 ère génération :
 - ☐ hiérarchique (IMS d'IBM)
 - ☐ réseau (DBTG CODASYL)
- 2eme génération :
 - ☐ entité-association
 - ☐ Relationnel
- 3eme génération :
 - ☐ modèle orienté objet
 - ☐ modèle objet-relationnel
 - ☐ UML
- 4eme génération (les données du Web)
 - ☐ XML

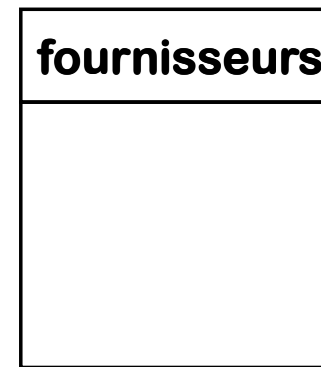
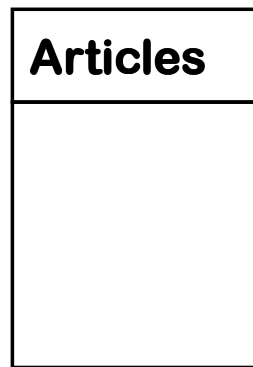
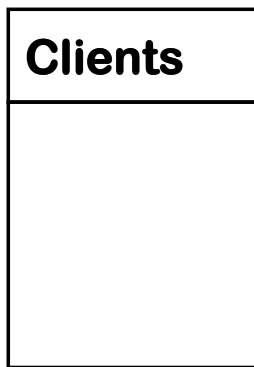
Modèles entité – association

- Dans les années 70,
- Le plus connu des modèles conceptuels pour le développement d'application de BDD,
- Simple et puissant (représentation graphique),

Modèles entité – association

Les éléments de base :

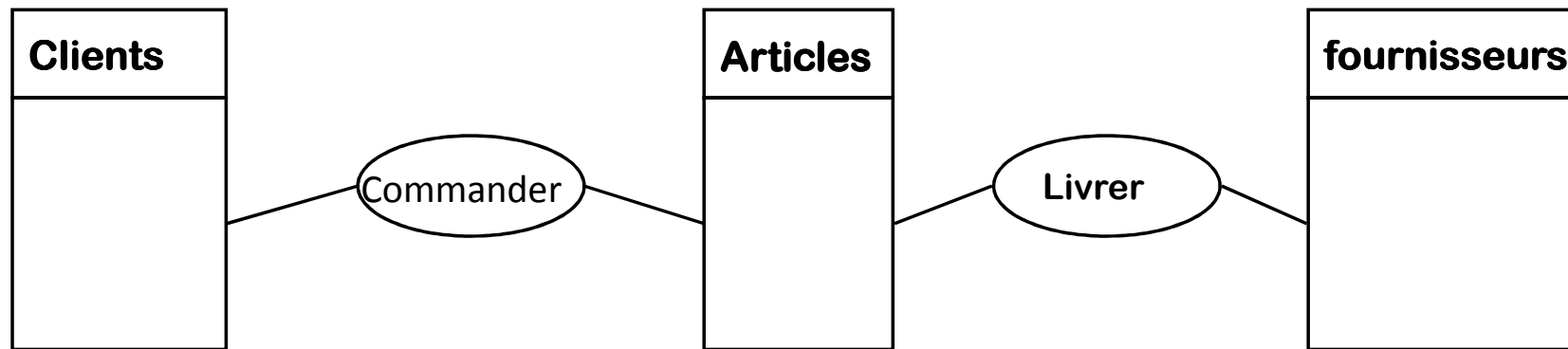
- **Entité** : un objet, un elt du monde qui existe et se distingue des autres. (=un ensemble d'attributs).



Modèles entité – association

Les éléments de base :

- **Association** : met en relation ++ entités. Elle peut avoir des attributs.



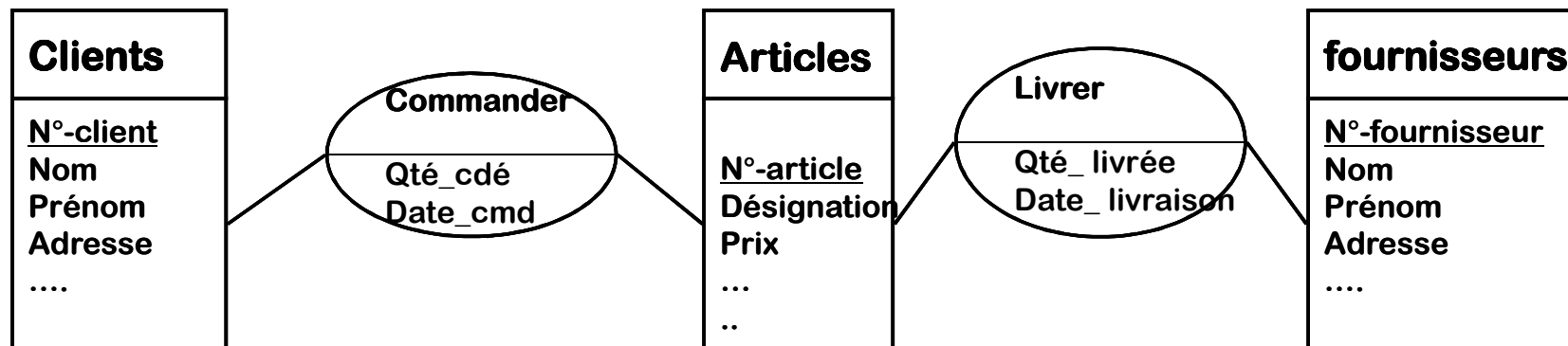
Modèles entité – association

Les éléments de base :

- **Attribut**: associé à un domaine.
- **Identificateur** :un attribut ou un ensemble d'attribut de l'entité permettant de distinguer une entité d'une autre du même ensemble.

