Chapitre 1: Introduction

Historique et SQL avancé

Dr. FERRAG Mohamed Amine (ferrag.mohamedamine@univ-guelma.dz)

Module: Bases de données avancées
Master STIC (Semestre 1)
Département Informatique, Université de Guelma
2020/2021

Plan

1) Syllabus
2) Historique des bases de données
3) Lien entre le modèle Entité Association et le modèle relationnel
4) Le modèle relationnel
5) Le modèle relationnel étendu
6) Le langage SQL

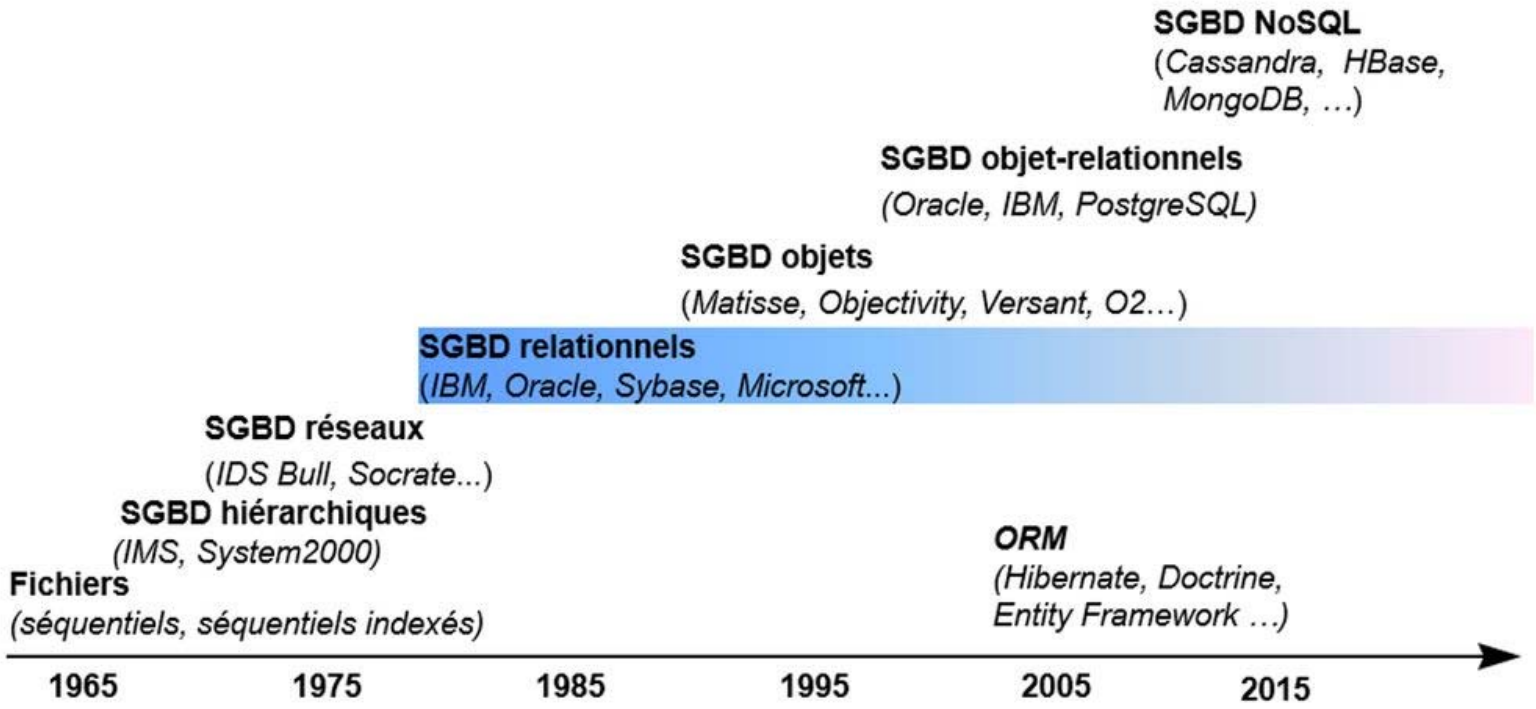
1) Syllabus

1 Cours , 1 TD, et 1 TP

- Chapitre 1: Historique et SQL avancé
- Chapitre 2: Les bases de données réparties
- Chapitre 3: Les bases de données orientée objet
- Chapitre 4: Les bases de données semi-structurées
- Chapitre 5: Les bases de données NoSQL

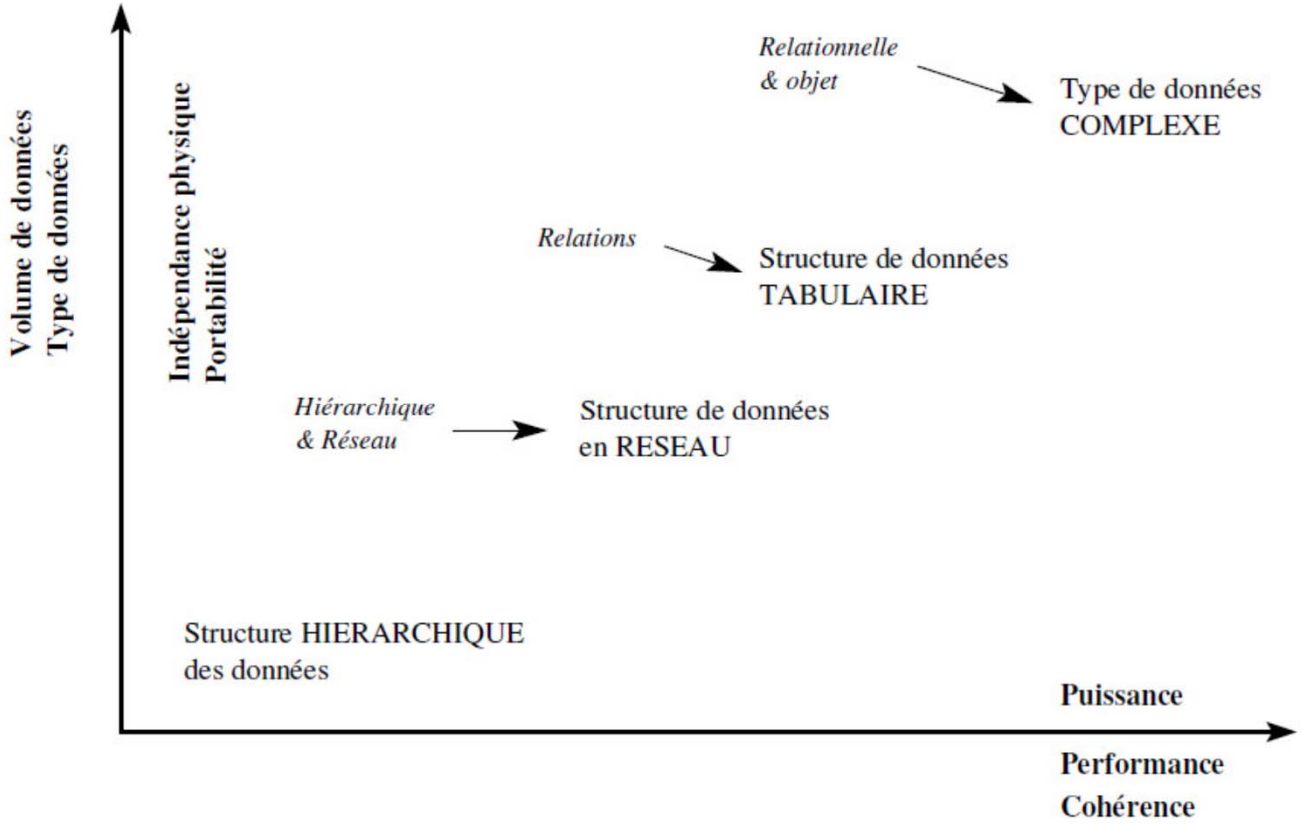
Note finale: 60% Examen, 20% TD, et 20% TP

2) Historique des bases de données



Source: Christian Soutou

Historique: Structure et type de données



2) Historique des bases de données (1)

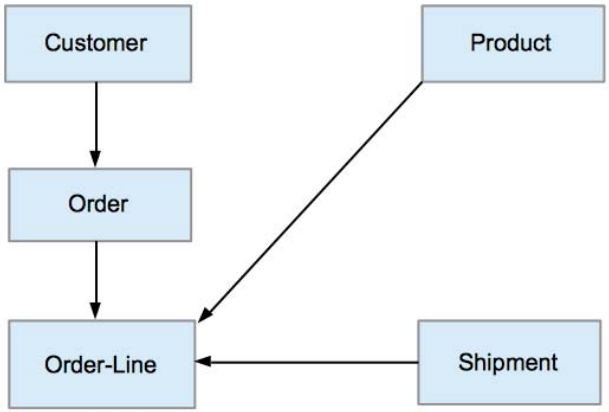
1960s IDS

- **Integrated Data Store (IDS) est un système de gestion de base de données réseau utilisé par l'industrie, réputé pour ses hautes performances.**

2) Historique des bases de données (2)

Base de données réseau

L'inventeur du modèle de réseau était Charles Bachman, et il a été développé dans une spécification standard publiée en 1969 par le Consortium CODASYL (Conference on Data Systems Languages).



