Université Mohammed V - Rabat
Ecole Mohammadia d'Ingénieurs

Filière Réseaux et Télécom

Bases de Données

Normalisation & SQL

Mr N.EL FADDOULI
nfaddouli@gmail.com

2022-2023

Plan

- Généralités:
  - Définition de Bases de Données
  - Le modèle relationnel
- Algèbre relationnelle
- SQL
- Conception de Bases de Données:
  - Besoins et principe de la normalisation
  - Dépendances fonctionnelles
  - Les 5 formes normales
- PL/SQL

Définitions: Base de Données (1)

Base de données:

- Une Base de données est un ensemble de données **structurées** avec le **minimum de redondance**, mémorisées sur un support **permanent** et qui peut être **partagée** par plusieurs applications et interrogeables de façon **sélective** par plusieurs utilisateurs.

- Une BD est gérée par un **S**ystème de **G**estion de **B**ases de **D**onnées (**SGBD**): Assure la **structuration**, le **stockage**, la **consultation** et la **mise à jour** des données.

Définitions: Base de Données (2)

Exemples d'utilisation des BDs:

- Bibliothèques: gestion des emprunts, …
- Banques: gestion des comptes, …
- Commerces: gestion des commandes, …
- Administrations: gestion du personnel, …
- Internet: moteur de recherche, …
- etc. (les BDs sont présentes partout!)

Définitions: Base de Données (3)

Exemple introductif: Gestion d'une Bibliothèque

Données:

- Livres
- Emprunteurs
- Emprunts

Quelques Traitements possibles:

- Stocker des livres de la bibliothèque;
- Modifier des livres;
- Retrouver un livre d'après sa cote, son titre ou son auteur;
- Enregistrer un emprunt (Créer une fiche d'emprunt), ….

Définitions: Base de Données (4)

Livres

Cote	Titre	Auteur	ISBN
15	Le Langage C	Claude Delanoy	2-266-0865-4
4	L'algèbre de base	Frank Shin	2-123-2301-5
24	Bases de données	Georges Gardarin	2-212-11281-5
25	Oracle 10g	Ian Abramson	2-7440-1778-7
67	Les BD XML	Georges Gardarin	2-266-01303-3

Définitions: Base de Données (5)

Emprunteurs

Matricule	Nom_Prenom	Date_De_Naissance
12308	Alami Samir	1981
12408	Semah Amal	1985
12508	Brahimi Said	1980

Emprunts

Cote	Matricule	Date_Emprunt	Date_Remise
15	12308	11/10/2007	2/11/2007
4	12408	1/11/2007	
24	12308	12/10/2007	7/11/2007

Définitions: SGBD (1)

SGBD:

Logiciel(s) assurant **structuration**, **stockage**, **maintenance**, **mise à jour et consultation** des données d'une BD en exécutant des requêtes exprimées selon une vue logique de la BD.

Vue logique: Représente la vision de l'utilisateur qui ne se soucie pas des détails de stockage et d'accès aux données (*Vue physique*).

Exemple: Vue logique d'une BD de bibliothèque

Il y a trois **tables**: livres, emprunteurs et emprunts contenant les données utilisées (cote, titre, ….)

Exemples de SGBD: Access, Oracle, Ingres, MySQL, …

Définitions: SGBD (2)

Fonctionnalités :

1- Description des données qui seront stockées (**LDD**)):

- Structure, Contraintes, Droits d'accès, ….

2- Manipulation des données (recherche, ajout, modification et suppression) avec le **LMD**.

3- Partage des données:

- Utilisation simultanée des données par différentes applications.
- Gestion des accès concurrents (techniques de verrouillage, …).
- Cohérence des données

Définitions: SGBD (3)

4- Satisfaction des contraintes d'intégrité pour conserver la validité des données (Exemple: un emprunt doit concerner un livre qui existe dans la BD).

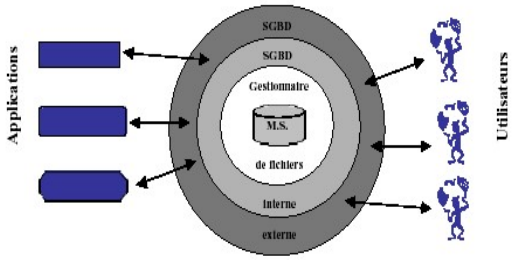
5- Sécurité des données:

- Les données doivent être protégées contre les accès non autorisés.
- Récupérer les données après panne.

6- Confidentialité des données: chaque utilisateur doit avoir accès aux données auxquelles il a droit.

Définitions: SGBD (4)

Architecture d'un SGBD:



• Gestionnaire de fichiers

- Stockage des données sur des supports physique.
- Mécanisme de recherche par le contenu.
- Optimisation du stockage.

Définitions: SGBD (5)

• SGBD interne (système d'accès au données)

- Assemblage des tables à partir des données dans les fichiers.
- Placement des tables dans les fichiers.
- Gère les liens et l'accès rapide.

• SGBD externe

- Analyse et interprétation des requêtes des applications et des utilisateurs interactifs.
- Mise en forme des données pour les échanges avec le monde extérieur.

Définitions: Organisation des fichiers (1)

Un fichier: Récipient d'informations sur disque.

- Il est caractérisé particulièrement par:
  - UN NOM
  - UN CREATEUR
  - UNE DATE DE CREATION
  - UN EMPLACEMENT EN MEMOIRE SECONDAIRE
  - **UNE ORGANISATION**

-Il est contient un ensemble d'articles (Exemple: des livres).

Méthode d'accès: Méthode d'exploitation du fichier utilisée par les programmes d'application pour sélectionner des articles.

Définitions: Organisation des fichiers (2)

1- Méthode d'Accès Séquentielle:

Pour accéder à un article donné: On doit lire tous les articles depuis le premier jusqu'à l'article recherché.

- ➔ Fichier séquentiel
- ➔ Beaucoup d'accès disque.

2- Méthodes d'Accès Sélectives:

Ensemble de méthodes permettant l'accès à un article donné au moyen de quelques accès disque (idéalement 1).

