Bases de Données – Conception, Modèle relationnel, SQL

IUT Paris Descartes

2019

Plan du cours

- Introduction
  - Concepts et définitions
  - Structures
- Conception et Modèle Entité-Association
  - Démarche
  - Modélisation
- Modèle relationnel
  - Éléments du modèle
  - Passage MCD–MRD
- Exemple
- Langage SQL

Plan

Applications classiques

- Gestion des personnels, étudiants, cours, inscriptions, ... d'une université
- Système de réservation (hotel, billet de trains etc.)
- Gestion des comptes clients d'une banque
- Gestion des commandes (Amazon)

Définitions

- **Base de données** : ensemble cohérent, intégré, partagé de données structurées défini pour les besoins d'une application
  - Ensemble de données non indépendantes,
  - Interrogeable par le contenu (selon des critères),

- **SGBD (Système de Gestion de Bases de Données)** : Ensemble de logiciels systèmes permettant aux utilisateurs d'insérer, de modifier et de rechercher efficacement des données spécifiques dans une grande masse d'informations partagée par de multiples utilisateurs.

Composition d'un SGBD

- Trois couches principales :
  1. **Couche Externe** : dialogue avec les utilisateurs , vues associées à chaque groupe d'utilisateurs
  2. **Couche Logique** : contrôle global et structure globale des données
  3. **Couche Interne** : stockage des données sur des supports physiques, gestion des structures de mémorisation (fichiers) et d'accès (gestion des index, des clés, ...)

Exemple de parcours d'une requête

1. Analyse syntaxique et sémantique d'une requête
2. Traduction au niveau logique
3. Contrôles de confidentialité, concurrence...
4. Si la requête est acceptée, optimisation et découpage en sous-requêtes élémentaires transférées au niveau interne
5. Au niveau interne, traduction des sous-requêtes en requêtes physiques correspondantes.

Objectifs des SGBD

- Objectifs principaux
  - Indépendance physique et logique des programmes aux données
  - Manipulation des données par des langages non procéduraux
  - Administration facilité des données

- Objectifs additionnels
  - Efficacité des accès aux données
  - Partage des données
  - Cohérence des données
  - Redondance contrôlée des données
  - Sécurité des données

BD, SGBD et Systèmes d'Information

- Définition : Un **système d'information** (noté SI) représente l'ensemble des éléments participant à la gestion, au stockage, au traitement, au transport et à la diffusion de l'information au sein d'une organisation (*Wikipedia*)

- Composé entre autres de
  - Bases de données et SGBD
  - Applications métiers
  - Réseau, dispositifs de sécurité, ...

- Système global permettant d'acquérir, de stocker, de structurer et de communiquer des informations sous forme de textes, images, sons ou données codées

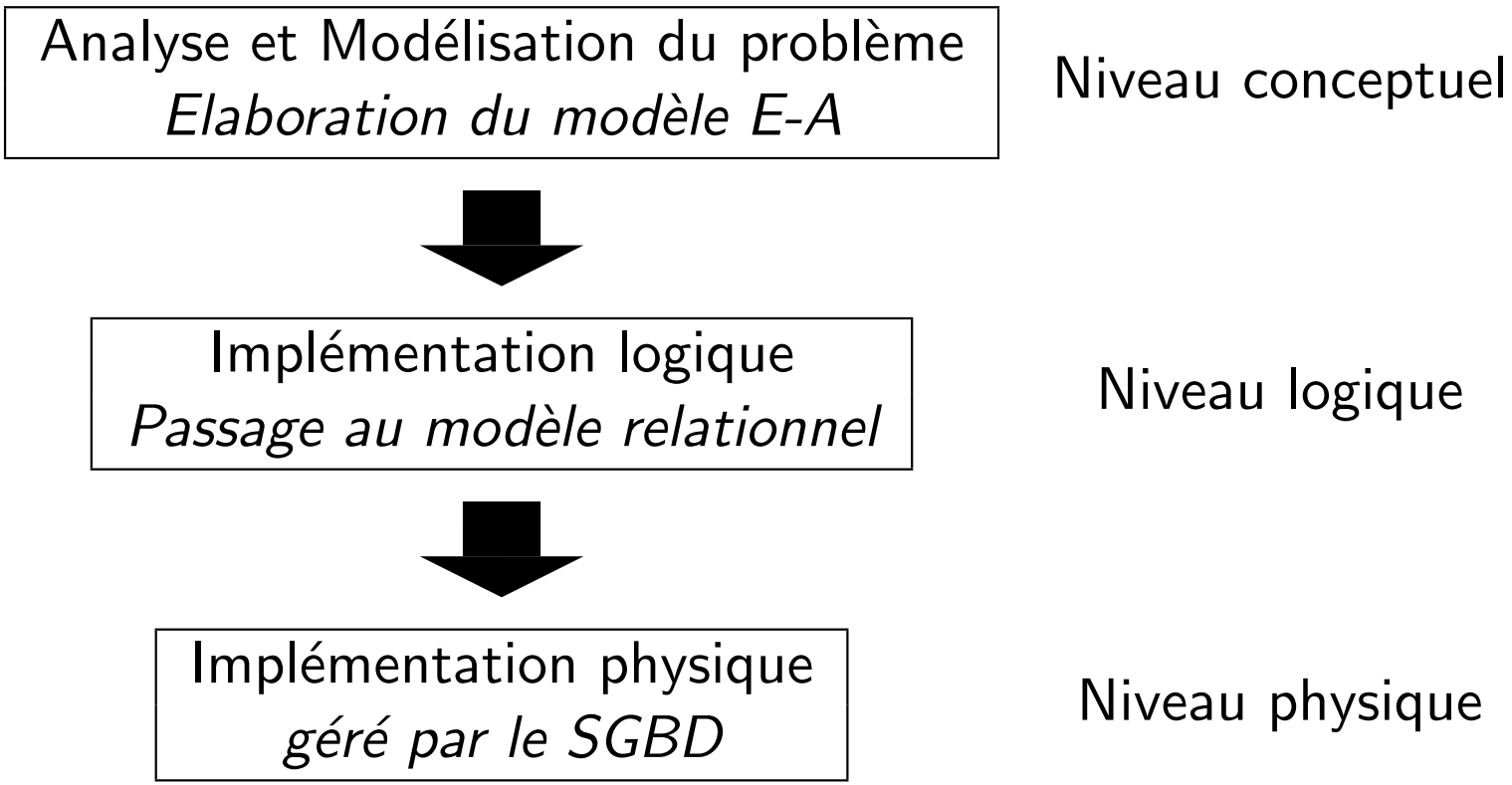
Différents types de BD

- Organisation possible
  - Hiérarchique
  - Réseau
  - Relationnel
  - Objet

- BD Relationnelles
  - Modèle le plus courant
  - Disponible dans les logiciels usuels
    - Simple : Access, FileMaker, FoxPro, ...
    - Gros sytèmes : Oracle, DB2, Sybase, SQL Server, ...
    - Gratuite : MySQL, MSQL, PostGre, InstantDB, ...

Démarche de conception

- Schéma de la démarche à utiliser pour concevoir une BD



Modèle E-A

- Modèle Entité-Association créé en 1976
  - Modèle Conceptuel des Données (MCD) avec MERISE en 1978-1979

- Objets représentés par des **Entités**
- Liens sémantiques représentés par des **Associations**

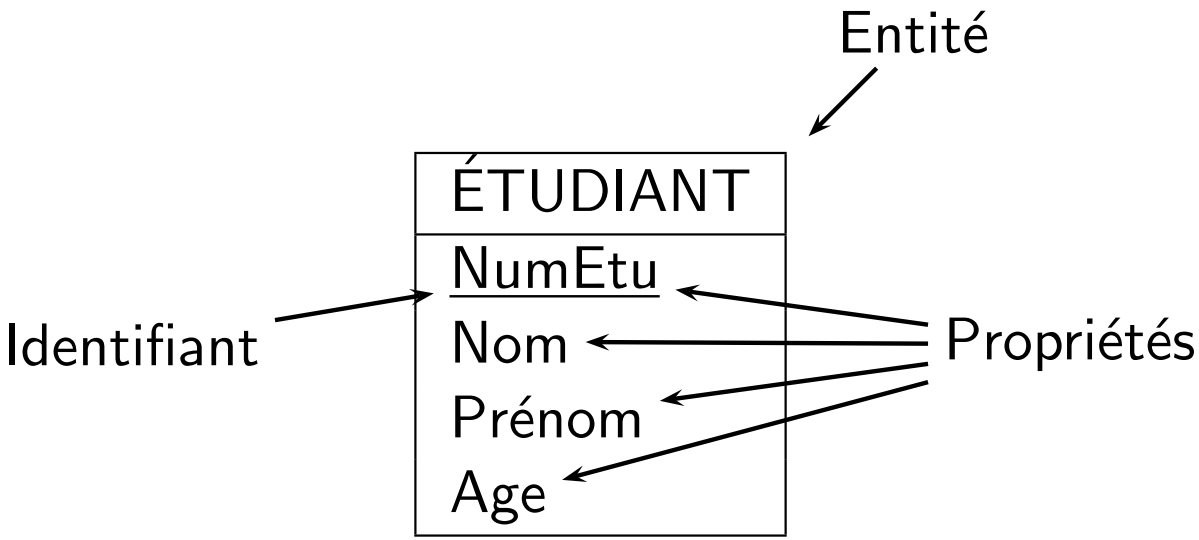
- Objectifs
  - Favoriser le dialogue avec les experts du domaine
    - Facilité de compréhension
    - Rigueur, pas d'ambigéité
  - Représenter graphiquement le modèle
    - Aucun lien avec le niveau logique ou physique