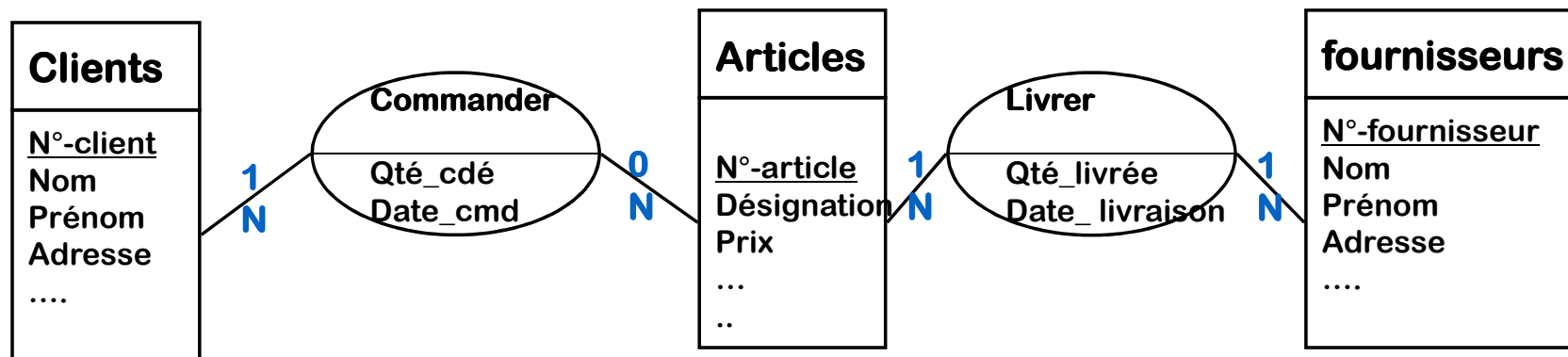
Modèles entité – association

Les éléments de base



Modèles entité – association

- **Les cardinalités:** d'une association indiquent, pour une entité, combien de fois une occurrence de cette entité peut participer à l'association en question.



Modèle relationnel

Modèle relationnel

- Il est basé sur une représentation très simple des données sous forme de tables constituées de lignes et de colonnes.
- Avec ce modèle la BDD= ensemble de tables relationnelles.

Modèle relationnel

- Une table est une relation ➔ la relation peut être:
 - Une relation binaire.
 - Une relation ternaire.
 - Une relation n – aire.

Modèle relationnel

Attributs

	Code	Nom	prénom	adresse
→	00134	Nom1	Pr1	Adr1
→	01045	Nom2	Pr2	Adr2
→	00033	Nom3	Pr3	Adr3
→	01190	Nom4	Pr4	Adr4

Tuples (occurrences)

Modèle relationnel

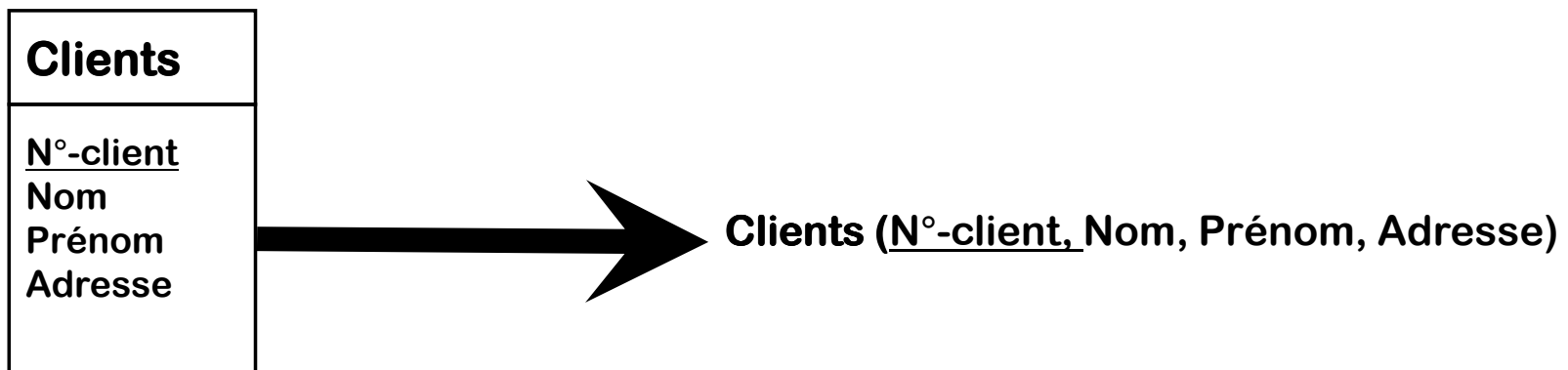
- Schéma de relation :
 - $R (A_1, A_2 \dots\dots\dots A_n)$
- Associer à chaque attribut un domaine:
 - $A_1: D_1; A_2:D_2;\dots\dots\dots A_n: D_n.$
- **Exemple:**
voiture (N°immat, couleur, modèle,marque).

Modèle relationnel

- **Clé d'une relation:** le + petit ensembles d'attributs qui détermine de manière unique chaque tuple de la relation.
- **Exemple:**
voiture (N°immat, couleur, modèle,marque).