Exemple: Méthodes d'accès par **index** (Arbre B, Arbre B+ , …)

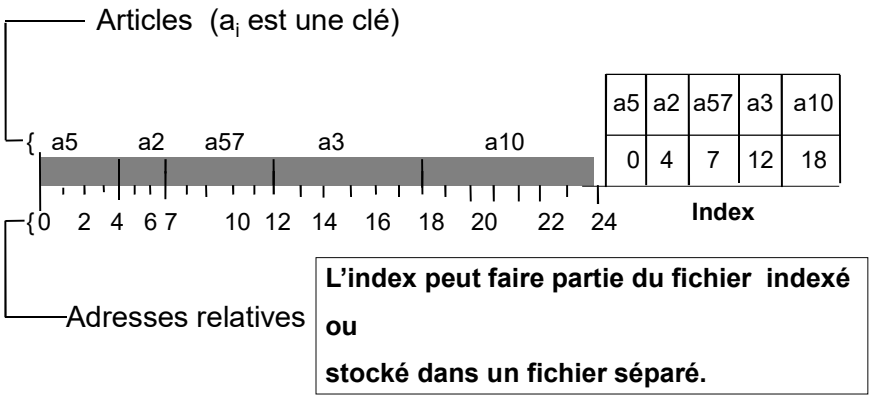
Définitions: Organisation des fichiers (3)

Méthodes d'Accès par Index:

- Chaque article possède une **clé** qui est une information (ou donnée). Une clé permet de:
  - Identifier un article (Par exemple: la cote d'un livre).
  - Sélectionner un article unique dans le fichier.
- Chaque article possède dans le fichier une **adresse relative** qui indique le déplacement par rapport au début du fichier.
- Principe de base de l'organisation et méthode d'accès par index: **Associer** à la **clé** d'un article son **adresse relative** dans le fichier.

Définitions: Organisation des fichiers (4)

Méthodes d'Accès par Index (Exemple):



L'index peut faire partie du fichier indexé ou stocké dans un fichier séparé.

Définitions: Cycle de Vie d'une BD

1- Conception de la base: Déterminer les **informations** qu'il conviendra de mettre dans la base de données ainsi que sa **structure**.

2- Implantation des données: Transmettre la **structure** de la BD et les contraintes d'intégrité au SGBD choisi (au moyen du LDD)

3- Utilisation: Interroger et mettre à jour les données (au moyen du LMD)

4- Maintenance : Correction, évolution

Définitions: Modélisation des données

- Représentation d'une partie du monde réel utilisant les concepts d'un modèle de représentation.

  Par exemple: la modélisation d'une «***personne***» pour une application de gestion des inscriptions d'étudiants sera différente de celle pour une application de gestion médicale. Les données choisies pour la modélisation diffèrent.

- Il existe plusieurs modèles de représentation:
  - **Modèle hiérarchique et modèle réseau** (1960).
  - **Modèle Relationnel** (1970).
  - **Modèle Objet** (1990)

Modèle Relationnel: Définition

- Le plus utilisé dans les BD et les SGBD (BDR et SGBDR).
- Fondé sur le concept mathématique de ***relation*** (théorie des ensembles):

  Sous-ensemble du produit cartésien de différents ensembles (***domaines***).
- Représentation des données sous forme de **tables** à deux dimensions contenant des valeurs simples.
- **Concepts:**
  - **Domaine, Relation, Attribut**
  - **Tuple**
  - **Schéma conceptuel**
  - **Clé**
  - **Contraintes**

Modèle Relationnel: Domaine

Domaine

- **Ensemble** de valeurs de **même type**.
- Caractérisé par un **nom**.

Exemple:

D1 = { Informatique, Électrique, Civil, Mécanique, Procédés, Industriel, MIS }

IN (ensemble des entiers)

Dates

Texte (suite de caractères)

Modèle Relationnel: Relation

Relation

- **SOUS-ENSEMBLE** du produit cartésien de ***n*** domaines D_i.
- Caractérisée par un nom.
- Une relation R: $\mathbf{R \subseteq D_1 \times D_2 \times D_3 \ldots D_n}$
- Les ***n*** domaines ne sont pas nécessairement distincts.
- Relation de ***n-aire***, d'***arité n*** ou de ***degré n***.

Exemple:

Livres $\subseteq$ **IN**$\times$ **Texte** $\times$ **Texte** $\times$ **Texte**

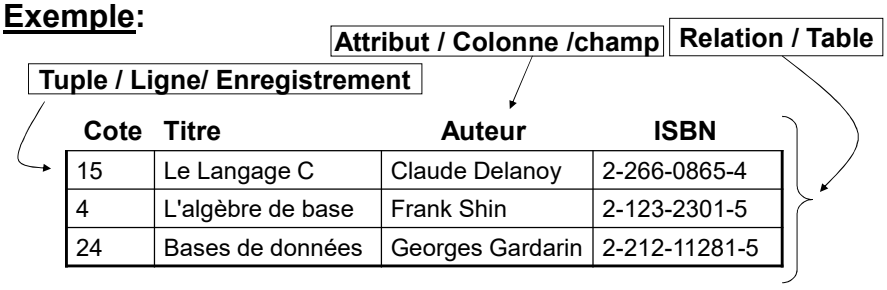
Livres est d'arité 4

Modèle Relationnel: Relation

Une relation est composée de vecteurs (lignes) qui forment une table à deux dimensions:

- Chaque ligne correspond à un vecteur.
- Chaque colonne correspond à un domaine du produit cartésien.

Exemple:



Cote	Titre	Auteur	ISBN
15	Le Langage C	Claude Delanoy	2-266-0865-4
4	L'algèbre de base	Frank Shin	2-123-2301-5
24	Bases de données	Georges Gardarin	2-212-11281-5

Modèle Relationnel: Attribut

Attribut

- Nom donné à une composante d'une relation.

Exemple: **Cote** est un attribut de la relation **Livres**

Tuple (ou n-uplet)

- Un élément d'une relation.
- Une ligne dans une table

Exemple:

t1 (Cote:15, Titre: 'Le Langage C', Auteur: 'Claude Delanoy', ISBN: 2-266-0865-4)

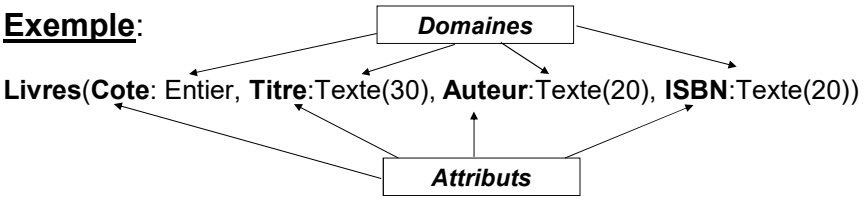
est un tuple de la relation **Livres**.

Modèle Relationnel: Schéma conceptuel

Schéma conceptuel de la relation :

Nom de la relation suivi de la liste des attributs et de leurs domaines.

Exemple:



Modèle Relationnel: Schéma conceptuel

- On peut donner aux domaines des noms relativement au SGBD utilisé.

Exemple:

Livres (**Cote**:INTEGER,**Titre**:CHAR(50),**Auteur**:CHAR(50),
ISBN:CHAR(50))

- La plupart du temps, les domaines sont implicites et découlent du nom de l'attribut

Exemple:

Livres(**Cote**, **Titre**, **Auteur**, **ISBN**)

Modèle Relationnel: Clé candidate

- Un ensemble minimal des attributs dont les valeurs identifient de manière unique un tuple.
- La valeur d'une clé candidate est donc distincte pour tous les tuples.
- Toute relation a *au moins une clé candidate* et peut en avoir plusieurs.