Entités

- Définitions :
  - une **Entité** est un ensemble d'objets du même type défini par un ensemble de propriétés
  - une **Propriété** est une données ayant un sens et pouvant être utilisée de manière autonome
  - une **Occurence** est une réalisation particulière d'une entité
  - un **Identifiant** (ou **clé**) est une propriété dont la valeur discrimine une occurence par rapport à toutes les autres

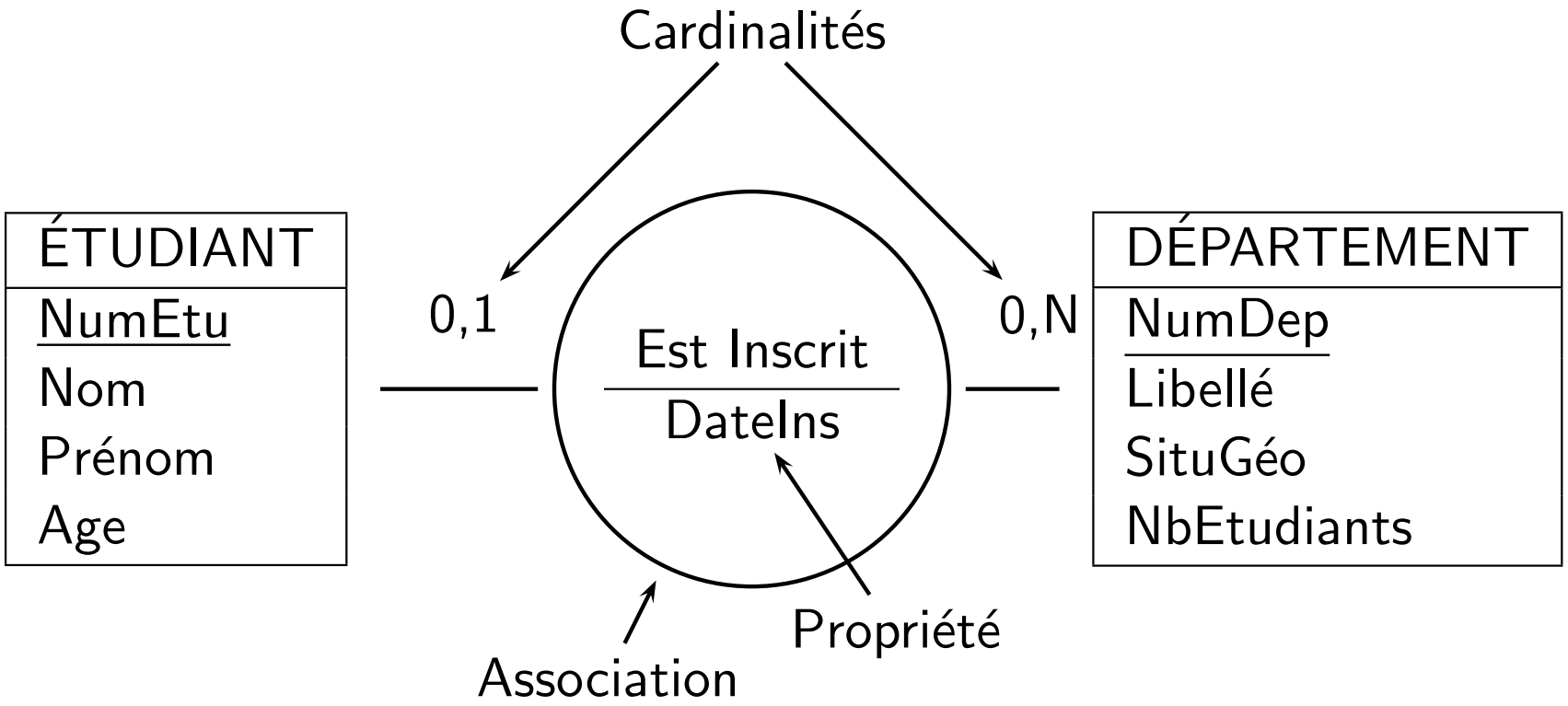
Entités

- Exemple
  - Entité : ETUDIANT
  - Propriétés : Numéro d'étudiant, Nom, Prénom, Age
  - Clé : Numéro d'étudiant
  - Occurence : Jacques Dupont, 23 ans, numéro d'étudiant 123456789



Associations

- Une **Association** est un lien sémantique reliant deux entités et pouvant avoir des propriétés
- Les **Cardinalités** sont les nombres *minimum* et *maximum* d'occurences d'une entité par rapport à une association



Exemple E-A

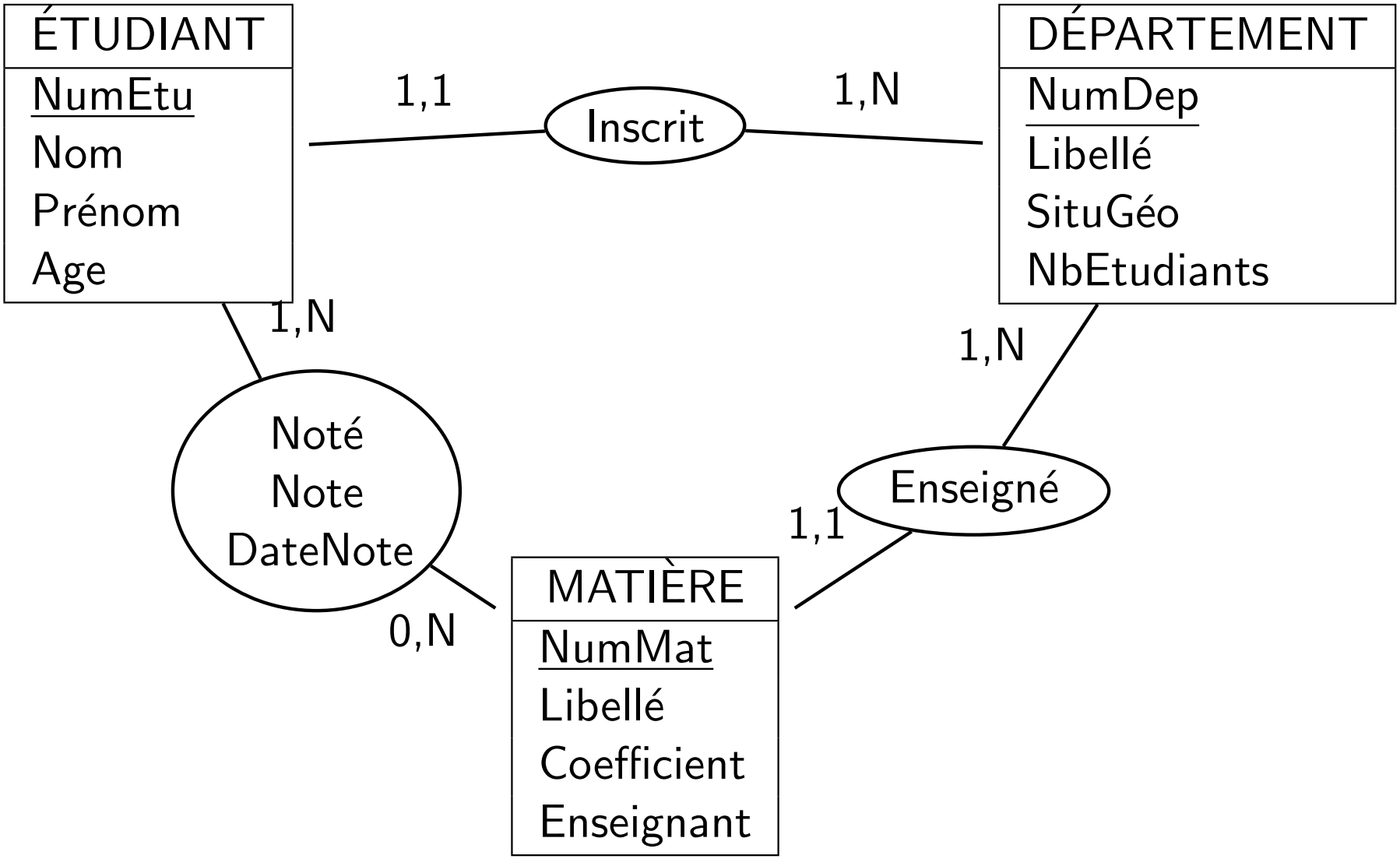
- Un étudiant est décrit par son numéro, son nom, son prénom et son âge
- Un département est décrit par son numéro, son libellé, sa situation géographique et son nombre d'étudiants maximum
- Une matière est décrite par son numéro, son libellé, son coefficient, son enseignant

