مخبر هندسة الأنظمة المعقدة
LABORATOIRE D'INGÉNIERIE DES SYSTÈMES COMPLEXES

جامعة باجي مختار-عنابة
Université Badji Mokhtar-Annaba

Administration des bases de données avancées

Master I: Ingénierie des Logiciels Complexes (ILC)

Dr Kamilia MENGHOUR
Laboratoire d'Ingénierie des Systèmes Complexes
Université Badji Mokhtar-Annaba
K_menghour@yahoo.fr

Année Universitaire : 2022- 2023

Chapitre 1

Rappel sur les bases de données et les SGBD

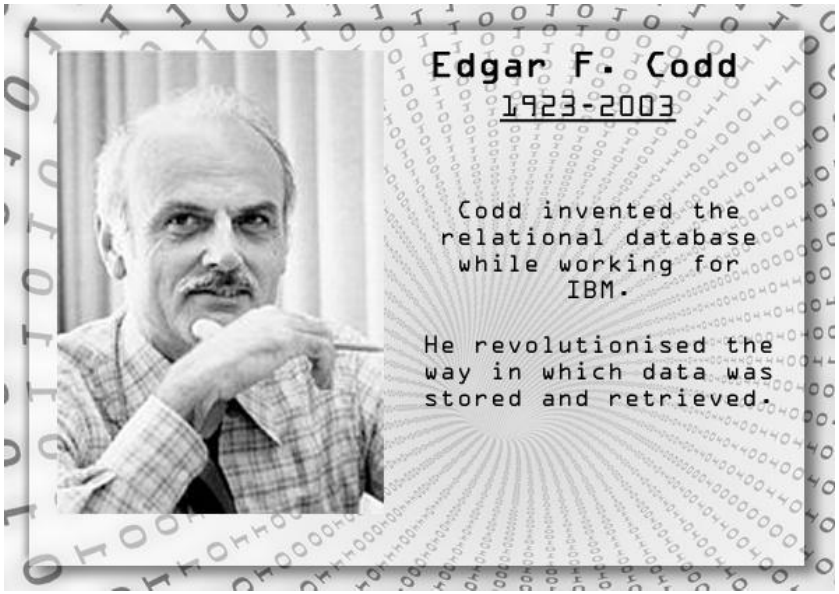
Plan du cours

Chapitre 1 : *Rappel sur les bases de données et les SGBD (10%)*

- Environnement d'un système de bases de données (composants, langages et acteurs)
- Classification des modèles de bases de données.
- Modèles, schémas et instance.
- Rappel sur modèle relationnel
  - Algèbre relationnelle
  - SQL

Modèle relationnel

- L'organisation des données au sein d'une BD a une importance essentielle pour faciliter l'accès et la mise à jour des données.
- Le ***modèle relationnel*** est fondé sur la théorie mathématique bien connue des RELATIONs. Cette théorie se construit à partir de la théorie des ensembles.
  - Il a été introduit par **Codd** en 1970 (centre de recherche IBM).
  - Les Données dans ce modèle sont organisées en tables (relations).
  - La stratégie d'accès aux données est déterminée par le SGBD.



Les avantages du modèle relationnel

- Simplicité de présentation : représentation sous forme de tables,
- Opérations relationnelles : algèbre relationnelle,
- Indépendance physique : optimisation des accès, stratégie d'accès déterminée par le système,
- Indépendance logique : concept de VUES,
- Maintien de l'intégrité : contraintes d'intégrité définies au niveau du schéma.

Définitions : Relations, Attributs, n-uplets(1/4)

- ***Domaine:*** ensemble de valeurs atomiques d'un certain type sémantique
- ***Exemple:***

$$NOM\_VILLE = \{ Alger,\ Annaba, Oran\}$$

- Les domaines sont les ensembles de valeurs possibles dans lesquels sont puisées les données
- deux ensembles peuvent avoir les mêmes valeurs bien que sémantiquement distincts
- ***Exemple:***

$$NUM\_CLIENT = \{ 1, 2, \dots , 2000 \}$$

$$NUM\_ANNEE = \{ 1, 2, \dots , 2000 \}$$

Définitions : Relations, Attributs, n-uplets(2/4)

- ***La Relation:*** sous ensemble du produit cartésien de plusieurs domaines.

$$R \subset D1 \times D2 \times \ldots \times Dn$$

-D1, D2, … , Dn sont les domaines de R. où n est appelé *degré* de la relation et R est une relation de degré n sur les domaines $D1, D2, \ldots, Dn$.

- Une ***relation*** R est un ensemble d'attributs $\{A1, A2, \ldots, An\}$.
- Dans une base relationnelle, on utilise toujours la représentation d'une relation sous forme de table.

Exemple: CLIENT (NumClient, Nom, Prenom, DateNaiss , Rue, CP, Ville).

Définitions : Relations, Attributs, n-uplets(3/4)

- ***Attributs:*** Les attributs nomment les colonnes d'une relation. Il servent à la fois à indiquer le contenu de cette colonne, et à la référencer quand on effectue des opérations. Un attribut est toujours associé à un domaine.
- Le nom d'un attribut peut apparaître dans plusieurs schémas de relations
- ***Schéma de relation:*** Un schéma de relation est simplement un nom suivi de la liste des attributs, chaque attribut étant associé à son domaine.
- La syntaxe est donc :

$$R(A1:D1,A2:D2,\ldots,An:Dn)$$

-Ou les Ai sont les noms d'attributs et Di les domaines.