Charles Bachman

2) Historique des bases de données (3)

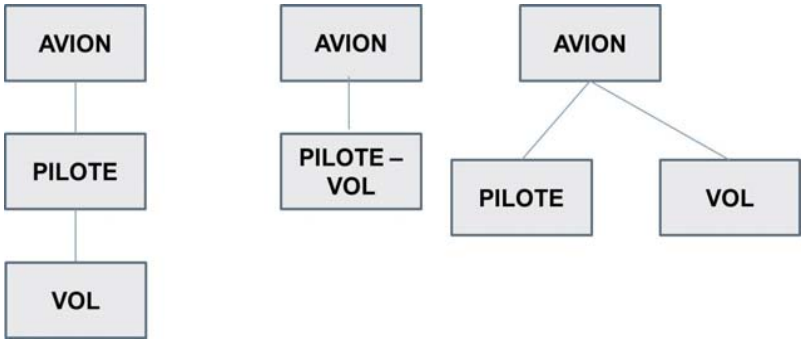
1960s - IBM IMS

- **IMS (pour Information Management System) a débuté comme une base de données hiérarchique créée par IBM en 1966 pour le compte de Rockwell et le programme Apollo.**

2) Historique des bases de données (4)

Base de données hiérarchique

Une base de données hiérarchique est une base de données dont le système de gestion lie les enregistrements dans une structure arborescente où chaque enregistrement n'a qu'un seul possesseur.



Il est particulièrement adapté aux organisations économiques et sociales.

Ses principaux points faibles sont :
- la recherche est difficile et coûteuse du fait que l'unique point d'entrée est la racine.
- une faible dépendance logique, qui induit un manque de sémantique.
- le changement de la structure des données implique une modification des applications qui accèdent aux données.

2) Historique des bases de données (5)

A noter

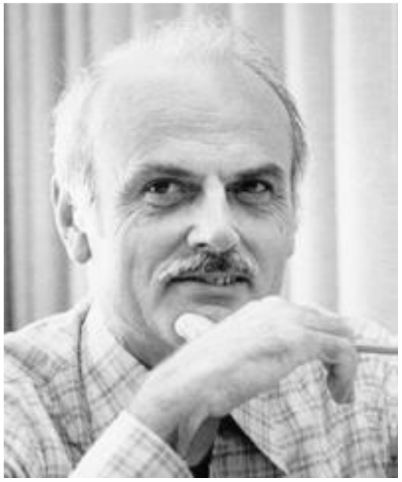
Ces deux modèles ne sont plus utilisés aujourd'hui, il me semblait intéressant de les évoquer du fait de leur valeur historique.

- Base de données réseau
- Base de données hiérarchique

2) Historique des bases de données (6)

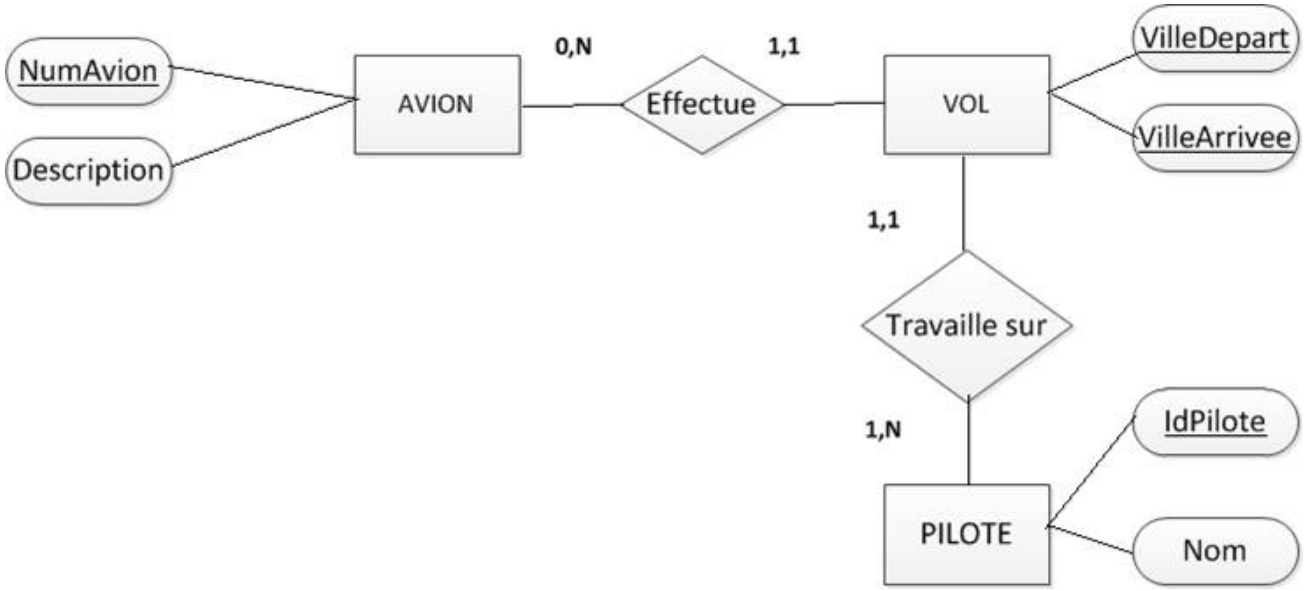
1970s – Le model relationnel

- **Le modèle relationnel est proposé dans les années 70 par E.F. Codd.**
- **Ce modèle permet de représenter à un plus bas niveau la structure d'une base de données.**



2) Historique des bases de données (7)

Le modèle relationnel



2) Historique des bases de données (6)

1970s – Le model relationnel

- **Premières implémentations de SGBD relationnel:**

→ System R - IBM Research

→ INGRES - U.C. Berkeley

→ Oracle - Larry Ellison


Gray


Stonebraker


Ellison

2) Historique des bases de données (7)

1980s – Le model relationnel

- **Le modèle relationnel gagne.**

→ IBM sort avec DB2 en 1983

→ “SEQUEL” devient la norme (SQL).

→ Beaucoup de nouveaux SGBD «d'entreprise»,
mais Oracle remporte le marché.

→ Stonebraker crée Postgres.

2) Historique des bases de données (7)

1980s – Base de données orientée objet

→ Dans une base de données à objets les informations sont regroupées sous forme d'objets.

→ Peu de ces SGBD originaux des années 1980 existent encore aujourd'hui, mais de nombreuses technologies existent sous d'autres formes (JSON, XML).

→ Plusieurs approchent existent pour manipuler les objets: Langages OQL

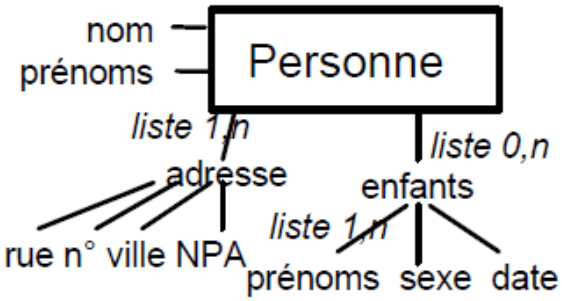
→ OQL est un langage de requête ayant une syntaxe proche de SQL mais qui permet une navigation entre objets.