→ Trois entités : ETUDIANT, DEPARTEMENT, MATIERE

- Un étudiant est inscrit dans un département maximum ; dans un département, il y a plusieurs étudiants inscrits
- Un étudiant est noté dans plusieurs matières ; dans une matière, plusieurs étudiants sont notés
- Une matière est enseignée dans un et un seul département ; dans un département sont enseignés plusieurs matières

→ Trois associations : *Inscrit*, *Noté*, *Enseigné*

Exemple E-A


ÉTUDIANT
NumEtu
Nom
Prénom
Age
1,1
Inscrit
1,N
DÉPARTEMENT
NumDep
Libellé
SituGéo
NbEtudiants
1,N
1,N
Noté
Note
DateNote
0,N
Enseigné
1,1
MATIÈRE
NumMat
Libellé
Coefficient
Enseignant

Plan

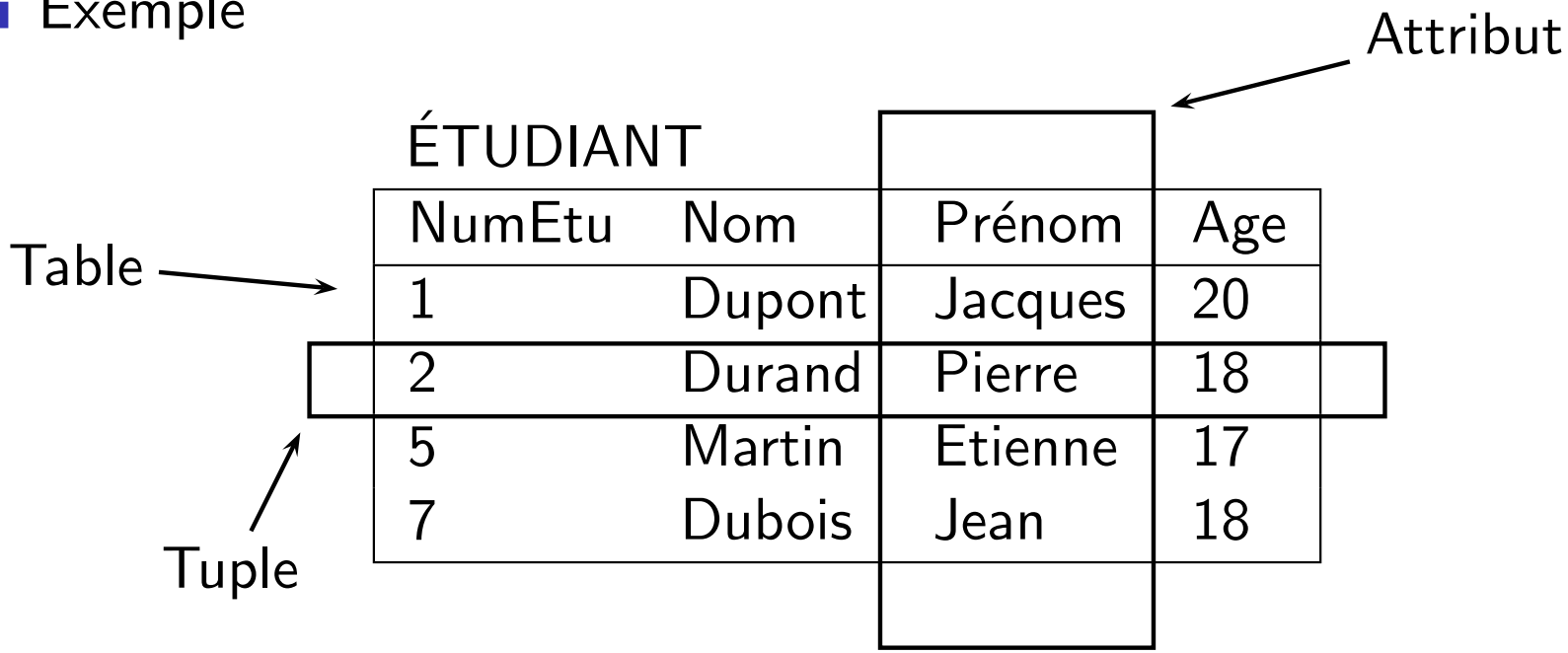
Modèle relationnel

- Formalisé par Edgar Frank Codd en 1969, il est aussi noté aussi MRD
- Basé sur la théorie des relations et l'algèbre relationnelle
- Principe
  - Tables contenant les données sans connaissance de la représentation physique dans la machine
  - Chaque table représentant un ensemble ou **relation**
- Succès grâce à la simplicité des concepts
- Opérations se résumant à transformer une ou deux tables en une nouvelle table

Éléments du modèle relationnel

- Éléments d'une base de données relationnelle
  - Une **Relation** est une table contenant les données
  - Un **Tuple** est une ligne d'une table
  - Un **Attribut** est une colonne d'une table

- Exemple



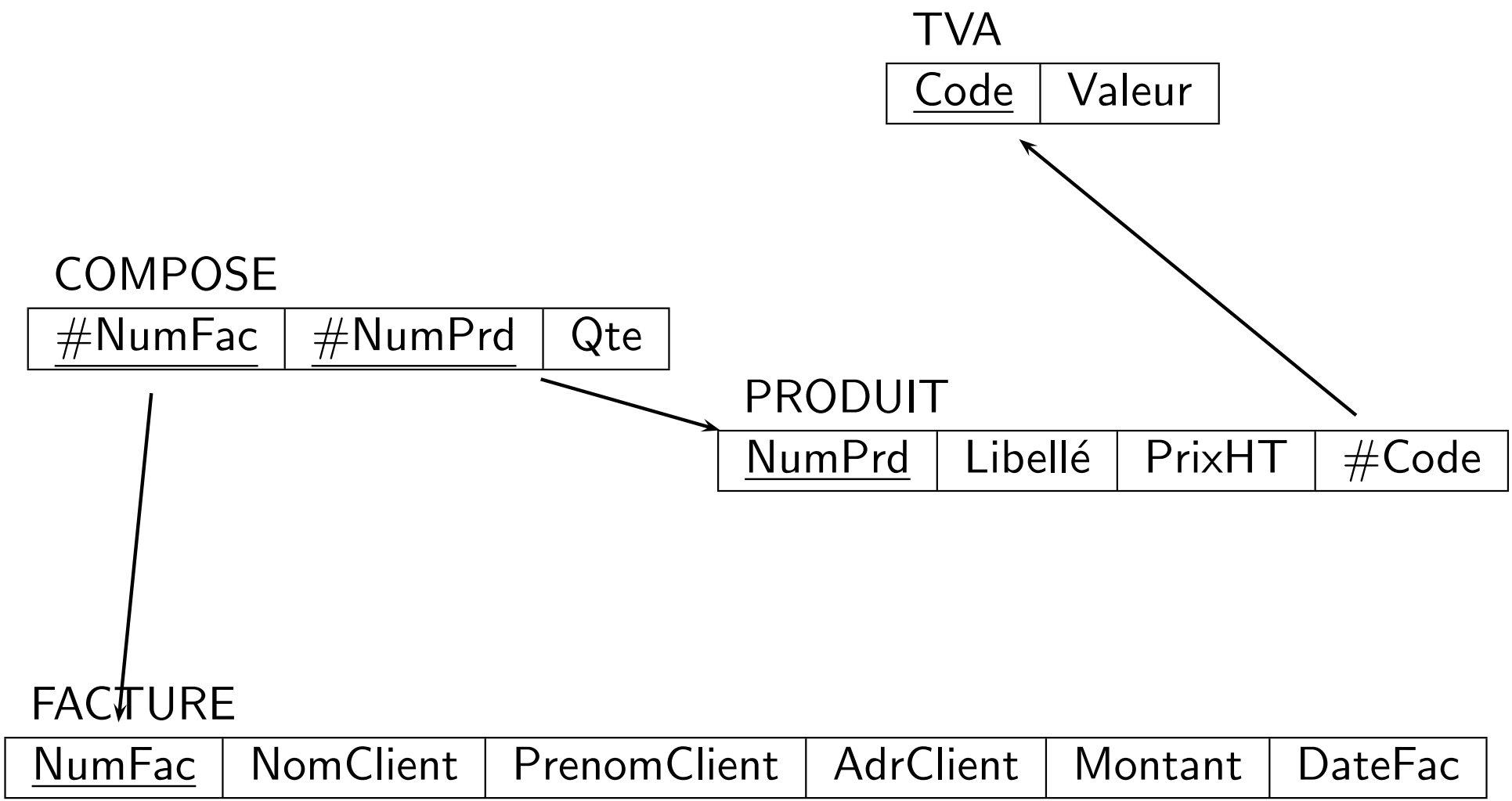
Règles d'intégrité structurelles

- **Unicité de la clé** : attribut(s) permettant d'identifier chaque tuple de la relation de manière unique
- **Contraintes de références** : attribut(s) d'une relation devant apparaître comme clé dans une autre relation
- **Contrainte d'entité** : une clé ne peut pas avoir de valeur nulle (non-présence de l'information)
- **Contrainte de domaine** : permet de définir l'ensemble de valeurs possibles pour chaque attribut

