Cours

# Systèmes de Gestion des Bases Données Avancées

**Master 1, ISI.**
**2023-2024**

**Dilekh Tahar**

**tahar.dilekh@univ-batna2.dz**



Département d'informatique. Faculté des mathématiques et informatique. Université Batna 2

- **Chapitre 3:**

**Le modèle relationnel formel & Le Modèle logique (le langage de définition des données de SQL - LDD ou DDL)**

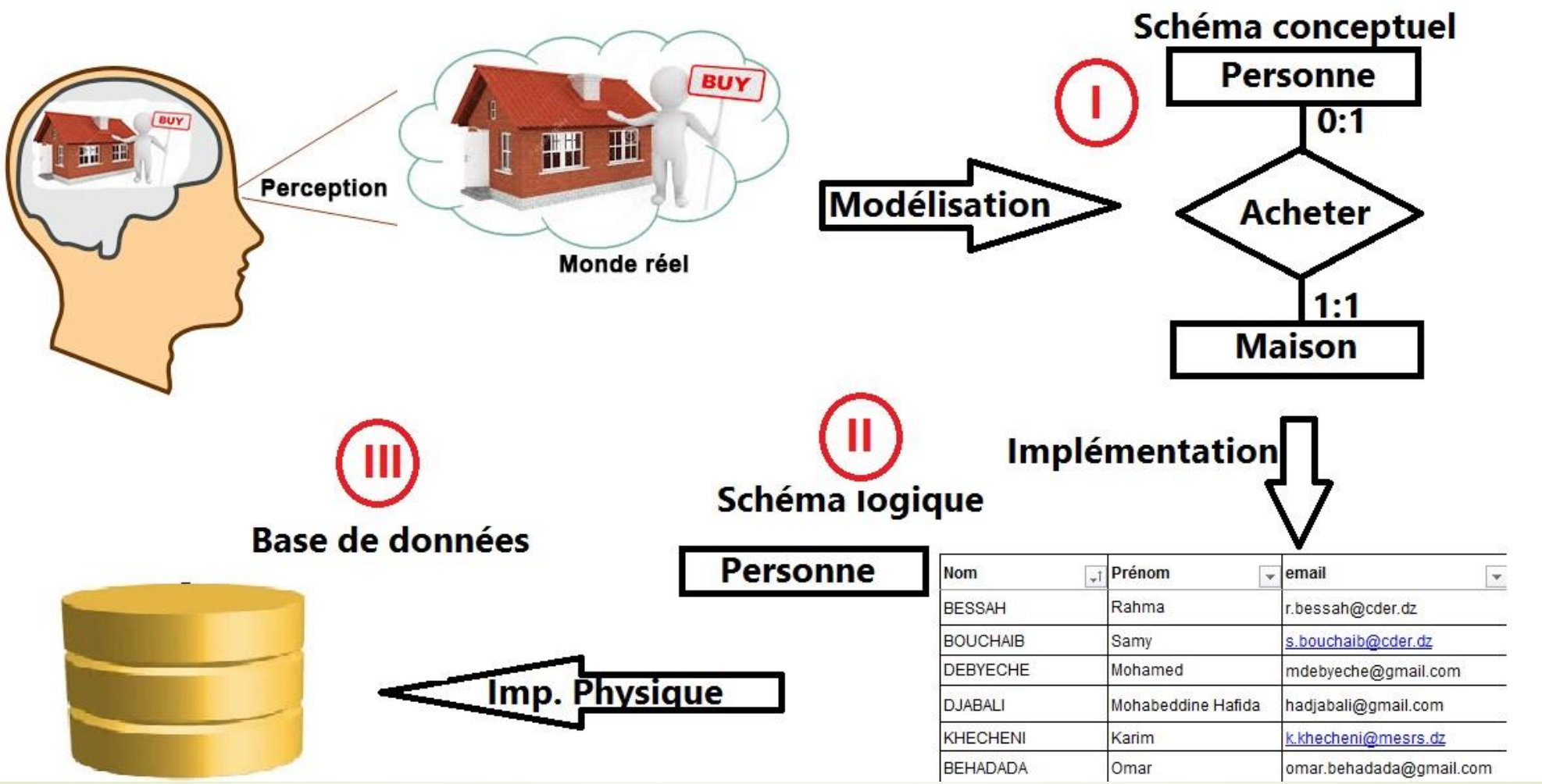
# Plan de cours (Partie 1 : Le modèle relationnel  formel)