2) Historique des bases de données (8)

1980s – Base de données orientée objet

En OO : **un** seul objet

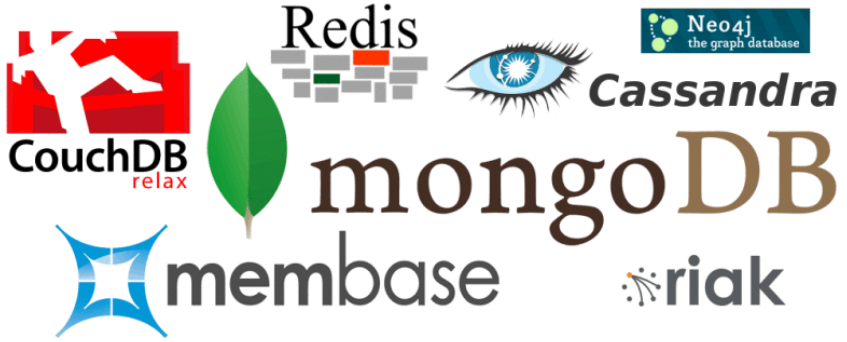
```
CLASS Personne
   { ATTRIBUTE nom : STRING ,
   ATTRIBUTE prénoms : LIST STRING ,
   ATTRIBUTE adresse : STRUCT  adr
        { rue : STRING  ,
        n° : STRING ,
        ville : STRING ,
        codeNPA : INT }
   ATTRIBUTE enfants : LIST STRUCT enfant
        { prénoms : LIST STRING ,
        sexe : ENUM {'M', 'F'} ,
        date : DATE }
   }
```



2) Historique des bases de données (9)

2010 – Base de données NoSQL

- **Le NoSQL, pour "not only SQL", désigne les bases de données qui ne sont pas fondées sur l'architecture classique des bases de données relationnelles.**
- **Développé à l'origine pour gérer du big data, l'utilisation de base de données NoSQL a explosée depuis quelques années.**
- **Lorsque l'on parle de NoSQL, on regroupe des systèmes de base de données qui ne sont pas relationnels, mais il faut savoir qu'il existe plusieurs types de bases de données "NoSQL".**



2) Historique des bases de données (10)

2010 – Base de données NoSQL

- **Les bases clef/valeur, permettent de stocker des informations sous forme d'un couple clef/valeur où la valeur peut être une chaine de caractère, un entier ou un objet sérialisé.**
- **Une base de données Redis.**
- **Ce type de base de données offre de très bonne performances par sa simplicité et peut même être utilisé pour stocker les sessions utilisateur ou le cache de votre site par exemple.**

redis

2) Historique des bases de données (11)

2010 – Base de données NoSQL

- **Les bases orientées colonnes, ressemble aux bases de données relationnelles, car les données sont sauvegardées sous forme de ligne avec des colonnes, mais se distingue par le fait que le nombre de colonnes peut varier d'une ligne à l'autre.**

- **Les solutions les plus connues sont HBase ou Cassandra.**

2) Historique des bases de données (12)

2010 – Base de données NoSQL

- **Les bases orientées document, représente les informations sous forme d'objet XML ou JSON.**
- **L'avantage est de pouvoir récupérer simplement des informations structurées de manière hiérarchique.**
- **Les solutions les plus connues sont CouchDB, RavenDB et MongoDB.**

2) Historique des bases de données (13)

2010 – Base de données NoSQL

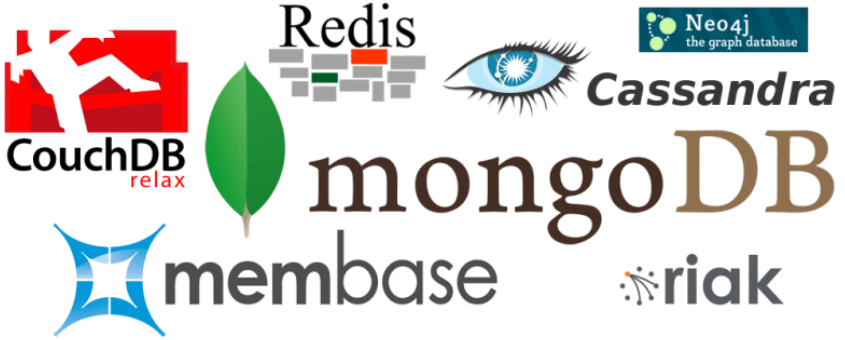
- **Les bases orientées graphe, présentent les données sous forme de noeud et de relation.**
- **Cette structure permet de récupérer simplement des relations complexes.**
- **Un exemple de base graphe est Neo4J.**

2) Historique des bases de données (14)

2010 – Base de données NoSQL

- **Les performances**
- **On lit assez souvent que les bases de données non relationnelles sont plus "performantes" et plus "scalable".**

- **Le problème des bases de données relationnelles est qu'elles sont difficiles à distribuer sur plusieurs serveurs forçant souvent à faire du "vertical scaling". On augmente la puissance du serveur qui accueille la base de données afin d'obtenir de meilleures performances et mieux tenir la charge. Les bases de données NoSQL supportent "l'horizontal scaling", une base de données peut être distribuée sur une "pool" de serveurs ce qui permet de mutualiser les performances. L'avantage de cette méthode est qu'elle permet d'adapter l'architecture en fonction de la charge en distribuant sur plus ou moins de serveurs suivant les cas.**

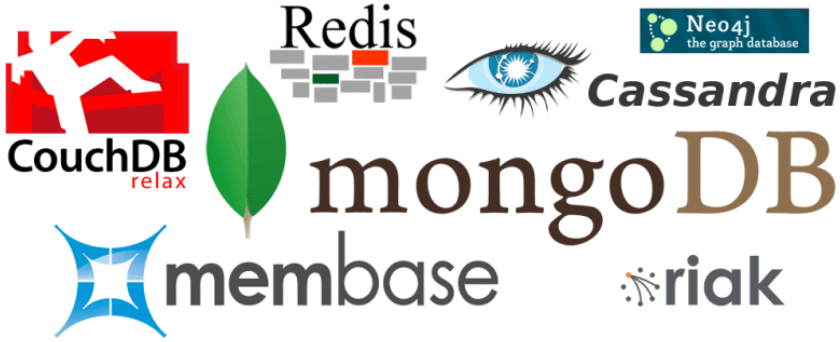


2) Historique des bases de données (15)

2010 – Base de données NoSQL

- **NoSQL > SQL ?**
- **En lisant quelques benchmarks, on peut alors se demander : pourquoi continuer à utiliser des bases de données relationnelles si le NoSQL est plus performant et plus scalable ?**

- **Afin d'obtenir de meilleures performances, les bases de données NoSQL ont abandonné certaines fonctionnalités proposées par défaut par les bases relationnelles comme les transactions ou encore les vérifications d'intégrités. On peut ici faire la même analogie que Dare Obasanjo avec les boites de vitesses. Le NoSQL est l'équivalent d'une boîte de vitesse manuelle, elle permet d'obtenir de meilleure performance à condition de savoir quand et comment passer les vitesses. Les bases de données relationnelles, comme les boites automatiques permettent de ne pas avoir à se soucier de ce qui se passe derrière en automatisant les choses.**

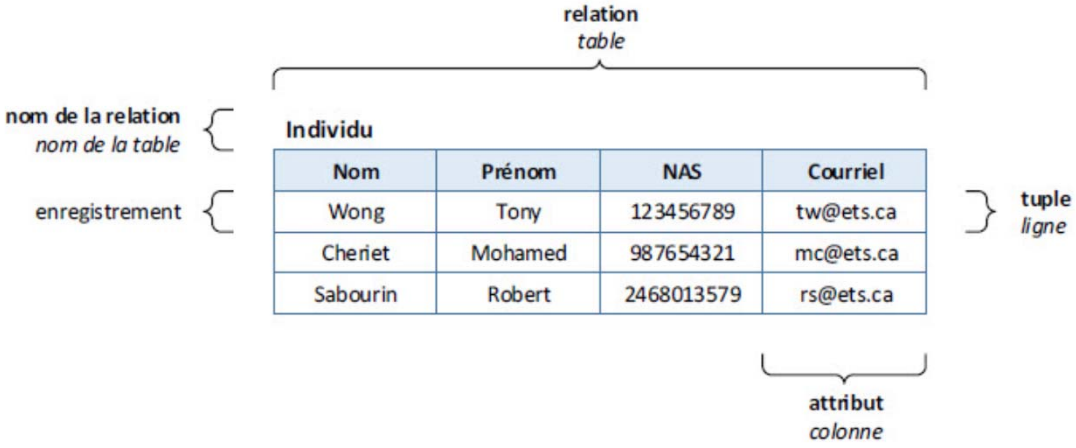


3) Lien entre le modèle Entité Association et le modèle relationnel

Modèle E/A	**Modèle relationnel**
Association	Entité Table(relation)
Propriété	Attribut
Identifiant	Clé Primaire

Le modèle relationnel

- Les données sont organisées en relations
    - Tables: relations
    - Colonnes: attributs
    - Lignes: n-uplets (ou tuples)



Le modèle relationnel

- Schéma d'une base de données relationnel
    - Ensemble de noms de tables
    - Ensemble d'attributs pour chaque table

STUDENT [Num, FirstName, LastName, BirthYear]

INSCRIPTION[Num, CourseCode, Year]

- **Instance d'une bases de données**
- **Ensemble de valeurs dans une table (ensemble de n-uplets)**

{h2008120,Dumont,Marie, 1980i, h2008122,Dubois, Paul, 1980}

Le modèle relationnel étendu

- Le modèle MRE est une extension du modèle MR. Ainsi, il inclus tout ce qui existe dans le modèle de base et y ajoute les concepts suivants :
    - La généralisation et la spécialisation;
    - Attribues composites / multi values;
    - Associations;
    - L'agrégation.