Définitions : Relations, Attributs, n-uplets (4/4)

- ***Les n-uplets:***
  - Un **n-uplet** - ou **tuple** (en anglais), **instance.**
  - Un élément d'une relation de dimension n ($a1, a2, \ldots, an$).
  - Dans la représentation par table, un n-uplet est une ligne.
- ***Exemples :***

PERSONNE	Num_securite_sociale	Nom	Prenom	Code_postal	Telephone
n-uplet1	1 76 02 99 167 098	Benmalek	amar	41500	06 08 78 65 88
n-uplet2	2 76 04 95 165 008	Salhi	Anis	31900	02 99 167 098
n-uplet3	1 78 12 38 122 4332	Amara	mohamed	38700	04 38 56 45 32
n-uplet4	1 68 02 99 5649 876	Bahi	ishak	75016	01 55 45 34 87

STAGE	Num_securite_sociale	D_type_stage	Titre	Date-debut
n-uplet1	1 76 02 99 167 098	Inge_Adjoint	Définition d'une politique Qualité	01/02/2006
n-uplet2	2 76 04 95 165 008	Inge_Adjoint	Mise en place d'un SI pour la maintenanc	02/02/2006
n-uplet3	1 68 02 99 5649 876	EDT	Reconfiguration des achats	15/03/2005
n-uplet4	2 76 04 95 165 008	EDT	Reconfiguration des achats	15/03/2005
n-uplet5	1 76 02 99 167 098	PFE	Mise en place d'un ERP	15/03/2005

Contraintes d'intégrité

- ***Clé primaire :*** La clé primaire d'une relation est le plus petit sous-ensemble des attributs qui permet d'identifier chaque ligne de manière unique.
- ***Clé étrangère :*** Attribut qui est clé primaire d'une autre relation.

Notations :Clés primaires soulignées, clés étrangères en *italiques*.

Exemple: PRODUIT (NumProd, Dési, PrixUni, *NumFour*)

*NumProd clé primaire de la relation PRODUIT.

*Connaître le fournisseur de chaque produit ⇒ajout de l'attribut *NumFour* à la relation PRODUIT.

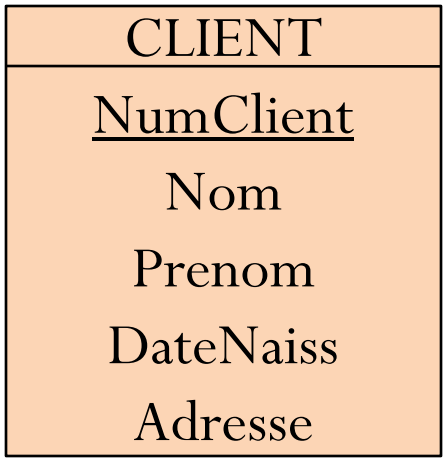
- ***Contraintes de domaine :*** les attributs doivent respecter une condition logique .

Exemple: PrixUni > 0 ET PrixUni ≤ 10000.

Passage d'un schéma E/A à un schéma relationnel (1/4)

- Les règles de passage d'un schéma Entité/Association à un Schéma relationnel:
- ***Règle 1 :*** Traduction des entités.
  - Chaque entité devient une relation.
  - Les attributs de l'entité deviennent attributs de la relation.
  - Seuls les attributs simples des attributs composés sont inclus.
  - L'identifiant de l'entité devient clé primaire de la relation.

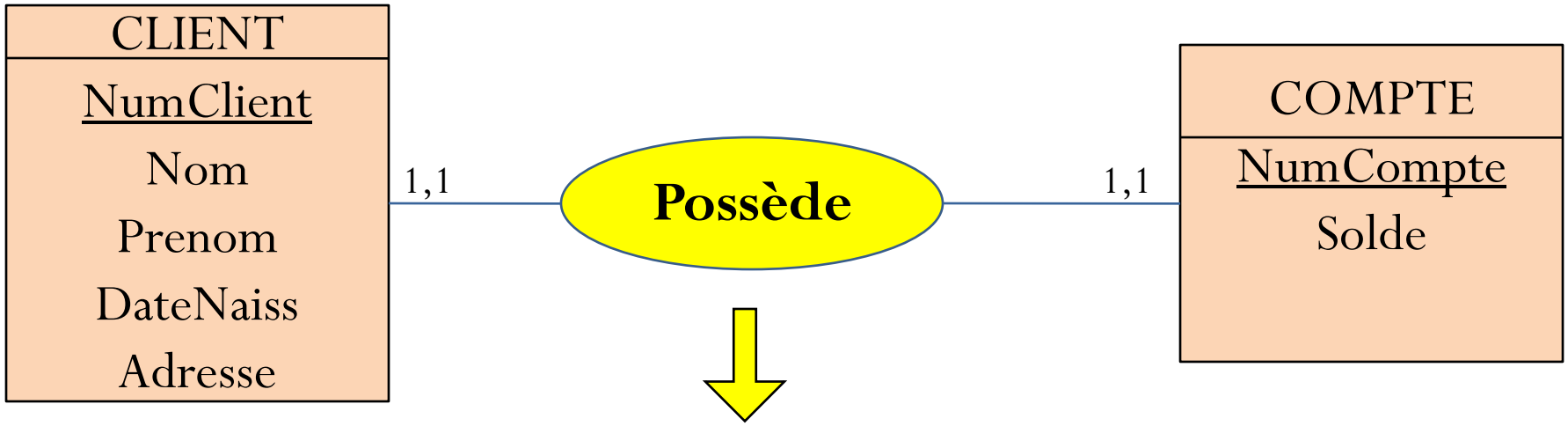
Exemple :



CLIENT(NumClient, Nom, Prenom, DateNaiss , Rue, CP, Ville)

Passage d'un schéma E/A à un schéma relationnel (2/4)

- ***Règle 2 :*** Traduction des associations 1,1.
  - Chaque association 1-1 est prise en compte en incluant la clé primaire d'une des relations comme clé étrangère dans l'autre relation.
  - ***Exemple***:

