

The slide contains a pink title "Bases de données". To its right is a purple pen nib drawing a wavy line. Below the title is a bulleted list of topics:

- Définitions
- Glossaire des SGBD relationnels
- Démarche
- Structures physiques

In the bottom left corner, there are illustrations of colored pencils (yellow, green, red) and a small blue pen nib. The number "2" is located in the bottom right corner.

## BDD : définitions

**BDD = Bases Des Données :**

- **Fichier**
- **Attribut**
- **Identifiant**
- **Bases de données**
- **SGBD**

3

## BDD : glossaire SGBDR

**SGBDR = Système de Gestion des  
Bases de Données Relationnel**

- **Table ou relation**
- **Clé primaire**
- **Clé externe(étrangère)**
- **Champ ou attribut**
- **Domaine**
- **Tuple**

4

## BDD : démarche

- Conception : modélisation, normalisation
- Développement : création, optimisation
- Utilisation
- Maintenance



5



## BDD : Conception

- Méthode Merise : MCD, MLD, MPD
  - MCD = Modèle Conceptuel des Données
  - MLD = Modèle Logique des Données
  - MPD = Modèle Physique des Données

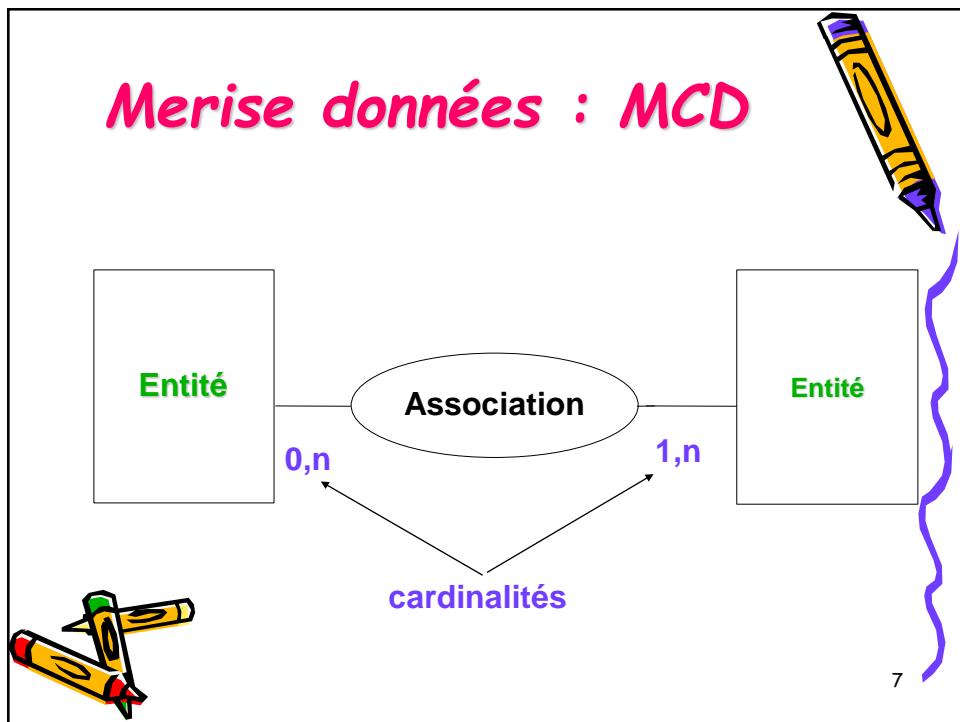
- Normalisation : formes normales



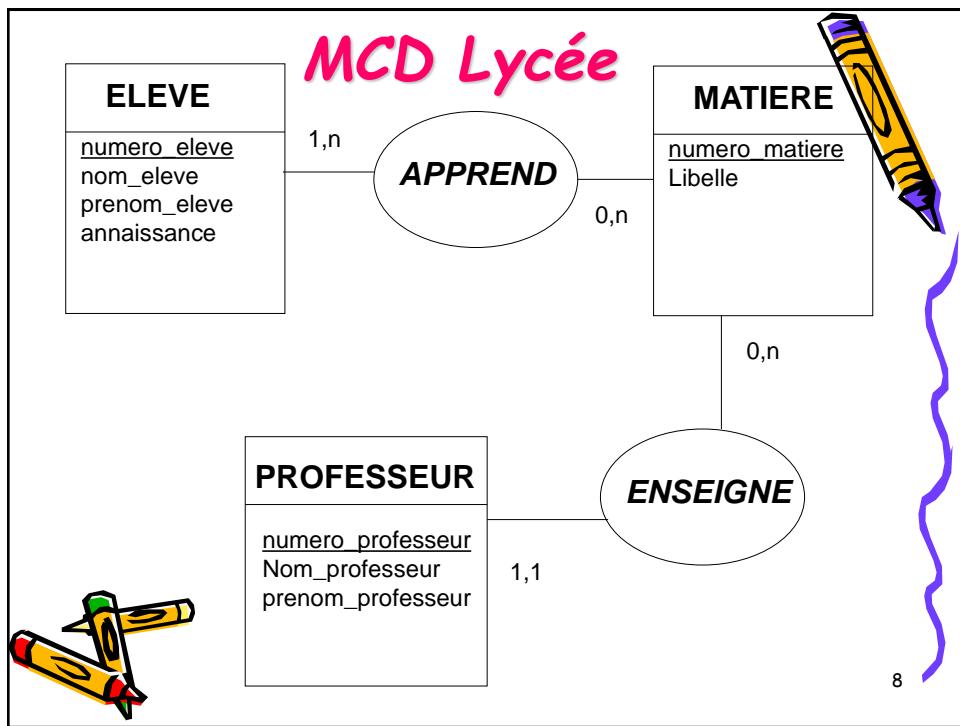
6

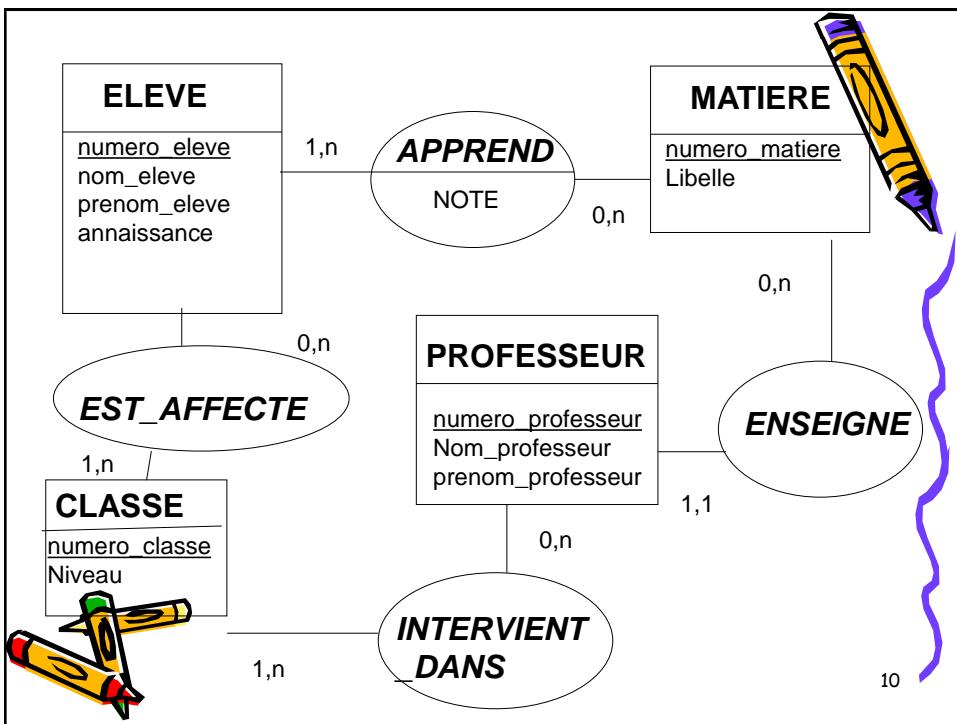
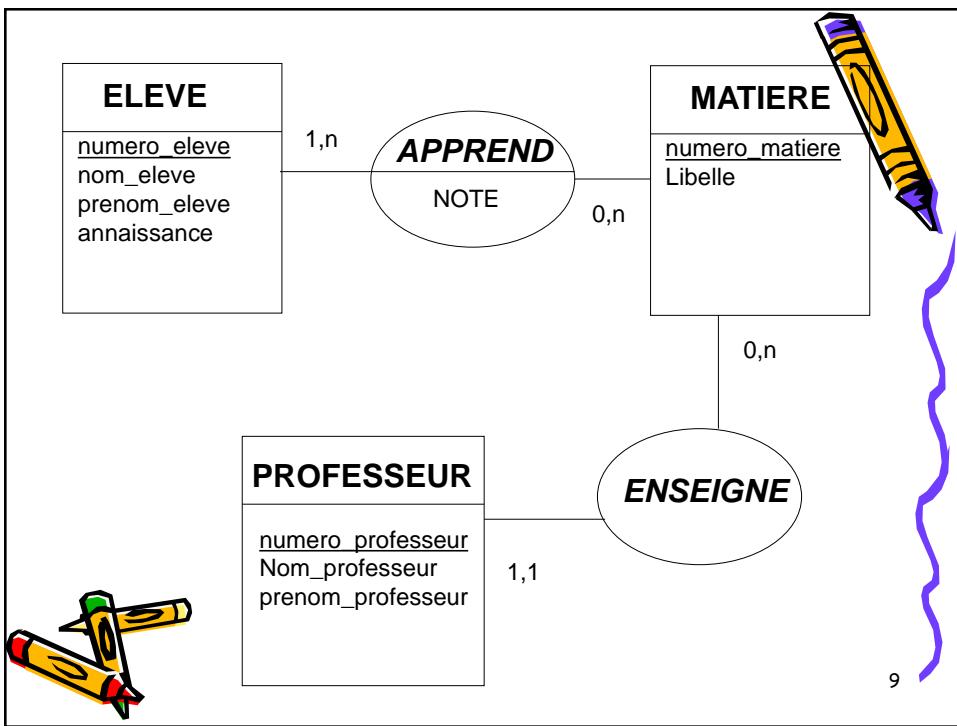