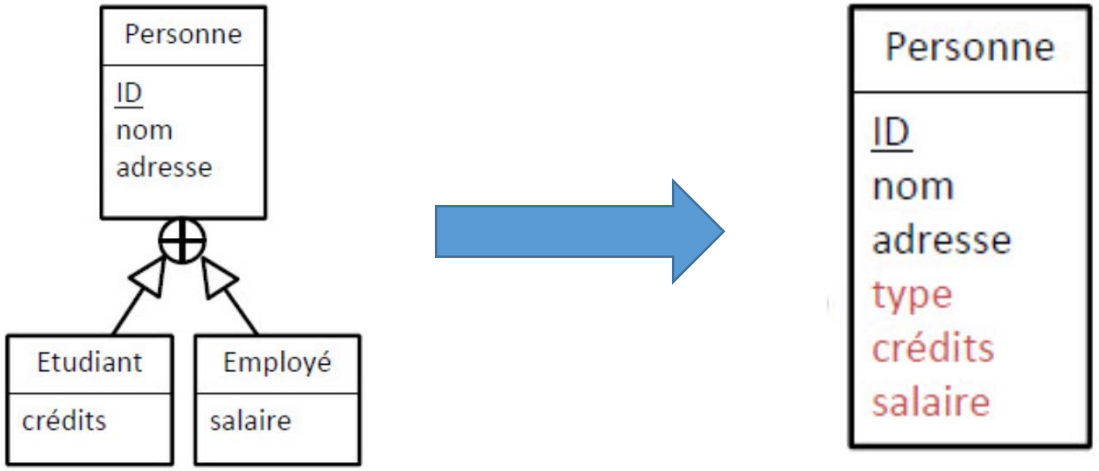
Le modèle relationnel étendu

La généralisation et la spécialisation

- Trois solutions possibles

1- Première solution : conserver uniquement la super-entité

- Exemple



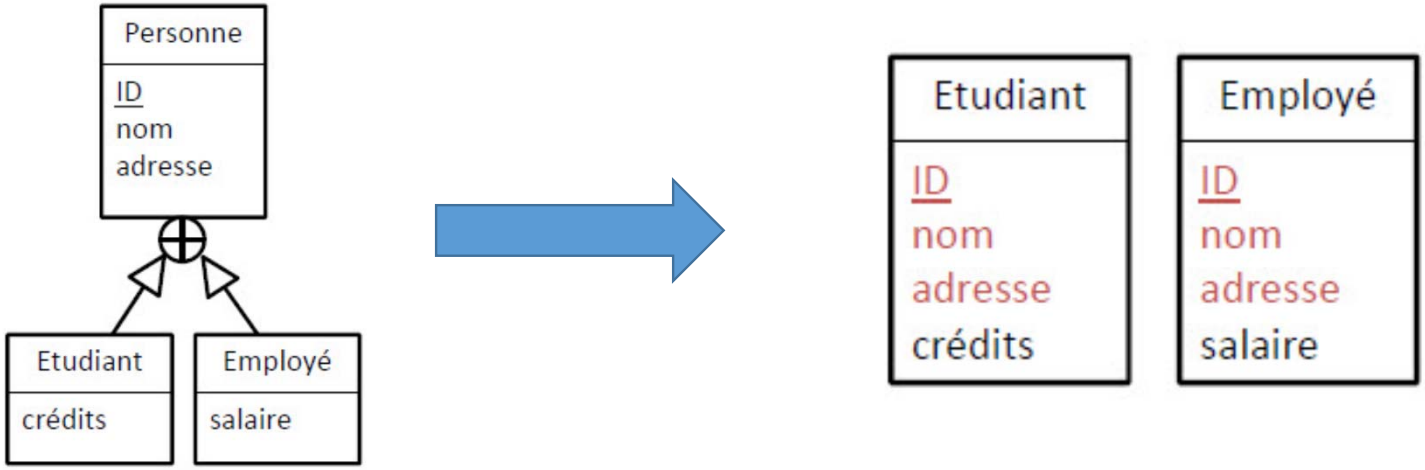
Le modèle relationnel étendu

La généralisation et la spécialisation

- Trois solutions possibles

2- Deuxième solution: Conserver uniquement les spécialisations Exemple

- Exemple



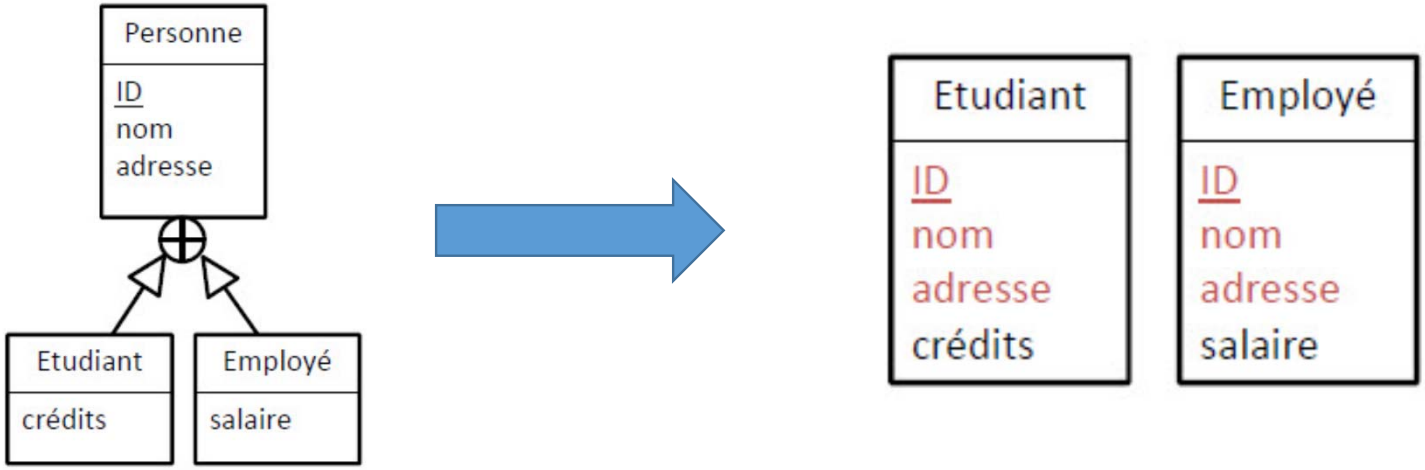
Le modèle relationnel étendu

La généralisation et la spécialisation

- Trois solutions possibles

2- Deuxième solution: Conserver uniquement les spécialisations Exemple

- Exemple



Le modèle relationnel étendu

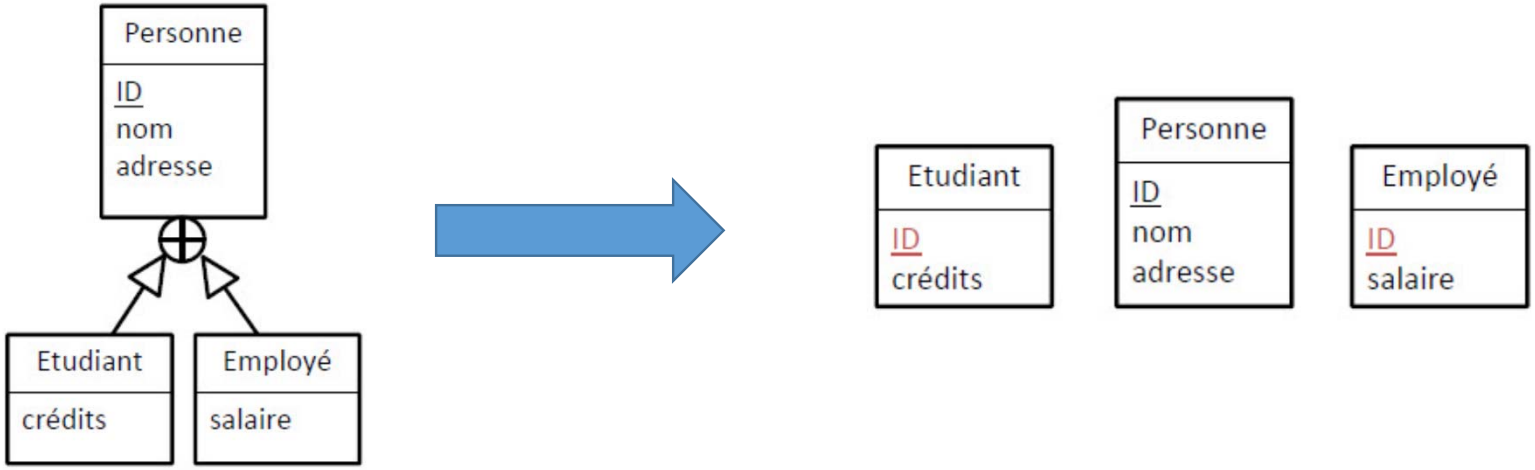