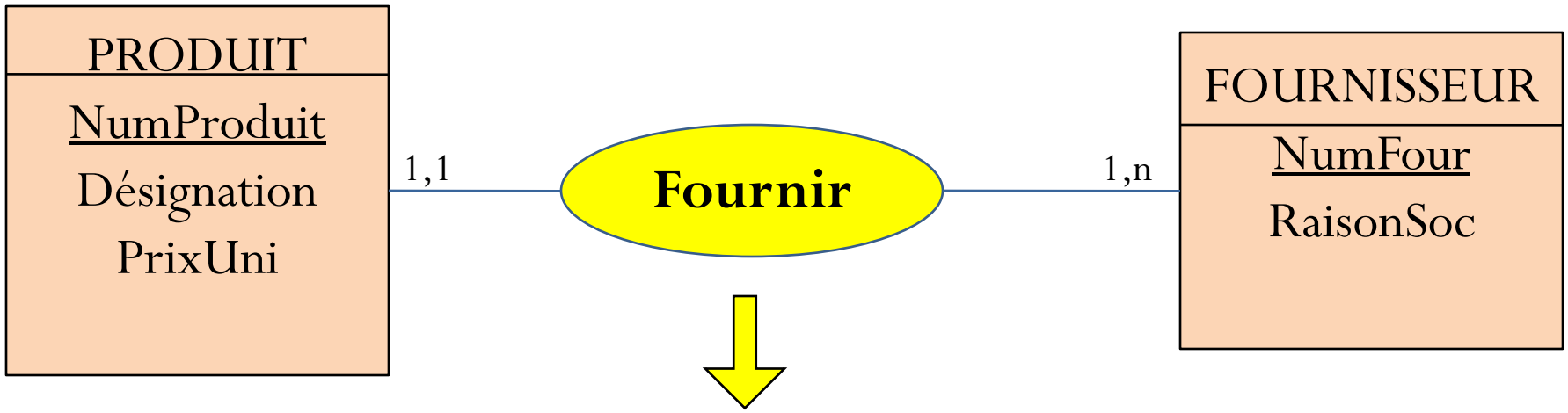


CLIENT(NumClient, Nom, Prenom, DateNaiss , Rue, CP, Ville)
COMPTE(NumCompte, solde, *NumClient*)

(*NumClient*) clé étrangère de COMPTE référençant CLIENT.

Passage d'un schéma E/A à un schéma relationnel (3/4)

- ***Règle 3 :*** Traduction des associations 1,n
  - Chaque association 1,n est prise en compte en incluant la clé primaire de la relation dont la cardinalité maximale est N comme clé étrangère dans l'autre relation.
  - ***Exemple***:



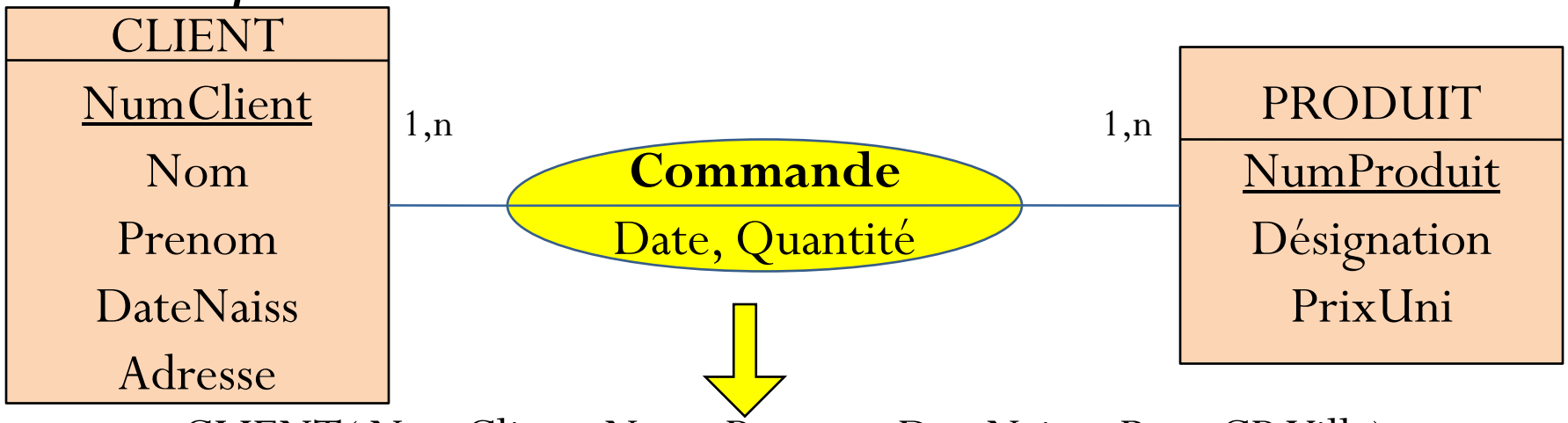
PRODUIT (NumProd, Dési, PrixUni, *NumFour*)

FOURNISSEUR (NumFour, RaisonSoc)

(*NumFour*) clé étrangère de PRODUIT référençant FOURNISSEUR.