- Concepts de base
- Schéma relationnel
- Règles de structuration
- Valeurs nulles
- Identifiant et Identifiants externes
- Vérification de l'intégrité référentielle
- Domaines de valeurs
- Définition d'une relation
- Contraintes de modélisation
- Représentation d'attribut monovalué  complexe
- Représentation d'attribut multivalué
- Cas d'attribut multivalué et complexe

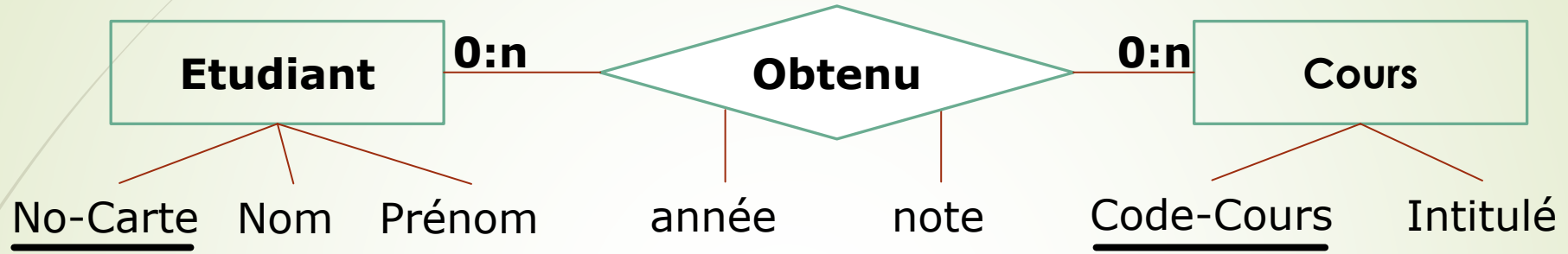
# Introduction

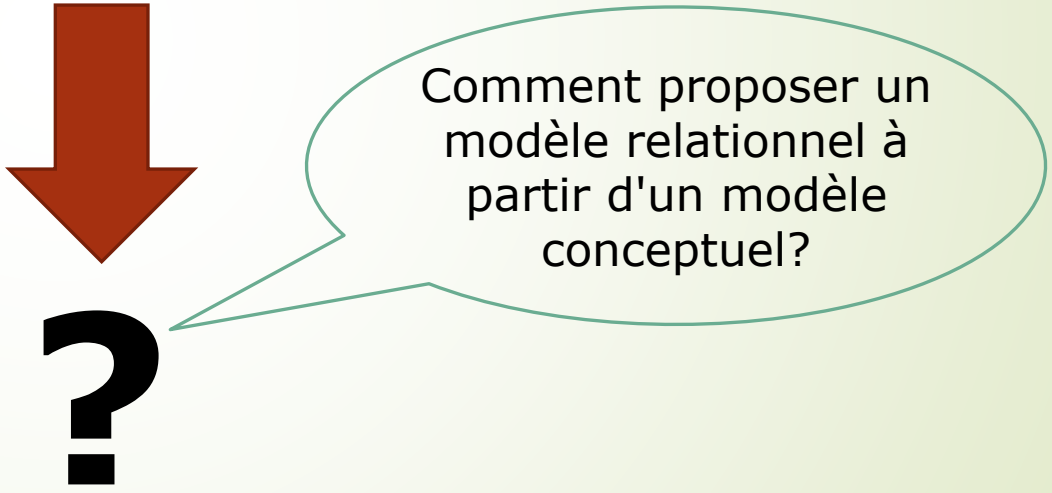
- Modèle de niveau logique
- modèle simple : deux concepts
  - Relation (table)
  - attribut (colonne)
- défini par Ted Codd en 1970 ; prix Turing en 1986. Développé par IBM lab.
- support théorique très solide.
- aujourd'hui utilisé par beaucoup de SGBD commerciaux (Oracle, Informix, DB2, Ingres, Sybase, dBase, Access …)

