Bases de données relationnelles

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Cours 1 : Introduction aux bases de données

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Evaluation

- 1 note de suivi de TP et TD (50%)
- 1 examen final de 3h (50%)

Récapitulatif de l'ensemble des cours

- Cours 1 : Introduction aux bases de données
- Cours 2 : Langage SQL (partie 1) requêtes et opérateurs
- Cours 3 : Langage SQL (partie 2) sous-requêtes et structure
- Cours 4 : Langage algébrique
- Cours 5 : Normalisation
- Cours 6 : Transactions

Récapitulatif de l'ensemble des TP et TD

- TP1 : Manipulation graphique d'une BDD
- TP2 : Utilisation du langage SQL
- TD1 : Utilisation du langage algébrique
- TD2 : Dépendances fonctionnelles

Objectifs

- Savoir ce qu'est un SGBD
- Comprendre l'apport des bases de données
- Connaitre le modèle relationnel
- Apprendre plusieurs styles de langage (Graphique, algébrique, textuel)
- Apprendre des mécanismes avancées: journalisation, transaction...
- Mettre en œuvre une base de données : prise en compte des problèmes d'optimisation, de gestion des droits d'accès
- Comprendre les problèmes liés à la conception de gros systèmes d'information

Exemples de base de données

- Produits et achats dans un supermarché
- Réservation d'un vol
- Transactions bancaires
- Gestion d'un établissement scolaire :
  liste des étudiants,
  liste des profs,
  emplois du temps,
  livres dans la bibliothèque

...

Système de gestion de base de données (SGBD)

Définition : **Base de donnée**

- Des données en relation logique
- Des descriptions de données

Définition : **SGBD** (Système de gestion de base de données)

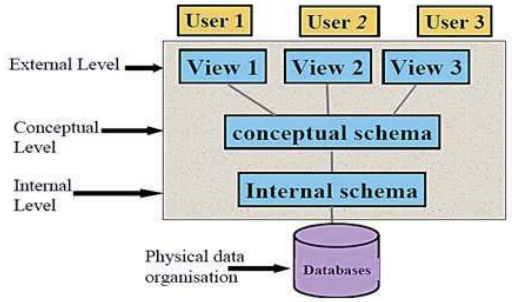
Le système logiciel qui permet de définir, créer, mettre à jour et contrôler l'accès d'une base de donnée

- LDD : langage de définition de données (UML, langage algébrique)
- LMD : langage de manipulation de données (requêtes SQL)

Spécificités d'un SGBD

- Très grande quantité de données a gérer
- Besoin d'interroger, mettre a jour souvent, rapidement et efficacement ces données
- Contrôler la redondance d'information
- Partage des données / Accès concurrents
- Gérer les autorisation d'accès / Sécurité des données
- Offrir des interfaces d'accès multiples
- Vérifier les contraintes d'intégrité
- Assurer la reprise après panne

Architecture à 3 niveaux



Principaux concepts

- Relation/Table
- Identifiant/Clé primaire
- Identifiant externe/Clé étrangère
- Domaine

Modèle relationnelle

- Modèle logique propose en 1970 par Tedd Codd (IBM lab.)
- Basé sur la notion de relations au sens mathématique, la théorie des ensembles et la logique des prédicats du 1 er ordre
- Premier système en 1980 : Oracle avec SQL/DS
- Actuellement : DB2, INFORMIX, ORACLE, SQL Server, Ingres, Sybase, Dbase, Access, MySQL, . . .

Terminologie (1)

Définition : **Relation**

Une table avec des colonnes et des lignes

Définition : **Attribut**

Le nom d'une colonne de la relation

Définition : **Domaine**

Valeurs admissibles pour un (ou plusieurs) attributs

Définition : **Tuple**

Une ligne dans une relation

Terminologie (2)

Définition : **Degré**

Nombre d'attributs d'une relation

Définition : **Cardinalité**

Nombre de tuples d'une relation

Définition : **Schéma de relation**

Une relation définie par des paires attribut /domaine

Définition : **Instance de relation**

Ensemble des tuples d'une relation

Terminologie (3)

Définition : **Clé candidate**

Ensemble minimum d'attributs pour identifier de façon unique les tuples d'une relation

Définition : **Clé primaire**

La clé candidate choisie pour identifier de façon unique les tuples d'une relation