Passage d'un schéma E/A à un schéma relationnel (4/4)

- ***Règle 4 :*** Traduction des associations M,N.
  - Chaque association M-N est prise en compte en créant une nouvelle relation dont la clé primaire et la concaténation des clés primaires des relations participantes. Les attributs de l'association sont insérés dans cette nouvelle relation.
  - ***Exemple:***



CLIENT(NumClient, Nom, Prenom, DateNaiss , Rue, CP, Ville)

PRODUIT (NumProd, Dési, PrixUni)

COMMANDE (*NumClient, NumProd*, Date, Quantité*)*

Les règles d'intégrité

- ***Les règles d'intégrité*** sont les assertions qui doivent être vérifiées par les données contenues dans une base.
- Le modèle relationnel impose les contraintes structurelles suivantes :
  - ***Intégrité de domaine***: Les valeurs d'une colonne de relation doivent appartenir au domaine correspondant.
  - ***Intégrité de clé*** :Les valeurs de clés primaires doivent être uniques et non NULL.
  - ***Intégrité référentielle***: Permet de vérifier la présence de données référencées dans des tables différentes. Une contrainte d'intégrité référentielle peut s'utiliser dès qu'une clé primaire d'une table est utilisée comme référence dans une autre table « **clé étrangère»**.
- La gestion automatique des contraintes d'intégrité est l'un des outils les plus importants d'une base de données.

La normalisation

- Les ***formes normales*** sont les différents stades de qualité qui permettent d'éviter la redondance dans les bases de données relationnelles afin d'éviter ou de limiter : les pertes de données, les incohérences au sein des données, l' effondrement des performances des traitements.

- ***Objectifs de la normalisation :***
  - Suppression des problèmes de mise à jour.
  - Minimisation de l'espace de stockage (élimination des redondances).

Les dépendances fonctionnelles (DF)

- Soit R (X, Y, Z) une relation où X, Y, et Z sont des ensembles d'attributs. Z peut être vide.

- ***Définition*** : Y dépend fonctionnellement de X (X → Y) si c'est toujours la même valeur deY qui est associée à X dans la relation R.
- ***Exemple***:

  PRODUIT (NumProd, Désі, PrixUni)

  NumProd → Désі

  Désі → PrixUni

Propriétés des dépendances fonctionnelles (DF)

- ***Réflexivité*** : Si $Y \subseteq X$ alors $X \rightarrow Y$.
- ***Augmentation*** : Si $W \subseteq Z$ et $X \rightarrow Y$ alors $X, Z \rightarrow Y, W$.
- ***Transitivité*** : Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$.
- ***Pseudo-transitivité*** : Si $X \rightarrow Y$ et $Y, Z \rightarrow T$ alors $X, Z \rightarrow T$.
- ***Union*** : Si $X \rightarrow Y$ et $X \rightarrow Z$ alors $X \rightarrow Y, Z$.
- ***Décomposition*** : Si $Z \subseteq Y$ et $X \rightarrow Y$ alors $X \rightarrow Z$.
- NB : La notation X, Y signifie $X \cup Y$

Première forme normale (1FN)

- Une relation est en ***première forme normale*** si, et seulement si, tout attribut contient une valeur ***atomique*** (non multiples, non composées).

- ***Exemple:*** le pseudo schéma de relation

  Personne(num-personne, nom, prenom, rue-et-ville, prenomsenfants)

  n'est pas en première forme normale. Il faut le décomposer en :

  - Personne(num-personne, nom, prenom, rue, ville)
  - Prenoms-enfants(*num-personne, num-prenom*)
  - Prenoms(num-prenom, prenom)

Deuxième forme normale (2FN)

- Une relation est en ***deuxième forme normale*** si , et seulement si:
  - Elle est en 1FN ;
  - Tout attribut non *clé primaire* est dépendant de la clé primaire entière.
- ***Exemple:***

1. La relation CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville) est en 2FN.
2. La relation COMMANDE_PRODUIT (NumProd, Quantité, NumFour, Ville) n'est pas en 2FN car on a :

   NumProd, NumFour → Quantité et NumFour → Ville

   - La décomposition suivante donne deux relations en 2FN :
     - COMMANDE (*NumProd, NumFour*, Quantité) ;
     - FOURNISSEUR (NumFour, Ville).

Troisième forme normale (3FN)

- Une relation est en ***troisième forme normale*** si , et seulement si:
  - Elle est en 2FN ;
  - Il n'existe aucune DF entre deux attributs non clé primaire.
- ***Exemple:***
- La relation COMPAGNIE (Vol, Avion, Pilote) avec les DF: Vol→Avion, Avion → Pilote et Vol → Pilote est en 2FN, mais pas en 3FN.
- Anomalies de mise à jour sur la relation COMPAGNIE : il n'est pas possible d'introduire un nouvel avion sur un nouveau vol sans préciser le pilote correspondant.
- La décomposition suivante donne deux relations en 3FN qui permettent de retrouver (par transitivité) toutes les DF :
- R1 (Vol, Avion) ; R2 (Avion, Pilote).

Plan du cours

Chapitre 1 : *Rappel sur les bases de données et les SGBD (10%)*

- Environnement d'un système de bases de données (composants, langages et acteurs)
- Classification des modèles de bases de données.
- Modèles, schémas et instance.
- Rappel sur modèle relationnel
  - Algèbre relationnelle
  - SQL

L'algèbre relationnelle

- ***L'algèbre relationnelle :*** a été inventée par E. Codd comme une collection d'***opérations*** formelles qui agissent sur des relations et produisent les relations en résultats.
- Ces opérateurs se divisent en trois classes:
  - *Les opérateurs unaires (Sélection, Projection):* ce sont les opérateurs les plus simples, ils permettent de produire une nouvelle table à partir d'une autre table.
  - *Les opérateurs binaires ensemblistes (Union, Intersection Différence):* ces opérateurs permettent de produire une nouvelle relation à partir de deux relations de même degré et de même domaine.
  - *Les opérateurs binaires ou n-aires (Produit cartésien, Jointure, Division):* ils permettent de produire une nouvelle table à partir de deux ou plusieurs autres tables.