La généralisation et la spécialisation

- Trois solutions possibles

3- Troisième solution: conserver toutes les entités

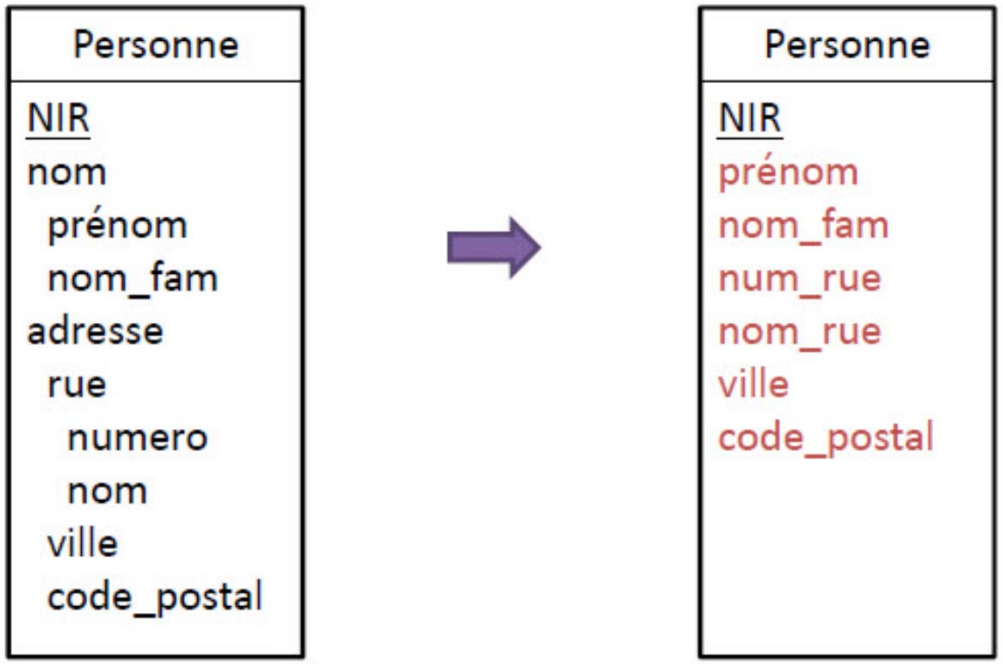
Le schéma est factorisé (seule la clé est partagée)

Exemple



Le modèle relationnel étendu

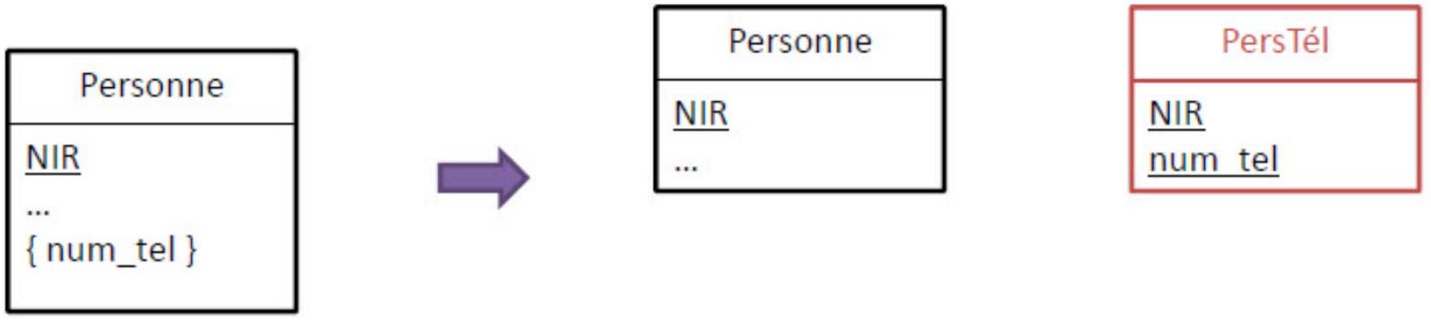
Attribues composites



Le modèle relationnel étendu

Attribues multi values (1)

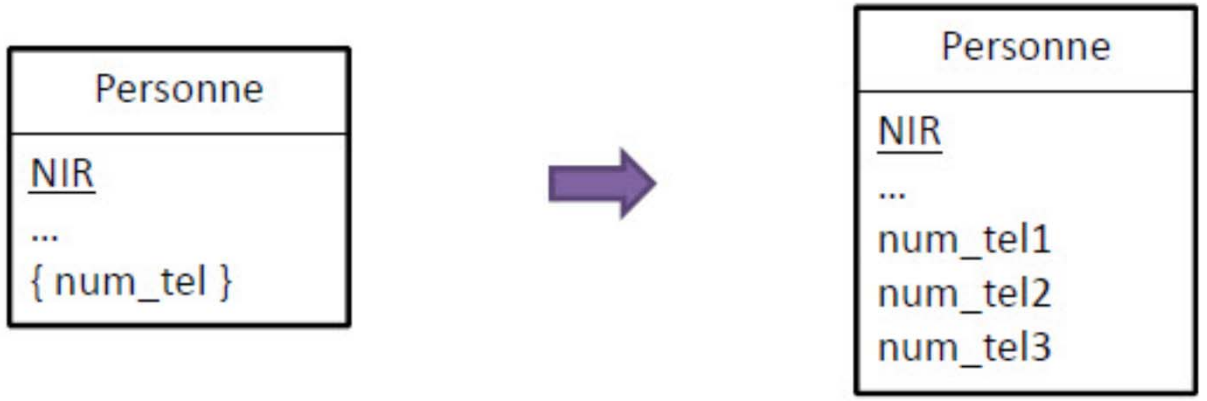
- Deux traductions possibles
- Cas général : création d'une nouvelle entité
- Exemple



Le modèle relationnel étendu

Attribues multi valués (2)

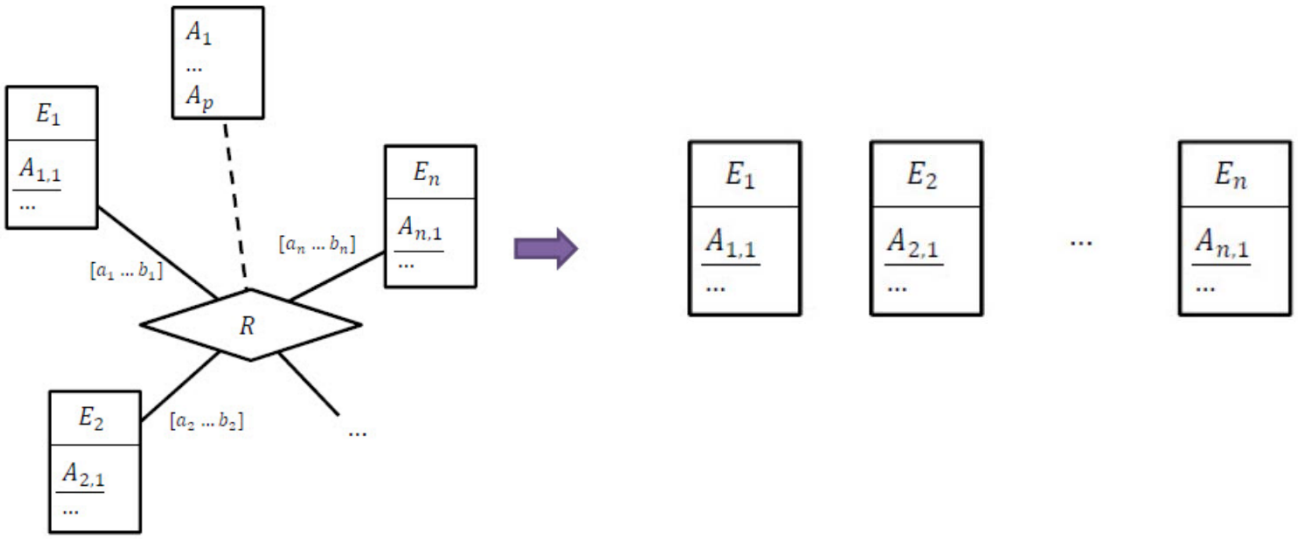
- Deux traductions possibles
- Cas général : multiplication de l'attribut
- Exemple