Règle de passage au relationnel

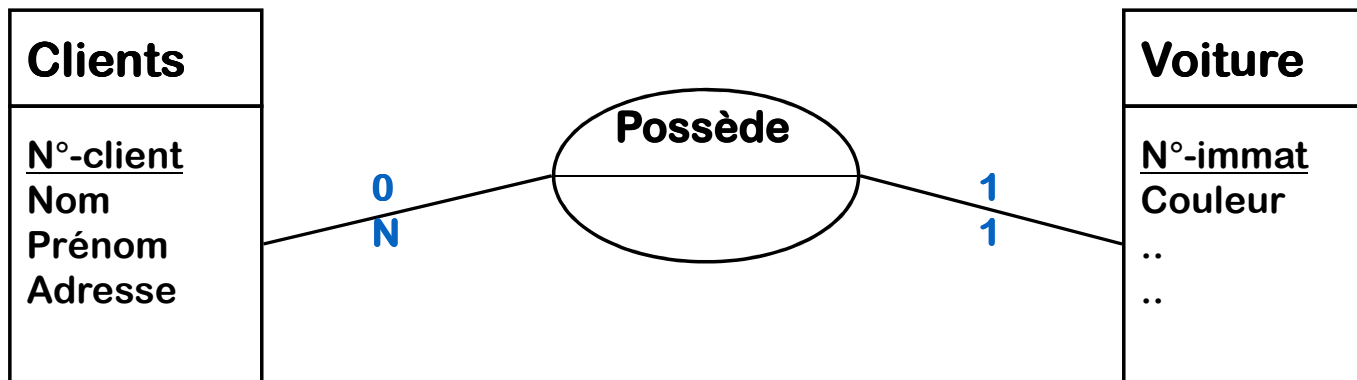
- **Entité:** une entité E est représentée par une relation R dont les attributs sont les attributs de l'entité E.
- La clé de R est l'identifiant de E.



Règle de passage au relationnel

- **Association père/fils:** dans le cas d'une relation 1:1/ 0:1, l'association n'est pas représentée par une relation, cependant la clé du père migre vers le fils comme clé étrangère.

Règle de passage au relationnel



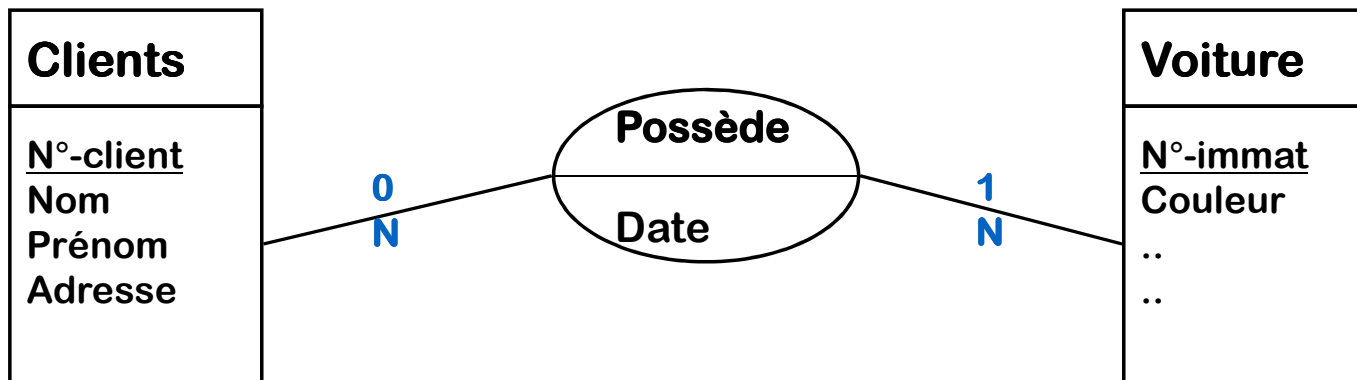
Clients (N°-client, Nom, Prénom, Adresse)

Voiture (N°- immat, couleur, modèle, marque, #N°- client)

Règle de passage au relationnel

- **Association ++ à ++:** dans le cas d'une relation N:N, l'association A est représentée par une relation T dont les attributs sont les attributs de A et la clé est la concaténation des clés des entités participants à l'association A.

Règle de passage au relationnel



Clients (N°-client, Nom, Prénom, Adresse)

Possède (N°-client, N°-immat, date)

Voiture (N°-immat, couleur, modèle, marque)

Normalisation

Normalisation

- **Objectifs:** éviter
 - ❑ la **redondance**, c'est-à-dire la duplication d'informations.
 - ❑ Les **incohérences** par ajout.
 - ❑ La **perte d'informations** par suppression.

Dépendances Fonctionnelles

- Soit $R(X, Y, Z)$ une relation.
 - ❑ Si à une valeur de X n'est associée qu'une seule valeur de Y ,
 - ❑ X **détermine** Y ($X \rightarrow Y$).
 - ❑ On dit encore que Y **dépend fonctionnellement** de X .

Exemple: num – client \rightarrow nom – client.

Axiomes d' Armstrong

Soit un ensemble F de dépendances fonctionnelles entre les attributs d'une relation R . On peut déduire d'autres dépendances en utilisant les axiomes d'Armstrong :

réflexivité : si X contient Y , alors $X \rightarrow Y$.

transitivité : si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$.

augmentation : si $X \rightarrow Y$, alors $XZ \rightarrow Y$ pour tout groupe Z d'attributs de la relation.

Axiomes d' Armstrong

- Autres règles, déduites des axiomes:

union : si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow YZ$.

pseudo-transitivité : si $X \rightarrow Y$ et $WY \rightarrow Z$, alors $WX \rightarrow Z$.

décomposition : si $X \rightarrow Y$ et Y contient Z , alors $X \rightarrow Z$.

1ère Forme Normale

Une relation est en 1^{ère} forme normale si et seulement si elle ne contient que des valeurs atomiques.

AVION	construc	type	capacité	compagnie
	Airbus	A340	228	Air Algérie
	Boing	B747	432	British Airways Qantas

La relation AVION n'est pas en 1^{ère} forme normale.

1ère Forme Normale

AVION

construc	type	capacité	compagnie
Airbus	A340	228	Air Algérie
Boing	B747	432	British Airways Qantas

La relation AVION n'est pas en 1^{ère} forme normale.