# Introduction



# Introduction



# Introduction

1- Modèle formel
- relation, attribut, tuple, identifiant…
- Normalisation
- algèbre relationnelle
- calculs relationnels

2- Modèle logique, implémenté : SQL
- table, colonne, ligne, clé primaire…
- SQL - définition et modification du schéma
- SQL - entrée et mise à jour des données
- SQL - requêtes

# Le modèle relationnel  formel

# Concepts de base

1. Monde réel -> Relationnel
2. Objet -> Relation
3. Propriété
   - simple, monovaluée -> Attribut
4. lien
   - binaire, sans attribut -> Identifiant externe

# Concepts de base

**Etudiant (N°Etud, nom, prénom, âge)**

nom de la relation

noms des attributs

- schéma

| Etudiant | | | |
|---|---|---|---|
| **N°Etud** | **Nom** | **Prénom** | **Age** |
| 136 | Rochat | Jean | 19 |
| 253 | Aubry | Annie | 20 |
| 101 | Duval | André | 21 |
| 147 | Rochat | Marc | 21 |

tuple ou occurrence

Population

# Schéma relationnel

- schéma d'une relation = un ensemble d'attributs avec leurs domaines

  R = (A1/d1, A2/d2, …, Ap/dp)  ou, plus simplement,

  R = (A1, A2, …, Ap)

- schéma d'une BD relationnelle = l'ensemble des schémas de ses relations : R1, R2, …, Rn
- uneBD = l'ensemble des populations de toutes ses relations

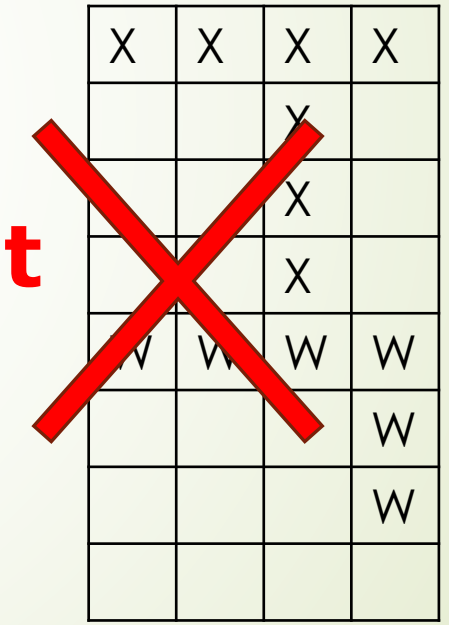
# Règles de structuration

- Attributs : simples et monovalués (domaine de valeurs atomiques)
- structure plate régulière

| | | | |
|---|---|---|---|
| X | X | X | X |
| Y | Y | Y | Y |
| … | | | |
| | | | |
| | | | |
| | | | |

x, y … : une et une seule valeur atomique par attribut

**Interdit**



# Valeurs nulles

- Un attribut peut ne pas être value pour un tuple.  On dit alors qu'il a une valeur nulle
  - exemple : on ne connaît ni l'âge de NE2 ni le  prénom de NE3

| Etudiant | | | |
|---|---|---|---|
| **N°Etud** | **Nom** | **Prénom** | **Age** |
| 136 | NE1 | PNE1 | 19 |
| 253 | NE2 | PNE2 | NULL |
| 101 | NE3 | NULL | 21 |
| 147 | NE4 | PNE4 | 21 |

# Identifiant

- Rappel : ensemble minimum d'attributs tel qu'il n'existe jamais 2 tuples ayant mêmes valeurs pour tous ces attributs.
- L'identifiant n'admet pas de valeurs nulles.

| Etudiant | | | |
|---|---|---|---|
| **N°Etud** | **Nom** | **Prénom** | **Age** |
| 136 | NE1 | PNE1 | 19 |
| 253 | NE2 | PNE2 | NULL |
| 101 | NE3 | NULL | 21 |
| 147 | NE4 | PNE4 | 21 |