Notation

- Relation : nom de la relation suivi des attributs et de leur domaine entre parenthèses
- Clé primaire : nom de l'attribut (ou des attributs) souligné
- Clé externe : ajout du caractère # devant le nom de l'attribut concerné
- Exemple
  - TVA (Code : *entier*, Valeur : *réel*)
  - PRODUIT(NumPrd : *entier*, Libellé : *chaîne*, PrixHT : *réel*, #Code : *entier*)

MRD


TVA
Code
Valeur
COMPOSE
#NumFac
#NumPrd
Qte
PRODUIT
NumPrd
Libellé
PrixHT
#Code
FACTURE
NumFac
NomClient
PrenomClient
AdrClient
Montant
DateFac

MCD vers MRD

- Passage du modèle conceptuel des données (MCD) au modèle relationnel des données (MRD)

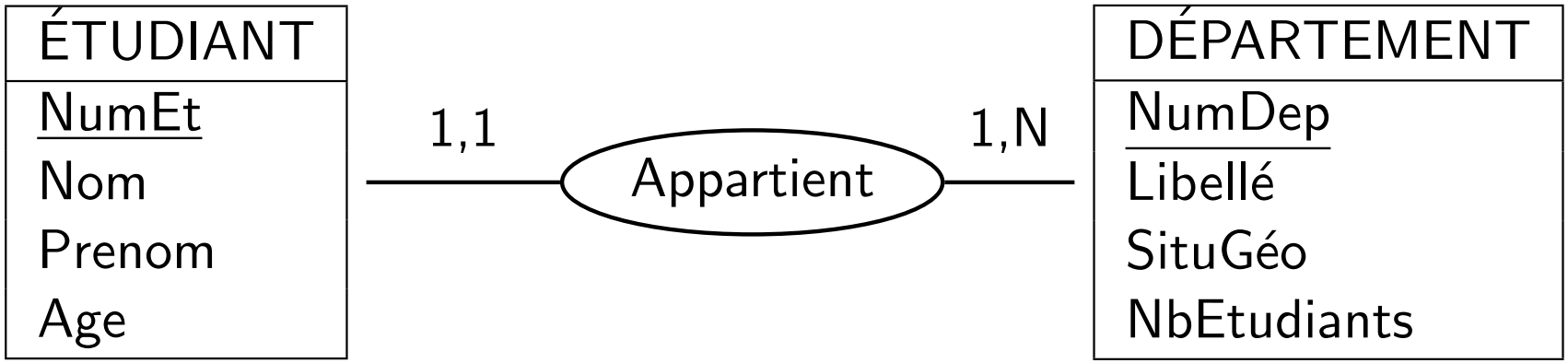
- Entité seule

ÉTUDIANT
NumEt
Nom
Prenom
Age

  - ETUDIANT(NumEt, Nom, Prenom, Age)

MCD vers MRD

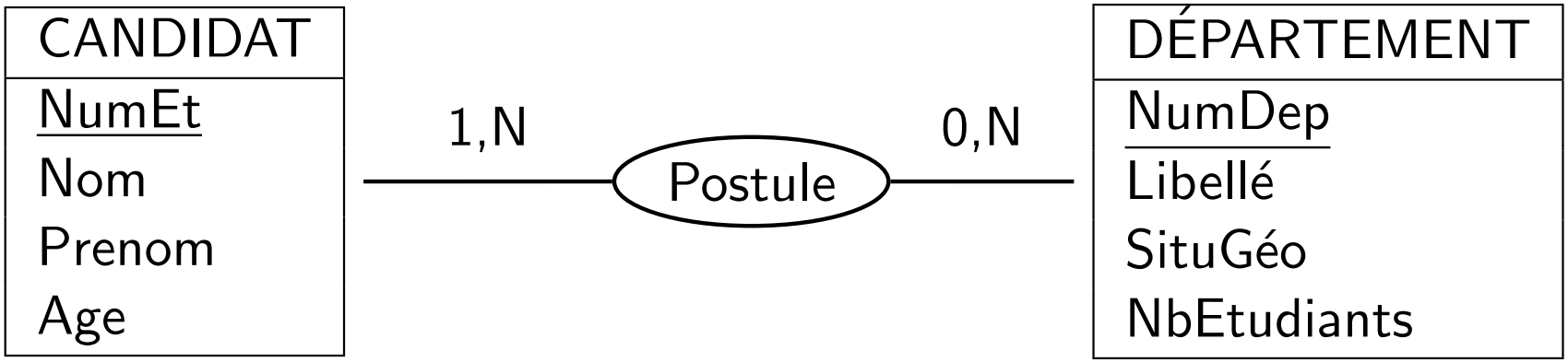
- Association binaire 1,1 - 1,N



- ETUDIANT(NumEt, Nom, Prenom, Age, #NumDep)
- DEPARTEMENT(NumDep, Libellé, SituGéo, NbEtudiants)

MCD vers MRD

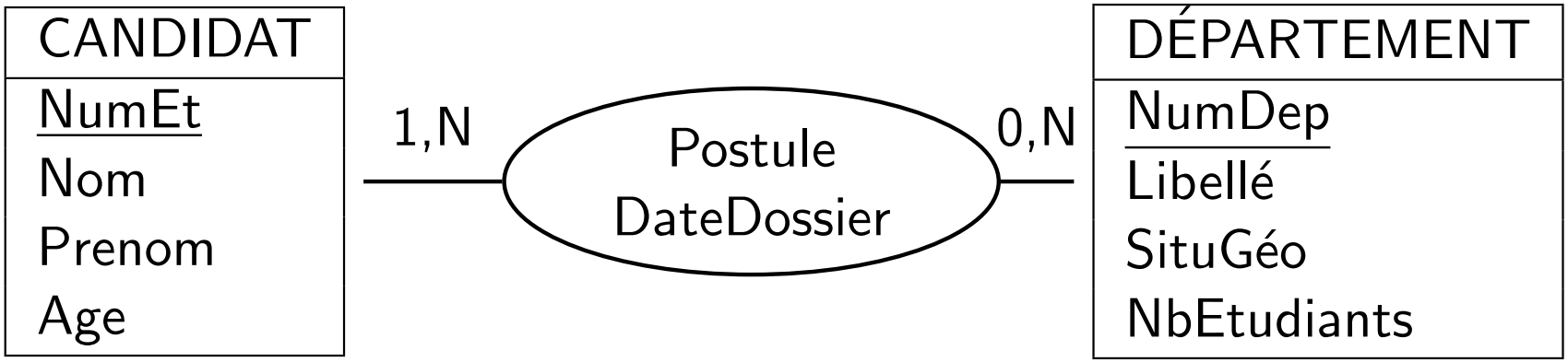
- Association binaire 0/1,N - 0/1,N



- CANDIDAT(NumEt, Nom, Prenom, Age)
- DÉPARTEMENT(NumDep, Libellé, SituGéo, NbEtudiants)
- POSTULE(#NumEt, #NumDep)

MCD vers MRD

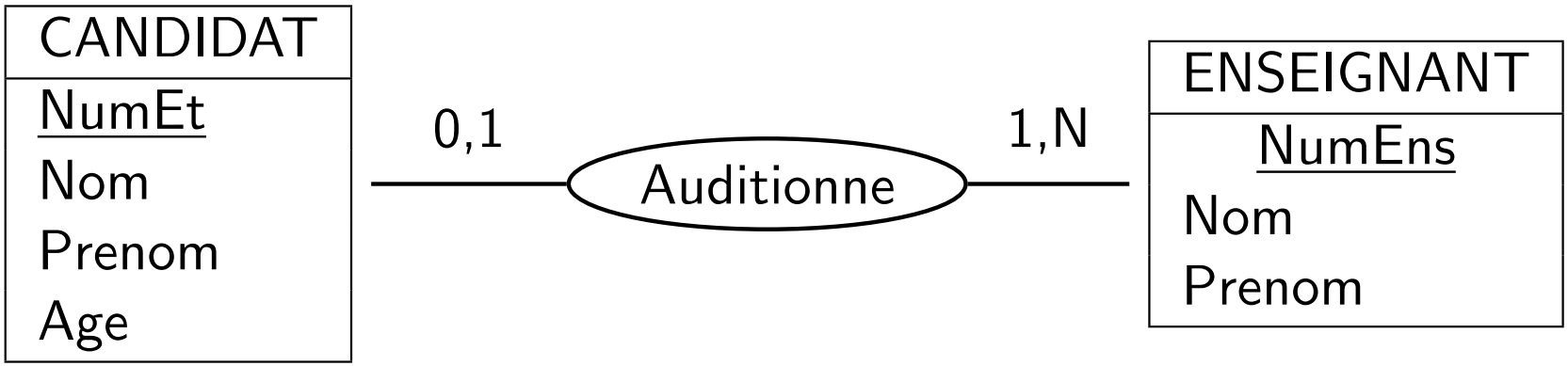
- Association binaire 0/1,N - 0/1,N (avec propriétés)