Les opérateurs unaires: la sélection

- ***La sélection*** (parfois appelée restriction) génère une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont l'expression logique **E**, on la note

$$\sigma(E)R$$

- Elle permet de choisir (i.e. sélectionner) des lignes dans la table. Le résultat de la sélection est donc une nouvelle relation qui a les mêmes attributs que R.

-Si R est vide (i.e. ne contient aucune occurrence), la relation qui résulte de la sélection est vide.

Les opérateurs unaires: la sélection

- ***Exemple :***

PRODUIT

NumProd	Désignation	PrixUni
1	P1	100DZD
2	P2	500DZD
3	P3	1000DZD
4	P3	8100DZD
5	P4	20DZD
6	P5	2600DZD

R1

NumProd	Désignation	PrixUni
3	P3	1000DZD
4	P3	8100DZD
6	P5	2600DZD

$$R1 = \sigma(PrixUni \geq 1000DZD)PRODUIT$$

Les opérateurs unaires: la projection

- La ***projection*** consiste à supprimer les attributs autres que A1, … An d'une relation et à éliminer les n-uplets en double apparaissant dans la nouvelle relation ; on la note

Π(A1, … An)R.

- Elle permet de choisir des colonnes dans le tableau.
- Si R est vide, la relation qui résulte de la projection est vide, mais pas forcément équivalente (elle contient généralement moins d'attributs).

Les opérateurs unaires: la projection

- ***Exemple :***

PRODUIT

NumProd	Désignation	PrixUni
1	P1	100DZD
2	P2	500DZD
3	P3	1000DZD
4	P3	8100DZD
5	P4	20DZD
6	P5	2600DZD

R2

Désignation
P1
P2
P3
P4
P5

$R2 = \Pi_{(Désignation)}\ PRODUIT$

Les opérateurs binaires ensemblistes :l'union

- ***L'union*** est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation constituée des n-uplets appartenant à chacune des deux relations R1 et R2 sans doublon, on la note

R1 U R2.

- Le résultat de l'union est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 et R2 sont vides, la relation qui résulte de l'union est vide.

Les opérateurs binaires ensemblistes : l'union

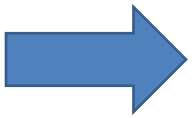
- ***Exemple :***

PRODUIT1

Désignation	PrixUni
P1	100DZD
P2	500DZD
P3	1000DZD
P3	8100DZD
P4	20DZD
P5	2600DZD

PRODUIT2

Désignation	PrixUni
P9	1050DZD
P12	4400DZD
P53	1080DZD
P63	440DZD



PRODUIT3

Désignation	PrixUni
P1	100DZD
P2	500DZD
P3	1000DZD
P3	8100DZD
P4	20DZD
P5	2600DZD
P9	1050DZD
P12	4400DZD
P53	1080DZD
P63	440DZD

PRODUIT3= PRODUIT1 ∪ PRODUIT2

Les opérateurs binaires ensemblistes :l'intersection

- ***L'intersection*** est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux appartenant aux deux relations, on la note

R1 ∩ R2.

- Le résultat de l'intersection est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 ou R2 ou les deux sont vides, la relation qui résulte de l'intersection est vide.

Les opérateurs binaires ensemblistes : l'intersection

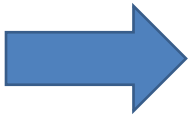
- ***Exemple :***

PRODUIT1

Désignation	PrixUni
P1	100DZD
P2	500DZD
P3	1000DZD
P3	8100DZD
P4	20DZD
P5	2600DZD
P12	4400DZD

PRODUIT2

Désignation	PrixUni
P9	1050DZD
P12	4400DZD
P53	1080DZD
P63	440DZD
P3	1000DZD
P4	20DZD



PRODUIT3

Désignation	PrixUni
P3	1000DZD
P4	20DZD
P12	4400DZD

PRODUIT3= PRODUIT1 ∩ PRODUIT2

Les opérateurs binaires ensemblistes : la différence

- La ***différence*** est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux ne se trouvant que dans la relation R1 ; on la note

R1 − R2.

- Le résultat de la différence est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 est vide, la relation qui résulte de la différence est vide. Si R2 est vide, la relation qui résulte de la différence est identique à R1.

Les opérateurs binaires ensemblistes : la différence

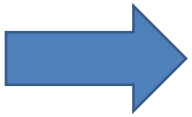
- ***Exemple :***

PRODUIT1

Désignation	PrixUni
P1	100DZD
P2	500DZD
P3	1000DZD
P3	8100DZD
P4	20DZD
P5	2600DZD
P12	4400DZD

PRODUIT2

Désignation	PrixUni
P9	1050DZD
P12	4400DZD
P53	1080DZD
P63	440DZD
P3	1000DZD
P4	20DZD



PRODUIT3

Désignation	PrixUni
P1	100DZD
P2	500DZD
P3	8100DZD
P5	2600DZD

PRODUIT3= PRODUIT1 − PRODUIT2

Les opérateurs binaires ou n-aires : produit cartésien

- Le ***produit cartésien*** est une opération portant sur deux relations R1 et R2 et qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2, on la note

R1 × R2.

- Le résultat du produit cartésien est une nouvelle relation qui a tous les attributs de R1 et tous ceux de R2. Si R1 ou R2 ou les deux sont vides, la relation qui résulte du produit cartésien est vide. Le nombre d'occurrences de la relation qui résulte du produit cartésien est le nombre d'occurrences de R1 multiplié par le nombre d'occurrences de R2.