## Merise données : MCD



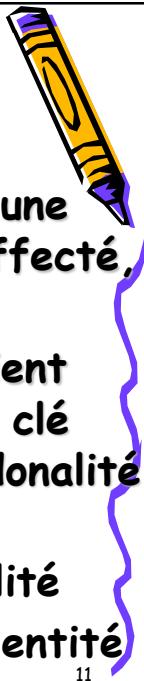
## MCD Lycée





## Merise : MLD relationnel

- Chaque entité devient une table
- Chaque association porteuse devient une table (associations : apprend, est\_affecté, intervient: 1,n ou 0,n to 0,n ou 1,n)
- Chaque association non porteuse devient soit une table de pointeurs, soit une clé étrangère dans la table qui a la cardinalité 0,1 ou 1,1
- Ici, nous pouvons voir que la cardinalité (1,1) ou (0,1) va nous indiquer l'entité qui va recevoir la clé étrangère.



11

## Merise : MLD relationnel

1. **Elève** (numero\_eleve, nom\_eleve, prenom\_eleve, annaissance)
2. **Apprend** (numero\_eleve, numero\_matiere, Note)
3. **matiere** (numero\_matiere, Libelle)
4. **Professeur** (numero\_professeur, Nom\_professeur, prenom\_professeur, numero\_matiere)
5. **Classe** (numero\_classe, Niveau)
6. **Intervient\_dans** (numero\_classe, numero\_professeur)
7. **Est\_affecté** (numero\_eleve, numero\_classe)

Le nouvel identifiant des nouvelles entités

Intervient\_dans, Apprend, Est\_affecté, seront une concaténation des deux clés étrangères<sup>12</sup>



12

## **Normalisation de Codd : point de départ**



- **Liste des attributs**
- **Dépendances fonctionnelles**
- **Dépendances multivaluées**



13

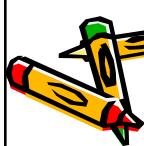
## **Normalisation : processus**

**Normaliser c'est  
regrouper la liste des  
attributs en relations**

**- ayant du sens**

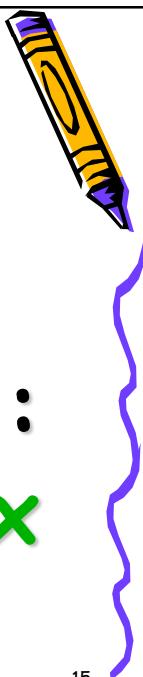
**- cohérentes**

**- sans redondance**



14

Normalisation : Point d'arrivée



On obtient une  
série de tables :  
fichiers et index



15

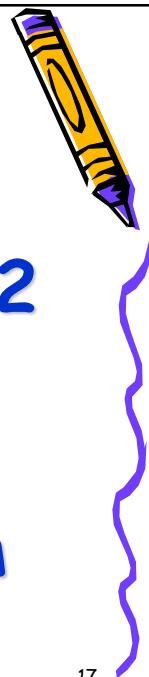
Normalisation : 1FN =  
Forme Normale

- Tout attribut de R  
contient:  
une valeur monovaluée  
et non composée



16

Normalisation : 1FN

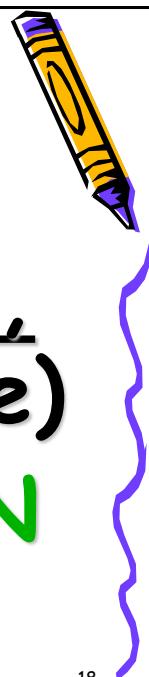


## • Exemples :

- prenom 1, prenom 2
- Adresse:  
numero\_voie, type  
voie, nom voie, nom  
ville, code postal