AVION1

construc	type	capacité	compagnie
Airbus	A340	228	Air Algérie
Boing	B747	432	British Airways
Boing	B747	432	Qantas

La relation AVION1 est en 1^{ère} forme normale.

2ème Forme Normale

Une relation est en 2^{ème} forme normale si et seulement si :

- elle est en 1^{ère} forme normale et
- tout attribut n'appartenant pas à la clé ne dépend pas d'une partie de la clé.

Exemple

AVION

<u>Num_avion</u>	<u>construc</u>	type	capacité	compagnie
10	Airbus	A340	228	Air Algérie
20	Boing	B747	432	British Airways
30	Boing	B747	432	Qantas

Dépendances fonctionnelles :

num_avion → type,

type → capacité,

type → constructeur,

num_avion → compagnie

La relation AVION n'est pas en 2^{ème} forme normale.

Exemple

AVION

<u>Num_avion</u>	construc	type	capacité	compagnie
10	Airbus	A340	228	Air Algérie
20	Boing	B747	432	British Airways
30	Boing	B747	432	Qantas

Dépendances fonctionnelles :

num_avion → type,
type → capacité,
type → constructeur,
num_avion → compagnie

La relation AVION est en 2^{ème} forme normale.

3ème Forme Normale

Une relation est en 3^{ème} forme normale si et seulement si :

- elle est en 2^{ème} forme normale et
- tout attribut n'appartenant pas à la clé ne dépend pas d'un attribut non clé.

Exemple

AVION

<u>Num_avion</u>	construc	type	capacité	compagnie
10	Airbus	A340	228	Air Algérie
20	Boing	B747	432	British Airways
30	Boing	B747	432	Qantas

Dépendances fonctionnelles :

num_avion \rightarrow type,

type \rightarrow capacité,

type \rightarrow constructeur,

num_avion \rightarrow compagnie

La relation AVION n'est pas en 3^{ème} forme normale.

Exemple

AVION

<u>Num_avion</u>	type	compagnie
10	A340	Air Algérie
20	B747	British Airways
30	B747	Qantas

Modèle

<u>type</u>	construc	capacité
A340	Airbus	228
B747	Boing	432

3ème forme normale



Dépendances fonctionnelles :

- num_avion → type,
- num_avion → compagnie,
- type → constructeur,
- type → capacité

Forme Normale de Boyce-Codd

Une dépendance fonctionnelle $X \rightarrow Y$ est **élémentaire** si et seulement si $\forall X' \subset X : X' \not\rightarrow Y$.

Une relation est en **BCNF** si :

- elle est en 3^{ème} forme normale et
- les seules dépendances fonctionnelles élémentaires sont celles où une clé détermine un attribut.

Algorithme de Décomposition

(Rendre une relation en 3^{ème} forme normale)

- Soit F , l'ensemble de toutes les dépendances fonctionnelles définies sur les attributs d'une relation R .
- Les étapes de l'algorithme sont les suivantes:
 - 1. Trouvez l'ensemble contenant la couverture minimale.**
 2. s'il y a plusieurs dépendances fonctionnelles $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n$, regrouper tous ces attributs dans une seule relation $R_j(X, A_1, A_2, \dots, A_n)$
 3. créer une relation pour tous les attributs n'apparaissant pas dans celles obtenues jusqu'à présent.

Algorithme de Décomposition

(Rendre une relation en 3^{ème} forme normale)

- Couverture minimale (irréductible): est un sous ensemble minimum de DF élémentaires permettant de générer toutes les autres.
- Théorème: Tout ensemble de DF admet une couverture minimale, en général non unique

Algorithme de Décomposition

(Rendre une relation en 3^{ème} forme normale)

- Exemple :
- 1- code_mod → cod_filière
- 2- cod_filière → libélé_filière
- 3- ~~Code_mod → libélé_filière~~ → **Transitivité de 1 et 2**
- 4- Jour, heure, local → num_ens
- 5- Jour, heure, local → cod_filière **Transitivité de 1 et 9**
- 6- Jour, heure, local → section
- 7- Jour, heure, local → groupe
- 8- Jour, heure, local → an_étude
- 9- Jour, heure, local → cod_mod
- 10 - ~~Jour, heure, local , num_ens → cod_filière~~
Augmentation de 5

Algorithme de Décomposition

(Rendre une relation en 3^{ème} forme normale)

- La couverture minimale est la suivante :
- 1- code_mod \rightarrow cod_filière
- 2- cod_filière \rightarrow libélé_filière
- 3- Jour, heure, local \rightarrow num_ens
- 4- Jour, heure, local \rightarrow section
- 5- Jour, heure, local \rightarrow groupe
- 6- Jour, heure, local \rightarrow an_étude
- 7- Jour, heure, local \rightarrow cod_mod

• Rendre R en 3ème forme normale:

R (code_mod, cod_filière, libélé_filière, jour, heure, local, num_ens, section, groupe, an_étude)

L'ensemble de DF:

- 1- code_mod → cod_filière
- 2- cod_filière → libélé_filière
- 3- ~~Code_mod~~ → ~~libélé_filière~~ → **Transitivité de 1 et 2**
- 4- Jour, heure, local → num_ens
- 5- Jour, heure, local → cod_filière **Transitivité de 1 et 9**
- 6- Jour, heure, local → section
- 7- Jour, heure, local → groupe
- 8- Jour, heure, local → an_étude
- 9- Jour, heure, local → cod_mod
- 10 - Jour, heure, local, num_ens → cod-filière
Augmentation de 5



• La couverture minimale est la suivante :