Exemple:

Salarié (Matricule, nom, prénom, CIN, salaire, fonction)

Les clés candidates sont: Matricule et CIN

Modèle Relationnel: Clé primaire

- Attribut ou ensemble minimale d'attributs dont les valeurs **identifient de manière unique** chaque **tuple** de la relation.

Exemple:

Livres(Cote, **Titre**, **Auteur**, **ISBN**)

Emprunteurs(Matricule,**Nom_Prenom, Date_De_Naissance)**

Emprunt (Matricule, Cote, Date_Emprunt, **Date_Remise**)

Remarque: La clé primaire fait partie de l'ensemble des clés candidates.

Modèle Relationnel: Clé Étrangère

- Une clé étrangère dans une relation **R1** est un attribut ou ensemble d'attributs qui représentent une **clé primaire** dans une autre relation **R2**.
- On précède une clé étrangère par le caractère dièse (#)

Exemple:

Emprunt(#Matricule, #Cote, Date_Emprunt, Date_Remise)

Remarque: La clé étrangère peut avoir un nom différent de la clé primaire correspondante.

Modèle Relationnel: Contrainte d'intégrité

- Toute règle spécifiant les valeurs permises pour certaines données, éventuellement en fonction d'autres données, et permettant d'assurer une certaine cohérence de la BD:
  - **Contrainte de l'unicité de clé.**
  - **Contrainte référentielle.**
  - **Contrainte de domaine.**
  - **Contrainte d'entité.**

Modèle Relationnel: Contrainte (1)

Contrainte d'unicité de clé

- Elle permet d'assurer l'unicité de la clé primaire: Deux tuples dans une relation ne peuvent pas avoir la même valeur de la clé primaire.

Contrainte référentielle

- Une contrainte d'intégrité portant sur une relation R1, consiste à imposer que **la valeur d'un ou de plusieurs attributs** apparaissent comme **valeur de clé primaire** dans une autre relation R2.

Exemple: Pour chaque **tuple** de la relation **Emprunt**, il faut que la valeur de l'attribut **Cote** apparaisse dans **un tuple** de la relation **Livres** comme valeur de sa clé primaire **Cote**.

Modèle Relationnel: Contrainte (2)

Contrainte de domaine

- Elle impose qu'une colonne d'une relation doit comporter des valeurs appartenant à une plage de valeurs ou à une liste de valeurs.

Exemple:

Salaire $\geq$5000 et $\leq$30000

Couleur $\in$ {Bleu, Blanc, Rouge}

Modèle Relationnel: Contrainte (3)

Valeurs Nulles et Clés

- Si la valeur d'un attribut est **inconnue** au moment de l'insertion d'un tuple dans une relation: on lui attribue une valeur conventionnelle, appelée **valeur nulle** (**Null**)

Exemple: Dans la relation Emprunt, Dat_Emprunt est égale à NULL si le livre emprunté n'est pas encore remis.

Contrainte d'entité

- Une contrainte d'intégrité imposant que **toute relation possède une clé primaire** et que **tout attribut participant à cette clé primaire soit non nul.**

Modèle Relationnel: Instance

Instance (Extension)

- Le schéma d'une relation représente les propriétés (attributs) des tuples qu'elle va contenir au cours du temps.
- Une **table** représente une **instance** d'une relation, càd une vue des **tuples** qu'elle contient **à un instant donné**.
- **Instance d'une BD** est l'ensembles des instances des relations de la BD: l'ensemble des table de la BD.

Algèbre Relationnelle

- Le modèle relationnel (algèbre relationnel) basé sur le concept de relation possède un ensemble d'opérations qu'on peut appliquer aux relations.
- Ces opérations sont le fondement du langage SQL.
- Les opérations traitées sont:
  - **Projection**
  - **Restriction**
  - **Produit Cartésien**
  - **Jointure.**

Algèbre Relationnelle: Projection (1)

Projection: Opération sur une relation R1 consistant à composer une relation R2 en relevant à la relation initiale (R1) tous les attributs non mentionnés en opérandes, et en éliminant les éléments en double qui sont conservés une seule fois.

- Soit la relation **R1(A1, A2,….Am)**, la **projection** de R1 comportant les attributs **Ai, Aj, …Ap** est notée par l'une des notations suivantes: **PROJECT** (**R1, Ai, Aj,…,Ap**)

∏Ai, Aj,…,Ap (R1)

Algèbre Relationnelle: Projection(2)

Exemple:

Livres

Cote	Titre	Auteur	ISBN
15	Le Langage C	Claude Delanoy	2-266-0865-4
4	L'analyse numérique	Frank Shin	2-123-2301-5
24	Bases de données	Georges Gardarin	2-212-11281-5

Algèbre Relationnell: Projection(3)

Exemple:

PROJECT(Livres ,Titre, Auteur**)**

Titre	**Auteur**
Le Langage C	Claude Delanoy
L'algèbre de base	Frank Shin
Bases de données	Georges Gardarin

Algèbre Relationnelle: Restriction(1)

Restriction: Opération sur une relation R1 produisant une relation R2 de même schéma (***mêmes attributs***), mais comportant seulement les tuples qui vérifient une **condition** précisée en argument.

- Soit la relation **R1,** la **Restriction** de **R1** selon une **condition** donnée est notée par l'une des notations suivantes:

  RESTRICT (**R1, Condition**)

  $\sigma_{\text{Condition}}$ **(R1)**

Algèbre Relationnelle: Restriction(2)

Exemple:

Emprunts

Cote	Matricule	Date_Emprunt	Date_Remise
15	12308	11/10/2007	2/11/2007
4	12308	1/11/2007	
30	12408	4/11/2007	11/11/2007
24	12308	12/10/2007	7/11/2007

Algèbre Relationnelle: Restriction(3)

Exemple:

Restrict(Emprunts, Matricule=12308)

Cote	Matricule	Date_Emprunt	Date_Remise
15	12308	11/10/2007	2/11/2007
4	12308	1/11/2007	
24	12308	12/10/2007	7/11/2007

Algèbre Relationnelle:
Combiner Restriction et projection

- Sélectionner d'une relations R1 ***certains attributs*** des tuples vérifiant une ***condition***.
- Notation algébrique: $\prod Ai, Aj,\ldots,Ap\ (\sigma_{Condition}(R1))$

Exemple:

Le titre et l'auteur du livre dont la cote est 24

$\prod$ Titre, Auteur ($\sigma_{Cote=24}$(Livres))

Algèbre Relationnelle:
Produit Cartésien (1)

- Opération portant sur deux relations **R1** et **R2**, ou plusieurs, consistant à construire une troisième relation ayant pour **schéma (colonnes) la concaténation de ces deux relations** et pour **tuples toutes les combinaisons des tuples des relations R1 et R2**.
- Notations possibles:

  R1 × R2

  PRODUCT (R1, R2)