- CANDIDAT(NumEt, Nom, Prenom, Age)
- DÉPARTEMENT(NumDep, Libellé, SituGéo, NbEtudiants)
- POSTULE(#NumEt, #NumDep, DateDossier)

MCD vers MRD

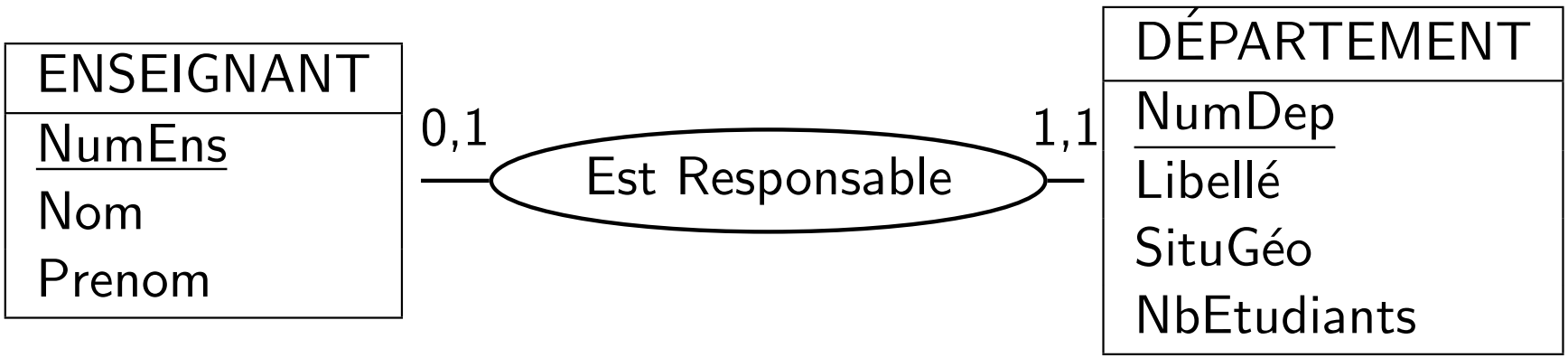
- Association binaire 0,1 - 1,N



- CANDIDAT(NumEt, Nom, Prenom, Age)
- ENSEIGNANT(NumEns, Nom, Prenom)
- AUDITIONNE(#NumEt, #NumEns)

MCD vers MRD

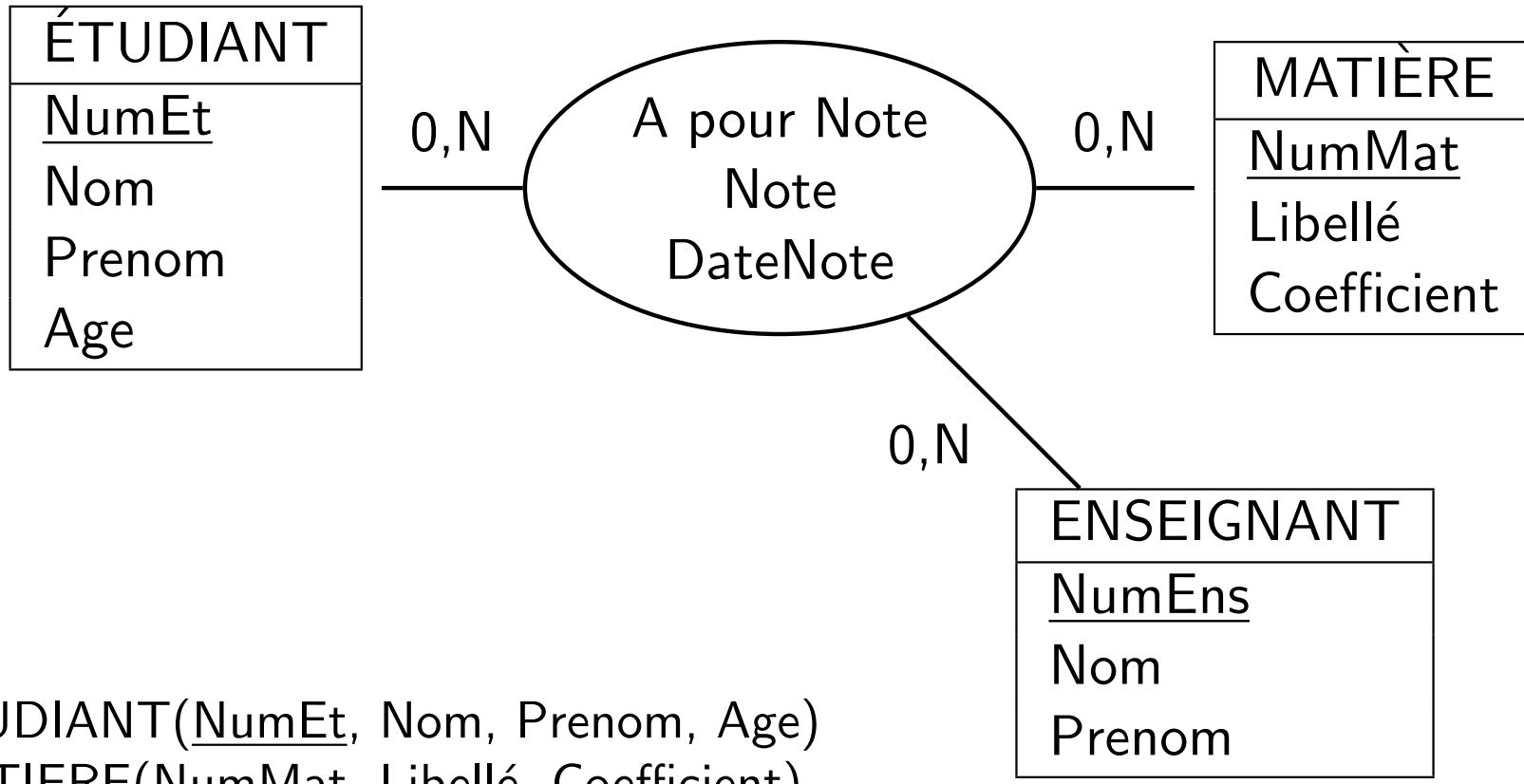
- Association binaire 1,1 - 0,1



- ENSEIGNANT(NumEns, Nom, Prenom)
- DÉPARTEMENT(NumDep, Libellé, SituGéo, NbEtudiants, #NumEns)

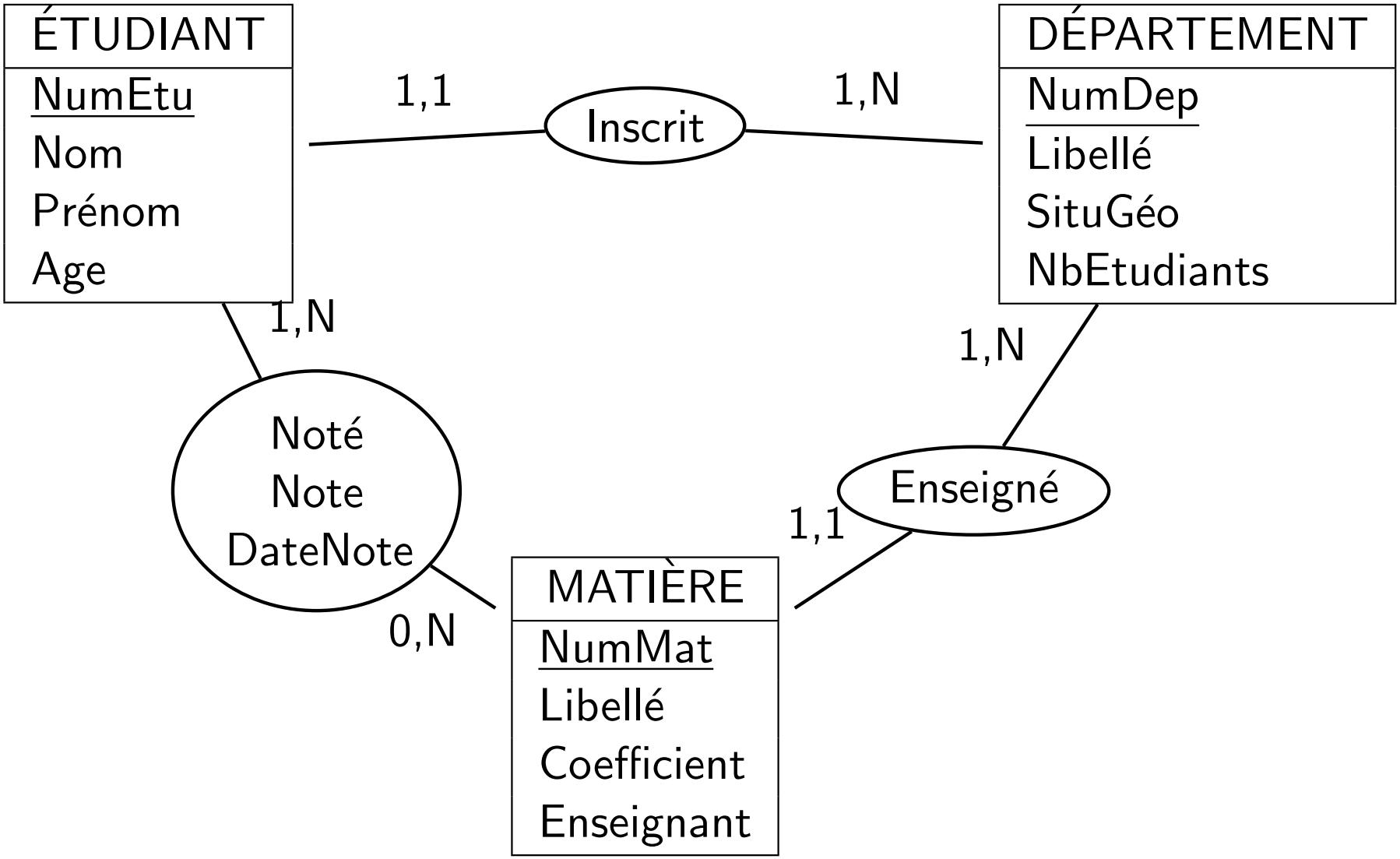
MCD vers MRD

- Association ternaire 0,N - 0,N - 0,N (avec propriétés)