- 1- code_mod → cod_filière
- 2- cod_filière → libélé_filière
- 3- Jour, heure, local → num_ens
- 4- Jour, heure, local → section
- 5- Jour, heure, local → groupe
- 6- Jour, heure, local → an_étude
- 7- Jour, heure, local → cod_mod

R1 (code_mod, cod_filière) **3FN**

R2 (cod_filière, libélé_filière) **3FN**

R3 (jour, heure, local, num_ens, section, groupe, an_étude, cod_mod) **3FN**

Règles sur les Relations

1. chaque **attribut** est **atomique**
2. chaque **tuple** est **unique** (les doublons ne sont pas autorisés)
3. l'ordre des attributs n'a pas de signification
4. l'ordre des tuples n'a pas de signification
5. chaque **attribut** prend ses valeurs dans **un seul domaine**
6. un **domaine** peut correspondre à **plusieurs attributs**

Algèbre relationnelle

L'algèbre relationnelle

L'**algèbre relationnelle** utilise une collection d'opérateurs qui agissent sur des relations et produisent des relations comme résultat.

Opérateurs :

- unaires : **projection**, **sélection**
- ensemblistes : **union**, **produit**, **différence**, **intersection**
- binaires (ou n-aires) : **jointure**, **division**

L'algèbre relationnelle

La projection

La **projection** d'une relation R sur un ensemble d'attributs $\{A_i\}$ de R est un sous-ensemble de R réduit aux sous-tuples ne contenant que les attributs de $\{A_i\}$.

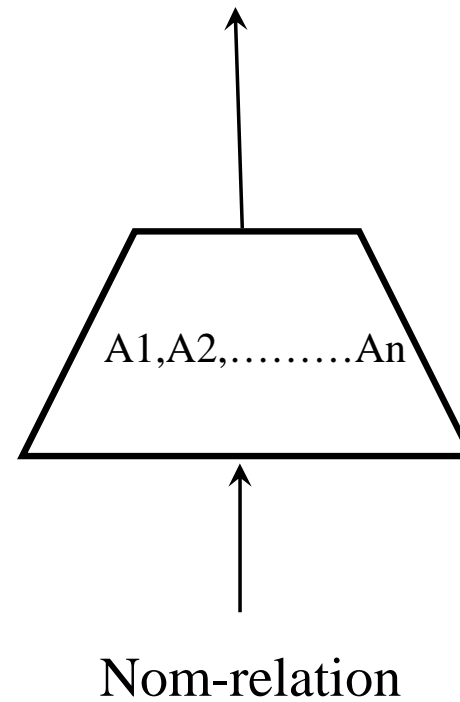
Suppression dans les tuples de tous les attributs n'appartenant pas à $\{A_i\}$, puis suppression des doublons.

Notation : $\Pi_{A_1, \dots, A_n}(\text{nom-relation})$

L'algèbre relationnelle

La projection

Représentation graphique



L'algèbre relationnelle

La projection

$\Pi_{\text{code, adresse}} (\text{Étudiant})$

Étudiant

Code	Adresse
00134	Adr1
00134	Adr1
01045	Adr2
01045	Adr2

L'algèbre relationnelle

La projection

$\Pi_{\text{code,adresse}} (\text{Étudiant})$

Code	Adresse
00134	Adr1
00134	Adr1
01045	Adr2
01045	Adr2

Code	Adresse
00134	Adr1
01045	Adr2

L'algèbre relationnelle

La sélection

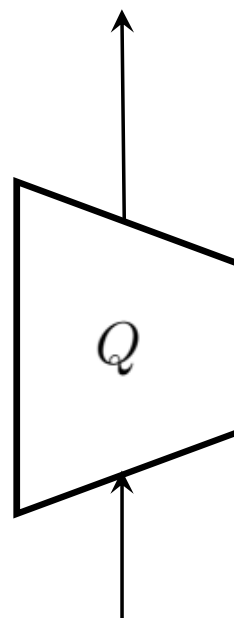
La **sélection** sur une relation R suivant une qualification Q portant sur des attributs de R est un sous-ensemble de R dont les tuples satisfont Q .

Notation : $\sigma_Q(\text{nom-relation})$

L'algèbre relationnelle

La sélection

Représentation graphique



Nom-relation

L'algèbre relationnelle

La sélection

Étudiant

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3
01190	Nom4	Pr4	Adr4

L'algèbre relationnelle

La sélection

$\sigma_{\text{Nom} = \text{'Nom2'}} (\text{Étudiant})$

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3
01190	Nom4	Pr4	Adr4

L'algèbre relationnelle

La sélection

$\sigma_{\text{Nom} = \text{'Nom2'}} (\text{Étudiant})$

Code	Nom	prénom	adresse
01045	Nom2	Pr2	Adr2

L'algèbre relationnelle

L'union, U

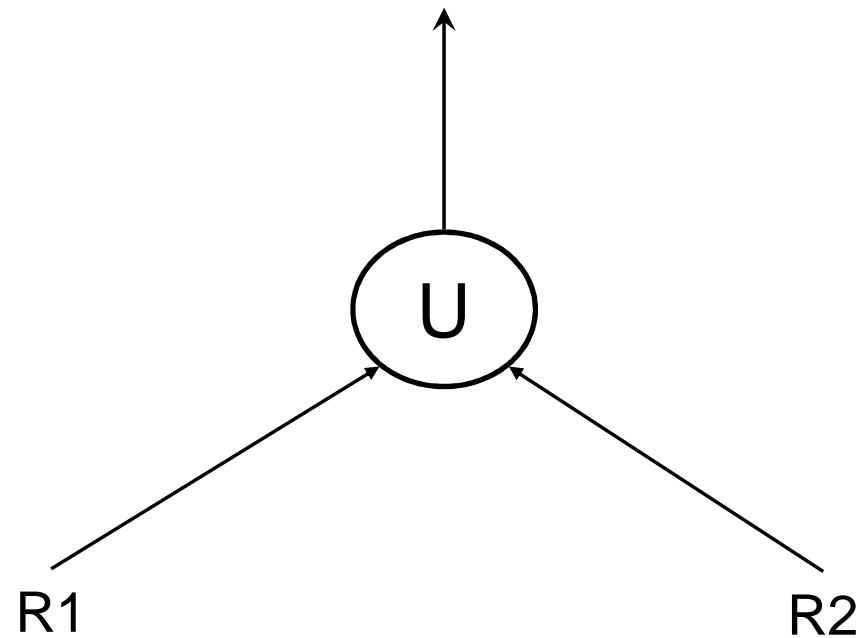
L'**union** est une opération ensembliste portant sur 2 relations R_1 et R_2 de même schéma. $R_1 \cup R_2$ est la relation de même schéma contenant les tuples de R_1 et ceux de R_2 .