Algèbre Relationnelle:
Produit Cartésien (2)

Exemple :

Département

Code	Nom_Dept
1	Informatique
2	Civil
3	Électrique

Étudiant

Matricule	Nom_Prenom	Code_Dept
12308	Alami Samir	1
12408	Semah Amal	3

Algèbre Relationnelle:
Produit Cartésien (3)

Département × Étudiant

Code	Nom_Dept	Matricule	Nom_Prenom	Code_Dept
1	Informatique	12308	Alami Samir	1
1	Informatique	12408	Semah Amal	3
2	Civil	12308	Alami Samir	1
2	Civil	12408	Semah Amal	3
3	Électrique	12308	Alami Samir	1
3	Électrique	12408	Semah Amal	3

Algèbre Relationnelle: Jointure (1)

• Consiste à combiner deux relations **R1** et **R2** tuple à tuple par produit cartésien (ligne à ligne) en vérifiant la concordance entre certains attributs (colonnes) des deux relations (en général: une **clé primaire** avec une **clé étrangère**).

• C'est une projection d'une restriction sur un produit cartésien entre plusieurs relations afin de: sélectionner **certains attributs** des **tuples** du produit cartésien **vérifiant une condition** donnée.

• Notations possibles: R1 ▷◁ R2 (*Condition*)

Join (R1,R2,*Condition*)

Algèbre Relationnelle : Jointure (2)

Exemple:

Join (**Département** , **Étudiant**, *Code* **=** *Code_Dept*)

ou

Join(**Département**,**Étudiant**,*Département*.*Code*=
Étudiant.*Code_Dept*)

Algèbre Relationnelle : Jointure (3)

Département × Étudiant

Code	Nom_Dept	Matricule	Nom_Prenom	Code_Dept
1	Informatique	12308	Alami Samir	1
1	Informatique	12408	Semah Amal	3 ←
2	Civil	12308	Alami Samir	1
2	Civil	12408	Semah Amal	3
3	Électrique	12308	Alami Samir	1
3	Électrique	12408	Semah Amal	3 ←

Algèbre Relationnelle : Jointure (4)

Join (**Département** , **Étudiant**, *Code* = *Code_Dept*)

Code	Nom_Dept	Matricule	Nom_Prenom	Code_Dept
1	Informatique	12308	Alami Samir	1
3	Électrique	12408	Semah Amal	3

Algèbre Relationnelle : Jointure (5)

Exercice:

Project (Join (Département, Étudiant,Code=Code_Dept),

Nom_Dept,Nom_Prenom)

Résultat ?

……

……

Algèbre Relationnelle : Jointure (6)

• S'il y des attributs de même nom dans les relations de la jointure, on doit **préfixer** ces attributs par les noms de leurs relations respectives comme suit: ***Relation.Attribut***

Exemple: **Ville** **(Code_V, Nom_Ville)**

Ecole (Code_E, Nom_Ecole, NbSalle, #Code_V)

Ville

Code_V	Nom_Ville
1	Rabat
2	Casa
3	Tanger

Ecole

Code_E	Nom_Ecole	NbSalle	Code_V
120	Mohammed V	20	1
130	Imam Malek	15	1
140	Al Massira	25	3

Join (**Ville**, **Ecole**, *Ville.Code_V* = *Ecole.Code_V*) ?

Algèbre Relationnelle : Jointure (7)

Join (**Ville**, **Ecole**, *Ville.Code_V* = *Ecole.Code_V*)

Code_V	Nom_Ville	Code_E	Nom_Ecole	NbSalle	Code_V
1	Rabat	120	Mohammed V	20	1
1	Rabat	130	Imam Malek	15	1
3	Tanger	140	Al Massira	25	3

SQL: Définition

Structured Query Langage(SQL):

- Langage de base dans les SGBD
- Langage de Définition des Données (**LDD**):

  Il permet la définition des structures des tables et des autres objets de la BD (index, …)
- Langage de Manipulation des Données (**LMD**):

  Il permet de faire des recherches (sélection) et des mises à jour (ajout, suppression, modification) des données de la base.

SQL: LDD Définition des schémas (1)

Création de tables (relations)

CREATE TABLE *nom_de_table* **(** *colonnes et contraintes* **)**

Exemple 1:

```
CREATE TABLE Emprunteurs
        ( Matricule  NUMBER(5) PRIMARY KEY,
          Nom_Prenom VARCHAR2(60) NOT NULL,
          Dat_Nais  DATE );
```


SQL: LDD Définition des schémas (2)

Exemple 2:

```
CREATE TABLE Livres
        ( Cote  NUMBER(5) PRIMARY KEY,
          Titre VARCHAR2(255) NOT NULL,
          Auteur VARCHAR2 (255),
          ISBN  VARCHAR2 (100)
          );
```

SQL: LDD Définition des schémas (3)

Exemple 3:

```
CREATE TABLE Emprunt
(   Matricule NUMBER(5),
    Cote NUMBER(5),
    Dat_Emprunt DATE NOT NULL,
    Dat_Remise  DATE,
PRIMARY KEY (Matricule, Cote, Dat_Emprunt),
FOREIGN KEY(Matricule) REFERENCES Emprunteurs (Matricule),
 FOREIGN KEY (Cote) REFERENCES Livres (Cote)     ) ;
```


SQL: LDD Définition des schémas (4)

Exemple 4: Personne_Eau(CIN, Nom, Dat_B,Quantité)

```
CREATE TABLE Personne_Eau
  ( CIN  VARCHAR2(10)  PRIMARY KEY,
    Nom  VARCHAR2(50) NOT NULL,
    Dat_B  DATE,
    Quantité  NUMBER(2) DEFAULT 1,
    CHECK (Quantité BETWEEN 1 AND 5)
)
```

SQL: LDD Définition des schémas (5)

Exemple 5: Créer une table à partir du résultat d'une requête

```
CREATE TABLE Algebre AS SELECT * FROM Livre
  WHERE titre like '%Algèbre%';

CREATE TABLE Algebre AS SELECT * FROM Livre
  WHERE 1=2;
```


SQL: LDD Définition des schémas (6)

Exercice:

Donnez le code SQL permettant de créer les tables de la BD dont le schéma conceptuel est le suivant:

Etudiant (Matricule, Nom, Prénom, DatN, Niveau)

Matière (CodMat, NomMat, Coefficient, VHoraire)

Notes(#Matricule, #CodMat, Note)

N.B: Les coefficients sont entre 1 et 5 (valeur par défaut = 1)

Les notes sont entre 0 et 20 (valeur par défaut = 0) .

SQL: Mise à jours de données(1)

Insertion de données

- **Insertion d'une ligne complète**

INSERT INTO *Table* VALUES (*Val_Attr1*, *Val_Attr2*, ….)