- ETUDIANT(NumEt, Nom, Prenom, Age)
- MATIERE(NumMat, Libellé, Coefficient)
- ENSEIGNANT(NumEns, Nom, Prenom)
- NOTE(#NumEt, #NumMat, #NumEns, Note, DateNote)

Exemple MRD


ÉTUDIANT
NumEtu
Nom
Prénom
Age
1,1
Inscrit
1,N
DÉPARTEMENT
NumDep
Libellé
SituGéo
NbEtudiants
1,N
1,N
Noté
Note
DateNote
Enseigné
1,1
0,N
MATIÈRE
NumMat
Libellé
Coefficient
Enseignant

Exemple MRD

- Trois entités de base : ÉTUDIANT, DÉPARTEMENT, MATIÈRE
- Association Inscrit à intégrer à ÉTUDIANT
- Association Enseigné à intégrer à MATIÈRE

- DÉPARTEMENT(NumDep, Libellé, SituGéo, NbEtudiants)
- ETUDIANT(NumEt, Nom, Prenom, Age, #NumDep)
- MATIERE(NumMat, Libellé, Coefficient, Enseignant, #NumDep)
- NOTÉ(#NumEt, #NumMat, Note, DateNote)

Algèbre relationnelle : Opérateurs

- **Union** : R = R1 ∪ R2
- Ensemble des tuples distincts des deux relations R1 et R2
- R1 et R2 obligatoirement de même structure
- Exemple

R1	A	B
	1	1
	1	2

R2	A	B
	1	2
	2	5
	3	4

R = R1 ∪ R2	A	B
	1	1
	1	2
	2	5
	3	4

Algèbre relationnelle : Opèrateurs

- **Intersection** : $R = R1 \cap R2$
- Ensemble des tuples présents conjoitement dans les deux relations R1 et R2
- R1 et R2 obligatoirement de même structure
- Exemple

R1	A	B
	1	1
	1	2

R2	A	B
	1	2
	2	5
	3	4

$R = R1 \cap R2$	A	B
	1	2

Algèbre relationnelle : Opérateurs

- **Différence** : R = R1 − R2
- Ensemble des tuples de la relation R1 qui ne sont pas présent dans la relation R2 (opération non commutative)
- R1 et R2 obligatoirement de même structure
- Exemple

R1	A	B
	1	1
	1	2

R2	A	B
	1	2
	2	5
	3	4

R = R1 − R2	A	B
	1	1

R = R2 − R1	A	B
	2	5
	3	4

Algèbre relationnelle : Opérateurs

- **Restriction** : $R = \sigma_{condition}(R1)$
- Ensemble des tuples de la relation R1 qui vérifient la condition donnée
- Exemple

R1	A	B
	1	1
	1	2
	2	5
	3	4

$R = \sigma_{A \geq 2}(R1)$	A	B
	2	5
	3	4

Algèbre relationnelle : Opérateurs

- **Projection** : $R = \pi_{attribut(s)}\ (R)$
- Ensemble des n-uplets distincts composés par les attributs fournis
- Exemple

R1	A	B
	1	1
	1	2
	2	5
	3	4

$R = \pi_A(R1)$	A
	1
	2
	3

Algèbre relationnelle : Opérateurs

- **Produit cartésien** : $R = R1 \times R2$

- Relation ayant comme schéma tous les attributs des deux relations R1 et R2, et comme tuples l'association de chaque ligne de R1 avec chaque ligne de R2

- Exemple

R1	A	B
	1	1
	1	2

R2	C	D
	1	2
	2	5
	3	4

$R = R1 \times R2$	A	B	C	D
	1	1	1	2
	1	2	1	2
	1	1	2	5
	1	2	2	5
	1	1	3	4
	1	2	3	4

Algèbre relationnelle : Opérateurs

- **Jointures** : R1 $\bowtie_{condition}$ R2
- Restriction selon la condition fournie sur le produit cartésien de R1 et R2
- Exemple

R1	A	B
	1	1
	1	2

R2	C	D
	1	2
	2	5
	3	4

$R = R1 \bowtie_{A=C} R2$

R	A	B	C	D
	1	1	1	2
	1	2	1	2

Algèbre relationnelle : Opérateurs

- **Theta-jointure** : La condition est une comparaison de deux attributs

  $R1 \bowtie_{A<C} R2$

- **Equi-jointure** : La condition porte sur l'égalité entre deux attributs

  $R1 \bowtie_{A=C} R2$

- **Jointure naturelle** : la plus utilisée : C'est une équi-jointure entre les attributs portant le m ême nom

  $R1 \bowtie R3$

Algèbre relationnelle : Opérateurs

- **Division** : R1 / R2

- Relation ayant le schéma de R1 sans les attributs de R2 dont les tuples sont toutes les occurences de R1 associées avec toutes les occurences de R2. Il faut que le schéma de R2 soit strictement inclu dans celui de R1. Cet opérateur permets de répondre aux questions du type “les . . . tels que tous les . . . ”.

- Exemple : quels sont les voyages faits par tous les clients

R1	Destination	Client
	Tunisie	c1
	USA	c2
	Tunisie	c2
	Espagne	c1
	Espagne	c2
	Grâce	c1

R2	Client
	c1
	c2

R = R1 / R2

R	Destination
	Tunisie
	Espagne

Algèbre relationnelle : Fonctions

- **Calcul élémentaire** :

  $R = \pi_{\text{attribut(s)},\text{variable}=\text{expression},\ldots}\ R1$

- Projection sur une relation associée à un calcul portant sur chaque tuple afin de créer un ou plusieurs nouveaux attributs. L'expression peut être
  - une opération arithmétique,
  - une fonction mathématique,
  - une fonction portant sur une chaîne de caractères.

- Exemples
  - $R = \pi_{A,\ B,\ Z\ =\ B\times 1.5}(R1)$
  - $R = \pi_{C,\ COSI\ =\ COS(D),\ SINU\ =\ SIN(D)}(R2)$

Algèbre relationnelle : Fonctions

- **Calcul d'agrégats** : $R = \pi_{\text{attribut(s), N1 = fonction,...}}\ R1$

- Projection sur une relation associée à un ou plusieurs calculs statistiques portant sur un attribut, pour tous les éléments de la relation ou du regroupement lié à la projection. Les fonctions sont

COUNT(*)	nombre de lignes
COUNT(attribut)	nombre de valeurs non nulles
SUM(attribut)	somme des valeurs non nulles
AVG(attribut)	moyenne des valeurs non nulles
MIN(attribut)	valeur maximum (non nulle)
MAX(attribut)	valeur minimum (non nulle)

- Exemple
  - $R = \pi_{\text{Sexe, AgeMoyen = AVG(Age)}}(R1)$

Exemple

- Fonctionnement d'un département de l'IUT (*STID* par exemple)
- Modélisation des étudiants, des matières et des notes obtenues par les étudiants dans chaque matière
- Modèle Relationnel des Données (ou MRD) obtenu :
  - ETUDIANT(NumEt : *entier*, Nom : *chaîne*, Prenom : *chaîne*, Adresse : *chaîne*, Age : *réel*, Sexe : *caractère*)
  - MATIERE(NumMat : *entier*, Libellé : *chaîne*, Enseignant : *chaîne*, Coefficient : *entier*)
  - NOTES(#NumEt : *entier*, #NumMat : *entier*, Note : *réel*)

Exemple

Quelques exemples d'utilisation du langage algébrique

- Liste des étudiants majeurs
  - R1 = $\sigma_{\text{Age} \geq 18}(\text{ETUDIANT})$
- Noms des étudiants
  - R2 = $\pi_{\text{Nom}}(\text{ETUDIANT})$
- Enseignant de la matière *Base de Données*
  - R1 = $\sigma_{\text{Libelle='Base de Donnees'}}(\text{MATIERE})$
  - R2 = $\pi_{\text{Enseignant}}(\text{R1})$
- Moyenne pondérée pour chaque étudiant (nom et prénom)
  - R1 = ETUDIANT $\bowtie$ NOTES
  - R2 = R1 $\bowtie$ MATIERE
  - R3 = $\pi_{\text{Nom, Prenom, SUM(Note * Coefficient) / SUM(Coefficient)}}(\text{R2})$