# Identifiant

- Modèle formel : jamais de tuple en double
- Il existe donc toujours un identifiant : dans le pire cas, c'est l'ensemble des attributs de la relation
- Il peut exister plusieurs identifiants
  - Etudiant ( AVS, N°inscription, nom, prénom, adresse, tél )
    - AVS
    - N°inscription
    - ( nom + prénom )

# Identifiants externes

- Décrivent les liens binaires entre les relations
- Etudiant ( N° , nom , prénoms , adresse , tél)
- Cours ( Nom , horaire, prof )
- Suit ( N°Etud , NomC )
  - N°Etud **référence un** Etudiant
  - NomC **référence un** Cours
- La valeur de N°Etud dans Suit est :
  - celle de l'identifiant d'un tuple **existant** de Etudiant (intégrité référentielle)
  - ou éventuellement nulle (si attribut facultatif)
- Même contrainte pour NomC de Suit

# Identifiant externe - Exemple

Cours ( **NomC**, **horaire**, **prof**)

| | | |
|---|---|---|
| BD | Mercredi 15-17 | prof1 |
| SI | Mardi 16-19 | Prof2 |
| Réseaux | Jeudi 16-19 | prof2 |

Suit ( **N°Etud**, **NomC**)

| | |
|---|---|
| 253 | SI |
| 136 | BD |
| 253 | BD |
| 101 | SI |

**Suit** traduit un TA entre Etudiant et Cours.
Suit contient les identifiants de Etudiant et de Cours.
Suit.NomC est un identifiant externe sur Cours.

# Identifiants externes (suite)

- Si la relation référencée possède plusieurs identifiants, il faut préciser lequel :
- Etudiant (AVS, N°inscript, nom, prénoms, adresse)
  - identifiants : AVS
    N°inscript
- Suit (N°Etud, NomC)
- identifiants externes :

  N°Etud **référence un** Etudiant.N°inscript

  NomC **référence un** Cours

# Vérification de l'intégrité référentielle

- automatiquement, par le SGBD
- Si un utilisateur veut entrer (INSERT) un tuple dans Suit avec un NomC qui n'existe pas dans Cours
  - Refus
- Si un utilisateur veut modifier (UPDATE) le nom du cours d'un tuple dans Suit avec un NomC qui n'existe pas dans Cours
  - refus

# Vérification de l'intégrité référentielle

- Si un utilisateur veut supprimer (DELETE) un tuple de Cours pour lequel il existe des tuples dans Suit
  - refus ?
  - détruire les tuples de Suit correspondants ?
  - mettre à NULL la valeur de NomC dans Suit ?
- Si un utilisateur veut mettre à jour (UPDATE) le nom d'un cours pour lequel il existe des tuples dans Suit
  - refus ?
  - propager la mise à jour de NomC dans Suit ?

# Domaines de valeurs

- Un domaine est un ensemble de valeurs atomiques que peut prendre un attribut
- Exemples de domaines :
  - Dnom : chaînes de caractères de longueur maximale  30
  - Dâge : entiers compris entre 16 et 65
  - Dcouleur : {"bleu", "vert", "jaune"}
  - DétatCivil : {"célibataire", "marié", "veuf", "divorcé"}

# Définition d'une relation

- Une relation est définie par :
  - son nom
  - sa liste de couples <nom d'attribut : domaine>
  - son (ses) identifiant(s)
  - ses identifiants externes s'il en existe
  - la définition de sa sémantique (phrase en français)
- Exemple :
  - Etudiant (N°Etud : Dnum, Nom : Dnom,Prénom : Dnom, Age : Dâge)

  Identifiant : N°Etud

  Définition : tout étudiant actuellement inscrit

# Contraintes de modélisation

- Les notions d'attribut multivalué ou complexe n'existent pas dans le modèle relationnel. Il faut donc les modéliser autrement.
- Pour un attribut monovalué complexe, il faut choisir entre le composé ou les composants
- Pour un attribut multivalué, il faut créer une autre relation (ceci pour chaque attribut multivalué)