Respecter l'ordre et le type des colonnes

Exemple : **INSERT INTO** Ville **VALUES (**234, 'Kenitra'**)**

- **Insertion d'une ligne incomplète**

INSERT INTO *Table* (*Attr1*, *Attr2*, …) **VALUES** (*Val_Attr1*, *Val_Attr2*, ….)

Les attributs non indiqués = la valeur par **défaut** ou **NULL**.

Exemple : **INSERT INTO** Livres (Cote, Titre)
VALUES (100, 'XML'**)**

SQL: Mise à jours de données(2)

Modification de données

- **Modification de toutes les lignes**

UPDATE *table* **SET** *attr1* **=** *expression1* **,** *attr2* **=** *expression2*, …

Exemple: incrémenter de 1 le nombre de salles

UPDATE Ecole **SET** NbSalle **=** NbSalle +1

- **Modification de lignes satisfaisant une condition**

UPDATE *table* **SET** *attr1* **=** *expression1* **,** *attr2* **=***expression2* **,**…
WHERE *condition*

Exemple: Doubler le nombre de salle de l'école N° 23

UPDATE Ecole **SET** NbSalle **=** 2*NbSalle
WHERE Code_Ecole = 23

SQL: Mise à jours de données(3)

Suppression de données

- **Suppression de toutes les lignes**

DELETE FROM *table* , **TRUNCATE***table*

N.B: On ne peut pas annuler (Rollback) la suppression par truncate.

Exemple : Vider la table Ville

DELETE FROM Ville

- **Suppression des lignes satisfaisant une condition**

DELETE FROM *table*

WHERE *condition*

Exemple: Supprimer les école dont le nombre de salles <10

DELETE FROM Ecole

WHERE NbSalle <10

SQL: Projection (1)

Syntaxe

SELECT ***Attributs***

FROM ***Table***

Exemple : Titres et auteurs de tous les livres disponibles

SELECT Titre, Auteur

FROM Livres

Remarque: Pour ne pas avoir des tuples en double, on utilise **DISTINCT**

Exemple: Sélection de tous les auteurs

SELECT **DISTINCT** Auteur

FROM Livres

SQL : Projection (2)

- Si on veut sélectionner toutes les colonnes (attributs) d'une table (relation)

```
SELECT *
FROM   Table
```

Exemple : Tous les attributs de tous les livres disponibles

```
SELECT *
FROM   Livres
```


SQL : Restriction (1)

Syntaxe:

```
SELECT *
FROM   table
WHERE  Condition
```

Exemple 1: Emprunts dont le matricule est 12308.

```
SELECT *
FROM   Emprunts
WHERE  Matricule = 12308
```

SQL: Restriction (2)

La condition (WHERE) peut être exprimée en fonction de:

- Opérateurs de comparaison: **=**, **<=**, **>=**, **<**, **>**, **<>**
- Opérateurs logiques: **AND**, **OR**, **NOT**
- Les mots clés:

- **BETWEEN** : tester si une la valeur d'une expression est comprise entre deux valeurs constantes.
- **IN** : tester si la valeur d'une expression appartient à une liste de valeurs.
- **LIKE** : tester si une chaîne de caractères contient une sous-chaîne.

SQL : Restriction (3)

Exemple 2: livres dont la cote est entre 130 et 160.

```
SELECT  *
FROM    Livres
WHERE   Cote >= 130  AND  Cote<=160
```

Cote **BETWEEN** 130 **AND** 160

Exemple 3: livres dont la cote n'est pas entre 130 et 160.

```
SELECT  *
FROM    Livres
WHERE   Cote < 130  OR  Cote>160
```

NOT(Cote **>=** 130 **AND** Cote**<=**160)

SQL : Restriction (4)

Exemple 4: livres dont la cote est 130, 140, 145 ou 160.

```
SELECT  *
FROM    Livres
WHERE  Cote IN (130, 140, 145, 160)
```

Exemple 5: livres dont le titre **contient** "BD".

```
SELECT  *
FROM    Livres
WHERE  Titre LIKE '%BD%'
```


SQL : Restriction (5)

Exemple 6: livres dont le titre **commence** par "Algèbre".

```
SELECT  *
FROM    Livres
WHERE  Titre LIKE 'Algèbre%'
```

Exemple 7: livres dont le titre **se termine** par "BD".

```
SELECT  *
FROM    Livres
WHERE  Titre LIKE '%BD'
```

SQL : Restriction (6)

Exemple 8: Livres dont le titre commence par **un caractère** suivi de "BD":

```
SELECT *
FROM Livres
WHERE Titre LIKE '_BD%'
```

Remarque: * ⇔ % , ? ⇔ _ et # ⇔ un chiffre

Exemple 9: emprunts dont la date de remise est indéterminé:

```
SELECT *
FROM Emprunt
WHERE Dat_Remise IS NULL
```


SQL : Restriction (10)

Exemple 10: Livres dont le titre contient deux chiffres qui se suivent:

```
SELECT *
FROM Livres
WHERE Titre LIKE '%##%'
```

Exemple 11: Livres dont le titre **ne contient pas** le mot "*algèbre*":

```
SELECT *
FROM Livres
WHERE Titre NOT LIKE '%algèbre%'
```

Diapositive 69

A2 https://www.techonthenet.com/oracle/regexp_like.php
Admin; 22/11/2022

SQL : Restriction (11)

Exemple 12: titre des livres de "Claude Delanoy".

```
SELECT   Titre
FROM     Livres
WHERE  Auteur = 'Claude Delanoy'
```

Exemple 13: titre et auteur des livres dont le titre contient "Algèbre" **ou** dont le nom de l'auteur contient "Gardarin".

```
SELECT   Titre , Auteur
FROM     Livres
WHERE Tite LIKE '%Algèbre%' OR Auteur LIKE '%Gardarin%'
```


SQL : Produit Cartésien(1)

Produit Cartésien en SQL

```
SELECT  *
FROM  Table1 , Table2 , ... , TableN
```

Exemple 1:

```
SELECT  *
FROM     Départements, Étudiants
```

SQL : Jointure(1)

Syntaxe:

SELECT colonnes

FROM $Table_1$, $Table_2$, … , $Table_N$

WHERE *Condition*

BD; Ville (Code_V, Nom_Ville)

Ecole (Code_E, Nom_Ecole, NbSalle, #Code_V)

Exemple 1: nom de l'école suivi du nom et du code de ville

```
SELECT   Nom_Ecole , Nom_Ville,  Ville. Code_V
FROM     Ville , Ecole
WHERE    Ville. Code_V = Ecole. Code_V
```


SQL : Jointure(2)

- On peut utiliser des **alias** comme préfixes des attributs au lieu des noms de tables.