Copie des tuples de R_1 et de R_2 , puis suppression des doublons.

L'algèbre relationnelle

L'union

Représentation graphique



L'algèbre relationnelle

L'union

Étudiant 1

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3

U

Étudiant 2

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01046	Nom4	Pr4	Adr4

L'algèbre relationnelle

L'union

Étudiant 1 \cup Étudiant 2

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3
01046	Nom4	Pr4	Adr4

L'algèbre relationnelle

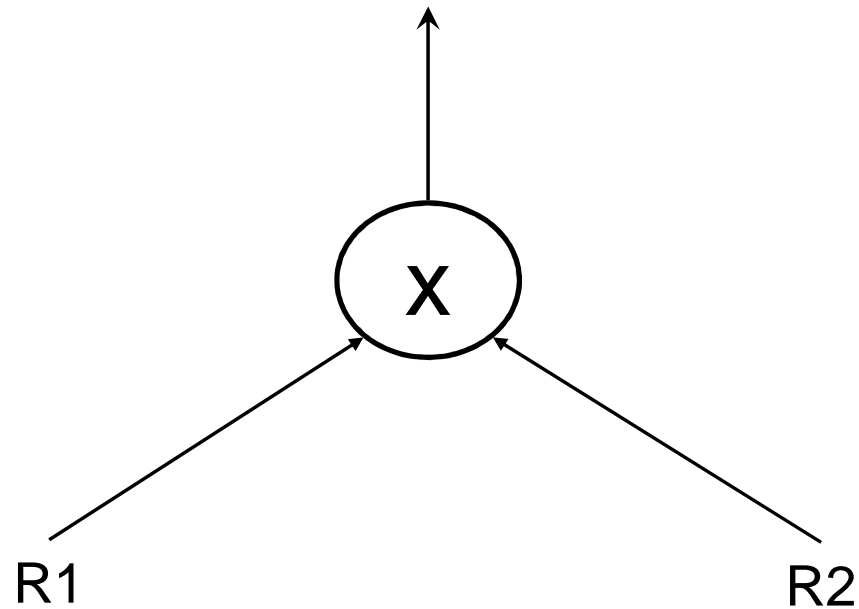
Le produit

Le **produit** est une opération ensembliste portant sur 2 relations R_1 et R_2 . $R_1 \times R_2$ est la relation ayant pour schéma la concaténation des schémas de R_1 et R_2 , et pour tuples toutes les combinaisons de tuples de R_1 et R_2 .

L'algèbre relationnelle

Le produit

Représentation graphique



L'algèbre relationnelle

Le produit

Étudiant

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2

Année

X

Année	Moyenne
2001	Bonne
2004	Passable

Cours BDD pour les développeurs

L'algèbre relationnelle

Le produit

Étudiant \times Année

Code	Nom	Prénom	Adresse	Année	Moyenne
00134	Nom1	Pr1	Adr1	2001	Bonne
00134	Nom1	Pr1	Adr1	2004	Passable
01045	Nom2	Pr2	Adr2	2001	Bonne
01045	Nom2	Pr2	Adr2	2004	Passable

L'algèbre relationnelle

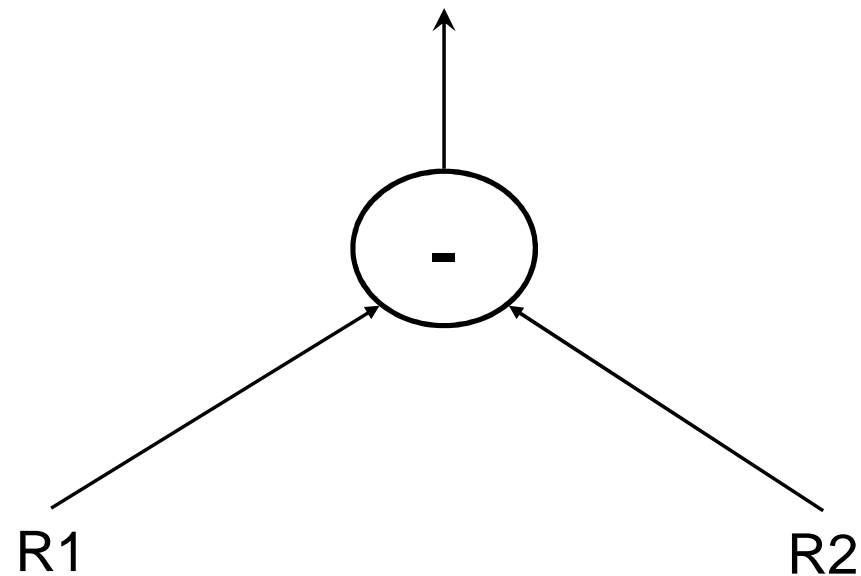
La différence

La **différence** est une opération ensembliste portant sur 2 relations R_1 et R_2 de même schéma. $R_1 - R_2$ est la relation ayant le même schéma que R_1 et R_2 , et pour tuples ceux appartenant à R_1 et pas à R_2 .