# Représentation d'attribut monovalué complexe

Soit pour Personne adresse : nom-rue , n° , ville , CP

- Solution 1 :
  - un attribut par composant :

  Personne (AVS, nom,…, nom-rue , n° , ville, CP)

  "Rue J", "2", " Batna", "05000"
  - il est éventuellement possible de définir par ailleurs une vue restituant la notion globale d'adresse
- Solution 2 :
  - un attribut adresse, de domaine : chaine de caractères

    Personne (AVS, nom,…, adresse)

  "Rue J N°2, Batna, 05000"
  - le système ignore nom-rue, n°, ville et CP

# Représentation d'attribut multivalué

- Exemple : mémoriser les différents prénoms des étudiants
- Mauvaise modélisation : plusieurs attributs

| **Etudiant** ( **N°Etud,** | **Nom,** | **Prénom1,** | **Prénom,** | …) |
|---|---|---|---|---|
| 255 | NE1 | Pre11 | Pre21 | |
| 214 | NE2 | Pre12 | NULL | |
| 212 | NE3 | Pre13 | Pre23 | |

- Bonne modélisation : créer une relation supplémentaire Etudiant (N°Etud, nom)

| **EtudPrénoms** | **( N°Etud,** | **Prénom)** |
|---|---|---|
| | 255 | Pre11 |
| | 255 | Pre21 |
| | 214 | Pre12 |
| | 214 | NULL |
| | 212 | Pre31 |
| | 212 | Pre32 |

identifiant externe : N°Etud référence Etudiant

# Représentation d'attribut multivalué (suite)

- Les prénoms sont ordonnés

  => Autre bonne modélisation

  **Etudiant** (N°Etud, nom)

  **EtudPrénoms2** ( N°Etud, N°prénom, prénom )

  identifiants : (N°Etud + N°prénom)

  (N°Etud + prénom)

  identifiant externe : N°Etud référence Etudiant

| **EtudPrénoms2** ( **N°Etud,** | **N°Prénom,** | **Prénom)** |
|---|---|---|
| 255 | 1 | Pre11 |
| 255 | 2 | Pre12 |
| … | | |

# Cas d'attribut multivalué et complexe

- Même solution que pour un attribut simple multivalué : création d'une nouvelle relation
- Exemple :
  - Personne avec téléphones (intitulé, numéro)
  - Personne ( AVS, nom, adresse, … )

    PersTel ( AVS, intitulé, numéro )

    identifiant externe : AVS **référence** Personne

# Récapitulatif

- Un schéma relationnel se compose :
  - pour chaque relation de :
    - nom de la relation
    - définition
    - attributs + domaines
    - identifiant(s)
    - éventuellement identifiant(s) externe(s)
    - contraintes d'intégrité associées
  - Et des autres contraintes d'intégrité qui portent sur plusieurs relations.

# Plan de cours (Partie 2: : Le modèle logique -SQL )

- Introduction
- SQL : Trois langages
- Terminologie
- Langage de Définition de Données
  - Créer une table/vue (CREATE TABLE/**VIEW**)
    - Domaines de valeurs
    - Contraintes
      - Identifiants : clé primaire / secondaire
      - Clé externe
      - Intégrité référentielle
      - Contrainte CHECK (condition)
  - Déclencheur (Trigger)
  - Supprimer une table/vue (DROP TABLE/**VIEW**)
  - Modifier une table/vue (ALTER TABLE/**VIEW**)

# Introduction

- SQL : Structured Query Language
- SQL 2: adopté (SQL 92)
- SQL 3: adopté (SQL 99)
- Standard d'accès aux bases de données relationnelles

# SQL : Trois langages

- Langage de définition de données (LDD/DDL)
  - création de relations : CREATE TABLE
  - modification de relations: ALTER TABLE
  - suppression de relations: DROP TABLE
  - vues, index …. : CREATE VIEW ...
- Langage de manipulation de données (LMD /DML)
  - insertion de tuples: INSERT
  - mise à jour des tuples: UPDATE
  - suppression de tuples: DELETE
- Langage de requêtes
  - SELECT ……. FROM ……. WHERE …….