Exemple 2:

```
SELECT   V.Code_V, Nom_Ville, Nom_Ecole
FROM     Ville  V , Ecole  E
WHERE    V. Code_V = E.Code_V
```

SQL : Jointure(3)

- Utiliser une table plusieurs fois dans la clause FROM (auto-jointure)

Exemple 3: **nom de compétition et noms des 2 premiers joueurs.**

Joueur(IdJ, NomJ)

Compétition (IdC, NomC, #IdJ1, #IdJ2, #IdJ3)

```
SELECT  NomC, J1.NomJ, J2.NomJ
FROM    Compétition C, Joueur J1, Joueur J2
WHERE   J1.IdJ = C.IdJ1 and  J2.IdJ = C.IdJ2
```


SQL : Jointure avec Join (1)

Exemple 1: nom d'école et nom de ville

```
Select nom_ville, nom_ecole
From ville v   Inner Join ecole e
                    On v.code_v = e.code_v ;
```

Remarque: Le mot clé **inner** est optionnel.

SQL : Jointure(5)

Exemple 2: Nom d'école ayant *plus que 4 salles* et nom de ville

```
Select      nom_ville, nom_ecole
From        ville  v , ecole  e
Where       v.Code_V =  e.Code_V and  nbsalle >4 ;
```

```
Select  nom_ville,  nom_ecole
From  ville  v  Join  ecole  e  On  v.code_v = e.code_v
Where  nbsalle >4 ;
```

```
Select  nom_ville,  nom_ecole
From  ville  v  Join  ecole  e
              On  v.code_v = e.code_v and  nbsalle >4 ;
```


SQL : Jointure avec Join (3)

Ville (Code_V, Nom_Ville)

Ecole (Code_E, Nom_Ecole, NbSalle, #Code_V)

Prof (Code_P, NomP, #Code_E, #Code_V)

Exemple 3: Nom de prof, nom de son école et la ville de travail.

Select NomP, Nom_Ecole, Nom_Ville

From **Ecole e Join Ville v On e.Code_V = v.Code_V**

Join Prof p On e.Code_E = p.Code_E;

SQL : Jointure avec Join (4)

Syntaxe générale:

SELECT *col1, col2, ….*

FROM *tab1* **JOIN** *tab2* **ON** *tab1.coli = tab2.colj*

JOIN *tab3* **ON** *tab2.colx=tab3.coly*

….

SQL : Classement des tuples (1)

- On peut classer les lignes d'une requête de sélection dans l'ordre croissant ou décroissant selon un ou plusieurs attributs.
- Syntaxe: SELECT Attribut(s)
  FROM Table(s)
  WHERE Condition
  ORDER BY *Attribut1* **DESC**, *Attribut2* **ASC**, …
- L'ordre par défaut: Croissant (ASC).

SQL : Classement des tuples (2)

Exemple 1:

```
SELECT    Nom_Ecole,  Nom_Ville
FROM      Ville  V , Ecole  E
WHERE     V.Code_V = E.Code_V
ORDER BY  Nom_Ecole, Nom_Ville DESC
```

Exemple 2:

```
SELECT    Nom_Ecole,  Nom_Ville
FROM      Ville  V , Ecole  E
WHERE     V.Code_V = E.Code_V
ORDER BY  1, 2 DESC
```


SQL : Fonctions de calcul(1)

Fonctions de Calcul (Fonctions d'agrégation)

- Elles prennent le nom d'un attribut comme argument.
- Elles fournissent une seule valeur en résultat. Cette valeur est calculée avec toutes les valeurs de l'attribut (colonne) spécifié.
- Fonctions de calcul:
  - **COUNT** : Nombre de valeurs d'une colonne.
  - **AVG** : Moyenne des valeurs d'une colonne.
  - **SUM** : Somme des valeurs d'une colonne.
  - **MIN** : Minimum des valeurs d'une colonne.
  - **MAX** : Maximum des valeurs d'une colonne.

SQL : Fonctions de calcul(2)

Exemple 1: Nombre d'écoles

Nbre de valeurs ≠ null

```
SELECT  COUNT(Nom_Ecole)
FROM    Ecole
```

Nbre de lignes

```
SELECT  COUNT(*)
FROM    Ecole
```

Remarque: Le résultat est composé d'une seule colonne sans nom explicite. On peut donner un nom explicite à une colonne dans le résultat comme suit:

```
SELECT   COUNT(Nom_Ecole) AS "Nombre d'écoles"
FROM     Ecole
```


SQL : Fonctions de calcul(3)

Exemple 2: Nombre d'écoles à Rabat

```
SELECT  COUNT(Nom_Ecole) AS "Nombre d'écoles à
                             Rabat"
FROM    Ville  V ,  Ecole  E
WHERE   V.Code_V = E.Code_V AND Nom_Ville = 'Rabat'
```

Exemple 3: Moyenne des salles par école.

```
SELECT  AVG(NbSalle) AS "Moyenne des salles"
FROM    Ecole
```

SQL : Fonctions de calcul(4)

Exemple 4: Le minimum de nombre de salles des écoles

```
SELECT  MIN(NbSalle) AS  "Minimum de nombre de salles"
FROM    Ecole
```

Exemple 5: Le nombre total de salles de toutes les écoles.

```
SELECT  SUM(NbSalle) AS  "Nombre total de salle"
FROM    Ecole
```


SQL : Regroupement (1)

Regroupement (ou agrégat):

- Partitionnement des lignes d'une table en plusieurs groupes selon les valeurs d'un ou de plusieurs attributs afin d'appliquer des fonctions de calcul sur chaque groupe: **GROUP BY** suivi d'une liste d'attributs de groupement.

Exemple 1: Nom de chaque ville et nombre de **ses** écoles

```
SELECT  Nom_Ville,  COUNT(Nom_Ecole)
FROM    VILLE  V, ECOLE  E
WHERE   V.Code_V = E.Code_V
GROUP  BY   V.Code_V, Nom_Ville
```

SQL : Regroupement (2)

Remarque:

Les attributs devant Select doivent apparaître dans la clause Group By.

Exercices:

- Nom de ville suivi du nombre total des salles de ses écoles.
- Titre du livre et le nombre d'emprunteurs qui l'ont emprunté.

SQL : Regroupement (3)

Clause HAVING

- Pour considérer seulement les groupes satisfaisant une condition.
- La condition doit comporter des **fonctions d'agrégation**.