L'algèbre relationnelle

La différence

Représentation graphique



L'algèbre relationnelle

La différence

Étudiant 1

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3

-

Étudiant 2

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01046	Nom4	Pr4	Adr4

L'algèbre relationnelle

La différence

Étudiant 1 - Étudiant 2

Code	Nom	prénom	adresse
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3

L'algèbre relationnelle

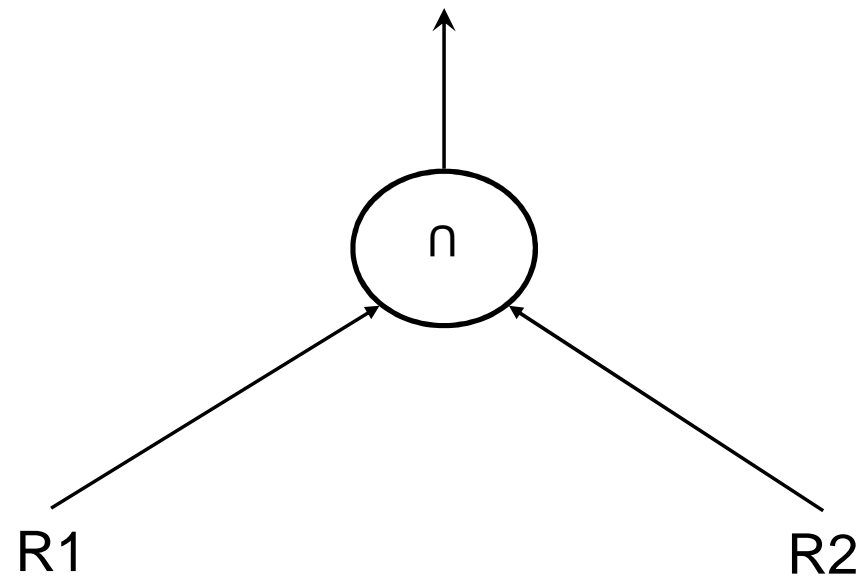
L'intersection

L'**intersection** est une opération ensembliste portant sur 2 relations R_1 et R_2 de même schéma. $R_1 \cap R_2$ est la relation ayant le même schéma que R_1 et R_2 , et pour tuples ceux appartenant à la fois à R_1 et à R_2 .

L'algèbre relationnelle

L'intersection

Représentation graphique



L'algèbre relationnelle

L'intersection

Étudiant 1

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01045	Nom2	Pr2	Adr2
00033	Nom3	Pr3	Adr3

n

Étudiant 2

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1
01046	Nom4	Pr4	Adr4

L'algèbre relationnelle

L'intersection

Étudiant 1 n Étudiant 2

Code	Nom	prénom	adresse
00134	Nom1	Pr1	Adr1

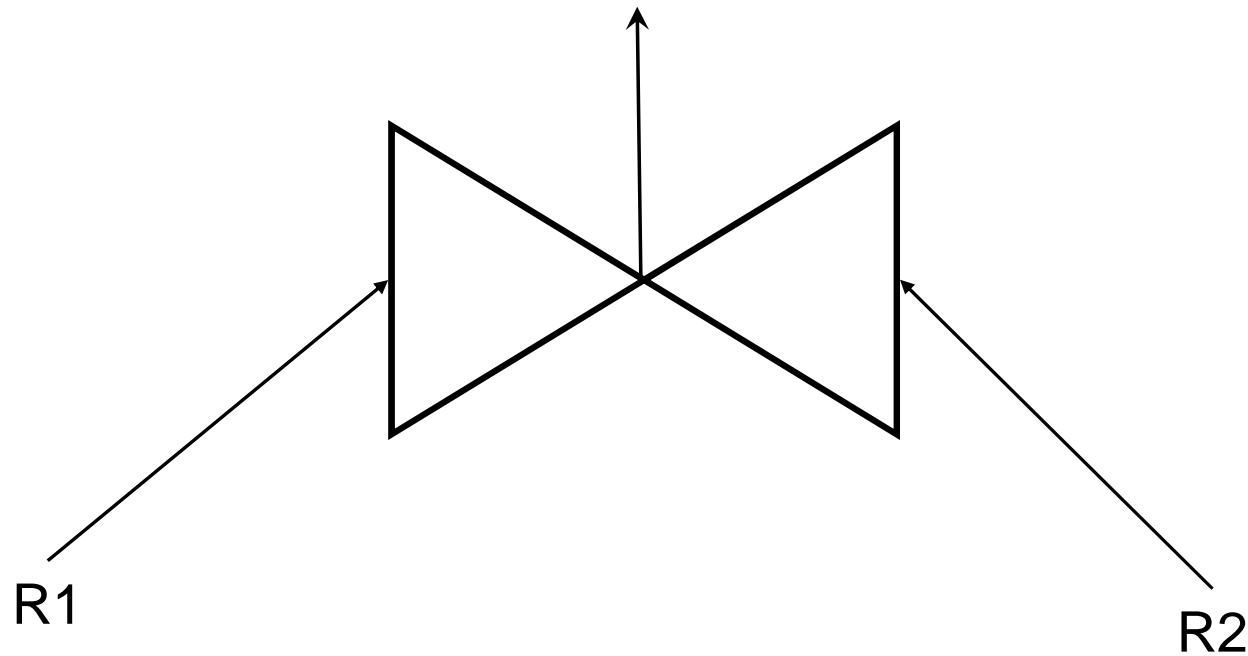
L'algèbre relationnelle

La Jointure

La **jointure** est une opération ensembliste portant sur 2 relations R_1 et R_2 . $R_1 \bowtie_{\text{attributs}} R_2$ est la relation dont les attributs sont ceux de R_1 et ceux de R_2 , et dont les tuples sont obtenus en composant un tuple de R_1 et un tuple de R_2 ayant la même valeur pour les attributs précisés.