Plan

Introduction à SQL

- SQL : *Structured Query Language*
- Besoin d'un langage standard pour la gestion et la manipulation des Bases de Données
- Articulation en plusieurs grandes parties :
  - **DDL** : *Data Definition Language*
  - **DML** : *Data Manipulation Language*
  - **TCL** : *Transaction Control Language*
  - ...
- Notation utilisée

SELECT	les mots clés SQL sont en majuscule et en gras
[]	indique un paramètre optionnel
{}	indique les choix possibles
...	indique un paramètre répétitif possible

DDL : Langage de Définition des Données

- Définition des tables
- Expression des contraintes d'intégrité
- Définition de vues
- Suppression des tables

Création de tables

Syntaxe :

```
CREATE TABLE tab (
  att type [DEFAULT exp] [[CONSTRAINT nom] ctr_att]...
  ...,
  [[CONSTRAINT nom] ctr_rel] ...
);
```

ou

```
CREATE TABLE tab (att, ...) AS
  requ\^ete;
```

Création de tables

- Quelques types

Caractère	**VARCHAR**(n), **CHAR**(n)
Numérique	**NUMBER**, **INT**
Temporel	**DATE**, **TIMESTAMP**

- Contraintes d'attributs

NULL/NOT NULL	Valeur NULL possible ou non
PRIMARY KEY	Définition de clé primaire unique
UNIQUE	Définition de clé secondaire
REFERENCES tab [(att)]	Référencement à une table (et un attribut particulier si le nom est différent)
CHECK(condition)	Contraintes sur l'attribut

Création de tables

- Contraintes de relations

PRIMARY KEY(att, ...)	Désigne le ou les attributs comme clé primaire
UNIQUE(att, ...)	Désigne le ou les attributs comme clé secondaire de la table
FOREIGN KEY (att, ...)	Indique une référence à une table
REFERENCES tab [(liste)]	(et un ou plusieurs attributs particuliers si leur nom différent)
CHECK(condition)	Vérifie la condition lors de l'insertion d'informations (celle-ci pouvant porter sur plusieurs attributs)

Création de tables : exemples

```
CREATE TABLE Etudiant (
  NumEt     INT NOT NULL PRIMARY KEY,
  Nom       VARCHAR(30),
  Prenom    VARCHAR(30),
  Sexe      CHAR(1) CHECK(Sexe IN ('H','F')),
  DateNais DATE
);

CREATE TABLE Notation (
  NumNot  INT NOT NULL
    CONSTRAINT pk_Notation PRIMARY KEY,
  NumEt   INT NOT NULL REFERENCES Etudiant,
  Matiere VARCHAR(50),
  Note    INT
);
```

Création de tables

- **`ON DELETE CASCADE`** : Demande la suppression des lignes de la table pour lesquelle la ligne correspondante dans la table référencée est supprimée. Si ce n'est pas précisé, la suppression sera impossible dans la table référencée.

- **`ON DELETE SET NULL`** : Demande la mise à NULL des attributs constituant la clé externe pour les lignes de la table pour lesquelle la ligne correspondante est supprimée, dans la table référencée. Si ce n'est pas précisé, la suppression sera impossible dans la table référencée.

Création de tables : exemples

```
CREATE TABLE Etudiant (
  NumEt     INT NOT NULL PRIMARY KEY,
  Nom       VARCHAR(30),
  Prenom    VARCHAR(30),

  Sexe      CHAR(1) CHECK(Sexe IN ('H','F')),

  DateNais DATE
);


CREATE TABLE Notation (
  NumNot  INT NOT NULL
    CONSTRAINT pk_Notation PRIMARY KEY,
  NumEt   INT NOT NULL REFERENCES Etudiant ON DELETE C
  Matiere VARCHAR(50),
```

Suppression de tables

- Syntaxe :

```
DROP TABLE tab [CASCADE CONSTRAINTS];
```

- La clause **CASCADE CONSTRAINTS** permet de supprimer les contraintes d'intégrités référentielles qui pointent sur la table supprimée

- Exemple

```
-- Cette commande produit une erreur
DROP TABLE Etudiant;

-- Voila comment faire
DROP TABLE Etudiant CASCADE CONSTRAINTS;
```

Modification d'une table

- Ajout ou modification d'attributs

```
ALTER TABLE tab [ADD/MODIFY] (
  att type [contraintes],
  ...
);
```

- Exemple

```
ALTER TABLE Etudiant ADD (
  Adresse VARCHAR(100),
  CP      INT,
  Ville   VARCHAR(50)
);

ALTER TABLE Notation MODIFY (Note NUMBER);
```

Modification d'une table

- Ajout d'une contrainte de relation

```
ALTER TABLE tab
  ADD [CONSTRAINT nom] contrainte;
```

- Exemple

```
ALTER TABLE Notation ADD CHECK (Note <= 20);

ALTER TABLE Notation ADD
  CONSTRAINT ck_Notation_Note_Sup0
  CHECK (Note >= 0);
```

Modification d'une table

- Suppression d'une contrainte de relation

```
ALTER TABLE tab
  {DROP PRIMARY KEY|
   DROP UNIQUE(att)|
   DROP CONSTRAINT nom};
```

- Exemple

```
ALTER TABLE Notation
  DROP CONSTRAINT ck_Notation_Note_Sup0;
```

Modification d'une table

- Suppression d'un attribut

```
ALTER TABLE tab
  DROP COLUMN att;

ALTER TABLE nom
  DROP (att, ...);
```

- Exemple

```
ALTER TABLE Etudiant
  DROP COLUMN Sexe;
```

Gestion des index

- Les index améliorent les performances des requêtes, on les créé généralement sur les clés externes et sur les critères de recherches courants.

- Syntaxe

```
CREATE INDEX nom
  ON tab (att [DESC], ...);
DROP INDEX nom;
```

- Exemple

```
CREATE INDEX ik_Notation_Et
  ON Notation (NumEt);
DROP INDEX ik_Notation_Et;
```

TCL : Langage de Contrôles des Transactions

- Transaction : ensemble d'opérations indivisibles
- Validation et annulation de modifications
- Définition de transactions
- Syntaxe

COMMIT	Validation de la transaction
ROLLBACK [TO pointsvg]	Annulation des transactions
SAVEPOINT pointsvg	Déclaration d'un point de sauvegarde

Contrôles des transactions

- Exemple (les `...` représentes des opérations)

```
...
SAVEPOINT S1;
...
COMMIT;
...
ROLLBACK;
...
ROLLBACK TO S1;
...
```

DML : Langage de Manipulation des Données

- Manipulation de base
  - Insertion
  - Modification
  - Suppression
  - Mise à jour
- Visualisation du contenu total ou partiel des informations

Les instructions

- Expression

constantes caractères	`'IUT', 'l''image'`
constantes de dates	`'19-jan-06', '19/01/06'`
constantes numériques	`5, -248.9`
noms d'attributs	`Etudiant.Nom, Notation.Note`
fonctions	`SQRT(12), SYSDATE,`
	`REPLACE('Bonjour','jour','soir')`

- Opérateurs

arithmétiques	`+ - / * ( )`
	`1.055 * PrixHT, SYSDATE + 23`
concaténation	`||`
	`'Bonjour␣'|| Prenom`

Les instructions

- Opérateurs de comparaison

comparaison simple	`exp1{=, <>, <, <=, >, >=}exp2`
appartenance	`exp1 IN (exp2, ...)`
	`exp1 BETWEEN exp2 AND exp3`
comparaison à un format	`exp1 LIKE 'format'`
	% (suite de 0 à caractères)
	_ (un et un seul caractère)
négation	`NOT exp`
combinaison	`exp1 \{AND OR\} exp2`

Les instructions

- Fonctions scalaires mathématiques

`ABS(n)`	valeur absolue
`SIGN(n)`	-1, 0 ou +1 selon que n est négatif, nul ou positif
`FLOOR(n), CEIL(n)`	premier entier inférieur ou supérieur ou égal à n
`ROUND(n,m)`	arrondi de n, avec m positions décimales
`MOD(n,m)`	reste de la division entière de n par m
`POWER(n,m)`	n exposant m
`LN(n), LOG(m,n)`	logarithme de n (népérien ou de base m)
`EXP(n)`	exponentielle de n
`SQRT(n)`	racine carré de n
`COS(n), COSH(n)`	cosinus (classique et hyperbolique) de n
`SIN(n), SINH(n)`	sinus (classique et hyperbolique) de n
`TAN(n), TANH(n)`	tangente (classique et hyperbolique) de n

Les instructions

- Fonctions scalaires chaînes de caractères

`CONCAT(c,d)`	concaténation de c et d
`LENGTH(c)`	longueur de c
`REPLACE(c,d,e)`	remplace dans c la chaîne d par e (e peut être nul)
`TRANSLATE``(c,d,e)`	idem mais e ne peut être nul
`SUBSTR``(c,n[,m])`	renvoie m caractères de c à partir de la position n
`INITCAP(c)`	renvoie c avec une majuscule à chaque mot
`INSTR(c,d[,n[,m]])`	position de la même occurence (par défaut, $m = 1$) de d dans c à partir de la position n (par défaut $n = 1$)
`LOWER``(c)`	renvoie c en minuscule
`UPPER``(c)`	renvoie c en majuscule

...