Les opérateurs binaires ou n-aires : produit cartésien

- ***Exemple :***

PRODUIT

NumProd	Désignation
1	P1
5	P4

FOURNISSEUR

NumFour	RaisonSoc
10	F1
15	F2
36	F3

R

NumProd	Désignation	NumFour	RaisonSoc
1	P1	10	F1
1	P1	15	F2
1	P1	36	F3
5	P4	10	F1
5	P4	15	F2
5	P4	36	F3

R= PRODUIT × FOURNISSEUR

Les opérateurs binaires ou n-aires : la jointure

- La ***jointure*** est une opération portant sur deux relations R1 et R2 qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2 qui satisfont l'expression logique E. La jointure est notée

$$\mathbf{R1 \triangleright\!\triangleleft_E R2.}$$

- Si R1 ou R2 ou les deux sont vides, la relation qui résulte de la jointure est vide.
- En fait, la jointure n'est rien d'autre qu'un produit cartésien suivi d'une sélection

$$\mathbf{R1 \triangleright\!\triangleleft_E R2 = \sigma_E (R1 \times R2)}$$

Les opérateurs binaires ou n-aires : la jointure

- ***Exemple :***

FAMILLE

Prénom	Age
Fatima	19
Youcef	23
Mouhamed	40

CADEAU

AgeC	Article	Prix
99	Livre	30
6	Poupée	60
20	Montre	45
10	Déguisement	15

R

Prénom	Age	AgeC	Article	Prix
Fatima	19	99	Livre	30
Fatima	19	20	Montre	45
Youcef	23	99	Livre	30
Mouhamed	40	99	Livre	30

$$R = FAMILLE \triangleright\triangleleft ((Age \leq AgeC) \wedge (Prix < 50)) \; CADEAU$$

Les opérateurs binaires ou n-aires : la jointure

- Une ***theta-jointure*** est une jointure dans laquelle l'expression logique E est une simple comparaison entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2. La theta-jointure est notée

R1 ⋈E R2.

- Une ***equi-jointure*** est une theta-jointure dans laquelle l'expression logique E est un test d'égalité entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2. L'equi-jointure est notée

R1 ⋈A1,A2 R2.

Ou

R1 ⋈A1=A2 R2

Les opérateurs binaires ou n-aires : la jointure

- Une ***jointure naturelle*** est une jointure dans laquelle l'expression logique E est un test d'égalité entre les attributs qui portent le même nom dans les relations R1 et R2. Dans la relation construite, ces attributs ne sont pas dupliqués mais fusionnés en une seul colonne par couple d'attributs. La jointure naturelle est notée

R1 ⋈ R2.

- On peut préciser explicitement les attributs communs à R1 et R2 sur lesquels porte la jointure : R1 ⋈A1, …, An R2.

Les opérateurs binaires ou n-aires : la jointure

- ***Exemple :***

FAMILLE

Prénom	Age
Mouhamed	50
Youcef	23
Fatima	6

CADEAU

Age	Article	Prix
50	Livre	30
6	Poupée	60
23	Montre	45
10	Déguisement	15

R

Prénom	Age	Article	Prix
Mouhamed	50	Livre	30
Youcef	23	Montre	45
Fatima	6	Poupée	60

$$R = FAMILLE \triangleright\triangleleft CADEAU$$

ou

$$R = FAMILLE \triangleright\triangleleft_{Age} CADEAU$$

Les opérateurs binaires ou n-aires : la division

- La ***division*** est une opération portant sur deux relations R1 et R2, telles que le schéma de R2 est strictement inclus dans celui de R1, qui génère une troisième relation regroupant toutes les parties d'occurrences de la relation R1 qui sont associées à toutes les occurrences de la relation R2 ; on la note

R1 ÷ R2

- la division de R1 par R2 (R1 ÷ R2) génère une relation qui regroupe tous les n-uplets qui, concaténés à chacun des n-uplets de R2, donne toujours un n-uplet de R1.
- La relation R2 ne peut pas être vide. Tous les attributs de R2 doivent être présents dans R1 et R1 doit posséder au moins un attribut de plus que R2 (inclusion stricte). Le résultat de la division est une nouvelle relation qui a tous les attributs de R1 sans aucun de ceux de R2. Si R1 est vide, la relation qui résulte de la division est vide.

Les opérateurs binaires ou n-aires : la division

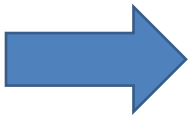
- ***Exemple :***

R1

NumProd	Num_Four
1	1
2	5
3	1
5	5
3	5
2	1

R2

Num_Four
1
5



R

Num_Prod
2
3

R= R1 ÷ R2

Plan du cours

Chapitre 1 : *Rappel sur les bases de données et les SGBD (10%)*

- Environnement d'un système de bases de données (composants, langages et acteurs)
- Classification des modèles de bases de données.
- Modèles, schémas et instance.
- Rappel sur modèle relationnel
  - Algèbre relationnelle
  - SQL

Présentation de SQL (Structured Query Language)

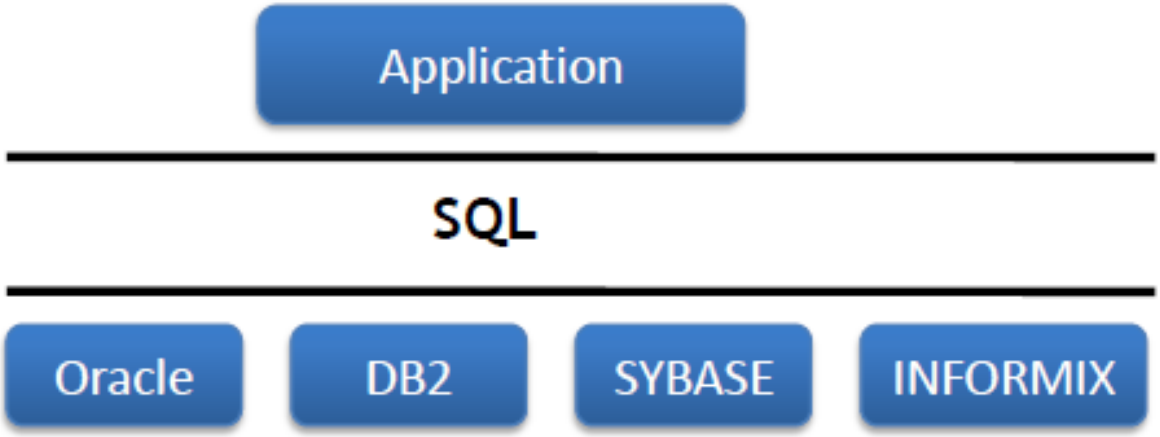
« ***Structured Query Language***» ou « ***Langage d'interrogation structuré***».