L'algèbre relationnelle

La jointure



$$\text{Jointure} = \sigma_{\text{condition}} (R1 \times R2)$$

L'algèbre relationnelle

La jointure

Étudiant \bowtie_{code} Prêt

Étudiant

Code	Nom	Prénom	Adresse
0134	Nom1	Pr1	Adr1
0178	Nom2	Pr2	Adr2
1045	Nom3	Pr3	Adr3
1196	Nom4	Pr4	Adr4

Prêt

Code	Cod-livre	Date
0178	MA12	12-03-2015
1182	IN08	12-03-2015

L'algèbre relationnelle

La jointure

Étudiant ⋈ Prêt
code

Code	Nom	Prénom	Adresse	Cod-livre	Date
0178	Nom2	Pr2	Adr2	MA12	12-03-2015

SQL (Structured Query Language)

SQL

- langage relationnel permettant de :
 - définir et modifier le schéma d'une base de données relationnelle
 - interroger et modifier une base de données relationnelle
 - contrôler la sécurité et l'intégrité de la base
- langage interactif ou intégré dans un langage de programmation

SQL

SQL est un **langage relationnel** \Rightarrow on manipule des tables et on obtient des tables

Une instruction SQL = une **requête**

3 familles d'**opérations** :

- **LDD = Langage de Définition des Données** : description de la structure de la base de données (tables, attributs)
- **LMD = Langage de Manipulation de Données** : manipulation des tables
- **LCD = Langage de Contrôle des Données** : gestion du contrôle et de la sécurité de la base de données

SQL

- Création d'une Table

1) **CREATE TABLE** Banque (Code_Agence CHAR(50), Ville CHAR(20) **NOT NULL DEFAULT** 'Nom_Ville', **PRIMARY KEY** (Code_Agence)) ;

2) **CREATE TABLE** Etudiant (Matricule CHAR(9),);

CREATE TABLE Module (Code_Mod CHAR(6),);

CREATE TABLE Etude_Mod (
Matricule CHAR(9),
Code_Mod CHAR(6),
Note Integer **CHECK (Note BETWEEN 0 and 20)**,
Decision CHAR(8) **CHECK (Decision IN ('Admis', 'Ajourné'))**,
PRIMARY KEY (Matricule , Cod_Mod),
FOREIGN KEY (Matricule) REFERENCES Etudiant,
FOREIGN KEY (Code_Mod) REFERENCES Module) ;

- Suppression d'une Table

DROP TABLE Banque;

SQL

- **Insertion de tuples**

INSERT INTO <NomTable> (A1, A2, ... An) VALUES (v1, v2, ... vn);

Exemple:

INSERT INTO Étudiant (code, nom, prénom, adresse) values
(0456,nom1, pr1,adr1);

SQL

- **Modification de tuples**

UPDATE <NomTable>

SET A1=v1, A2=v2, ... An=vn

WHERE condition;

Exemple :

UPDATE Étudiant

SET adresse='Alger'

WHERE code='0234';

SQL

- **Suppression de tuples**

DELETE FROM <NomTable>
WHERE condition;

Exemple :

DELETE FROM Étudiant
WHERE code='0234';

SQL

SELECT liste colonnes (projetées, calculées)

FROM listes tables

WHERE prédicat de jointure AND prédicat de sélection;

Exemple1

SELECT *

FROM Produit

WHERE codeP='P653';

Exemple2

SELECT Num_Emp , Nom_Emp

FROM EMPLOYES , DEPARTEMENTS

WHERE (EMPLOYES•Num_Dept=DEPARTEMENTS•Num_Dept) **AND** (Ville = 'ANNABA');

SQL

SELECT liste colonnes (projetées, calculées)

FROM listes tables

[**WHERE** prédicat de jointure AND prédicat de sélection]

[**GROUP BY** colonnes de groupement]

[**HAVING** prédicat de sélection de groupes]

[**ORDER BY** critères et ordre de tri];

SQL

- Prédicat de sélection composé par:
 - ☐ Comparateurs arithmétiques (<, <=, >, >=, != (<>), =)
 - ☐ Connecteurs logiques (AND, OR, NOT)
 - ☐ Comparateur de chaînes LIKE
 - ☐ Intervalle BETWEEN
 - ☐ Liste IN
 - ☐ Valeurs nulles IS NULL
 - ☐ Requête imbriquée IN, EXISTS

SQL

SQL offre aussi un certain nombre de fonctions dites fonctions de groupes qui peuvent être utilisées dans l'expression d'une requête. Les plus importantes sont :

- **AVG** : permet de calculer une **MOYENNE**
- **SUM** : permet de calculer une **SOMME**
- **COUNT** : permet de compter des tuples
- **MAX** : permet de calculer un **maximum**
- **MIN** : permet de calculer un **minimum**
- **GROUP BY** : permet de créer des sous-ensembles de tuples
- **HAVING** : permet de tester si une condition est vérifiée par un groupe de tuples

SQL

- **Tri du résultat d'une requête : la clause ORDER BY**

La clause ORDER BY permet de trier le résultat retourné par une requête. L'ordre peut être ascendant (ASC) ou descendant (DESC)

- **Exemple1**

Q: Liste des employés du département 10 par ordre décroissant de leur salaire?

```
SELECT Nom_Emp  
FROM EMPLOYES  
WHERE Num_Dept = 10  
ORDER BY Salaire DESC;
```

SQL

Exemple2 :

```
SELECT region, COUNT(nomStation)
FROM Station
GROUP BY region
HAVING COUNT(nomStation) >= 3;
```