Remarque: La condition de la clause WHERE ne doit pas comporter des fonctions d'agrégation. A1

Diapositive 88

A1 An aggregate function can be used in a WHERE clause only if that clause is part of a subquery of a HAVING clause and the column name specified in the expression is a correlated reference to a group

Admin; 22/11/2022

SQL : Regroupement (4)

Exemple 2: Les villes ayant plus qu'une école

```
SELECT  Nom_Ville, COUNT(Nom_Ecole)
FROM    Ville  V,  Ecole  E
WHERE   V.Code_V = E.Code_V
GROUP   BY  V.Code_V, Nom_Ville
HAVING  COUNT(Nom_Ecole)  >= 2
```


SQL : Jointure Externe (1)

Emp (**empno**, ename, sal, #deptno) **Dept** (**deptno**, dname)

Exemple 1: nom de l'employé et celui de son département

```
Select   ename, dname
From   Emp  e   Join   Dept  d   On   e.deptno= d.deptno  ;
```

Exemple 2: nom de l'employé et celui de son département

+ ceux sans département (deptno dans Emp est NULL)

```
Select   ename, dname
From    Emp  e   Left Outer  Join   Dept  d
                 On   e.deptno = d.deptno  ;
```

SQL : Jointure Externe (2)

Exemple 3: nom de département et ses employés + *département sans employés*.

```
Select   dname, ename
From  Emp  e  Right Outer Join  Dept  d
          On   e.deptno= d.deptno ;
```

Sur Oracle, on peut utiliser:

```
Select   dname, ename
From   Emp   e, Dept  d
Where d.deptno = e.deptno (+)
Where d.deptno(+)= e.deptno
```

Afficher tous les départements même ceux sans employés

Afficher tous les employés même ceux sans département

SQL : Jointure Externe (3)

Exemple 4: nom de département et ses employés + départements sans employés + employés sans département.

```
Select   dname, ename
From   Emp   e  Full Outer Join  Dept  d
          On   e.deptno= d.deptno  ;
```

Syntaxe de jointure externe:

```
Select  col1, col2, ….
From Tab1  LEFT OUTER | RIGHT OUTER | FULL OUTER
          JOIN  Tab2  ON  Tab1.coli = Tab2.colj
```

SQL : Requête Imbriquée (1)

- Sous-requête utilisée dans les clauses **From**, **Where** ou **Having** d'une autre requête (*principale*)

```
Select     ...
From       ...
Where      ...
Group By ...
Having     ...
Group By ...
```

- Exécutée en **premier** avant la requête principale
- Son résultat sert à déterminer celui de la requête principale
- Entourée de parenthèse.
- Son résultat peut être:
  - **une seule valeur** (**1 ligne** , **1 colonne**)
  - **plusieurs lignes** et **une colonne**
  - **une ligne** et **plusieurs colonne**
  - **plusieurs lignes** et **plusieurs colonnes**

SQL : Requête Imbriquée (2)

- **Le résultat de la requête est une seule valeur:**
  - Cette valeur sera comparée à une expression:

    expression Op_de_comp **(SELECT …..)**
  - Les opérateurs de comparaison: =, != ou <>, <, >, <=, >=

Exemple: les écoles ayant le plus grand nombre de salles.

```
Select   Nom_Ecole
From     Ecole
Where    nbsalle = (Select max(nbsalle) From Ecole);
```

SQL : Requête Imbriquée (3)

- **Le résultat est une ligne et plusieurs colonnes:**

  – Comparer une **ligne d'expressions** au résultat:

  (*exp1, exp2, …*) **= | != (SELECT …..)**

  Exemple: Que fait cette requête?

```
Select Nom_Ecole From Ecole
Where (Code_V, nbsalle)=(Select Code_V, max(nbsalle)
                         From Ecole
                         Where Code_V= 10 Group by Code_V);
```


SQL : Requête Imbriquée (4)

- **Le résultat est de plusieurs lignes et une colonne:**
  – Comparer une expression aux valeurs de la sous-requête:
  expression *Op_de_comp* **ANY | ALL (SELECT …..)**
  expression **IN | NOT IN (SELECT …..)**
  – ANY: au moins une valeur de la sous-requête
  – ALL: toutes les valeurs de la sous-requête
  Exemple: les villes ayant des écoles de plus que 10 salles.

```
Select Nom_Ville From Ville
Where
Code_V IN (Select Code_V From Ecole Where nbsalle >= 10);
```

Exercice: Les **noms** des écoles de **Rabat** dont le **nombre de salles** est **supérieur**, ou moins, au **nombre de salles d'une école de Casa**.

SQL : Requête Imbriquée (5)

- **Le résultat est de plusieurs lignes et plusieurs colonnes:**
  - Comparer une ligne d'expression aux lignes du résultat:

    (exp1, exp2, …) **IN | NOT IN (SELECT …..)**

    = ANY | !=ALL

  Exemple: les profs travaillant dans la ville de leurs résidences

```
Select  Nom_P
From    Prof
Where  (Code_E,Code_V)  IN (Select  Code_E, Code_V
                            From  Ecole );
```


Normalisation d'un schéma relationnel

Problèmes de la redondance

[En dehors des clés étrangères]

Exemple: Soit la relation **Produit_Stock**

NumProd	Quantité	NumFour	Adresse
101	200	**801**	Av Nassr Imble A N°1 Casa
106	1000	803	Av Mouahidin N°3 Casa
105	234	890	Rue Dakar N° 34 Tanger
123	55	**801**	Av Nassr Imble A N°1 Casa

Normalisation d'un schéma relationnel

Problèmes de la redondance

(Anomalies liées à la redondance)

- **Anomalies de modification**: Si l'on souhaite mettre à jour l'adresse d'un fournisseur, il faut le faire pour tous les tuples concernés.

- **Anomalies d'insertion:** Pour ajouter un nouveau fournisseur, il faut obligatoirement fournir des valeurs pour NumProd et Quantité.

- **Anomalies de suppression:** La suppression du produit 106, par exemple, fait perdre des informations concernant le fournisseur 803.

Normalisation d'un schéma relationnel

But de la normalisation des relations

- Éviter les problèmes de mise à jour liés à la redondance.
- Minimisation de l'espace de stockage.

Origine du Problème

- Les problèmes viennent en fait des **dépendances fonctionnelles** internes aux relations.

Normalisation d'un schéma relationnel

Dépendances fonctionnelles (DF)

- Soient X et Y deux attributs (ou groupe d'attributs) d'une même relation.
- Il y a dépendance fonctionnelle entre X et Y (ou Y dépend de X) si la **valeur de X détermine celle de Y**.
- On note: **X → Y**

Exemples: **Etudiant** (Matricule, Nom, Prénom, Moyenne)

Matricule → Nom, Prénom, Moyenne

Stock (NumProd, NumMagasin, Quantité)

NumProd , NumMagasin → Quantité

Normalisation d'un schéma relationnel

Pratique de la normalisation

- Normaliser un schéma relationnel d'une BD (l'ensemble des relations de la BD) consiste à remplacer chaque relation du schéma par des relations qui sont dans la **forme normale (FN)** voulue.
- Une **FN** représente une condition qu'une relation doit vérifier.
- Il existe plusieurs degrés de normalisation de la **1ère** forme normale à la **5ème**.

Normalisation d'un schéma relationnel

Première forme normale

- Une relation est en **1FN** si tout attribut n'est pas décomposable (multivaluée) càd que chaque attribut doit être atomique.

Exemple:

Livre (Cote, Titre, Auteurs) n'est pas en 1FN car l'attribut Auteurs est décomposable (il s'agit de plusieurs auteurs).