# Terminologie

- **Relation** –> table
- **Tuple** –> ligne
- **Attribut** –> colonne (column)
- **Identifiant** –> clé primaire (primary key)
  clé secondaire (unique)
- **Identifiant externe**
  –> clé externe (forgien key - external key)

# Langage de Définition de Données

- Commandes pour créer, modifier et supprimer les éléments du schéma (pour l'instant table et vue)
- **CREATE TABLE** : créer une table
- **CREATE VIEW** : créer une vue particulière sur les données à partir d'un SELECT

  table dérivée
- **DROP TABLE / VIEW** : supprimer une table ou une vue
- **ALTER TABLE / VIEW** : modifier une table ou une vue.

# Créer une table/vue (CREATE TABLE/VIEW)

- Commande créant une table en donnant son nom, ses attributs et ses contraintes
- CREATE TABLE nom-table { ( nom-col type-col [DEFAULT valeur] [ [CONSTRAINT] contrainte-col] )* [ [CONSTRAINT] contrainte-table ]*| AS requête-SQL };
- Légende :
  - {a | b} : a ou b
  - [option]
  - * : applicable autant de fois que souhaité
  - mot en capitale : mot clé

# Créer une table/vue (CREATE TABLE/VIEW)

- **CREATE TABLE** nom_table { ( nom-col type-col [DEFAULT val] [ [CONSTRAINT] contrainte-col] )* [ [CONSTRAINT] contrainte-table]* | AS requête-SQL };
- Exemples :
  - **CREATE TABLE** Doctorant ( nom VARCHAR(20), prénom VARCHAR(15), année_insc DECIMAL(4) DEFAULT 2003 ) ;
  - **CREATE TABLE** Doctorant **AS SELECT** nom, prénom, année_inscr FROM Etudiant WHERE statut='Doctorant' ;

# Domaines de valeurs

- CHAR(nb)
- VARCHAR(max)
- INTEGER
- NUMERIC(n,p)
- DATE
- TIME
- TIMESTAMP
- …

# Domaines de valeurs (…)

| Type | Description |
|---|---|
| INTEGER | Type des entiers relatifs |
| SMALLINT | Idem. |
| BIGINT | Idem. |
| FLOAT | Flottants simple précision |
| DOUBLE PRECISION | Flottants double précision |
| REAL | Synonyme de FLOAT |
| NUMERIC (*M*, *D*) | Numérique avec précision fixe. |
| DECIMAL (*M*, *D*) | Idem. |
| CHAR(*M*) | Chaînes de longueur fixe |
| VARCHAR(*M*) | Chaînes de longueur variable |
| BIT VARYING | Chaînes d'octets |
| DATE | Date (jour, mois, an) |
| TIME | Horaire (heure, minutes, secondes) |
| DATETIME | Date et heure |
| YEAR | Année |

# Contraintes

- contrainte-col : contrainte sur une colonne
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - REFERENCES nom-table [(nom-col)] [action]
  - CHECK ( condition)
- contrainte-table : contraintes sur une table
  - PRIMARY KEY (nom-col*)
  - UNIQUE (nom-col*)
  - FOREIGN KEY (nom-col*) REFERENCES nom-table [(nom-col*)] [action]
  - CHECK ( condition)

# Attribut obligatoire : NOT NULL

- Contrainte sur une colonne

- CREATE TABLE Pays
- (nom VARCHAR(20) NOT NULL ,
  capitale VARCHAR(20) NOT NULL ,
  surface INTEGER,
  … )

# Identifiants : clé primaire / secondaire

- PRIMARY KEY (A1, A2, …)
  - La clé primaire (si elle existe) . **En SQL les lignes doubles sont possibles** !
  - choisir l'identifiant le plus efficace
  - attribut référencé par défaut dans les identifiants externes
  - pas de valeur nulle possible

    c-à-d NOT NULL automatiquement
- UNIQUE (A1, A2, …)
  - une clé secondaire (s'il en existe)
  - contrainte d'intégrité pour les autres identifiants
  - **valeur nulle permise** (sauf si NOT NULL)