17

Normalisation : 1FN



● R (numero\_eleve,  
prenoms, Adresse)  
n'est pas en 1FN



18

**R normalisée en 1FN**

**R = Relation**

**• R (numero\_eleve,  
prenom1, prenom2,  
numero\_voie, Type\_voie,  
Nom\_voie, Nom\_ville,  
Code\_postal)**



19



**Normalisation : 2FN**

**1. R est en 1FN**

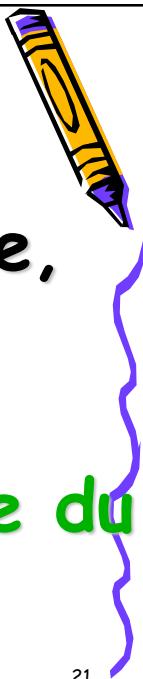
**2. Tout attribut non clé  
de R dépend de la  
totalité de la clé**



20



**Normalisation : 2FN**



- R (numero\_eleve,  
numero\_matiere, note,  
nom\_eleve)

Le Nom ne dépend  
Fonctionnellement que du  
numero\_eleve



21

100	005	12	Dupont
100	001	16	Dupont
100	002	08	Dupont
100	003	10	Dupont
200	004	12	Martin



22

## Normalisation : 2FN

● R n'est pas en 2FN

● On normalise :

R1 (numero\_eleve,  
nom\_eleve)

R2 (numero\_eleve,numero\_  
matiere, Note)



23



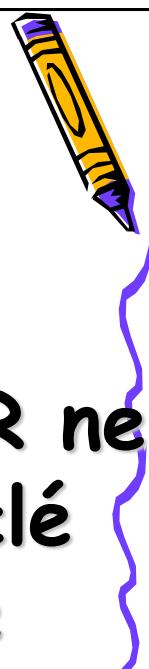
## Normalisation : 3FN

1. R est en 2FN

2. Tout attribut de R ne  
dépend pas de la clé  
par transitivité



24



## Normalisation : 3FN

● R (numero\_professeur,  
Nom\_professeur,  
numero\_matiere, Libelle  
matiere)

Libelle matière dépend  
fonctionnellement de numero\_  
professeur par transitivité, à  
travers le numero\_matiere

25

100	Duval	001	Anglais
101	Farenc	001	Anglais
102	Gervais	001	Anglais
103	Justin	001	Anglais
104	Loliée	002	Espagnol

26

On normalise en 3FN



R1 (numero\_professeur,  
Nom\_professeur,  
numero\_matiere)



R2 (numero\_matiere,  
Libelle matiere)

27

Normalisation 4FN et 5FN



● On traite les DM

Dépendances multivaluées



28

4FN : la relation ne contient qu'une seule DM

## R (Etudiant, Cours, Sport)

avec

Etudiant DM Cours

Etudiant DM Sport

29

R n'est pas en 4FN

Pierre	Maths	Karaté
Pierre	Anglais	Karaté
Pierre	Physique	Karaté
Jacques	Philo	Judo
Jacques	Philo	Natation
Jacques	Philo	Marathon

30

**Normalisation de R en 4FN**



**R1 (Etudiant, Cours)**

**R2 (Etudiant, Sport)**



31

**R1**

Pierre	Maths
Pierre	Anglais
Pierre	Physique
Jacques	Philo



32

**R2**

Pierre	Karaté
Jacques	Judo
Jacques	Natation
Jacques	Marathon

33

**5FN : DM de jointure**

**R (Revendeur, Marque, Type de produit)**

avec

**Revendeur DM Marque**

**Revendeur DM Type de produit**

**Marque DM Type de produit**

34



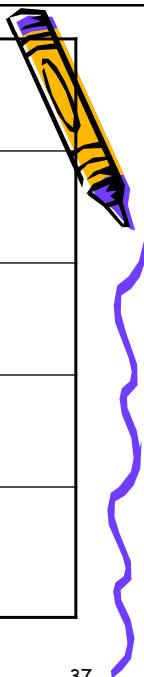



Pierre	Renault	Voiture
Pierre	Renault	Camion
Pierre	Volvo	Voiture
Paul	Renault	Camion
Jacques	Renault	Voiture
Jacques	Volvo	Voiture




<b>R1 (Revendeur, Marque)</b>
<b>R2 (Revendeur, Type de produit)</b>
<b>R3 (Marque, Type de produit)</b>