Les instructions

- Fonctions scalaires dates

`ADD_MONTHS(d,n)`	date d plus n mois
`CURRENT_DATE`	date et heure actuelle
`SYSDATE`	date système
`EXTRACT``(format` **`FROM`** `d)`	extrait un élément (jour, mois, ...) depuis d
`LAST_DAY(d)`	date du dernier jour du mois de d
`MONTHS_BETWEEN(d,e)`	différence (en mois) entre d et e
`NEXT_DAY(d,j)`	date du prochain jour j à partir de d

...

Les instructions

- Format de dates

`- / . ; : 'texte'`	éléments de séparation dans le résultat
`HH, HH12, HH24`	heure du jour (de 0 à 23 seulement HH24)
`MI`	minutes
`SS`	secondes
`SSSS`	nombre de secondes depuis minuit
`DAY`	nom du jour de la semaine sur 9 caractères
`DY`	nom abrégé du jour de la semaine
`DD`	numéro du jour dans le mois (1 à 31)
`DDD`	numéro du jour dans l'année (1 à 366)

Les instructions

- Format de dates (suite)

`IW`	numéro de la semaine dans l'année (selon la norme ISO)
`WW`	idem (la semaine 1 démarre le 01/01 et dure 7 jours)
`W`	numéro de le semaine dans le mois
`MM`	numéro du mois sur 2 caractères
`RM`	numéro du mois en chiffre romain
`MONTH`	nom du mois sur 9 caractères
`MON`	nom abrégé du mois
`Q`	numéro du trimestre
`Y, YY, YYY, YYYY`	année sur 1, 2, 3, 4 caractères

Les instructions

- Format de conversion

`TO_CHAR(n[,format])`	conversion d'un nombre n en chaîne de caractères selon un certain format
`TO_NUMBER(c)`	conversion d'une chaîne de caractères c en nombre (si cela possible)
`TO_CHAR(d,format)`	conversion d'une date d en chaîne de caractères selon un certain format
`TO_DATE(c,format)`	conversion d'une chaîne de caractères c en date selon un certain format

...

Insertion de valeurs

- L'insertion est effective si les contraintes de la table sont respectées.

- Syntaxe

```
INSERT INTO relation [(attribut, ...)]
  VALUES (exp, ...);
```

- Exemple

```
INSERT INTO Etudiant (NumEt, Nom, Prenom)
  VALUES (1,'Einstein', 'Albert');
INSERT INTO Notation VALUES (18,1,'Maths',01);
```

Suppression de valeurs

- La suppression est effective si les lignes supprimées ne sont pas ou plus référencées.

- Syntaxe

```
DELETE FROM relation
  [WHERE condition];
```

- Exemple

```
DELETE FROM Notation;
DELETE FROM Etudiant
  WHERE NumEt = 1;
DELETE FROM Etudiant
  WHERE Sexe = 'F';
```

Modification de valeurs

- La mise à jour de valeurs peut concerner toutes les lignes ou seulement celles répondant à un critère plus ou moins complexe

- Syntaxe

```
UPDATE relation
  SET attribut = exp, ...
  [WHERE condition];
```

- Exemple

```
UPDATE Notation
  SET  Note = Note + 2;
UPDATE  Etudiant
  SET   Sexe = 'H', Age = 18
  WHERE NumEt = 1;
```

Extraction de valeurs

- La requête **SELECT** est la plus importante et la plus utilisée en SQL

- Syntaxe

```
SELECT exp, ...
  FROM relation1, relation2, ...
  [WHERE condition];
```

- Exemple

```
SELECT * FROM Etudiant;
SELECT  NumEt, Matiere
  FROM  Notation
  WHERE Note = 20;
```

Traduction de l'algèbre relationnelle

- Nous avons abordé la théorie des bases de données relationnelles avec le modèle relationnel et l'algèbre adaptée. Nous allons maintenant traduire les opérations de cette algèbre en SQL.

- Bien que beaucoup de personnes résolvent les questions directement en SQL, il est parfois utile (et nécessaire) de passer par l'algèbre afin de pouvoir établir le schéma de raisonnement que l'on doit faire pour obtenir la réponse.

- On peux assimiler l'algèbre relationnelle à l'algorithmique des bases de données, SQL étant le langage de programmation à utiliser par la suite.

Restriction

- Syntaxe

```
SELECT *
  FROM  relation
  WHERE condition;
```

- Exemple

```
SELECT *
  FROM  Notation
  WHERE Note >=19;
```

Projection

- Syntaxe

```
SELECT [DISTINCT / ALL ] attribut
  FROM relation;
```

- Exemple

```
SELECT DISTINCT Note
  FROM Notation;
SELECT Nom, Prenom
  FROM Etudiant;
```

Calcul élémentaire

- Syntaxe

```
SELECT ..., expression
  FROM relation;
```

- Exemple

```
SELECT NumEt, Matiere, Note * 1.5
  FROM Notation;
```

Calcul d'agrégat

- La liste d'attribut de la clause **SELECT** et la liste d'attribut de la clause **GROUP BY** doivent être identique

- Syntaxe

```
SELECT liste, fonction
  FROM relation
  GROUP BY liste;
```

- Exemple

```
SELECT     Matiere, AVG(Note)
  FROM     Notation
  GROUP BY Matiere;
```

Calcul d'agrégat

- Fonctions de groupe

`AVG(att)`	Moyenne des valeurs de l'attribut
`STDDEV(att)`	Écart-type des valeurs de l'attribut
`VARIANCE(att)`	Variance des valeurs de l'attribut
`SUM(att)`	Somme des valeurs de l'attribut
`MIN(att), MAX(att)`	Minimum et Maximum des valeurs de l'attribut
`CORR(att1,att2)`	Coefficient de corrélation entre `att1` et `att2`
`COUNT(*)`	Nombre de lignes
`COUNT(att)`	Nombre de valeurs non NULL de `att` (donc nombre de valeurs présentes)

...

Restriction d'agrégat

- La clause **HAVING** ne doit concerner que des conditions sur un calcul d'agrégat

- Syntaxe

```
SELECT liste, fonction
  FROM relation
  GROUP BY liste
  HAVING condition;
```

- Exemple

```
SELECT     Matiere, AVG(Note)
  FROM     Notation
  GROUP BY Matiere
  HAVING   AVG(Note) <= 10;
```

Produit cartésien

- Syntaxe

```
SELECT ...
  FROM relation1, relation2;
```

- Exemple

```
SELECT Etudiant.NumEt, Notation.NumEt
  FROM Etudiant, Notation;
SELECT Nom, Matiere
  FROM Etudiant, Notation;
```

Jointures

- Syntaxe

```
SELECT ...
  FROM relation1, relation2
  WHERE condition;
```

- Exemple

```
SELECT  Nom, Matiere
  FROM  Etudiant, Notation
  WHERE Etudiant.NumEt = Notation.NumEt;
```

Union, Intersection et Différence

- Les deux requêtes doivent retourner exactement les mêmes attributs

- Syntaxe

```
requ\^ete
UNION [ALL] / INTERSECT / MINUS (ou EXCEPT)
requ\^ete;
```

- Exemple

```
SELECT  Nom, Prenom
  FROM  Etudiant
  WHERE Sexe = 'F'
MINUS
SELECT  Nom, Prenom
  FROM  Etudiant
  WHERE EXTRACT(YEAR FROM DateNais) <= 1986;
```

Division

- La division peut se faire de plusieurs manières. La méthode classique utilisent deux négations.
- Exemple : Passe (#NumCine, #NumFilm) / Film (NumFilm,. . .) (cinémas passant tous les films)
- = les cinémas pour lesquels il n'existe PAS de films qu'ils ne diffusent PAS

```
SELECT NumCine FROM Passe AS P1
  WHERE NOT EXISTS
    (SELECT * FROM Film AS F
    WHERE NOT EXISTS
      (SELECT * FROM Passe AS P2
      WHERE P1.NumCine = P2.NumCine
      AND P1.NumFilm = F.NumFilm )))
```

Tri du résultat

- Syntaxe

```
SELECT ...
  FROM ...
  ORDER BY attribut [DESC], ...;
```

- Exemple

```
SELECT      Nom, Prenom
  FROM      Etudiant
  ORDER BY Nom, Prenom;
SELECT      Matiere, AVG(Note)
  FROM      Notation
  GROUP BY Matiere
  ORDER BY 2 DESC;
```

Alias

- Nom alternatif donné à une colonne ou une table temporairement dans une requête.

- Syntaxe

```
SELECT attribut [AS] alias
  FROM relation alias ...;
```

- Exemple

```
SELECT      Nom, Prenom, AVG(Note) AS Moyenne
  FROM      Notation N, Etudiant E
  WHERE     N.NumEt = E.NumEt
  GROUP BY Nom, Prenom
  HAVING    Moyenne < 10;
```