- Introduit par IBM dans les années 70,
- l'évolution du langage SEQUEL (Structured English as a Query Language),
- Commercialisé par ORACLE.
- Devenu le langage standard des systèmes de gestion de bases de données relationnelles (SGBDR).

- Il a été normalisé dès 1986 :
  - SQL 1 initial : ANSI (1986).
  - SQL 1 intégrité référentielle : ANSI (1989).
  - SQL 2 : ANSI (1992), extension de SQL1.
  - SQL 3: ANSI (1999)(appelée aussi SQL99)implémentée dans Oracle.

Importance du langage SQL

- Standard d'accès aux serveurs de données relationnels, norme ISO:
  - SQL1: norme de base
  - SQL2: meilleur support des règles du relationnel
  - SQL3: intégration du modèle objet.
- SQL est le langage commun de nombreux systèmes commercialisés
- SQL est l'interface logiciel/logiciel entre les applications et les BDR.



Les commandes SQL (1/4)

SQL comporte trois parties :

1. ***Langage de définition de données*** (LDD): permet de spécifier le schéma d'une base de données relationnelle
2. ***Langage de manipulation des données*** (LMD) : c'est-à-dire du langage de mise à jour (LMJ) et du langage d'interrogation des données (LID).
3. ***Langage de contrôle des données*** (LCD): permet d'autoriser ou d'interdire l'accès aux données.

Les commandes SQL (2/4)

1. *Le Langage de définition de Données (LDD):*

- Ensemble de commandes qui définit une base de données et les objets qui la composent.
- La **définition** d'un objet inclut:
  - sa création: CREATE
  - sa modification ALTER
  - sa suppression DROP

Les commandes SQL (3/4)

2. *Le Langage de manipulation de Données (LMD):*

- Ensemble de commandes qui permet la *consultation* et la *mise à jour* des objets crées par le langage de définition de données.
  - **Consultation** : SELECT
- La **mise à jour** inclut :
  - l'insertion de nouvelles données : INSERT
  - la modification de données existantes : UPDATE
  - la suppression de données existantes : DELETE

Les commandes SQL (4/4)

1. *Le Langage de Contrôle de Données (LCD):*

- Ensemble de commandes de contrôle d'accès aux données.
- Le **contrôle** d'accès inclut :
  - l'autorisation à réaliser une opération : GRANT
  - l'interdiction de réaliser une opération : DENY
  - Annulation d'une commande de contrôle précédente : REVOKE
  - l'autorisation à modifier des enregistrements : UPDATE
  - l'interdiction de modifier des enregistrements : READ (consultation en lecture seulement)
  - l'autorisation a supprimer des enregistrements : DELETE

Langage de définition de Données (LDD)(1/3)

- Types de données principaux:
  - NUMBER(n) : nombre entier à n chiffres
  - NUMBER(n, m) : nombre réel à n chiffres au total (virgule comprise) et m chiffres après la virgule.
  - VARCHAR(n) : chaine de caractères de taille n
  - DATE : date au format 'JJ-MM-AAAA'.

Langage de définition de Données (LDD)(2/3)

- Contraintes d'intégrité:
  - Mot clé CONSTRAINT
  - Identification par un nom de contrainte
  - Clé primaire :

    PRIMARY KEY (clé)
  - Clé étrangère :

    FOREIGN KEY (clé) REFERENCES table(attribut)
  - Contrainte de domaine :

    CHECK (condition)

Langage de définition de Données (LDD)(3/3)

- ***Exemples :***

```
*CREATE TABLE Etudiant ( NumEtu NUMBER(8), Nom VARCHAR(255),
                Prenom VARCHAR(255), DateNaiss DATE, Rue VARCHAR(255),
                   CP NUMBER(5), Ville VARCHAR(255),
                CONSTRAINT EtuClePri PRIMARY KEY (NumEtu))
*CREATE TABLE Passer ( NumEtu NUMBER(8), CodeEpr VARCHAR(10),
                Note NUMBER(5, 2),
                CONSTRAINT PassClePri PRIMARY KEY (NumEtu, CodeEpr),
                CONSTRAINT PassCleEtrEtu FOREIGN KEY (NumEtu)
                REFERENCES Etudiant (NumEtu),
                CONSTRAINT PassCleEtrEpr FOREIGN KEY (CodeEpr)
                REFERENCES Epreuve (CodeEpr),
                CONSTRAINT NoteValide CHECK (Note >= 0 AND Note <= 20) )
```

Langage de Manipulation de Données (LMD)(1/9)

- La close WHERE **:** Elle permet de spécifier la ou les conditions que doivent remplir les lignes choisies.

Remarque : Dans la close WHERE, on peut utiliser que des propriétés qui sont dans la table sélectionnée.