Pierre	Renault
Pierre	Volvo
Jacques	Renault
Paul	Renault
Paul	Volvo

 **R1** 

37

Pierre	Voiture
Pierre	Camion
Paul	Camion
Jacques	Voiture

 **R2** 

38

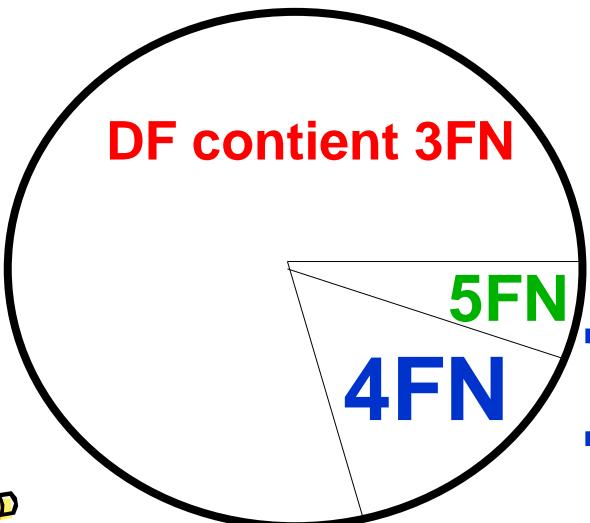
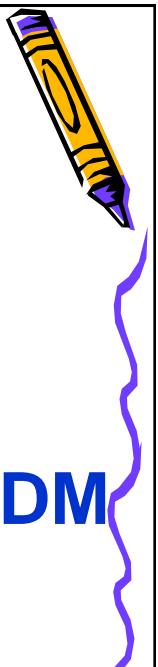


<i>Renault</i>	<i>Voiture</i>
<i>Renault</i>	<i>Camion</i>
<i>Volvo</i>	<i>Voiture</i>



*R3*

39



**DF contient 3FN**



**5FN**  
**4FN** } DM

40

## NORMALISATION : préparation

- numero\_eleve = A
  - nom\_eleve = B
  - prenom\_eleve = C
  - annaissance = D
  - numero\_matiere = E
  - Libelle\_matiere = F
  - Note = G
  - numero\_professeur = H
  - Nom\_professeur = I
  - prenom\_professeur = J
  - numero\_classe = K
  - Niveau = L
- A DF B, C, D, K
  - E DF F
  - AE DF G
  - H DF I, J, E
  - K DF L
- A DM E
  - H DM K
  - E DM H



41



## NORMALISATION : exécution

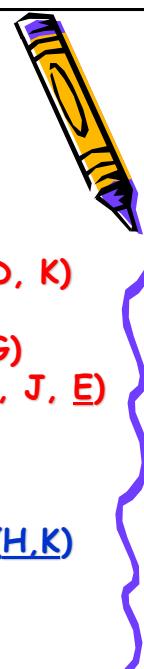
- A DF B, C, D, K
  - E DF F
  - AE DF G
  - H DF I, J, E
  - K DF L
- A DM E
  - H DM K
  - E DM H



- 1) DF
  - Elève (A, B, C, D, K)
  - matière (E, F)
  - Apprend (A, E, G)
  - Professeur (H, I, J, E)
  - Classe (K, L)

- 2) DM
  - Intervient dans (H, K)

42



## *Création & manipulation des BDD : requêtes SQL*



- « Structured Query Language » 1986

- Opérateurs de l'algèbre relationnelle



43

« LDD » :

*langage de définition des données*

**CREATE TABLE eleve  
(numero\_eleve NUM(6) [ UNIQUE] [NOT  
NULL] [PRIMARY KEY],  
nom\_eleve CHAR(30)[NOT NULL],  
prenom\_eleve CHAR(20) [NOT NULL],  
annaissance NUM(4)...)**



44

« LDD »

**CREATE TABLE** apprend  
**(numero\_eleve [ UNIQUE] [NOT NULL]  
[FOREIGN KEY],**  
**numero\_matiere [ UNIQUE] [NOT NULL]  
[FOREIGN KEY],**  
**note NUM(2,2) [VALEUR: 0,00 à 20,00])**



45

□ **INDEX : structures d'accès**

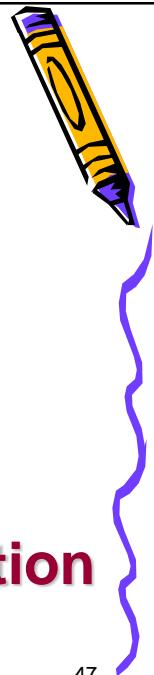
**CREATE [UNIQUE] INDEX i1  
ON eleve (numeroéléve)**