# PRIMARY KEY : exemples

1. CREATE TABLE Pays ( nom VARCHAR(20) PRIMARY KEY , capitale VARCHAR(20) … )

2. CREATE TABLE Employé ( nom VARCHAR(30) ,
   prenom VARCHAR(30) ,
   adresse VARCHAR(60) , …
   CONSTRAINT Pk_emp PRIMARY KEY (nom, prenom) )

- contrainte de **colonne (1)** et de **table (2)**

# UNIQUE : exemples

- CREATE TABLE Etudiant
  ( AVS CHAR(11) PRIMARY KEY,
  N°Etudiant CHAR(6) UNIQUE , -- (1)
  nom VARCHAR(20) ,
  prénom VARCHAR(30) , …
  CONSTRAINT UNIQUE (nom, prénom) ) -- (2)

- Contrainte de **colonne (1)** et de **table (2)**

- PRIMARY KEY et UNIQUE sont incompatibles

# Clé externe : FOREIGN KEY

- CREATE TABLE Etudiant (N°E …)
- CREATE TABLE Cours (NomCours …)
- CREATE TABLE Suit

```
( N°Etud CHAR(9) ,
NomC VARCHAR(25) ,
PRIMARY KEY (N°Etud , NomC) ,
FOREIGN KEY (N°Etud) REFERENCES Etudiant ,
FOREIGN KEY (NomC) REFERENCES Cours )
```

# Clé externe : FOREIGN KEY (suite)

- Les clés externes référencent par défaut la clé primaire de la table référencée
  - CREATE TABLE Employe
    (AVS CHAR(11) PRIMARY KEY,
    empN° CHAR(6) UNIQUE , … )
- CREATE TABLE Département
  (dpt_id VARCHAR(18) PRIMARY KEY,
  manager_id CHAR(11) REFERENCES Employe , … )
- Une clé externe peut référencer une clé secondaire de la table référencée => à préciser
- CREATE TABLE Departement2
    (dpt_id VARCHAR(18) PRIMARY KEY,
    manager_id CHAR(6) REFERENCES Employe (empN°) , … )

# Intégrité référentielle

- REFERENCES nom_table [(nom-col)] [action]
- Qu'est ce qui se passe quand on détruit/m.à.j. une clé primaire ou unique qui est référencée par un tuple (foreign key) d'une autre table?
- CREATE TABLE Departement

  (dpt_id VARCHAR(18) PRIMARY KEY,

  manager_id CHAR(11) REFERENCES Employe,… )
- Soit le tuple (dpt_id=Ventes, manager_id=12345, …) dans la table Département

  Que se passe-t-il si on détruit l'employé d'AVS 12345 dans la table Employé ?

# "Referential triggered action"

- Deux circonstances
  - ON DELETE
  - ON UPDATE
- Trois options
  - SET NULL
  - SET DEFAULT: valeur par défaut si elle existe, sinon NULL
  - CASCADE : on répercute la m.à.j.
- CREATE TABLE Département
  ( dpt_id VARCHAR(18) PRIMARY KEY,
  manager_id CHAR(11) REFERENCES Employé (emp_id)
  ON DELETE SET NULL
  ON UPDATE CASCADE ,
  … )
- Si la clause n'existe pas : refus

# Contraintes (rappel)

- contrainte-col : contrainte sur une colonne
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - REFERENCES nom-table [(nom-col)] [action]
  - CHECK ( condition)
- contrainte-table : contraintes sur une table
  - PRIMARY KEY (nom-col*)
  - UNIQUE (nom-col*)
  - FOREIGN KEY (nom-col*) REFERENCES nom-table [(nom-col*)] [action]
  - CHECK ( condition)