Le modèle relationnel étendu

Associations

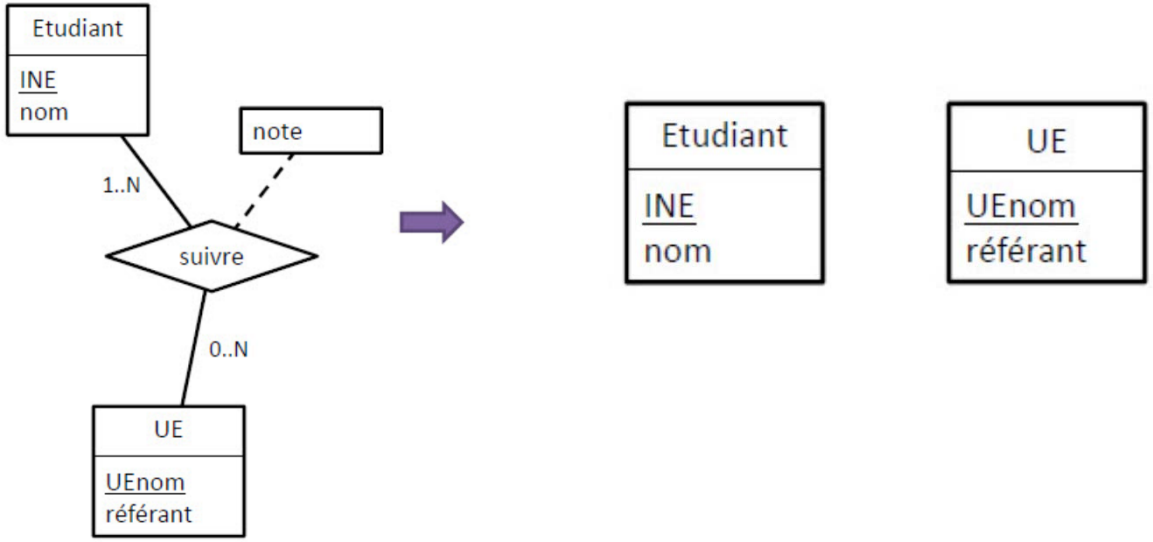
- Technique générale



Le modèle relationnel étendu

Associations

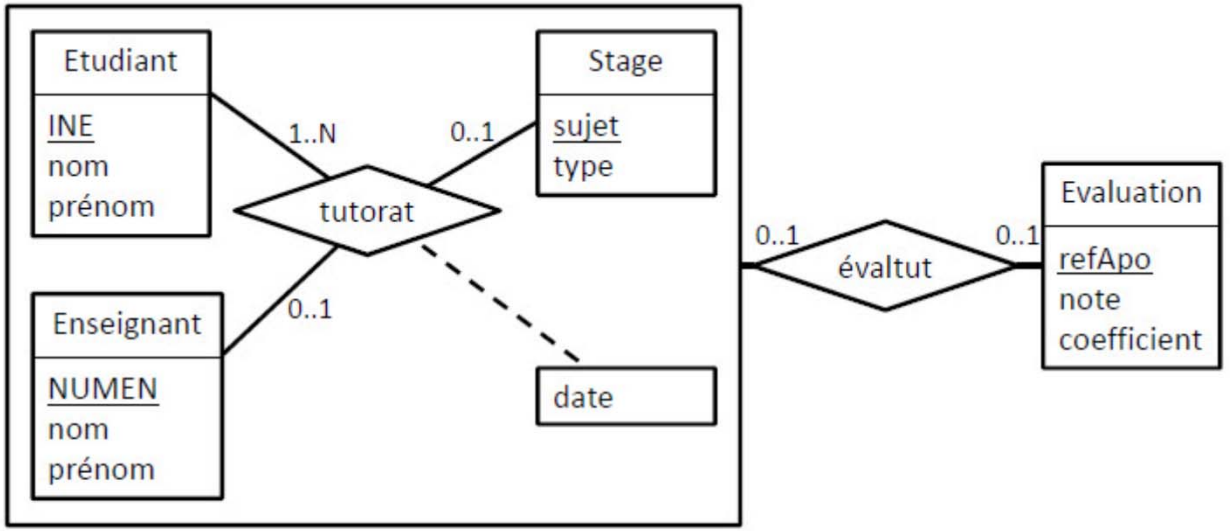
- Technique générale
- Exemple



Le modèle relationnel étendu

Agrégations

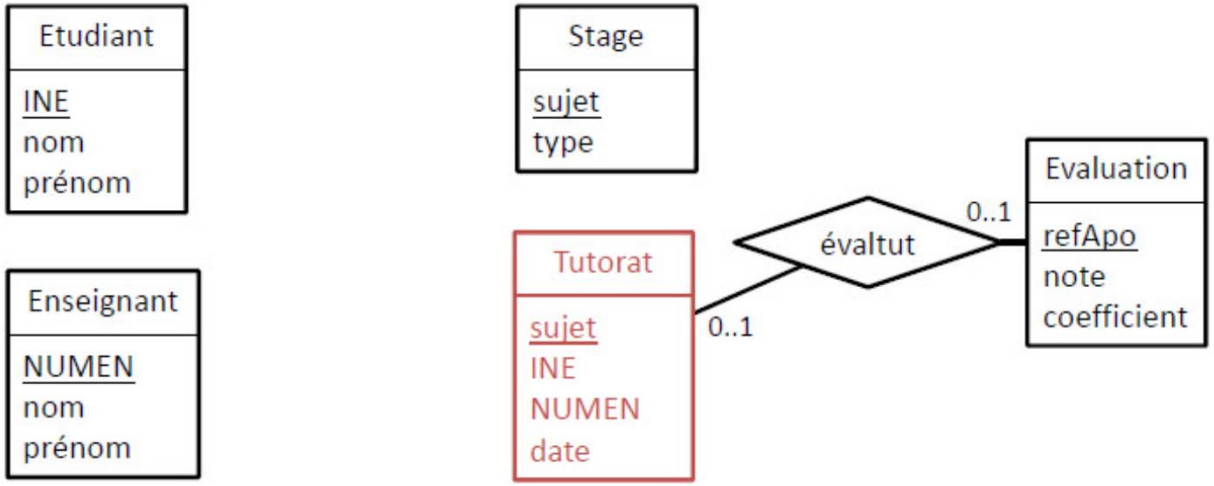
- Exemple



Le modèle relationnel étendu

Agrégations

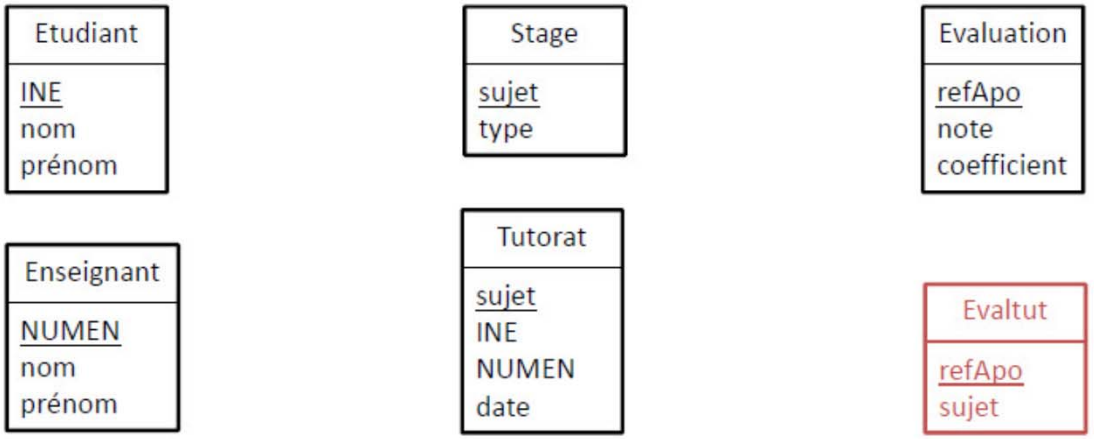
- Transformation
- Association agrégée traduite en entité
- Association d'ordre supérieur ramenée à une association de première ordre