**CREATE INDEX i2 ON eleve  
(annaissance)**



46

ALTER TABLE matiere  
ADD COLUMN coefficient  
NUM(1)

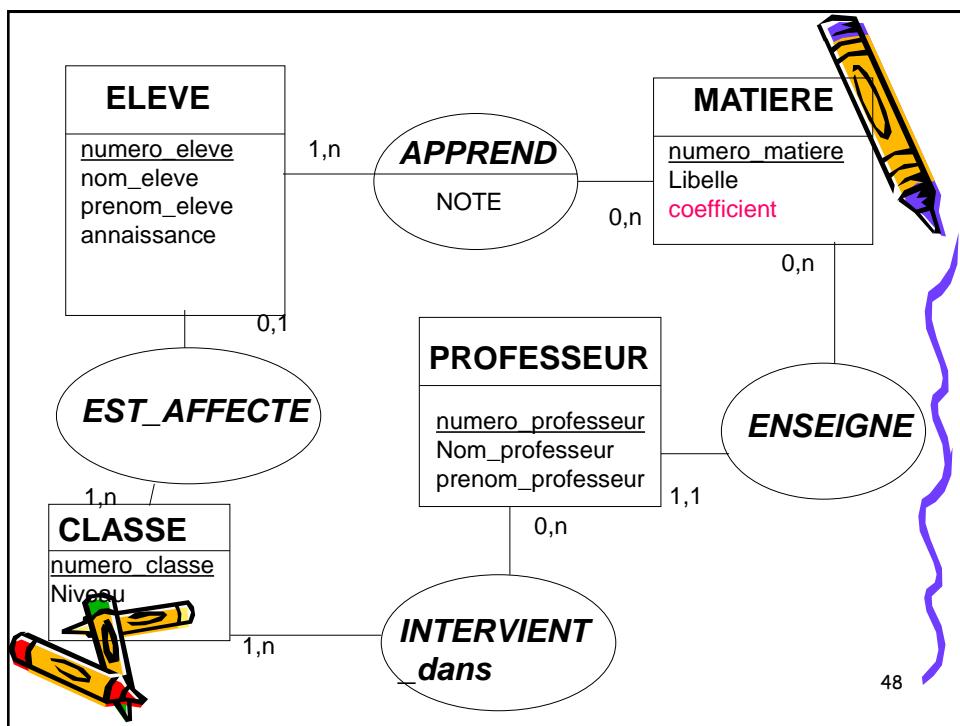


DROP TABLE apprend

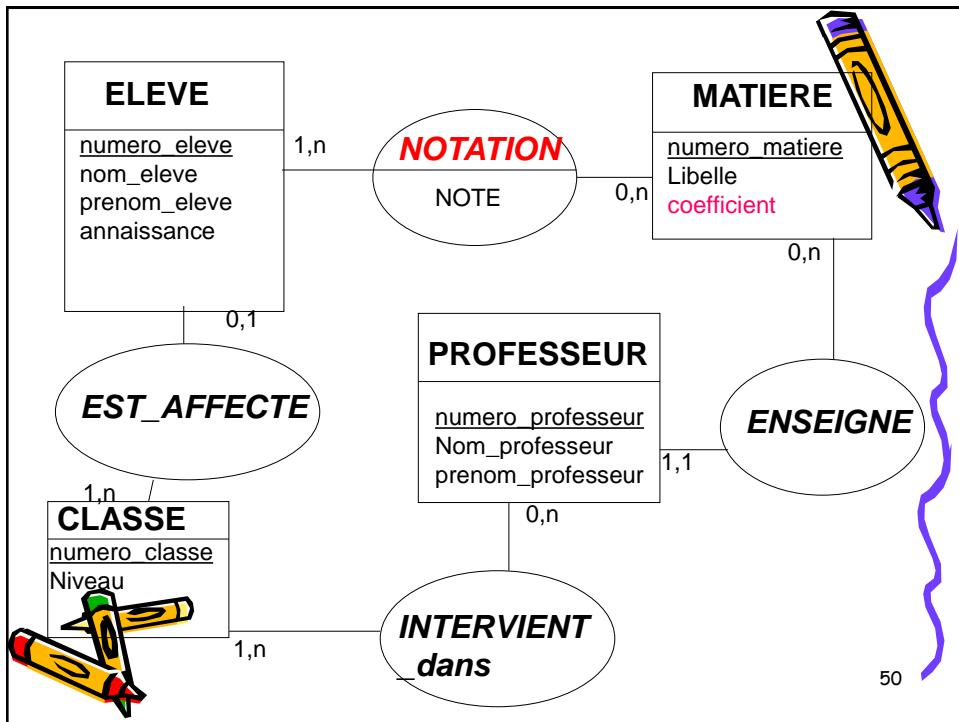
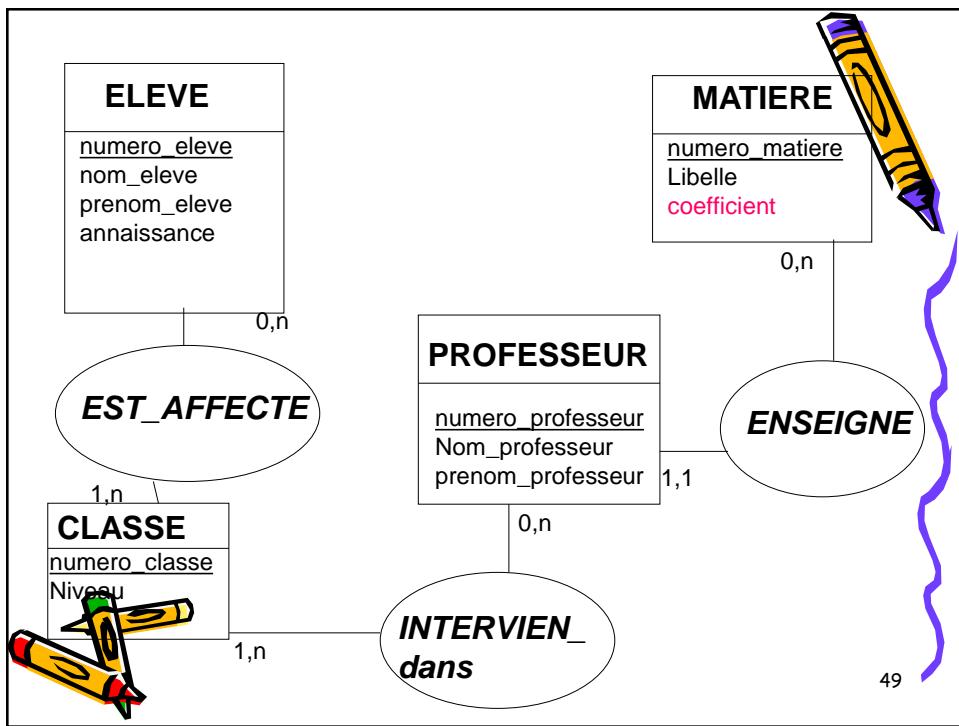
RENAME apprend to notation



47



48



« Les autorisations d'accès » :

## GRANT & REVOKE



**GRANT privilege ON objet TO public**

**Qui (public) a le droit de faire  
quoi (privilège)  
à quoi (objet)**



51

**GRANT INSERT  
ON apprend  
TO professeur**



**GRANT SELECT  
ON apprend  
TO PUBLIC**



52

« LMD » : langage de manipulation des données

**SELECT**



**INSERT**

**UPDATE**



**DELETE**

53

**SELECT**

**SELECT**

**nom\_eleve, prenom\_eleve**

**FROM eleve**

**WHERE annaissance = 1994**

**ORDER BY nom\_eleve**



54

```
SELECT nom_eleve, prenom_eleve  
FROM eleve, apprend  
WHERE annaissance = 1994  
AND eleve.numero_eleve=apprend.numero_eleve  
AND note NOT NUL  
ORDER BY nom_eleve
```

55

□**INSERT**

```
INSERT INTO eleve  
(numero_eleve, nom_eleve,  
prenom_eleve, annaissance)  
VALUES (105, durant, alexis,  
2000)
```

56

## UPDATE

UPDATE apprend

SET note = 14

WHERE numero\_eleve = 106

AND numero\_matiere = 5



57



## DELETE

DELETE FROM

eleve

WHERE

numero\_eleve = 78



58



## Les agrégats

### SUM

**GROUP BY** pour regrouper les données appartenant à la même entité(obligatoire dans SELECT).

**SELECT numero\_client, SUM(montant)**

**FROM commande**

**GROUP BY numero\_client**

59

### AVG

2 requêtes, Est-ce possible de les regroupées !