# Contrainte CHECK (condition)

- Condition que chaque ligne de la table doit vérifier
- Contrainte de **colonne (1)** et de **table (2)**
- CREATE TABLE Employe

```
( AVS CHAR(11) PRIMARY KEY ,
nom VARCHAR(20) NOT NULL,
prenoms VARCHAR(30) ,
age NUMBER CHECK (age BETWEEN 18 AND 70) , -- (1)
sexe CHAR CHECK (sexe IN ('M', 'F')) ,
salaire NUMBER ,
commission NUMBER ,
CONSTRAINT check_sal CHECK (salaire * commission <= 7000) ); -- (2)
```

# CREATE TRIGGER

- Contrainte d'intégrité
  - simple => clause CHECK dans CREATE TABLE
  - complexe => un TRIGGER
- CREATE TRIGGER
  - nouvelle instruction
  - QUAND événement
    - INSERT / DELETE /UPDATE

      SI condition-SQL

      ALORS action
      - refus / instructions SQL

# Déclencheur (TRIGGER) : exemple

- Pour la table Employes : Pour tout tuple de Employes <CodeEmploye, nomEmploye, PrenomEmploye, ageEmploye, adresseEmploye>, l'âge d'employé dans la table Employes doit être supérieur ou égal à 18 ans

  QUAND : INSERT INTO Employes

  SI : le ageEmploye dans Employes est inférieur de 18 ans

  ALORS : refuser l'insertion

# Déclencheurs : Syntaxe

```
CREATE [OR REPLACE] TRIGGER <nom_trigger>
{BEFORE | AFTER}
{INSERT | DELETE | UPDATE [OF colonnes]}
ON <nom_table>
[FOR EACH ROW]
[When condition ]
[DECLARE]
-- déclaration de variables, exceptions,
-- curseurs
BEGIN
-- bloc action
-- ordres SQL et PL/SQL
END;
```

# Déclencheurs

Prise de commande: (2) mise à jour  quantité en stock

```
CREATE TRIGGER t_a_i_detail_commandes
AFTER INSERT ON detail_commandes
FOR EACH ROW  BEGIN
        UPDATE Produits p
        SET p.qtstock = p.qtstock - :NEW.qtcom
        WHERE idprod = :NEW.idprod;
END;
```

# Supprimer une table/vue (DROP TABLE/VIEW)

- DROP : supprimer une table
  - supprime la table et tout son contenu
- DROP TABLE nom_table [CASCADE CONSTRAINTS]
- CASCADE CONSTRAINTS
  - Supprime toutes les contraintes de clé externe référençant cette table
  - Si on cherche à détruire une table dont certains attributs sont référencés sans spécifier CASCADE CONSTRAINT: refus

# Modifier une table/vue (ALTER TABLE/VIEW)

- Modifier la définition d'une table :
  - Changer le nom de la table

    mot clé : RENAME
  - Ajouter une colonne ou une contrainte

    mot clé : ADD
  - Modifier une colonne ou une contrainte

    mot clé : MODIFY
  - Supprimer une colonne ou une contrainte

    mot clé : DROP
  - renommer une colonne ou une contrainte

    mot clé : RENAME

# ALTER TABLE (format)

- **ALTER** TABLE nom-table

{ **RENAME TO** nouveau-nom-table |

**ADD** ( [ (nom-col type-col [DEFAULT valeur]

[contrainte-col])* |

**MODIFY** (nom-col [type-col] [DEFAULT valeur]

[contrainte-col])* |

**DROP COLUMN** nom-col [CASCADE CONSTRAINTS] |

**RENAME COLUMN** old-name TO new-name

}

# Récapitulatif

- LDD: commandes pour
  - La création des tables/vues en donnant ses noms, ses attributs et ses contraintes
    - Contraintes sur une colonne : NOT NULL, PRIMARY KEY, UNIQUE, REFERENCES nom-table [(nom-col)] [action], CHECK ( condition)
    - Et contraintes sur une table : PRIMARY KEY (nom-col*), UNIQUE (nom-col*), FOREIGN KEY (nom-col*) REFERENCES nom-table, [(nom-col*)] [action], CHECK ( condition).
  - La modification des tables/vues
  - La suppression des tables/vues
  - La vérifications d'intégrité des données, la mise à jour, … en utilisant les déclencheurs.

# Fin

**Questions ?**