Le modèle relationnel étendu

Agrégations

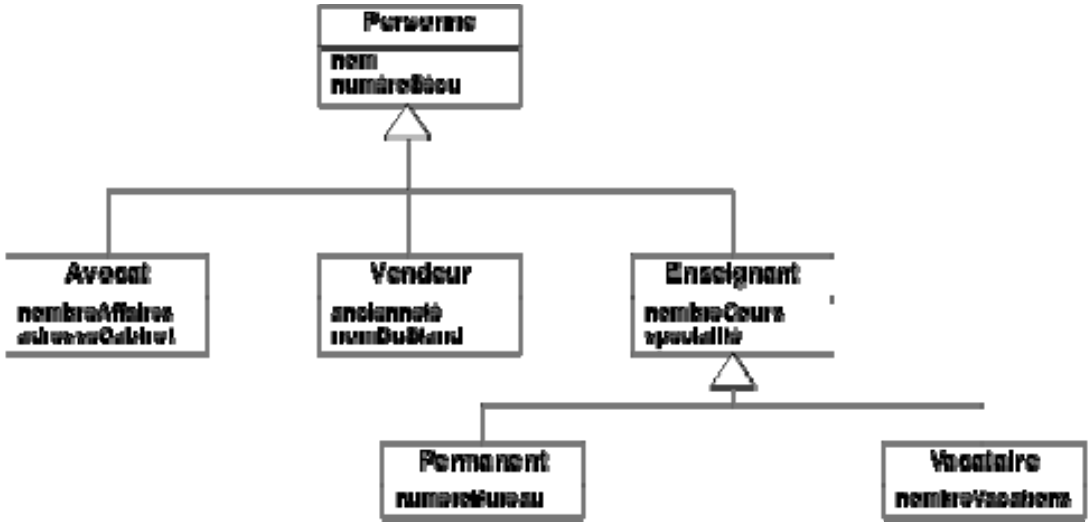
- Transformation trivial
- Association agrégée traduite en entité
- Association d'ordre supérieur ramenée à une association de première ordre



Exercice 1 : Le model relationnel étendu

Traduire en schéma relationnel le diagramme entité association suivant.

La généralisation et la spécialisation



Le SQL

Le SQL

- **Structured Query Language (SQL) :** sert à
  - Créer, modifier, supprimer des tables, des contraintes, et d'autres structures…
    - ✓ ……
  - Insérer, modifier, supprimer des tuples
    - ✓ ……
  - Récupérer le contenu d'une table, sélectionner des tuples vérifiant certains critères, effectuer des calculs…
    - ✓ ……
  - Gestion de la sécurité, droits accès…
    - ✓ ……

Le SQL

- **Structured Query Language (SQL) :** sert à
  - Créer, modifier, supprimer des tables, des contraintes, et d'autres structures…
    - ✓ Langage de **définition des données (LDD)**
  - Insérer, modifier, supprimer des tuples
    - ✓ Langage de **manipulation des données (LMD)**
  - Récupérer le contenu d'une table, sélectionner des tuples vérifiant certains critères, effectuer des calculs…
    - ✓ Le langage **d'interrogation des données (LID)**
  - Gestion de la sécurité, droits accès
    - ✓ Langage de **contrôle des données (LCD)**

Syntaxe générale

- Le LDD de SQL offre un ensemble de commandes pour **manipuler les structures**
  - Créer / Supprimer une base de données :
    - ……
  - Créer / Supprimer une table:
    - ……
  - Modifier une table:
    - ……

Syntaxe générale

- Le LDD de SQL offre un ensemble de commandes pour **manipuler les structures**
  - Créer / Supprimer une base de données :
    - **CREATE DATABASE et DROP DATABASE**
      - **Se fait généralement avec l'interface du SGBD**
  - Créer / Supprimer une table:
    - **CREATE TABLE et DROP TABLE <if exists>**
  - Modifier une table:
    - **ALTER TABLE**

Création de tables

- Syntaxe

  create table <nomtable> (

  <nom_attribut1> <type_att1> <contrainte_att1> <valeur_par_defaut>,

  <nom_attribut2> <type_att2> <contrainte_att2> <valeur_par_defaut>

  );

- Exemple :

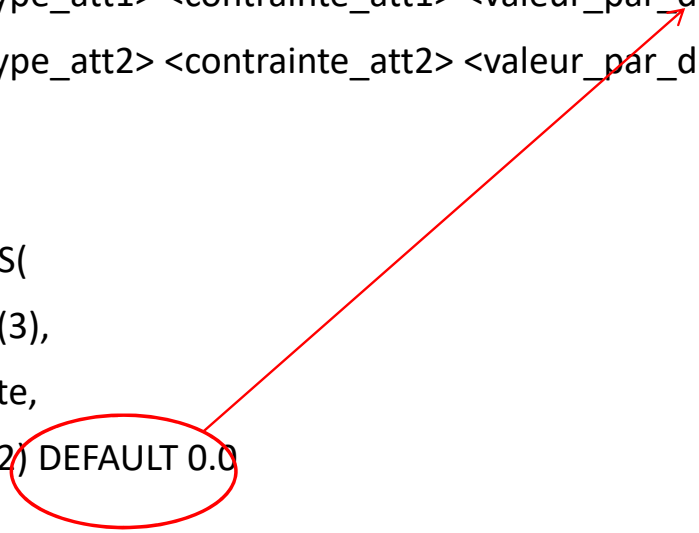
  create table COMMANDES(

  no_commande char(3),

  date_commande date,

  montant numeric(9,2) DEFAULT 0.0

  );



Expression de contraintes statiques

- Il existe différents types de contraintes
  - **Contrainte d'existence** (la valeur de l'attribut ne peut être nulle)
  - **Contrainte de clé primaire**
  - **Contrainte d'unicité** (deux tuples différents ne peuvent avoir la même valeur pour cet attribut)
  - **Contrainte d'intégrité référentielle** (clé étrangère)
  - **Contraintes de domaine** : la valeur de l'attribut doit appartenir à
    - Une liste
    - Un intervalle
    - Un format de données donné

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (1/3)

```
create table COMMANDES(
        no_commande char(3) ……,
        date_commande date,
        montant numeric(9,2) not null default 0.00
);
```

Contrainte d'existence

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (1/3)

```
create table COMMANDES(
        no_commande char(3) primary key,
        date_commande date,
        montant numeric(9,2) not null default 0.00
);
```

Contrainte d'existence

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (1/3)

```
create table FOURNISSEURS(
        no_four char(3) ……,
        raison_sociale varchar(20) not null,
        ville varchar(20) not null
);
```

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (1/3)

```
create table FOURNISSEURS(
        no_four char(3) primary key,
        raison_sociale varchar(20) not null,
        ville varchar(20) not null
);
```

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (2/3)

```
create table PRODUITS(
        ref_produit char(3),
        designation varchar(20) not null,
        prix_unitaire numeric(9,2) not null default 0.00,
        no_four char(3),
        ……
);
```

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (2/3)

```
create table PRODUITS(
        ref_produit char(3),
        designation varchar(20) not null,
        prix_unitaire numeric(9,2) not null default 0.00,
        no_four char(3),
        primary key (ref_produit)
);
```

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (3/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null,
        reference varchar(20) not null ,
        quantite integer not null default 0 ,
);
```

……

……

Expression de contraintes statiques

- **Clé primaire** : on peut la déclarer de 3 façons (3/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null,
        reference varchar(20) not null ,
        quantite integer not null default 0 ,
);
Alter table LIGNES_CDE
   add primary key (no_commande, reference);
```

Exercice 2
Expression de contraintes statiques

Ajouter la clé primaire « id » avec les trois façons…

CREATE TABLE utilisateur (

id INT NOT NULL AUTO_INCREMENT,

nom VARCHAR(50),

email VARCHAR(50),

date_inscription DATE,

);