**SELECT numero\_eleve, nom\_eleve**

**FROM eleve**

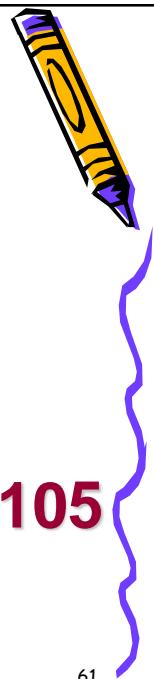
**SELECT numero\_eleve, AVG(note)**

**FROM apprend**

**GROUP BY numero\_eleve**

60

MIN ou MAX

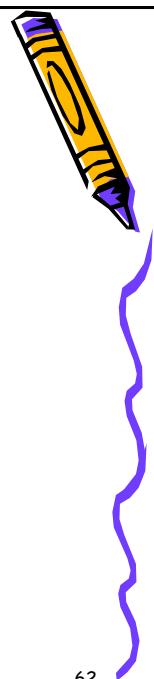


```
SELECT MAX(note)  
FROM apprend  
WHERE numero_eleve = 105
```



61

COUNT



```
SELECT COUNT(*)  
FROM eleve
```



62

# Opérateurs ensemblistes

EXISTS

UNION

INTERSECT

MINUS



63

EXISTS

la commande EXISTS s'utilise dans une clause conditionnelle pour savoir s'il y a une présence ou non de lignes lors de l'utilisation d'une sous-requête.

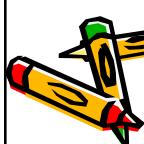
**SELECT nom\_eleve FROM eleve**

**WHERE EXISTS**

**(SELECT \* FROM apprend**

**WHERE note = 20**

**AND eleve.numero\_eleve =  
apprend.numero\_eleve)**



64

## ❑ NOT EXISTS



**SELECT nom\_eleve FROM eleve**

**WHERE NOT EXISTS**

**(SELECT \* FROM matiere,apprend**

**AND eleve.numero\_eleve =**  
**apprend.numero\_eleve**

**AND matiere.code\_matiere =**  
**apprend.code\_matiere)**

65

## ❑ UNION

Permet de concaténer les résultats de 2 requêtes ou plus. Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournent le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

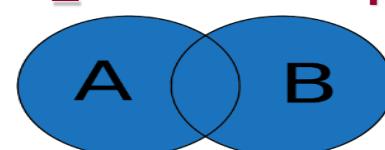
**apprend(numero\_eleve, code\_matiere)**

**pratique(numero\_eleve, code\_sport)**

**SELECT numero\_eleve FROM apprend**

**UNION**

**SELECT numero\_eleve FROM pratique**



66

## INTERSECT

Permet d'obtenir l'intersection des résultats de 2 requêtes. Cette commande permet donc de récupérer les enregistrements communs à 2 requêtes. Cela peut s'avérer utile lorsqu'il faut trouver s'il y a des données similaires sur 2 tables distinctes.

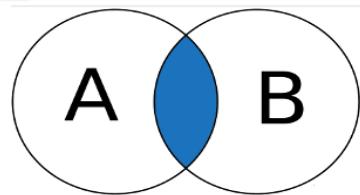
**SELECT numero\_eleve**

**FROM apprend**

**INTERSECT**

**SELECT numero\_eleve**

**FROM pratique**



67

## EXCEPT/MINUS

EXCEPT s'utilise entre 2 instructions pour récupérer les enregistrements de la première instruction sans inclure les résultats de la seconde requête. Si un même enregistrement devait être présent dans les résultats des 2 syntaxes, ils ne seront pas présent dans le résultat final.

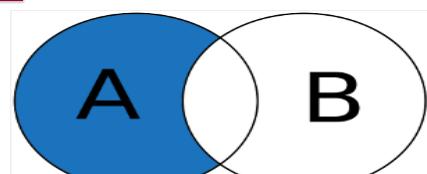
**SELECT numero\_eleve**

**FROM apprend**

**MINUS**

**SELECT numero\_eleve**

**FROM pratique**



68

## ❑ VUE : table virtuelle

Non stockée (temporaire)  
créeée par une requête  
sur une ou plusieurs tables  
(permanentes)



69

- **CREATE VIEW** [France\_Custom]  
**AS**  
**SELECT** CustomerName,  
ContactName  
**FROM** Customers  
**WHERE** Country = 'France';

- **SELECT \* FROM** [France\_Custom];



70

# DF : Dépendances Fonctionnelles



## Exercice 1 :

Considérons le schéma de la relation suivante : R(A, B, C, D, E)

Cette relation est définie en extension par les tuples suivants :

A	B	C	D	E
a1	b2	c2	d3	e2
a1	b2	c2	d1	e1
a2	b3	c2	d1	e5
a2	b4	c5	d1	e5

- 1) Parmi les dépendances fonctionnelles suivantes, lesquelles s'appliquent à l'extension de R ?

- E→D       E→A       {A, E}→C
- D→E       B→C       {A, E}→D
- C→A       B→D       {A, D}→E
- E→B       B→A       {A, B}→A