Cette relation peut être décomposée en deux relations:

Livre (Cote, Titre)

Auteurs (Cote, Auteur)

Normalisation d'un schéma relationnel

Première forme normale

Remarque

La jointure des deux nouvelles relations sur l'attribut **Cote** donnera l'équivalent de la relation de départ:

Join (Livre, Auteurs, Livre.Cote = Auteurs.Cote)

Cette jointure permettra d'avoir, pour chaque livre, la Cote, le titre et les auteurs.

Normalisation d'un schéma relationnel

Deuxième forme normale

- Une relation est en **2FN** si:
  - elle est en **1FN**.
  - chaque attribut non clé primaire est dépendant de la clé **primaire entière**.

Exemple: Client (NumCli, Nom, Prénom, RueNum, CodPostale, Ville)

est en 2FN

Employé (Mat, CodeProjet, Nom, FonctionProjet)

n'est pas en 2FN: **Mat → Nom**

Normalisation d'un schéma relationnel

Normalisation en deuxième forme normale

- Théorème de décomposition **sans perte d'informations (SPI)**:

Soit une relation R (A,B,C,D) où A, B et C sont des ensembles d'attributs, avec **B → C**, alors:

R1 = **Project**(R, A, B,D) ⇨ R1(A, B, D)

R2 = **Project**(R,B,C) ⇨ R2 (B, C)

R(A, B, C,D) = **Join**(R1, R2, R1.B = R2.B)

En décomposant R en deux relation R1 et R2, on dit que l'on a **"extrait" la DF de R**

Normalisation d'un schéma relationnel

Normalisation en deuxième forme normale

- **Exemple 1:** Normalisation en 2FN de la relation

  Employé (Mat, CodeProjet, Nom, FonctionProjet)

  Les dépendances de la relation sont:

  Mat → Nom

  CodeProjet, Mat → FonctionProjet

  Le résultat de normalisation en 2FN est:

  Employé(Mat, Nom)

  Participation(#Mat, **CodeProjet**, FonctionProjet)

Normalisation d'un schéma relationnel

Normalisation en deuxième forme normale

- **Exemple 2:** Normalisation en 2FN de la relation

  Magasin(NumProd, Quantité, NumFour, Ville)

  on a: NumProd, NumFour → Quantité

  NumFour → Ville ⇨ La relation n'est pas en 2FN

Le résultat de la normalisation:

Magasin (NumProd, NumFour, Quantité)

Fournisseur (NumFour, Ville)

Normalisation d'un schéma relationnel

Troisième forme normale

Une relation est en 3FN si:

- Elle est en 2FN.
- Il n'existe aucune DF entre deux attributs non clé primaire.

Exemple:

Musique(NoChanson, NoChanteur, NomChanteur)

Avec les DF: **NoChanson** → NoChnateur, NomChanteur

NoChanteur → NomChanteur

Cette relation est en 2FN mais pas en 3FN.

Normalisation d'un schéma relationnel

Algorithme de décomposition en 3FN

Soit un schéma de relation **R**.

- Pour chaque DF: X → A, on crée une relation Ri (X, A)
- Si on a plusieurs DF: X → A1, X → A2, …, X → An; on les regroupe toutes dans une même relation Rj (X, A1, A2, …, An)
- Les attributs n'appartenant à aucune DF seront regroupés dans une même relation.
- Pour avoir une décomposition **SPI**: il doit y avoir au moins **une clé** de **R** dans une des relations de décomposition.

  Si ce n'est pas le cas, on ajoute une relation contenant une clé de R, ou on ajoute des attributs dans une des relations de décomposition.

Normalisation d'un schéma relationnel

Algorithme de décomposition en 3FN

Exemple:

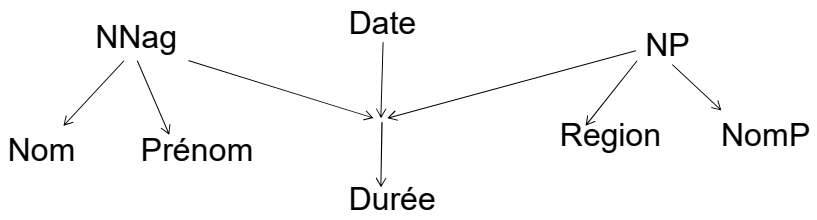
BAINS (NNag, Nom, Prénom, Durée, Date, NP, NomP, Region)

DF: NNag → Nom, Prénom
NP → NomP, Region
NP, NNag, Date → Durée

Résultat de normalisation: **NAGEUR** (**NNag**, NOM, PRENOM)
PLAGE (**NP**, NOMP, REGION)
BAIGNADE (#**NNag**, #**NP**, **DATE**, DUREE)

Normalisation d'un schéma relationnel

Graphe des DF



Normalisation d'un schéma relationnel

Algorithme de décomposition en 3FN

Exercice 1: Normaliser les relations suivantes

1) VOITURE (IMMATRICULATION, COULEUR, MODELE, MARQUE)

2) COMMANDE (NOM_FOURNISSEUR, ADRESSE_FOURNISSEUR, ARTICLE, QUANTITE, PRIX)

Normalisation d'un schéma relationnel

Exercice 2: Pour chaque relation ci-dessous:

- définir sa forme normale et la justifier
- établir un graphe de ses dépendances,
- proposer une décomposition optimale si nécessaire.

1) Pièce (Npièce, prix-unit, TVA, libellé, catégorie)

Une pièce est identifiée par un n° et a un prix unitaire, un libellé, une tva et une catégorie.

La tva est déterminée en fonction de la catégorie

Normalisation d'un schéma relationnel

2) Prime (Nmachine, Ntechn, atelier, montant-prime, nom-techn)

Une machine est identifiée par Nmachine et affectée à un seul atelier.

Un technicien est identifié par Ntechn et a un nom.

Une prime est donnée à un technicien ayant travaillé sur une machine donnée.

3) Employé (NEmp, NLab, NProj, NomEmp, NomProj, adresse)

Un employé est identifié par un n°, a un nom et une adresse et peut travailler sur plusieurs projets. Un projet identifié par NProj a un nom et concernne plusieurs laboratoire.

Exercice

Soit le schéma relationnel de la base d'une bibliothèque contenant les relations suivantes:

Livre (NumInv, Titre, Auteur, Qte)

Abonne (NumAb, Nom, Prénom)

Prêt (#NumInv, #NumAb, DatePret, DateRemise)

Exprimez les requêtes suivantes en algèbre relationnelle et en SQL:

1) Les titres et les quantités des livres.

2) Les livres disponibles en quantité supérieure ou égale à 2

3) Les titres et les quantités des livres dont la quantité ≥ 2

4) Les livres dont la quantité est entre 2 et 10 ou dont le titre est "Unix".

5) Les quantités maximale et minimale disponibles.