Expression de contraintes statiques

- **Clé étrangère** : on peut la déclarer de 3 façons (1/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null ……
        ……..,
        reference varchar(20) not null ……
        ……..,
        quantite integer not null default 0
);
```

Expression de contraintes statiques

- **Clé étrangère** : on peut la déclarer de 3 façons (1/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null foreign key references
           COMMANDES(no_commande),
        reference varchar(20) not null foreign key references
           PRODUITS(ref_produit),
        quantite integer not null default 0
);
```

Expression de contraintes statiques

- **Clé étrangère** : on peut la déclarer de 3 façons (2/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null,
        reference varchar(20) not null,
        quantite integer not null default 0,
        ……..
        ……..
);
```

Expression de contraintes statiques

- **Clé étrangère** : on peut la déclarer de 3 façons (2/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null,
        reference varchar(20) not null,
        quantite integer not null default 0,
        foreign key (no_commande) references COMMANDES(no_commande),
        foreign key (reference) references PRODUITS(ref_produit)
);
```

Expression de contraintes statiques

- **Clé étrangère** : on peut la déclarer de 3 façons (3/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null,
        reference varchar(20) not null,
        quantite integer not null default 0,
);
```

……

……

Expression de contraintes statiques

- **Clé étrangère** : on peut la déclarer de 3 façons (3/3)

```
create table LIGNES_CDE (
        no_commande char(3) not null,
        reference varchar(20) not null,
        quantite integer not null default 0,
);
```

Alter table LIGNES_CDE add foreign key (no_commande) references COMMANDES(no_commande);

Alter table LIGNES_CDE add foreign key (reference) references PRODUITS(ref_produit);

Exercice 3
Expression de contraintes statiques

Ajouter la clé étrangère «PersonID **» de la table** Persons(PersonID) **utilisant les trois façons…**

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   OrderNumber int NOT NULL,
   PersonID int,
   PRIMARY KEY (OrderID),
);
```

Expression de contraintes statiques

- **Clé unique** : 3 manières de la déclarer (remplacer « primary key » par « unique »). Par exemple :

```
create table MEMBRES(
        no_membre char(10) not null,
        nom varchar(50) not null,
        prenom varchar(30) not null,
        adresse varchar(60),
        no_licence char(20) not null,
        primary key(no_membre),
        ……
);
```

Expression de contraintes statiques

- **Clé unique** : 3 manières de la déclarer (remplacer « primary key » par « unique »). Par exemple :

```
create table MEMBRES(
        no_membre char(10) not null,
        nom varchar(50) not null,
        prenom varchar(30) not null,
        adresse varchar(60),
        no_licence char(20) not null,
        primary key(no_membre),
        unique (no_licence)
);
```

Expression des autres contraintes statiques

- **Contrainte de domaine :**
  - **Expression d'une liste : CHECK et IN**
  - **Expression d'un intervalle : CHECK et BETWEEN**

```
create table produits (
        reference char(3) primary key,
        tva float not null default 20.0 ....................,
        quantite integer default 1 ....................
);
```

Expression des autres contraintes statiques

- **Contrainte de domaine :**
  - **Expression d'une liste : CHECK et IN**
  - **Expression d'un intervalle : CHECK et BETWEEN**

```
create table produits (
        reference char(3) primary key,
        tva float not null default 20.0 check(tva in (20.0, 7.5)),
        quantite integer default 1 .....
);
```

Expression des autres contraintes statiques

- **Contrainte de domaine :**
  - **Expression d'une liste : CHECK et IN**
  - **Expression d'un intervalle : CHECK et BETWEEN**

```
create table produits (
        reference char(3) primary key,
        tva float not null default 20.0 check(tva in (20.0, 7.5)),
        quantite integer default 1 check (quantite between 1 and 100)
);
```

Expression des autres contraintes statiques

- **Contrainte de domaine :**
  - **Expression d'un format de données : CHECK, LIKE, _ , %**

create table produits (

reference char(3) primary key ……,

);

Expression des autres contraintes statiques

- **Contrainte de domaine :**
  - **Expression d'un format de données : CHECK, LIKE, _ , %**

```
create table produits (
        reference char(3) primary key check (reference like 'R__')
);
```

Création de domaine

•Lorsque la contrainte est complexe ou qu'elle est réutilisée plusieurs fois, il devient intéressant de créer un DOMAINE

•Création du domaine :

```
CREATE DOMAIN  domain_reference as text
 check ( VALUE LIKE 'DT__') );
```

•Utilisation du domaine :

```
CREATE TABLE PRODUITS(
   reference domain_reference primary key,
   designation text,
   PAchat float,
   PVente float ) ;
```

Création de domaine

•Exercice 1 : créer un domaine pour contraindre le résultat d'un jeu à être soit « échec », soit « succès »

……

……

Création de domaine

•Exercice 1 : créer un domaine pour contraindre le résultat d'un jeu à être soit « échec », soit « succès »

```
CREATE DOMAIN domain_resultat_jeu as varchar(6)
check  ( VALUE in ('échec','succès') );
```

Création de domaine

• Le domaine peut aussi représenter des contraintes plus complexes

•Les définitions de domaine peuvent utiliser les définitions d'expressions rationnelles (régulières) avec les notions d'atomes et de quantifieurs :

•Exemple :

```
CREATE DOMAIN  codepostal as text
check ( VALUE  ~ E'^\\d{5}$' );
```

^ indique le début de chaîne, $ la fin

\\d{5}: \d indique des CHIFFRES OBLIGATOIRES et {5} leur nombre

Création de domaine

Exercice 2 :

•Créez un domaine représentant une référence de magasin. Celle-ci sera composée obligatoirement de 'MA' suivi de 5 chiffres entre 1 et 9 et suivi par une lettre S ou C.

•Exemples : MA12585S, MA22222C …

……

……

•Construisez une relation MAGASIN (ref, CP) utilisant les deux domaines précédemment définis

……

Création de domaine

Exercice 2 :

•Créez un domaine représentant une référence de magasin. Celle-ci sera composée obligatoirement de 'MA' suivi de 5 chiffres entre 1 et 9 et suivi par une lettre S ou C.

•Exemples : MA12585S, MA22222C …

CREATE DOMAIN domain_reference as text

check (VALUE ~ E'^MA[1-9]{5}[SC]$');

•Construisez une relation MAGASIN (ref, CP) utilisant les deux domaines précédemment définis

Création de domaine

Exercice 2 :

•Créez un domaine représentant une référence de magasin. Celle-ci sera composée obligatoirement de 'MA' suivi de 5 chiffres entre 1 et 9 et suivi par une lettre S ou C.

•Exemples : MA12585S, MA22222C …

CREATE DOMAIN domain_reference as text

check (VALUE ~ E'^MA[1-9]{5}[SC]$');

•Construisez une relation MAGASIN (ref, CP) utilisant les deux domaines précédemment définis

create table MAGASIN (ref domain_reference, CP codepostal);

Modification de la structure d'une table

- **ALTER TABLE <nomtable> ACTION <modification>**
  - où l'action est :
    - ADD (ajouter un attribut, une contrainte…)
      - ……
    - ALTER (modifier la définition d'un attribut, une contrainte…)
      - ……
    - DROP pour supprimer un attribut, une contrainte…
      - **…….**

Modification de la structure d'une table

- **ALTER TABLE <nomtable> ACTION <modification>**
  - où l'action est :
    - ADD (ajouter un attribut, une contrainte…)
      - **ALTER TABLE commande ADD date_commande date;**
    - ALTER (modifier la définition d'un attribut, une contrainte…)
      - **…….**
      - **…….**
    - DROP pour supprimer un attribut, une contrainte…
      - **…….**

Modification de la structure d'une table

- **ALTER TABLE <nomtable> ACTION <modification>**
  - où l'action est :
    - ADD (ajouter un attribut, une contrainte…)
      - **ALTER TABLE commande ADD date_commande date;**
    - ALTER (modifier la définition d'un attribut, une contrainte…)
      - **ALTER TABLE commande ALTER date_commande SET default current_date;**
      - **ALTER table commande ALTER date_commande TYPE datetime;**
    - DROP pour supprimer un attribut, une contrainte…
      - **…….**

Modification de la structure d'une table

- **ALTER TABLE <nomtable> ACTION <modification>**
  - où l'action est :
    - ADD (ajouter un attribut, une contrainte…)
      - **ALTER TABLE commande ADD date_commande date;**
    - ALTER (modifier la définition d'un attribut, une contrainte…)
      - **ALTER TABLE commande ALTER date_commande SET default current_date;**
      - **ALTER table commande ALTER date_commande TYPE datetime;**
    - DROP pour supprimer un attribut, une contrainte…
      - **ALTER TABLE commande DROP date_commande;**