- La close GROUP BY: Un groupe est un sous-ensemble des lignes d'une table ayant la même valeur pour un attribut. Par exemple, on peut grouper les étudiants fonction de leur ville.
- La close HAVING : Elle ne s'utilise qu'avec le GROUP BY et permet de donner la ou les conditions que doivent remplir ces groupes.
- La close ORDER BY : Elle permet de spécifier l'ordre dans lequel vont être affichées les lignes.

Langage de Manipulation de Données (LMD)(2/9)

- Soit la base de données : ***ETUDIANTS***

ETUDIANT (NumEtu, Nom, Prenom, DateNaiss, Rue, CP, Ville)

MATIERE (CodeMat, Intitule)

EPREUVE (CodeEpr, DateEpr, Lieu, *CodeMat*)

PASSER (*NumEtu, CodeEpr*, Note)

Le Langage de Manipulation de Données (LMD)(3/9)

- ***Mise à jour des données***

  - **Ajout** d'un n-uplet
    - INSERT INTO **Matiere**
      VALUES ('BDDA', 'Bases de données avancées')
  - **Modification** de la valeur d'un attribut
    - UPDATE **Etudiant** SET Prenom='Mouhamed'
      WHERE NumEtu = 333333
    - UPDATE **Passer** SET Note = Note + 1
  - **Suppression** de n-uplets
    - DELETE FROM **Etudiant**
      WHERE Ville = 'Annaba'
    - DELETE FROM **Epreuve**

Le Langage de Manipulation de Données (LMD)(4/9)

- ***Interrogation des données***

  - **Projection**

    - Noms et Prénoms des étudiants, uniquement (pas les autres attributs)

```
SELECT Nom, prenom FROM Etudiant
```

  - **Sélection (Restriction)**

    - Liste des étudiants :

```
SELECT * FROM Etudiant
```

Le Langage de Manipulation de Données (LMD)(5/9)

- ***Interrogation des données***

  - **Projection +Sélection**

    - Liste des étudiantes qui habitent à Annaba:

```
SELECT Nom, Prenom FROM Etudiant
WHERE Ville = 'Annaba'
```

    - Liste des étudiants regroupés par ville où habitent plus de 10 étudiants

```
SELECT Nom, Ville FROM Etudiant
GROUP BY Ville
HAVING Count (Nom) > 10;
```

    - Liste des matières dans l'ordre alphabétique.:

```
SELECT Intitule FROM Matiere
ORDER BY Intitule
```

Le Langage de Manipulation de Données (LMD)(6/9)

- ***Interrogation des données***

  - **Jointure**

    - Liste des notes avec les noms des étudiants :

```
SELECT Nom, Note FROM Etudiant,Passer
WHERE Etudiant.NumEtu = Passer.NumEtu
```

  - Jointure exprimée avec le prédicat IN

    - Notes des épreuves passées le 15 septembre 2020

```
SELECT Note FROM Passer
WHERE  CodeEpr IN ( SELECT CodeEpr FROM Epreuve
                    WHERE DateEpr = '15-09-2020' )
```

Sous requête

NB : Il est possible d'imbriquer des requêtes.

Le Langage de Manipulation de Données (LMD)(7/9)

- ***Les fonctions agrégats*** (fonctions prédéfinis) : Il existe cinq
  - AVG : permet de calculer la moyenne d'un champ
    - Moyenne des notes:

```
SELECT AVG (Note) FROM Passer
WHERE Ville = 'Annaba'
```

  - COUNT : permet de compter les lignes
    - Nombre total de notes:

```
SELECT COUNT(*) FROM Passer
```

  - SUM : fait la somme des valeurs d'un champ
    - La somme des note de l'étudiant numéro «124 »:

```
SELECT SUM(Note) FROM Passer, Etudiant
WHERE NumEtu = '124'
```

  - MAX : fournit la valeur maximale d'un attribut
  - MIN : fournit la valeur minimale d'un attribut

Le Langage de Manipulation de Données (LMD)(8/9)

- ***Autres fonctions*** :
  - LIKE permet d'utiliser des jokers dans les chaînes de caractères ('_' veut dire un caractère quelconque, % veut dire un nombre quelconque de caractères)
    - Numéros des élèves dont le nom a pour deuxième lettre la lettre "r":

```
SELECT NumEtu FROM Erudiant
WHERE LIKE '_r%'
```

  - DISTINCT permet de n'obtenir qu'une seule fois chaque occurrence
    - Les salles ou se dérouler les épreuves:

```
SELECT DISTINCT Lieu FROM Epreuve
```

Le Langage de Manipulation de Données (LMD)(9/9)

- ***Autres fonctions*** :
  - ABS(n) : Valeur absolue de n
  - CEIL(n) : Plus petit entier ≥ n
  - FLOOR(n) : Plus grand entier ≤ n
  - MOD(m, n) : Reste de m/n
  - POWER(m, n) : m^n
  - SIGN(n) : Signe de n
  - SQRT(n) : Racine carrée de n
  - ROUND(n, m) : Arrondi à 10^-m
  - TRUNC(n, m) : Troncature à 10^-m
  - CHR(n) : Caractère ASCII n° n
  - INITCAP(ch) : 1re lettre en maj.
  - LOWER(ch) : c en minuscules
  - UPPER(ch) : c en majuscules
  - LTRIM(ch, n) : Troncature à gauche
  - RTRIM(ch, n) : Troncature à droite
  - REPLACE(ch, car) : Remplacement de caractère
  - SUBSTR(ch, pos, lg) : Extraction de chaîne
  - SOUNDEX(ch) : Représentation phonétique de ch
  - LPAD(ch, lg, car) : Compléter à gauche
  - RPAD(ch, lg, car) : Compléter à droite

Merci de votre attention

Des Questions

K_menghour@yahoo.fr