Définition : **Clé étrangère**

Ensemble d'attributs d'une relation qui correspond à une clé candidate d'une autre relation

Terminologie (4)

Table : Personne

Clé primaire, Attribut, Clés étrangères

NumP	Nom	Age	Sexe	NumA	CodeE
150	Elyoussefi	40	M		
19	Dahbi	25	F		
11	Marzouk	20	M	301	C443AX
40	Ouchkir	30	M		

Tuple

La table Personne :

Degré = 6

Cardinalité = 4

Domaine (Sexe) = 'M' ou 'F'

Domaine (Age) = 0-100

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
303	Casablanca	Maarif	Biranzaran	3
304	Marrakech	Gueliz	Mlyrachid	5

Table : Entreprise

CodeE	Nom
A343VX	CDG
A241BB	ONEE
C443AX	INWI

Attributs monovalués et multivalués (1)

Nom	Prenom	Age	Sexe
Elyoussefi	Abdelali	40	M
Dahbi	Sara	25	F
Marzouk	Moulay	20	M
Ouchkir	Wissam	30	M

Nom	Prenom	Age	Sexe
Elyoussefi	Abdelali	40	M
Dahbi	Sara	NULL	F
Marzouk	Moulay (Driss)	20	M
Ouchkir	Wissam	30	NULL

Problèmes :

NULL : Valeur inconnue (vide)

Prenom[] : Plusieurs valeurs

Constats :

Attributs monovalués ne sont pas adaptés

Attributs monovalués et multivalués (2)

L'attribut Prenom ne peut pas être monovalué
Monovalué = 1 seule valeur
Suppression de l'attribut Prenom

Nom	Prenom	Age	Sexe
Elyoussefi	Abdelali	40	M
Dahbi	Sara	NULL	F
Marzouk	Moulay (Driss)	20	M
Ouchkir	Wissam	30	NULL

Attributs monovalués et multivalués (3)

Nom	Prenom1	Prenom2	Age	Sexe
Elyoussefi	Abdelali	NULL	40	M
Dahbi	Sara	NULL	25	F
Marzouk	Moulay	Driss	20	M
Ouchkir	Wissam	NULL	30	M

Solution 1 :
Perte de place
Requêtes plus simples

Nom	NumP	Age	Sexe
Elyoussefi	150	40	M
Dahbi	19	25	F
Marzouk	11	20	M
Ouchkir	40	30	M

NumP	Prenom
150	Abdelali
19	Sara
11	Moulay
11	Driss
40	Wissam

Solution 2 :
Nouvelle table
Perte de l'ordre ?

Attributs complexes monovalués (1)

Table : Personne

NumP	Nom	Age	Sexe
150	Elyoussefi	40	M
19	Dahbi	25	F
11	Marzouk	20	M
40	Ouchkir	30	M

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
1	Rabat	HayRiad	Nakhil	120
2	Casablanca	Maarif	Atlas	7
3	Casablanca	Maarif	Biranzaran	3
4	Marrakech	Gueliz	Mlyrachid	5

Table : Entreprise

CodeE	Nom
A343VX	CDG
A241BB	ONEE
C443AX	INWI

Attributs complexes monovalués (2)

Table : Personne

NumP	Nom	Age	Sexe	NumA (N:1)	CodeE (N:1)
150	Elyoussefi	40	M		
19	Dahbi	25	F		
11	Marzouk	20	M	301	C443AX
40	Ouchkir	30	M		

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
303	Casablanca	Maarif	Biranzaran	3
304	Marrakech	Gueliz	Mlyrachid	5

Table : Entreprise

CodeE	Nom
A343VX	CDG
A241BB	ONEE
C443AX	INWI

Attributs complexes monovalués (3)

Table : Personne

N:1

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	
19	Dahbi	25	F	
11	Marzouk	20	M	301
40	Ouchkir	30	M	

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
303	Casablanca	Maarif	Biranzaran	3
304	Marrakech	Gueliz	Mlyrachid	5

Problème :

Marzouk possède plusieurs adresses

2 adresses ne peuvent pas être affectées à 1 personne

Attributs complexes monovalués (4)

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	
19	Dahbi	25	F	
11	Marzouk	20	M	301
40	Ouchkir	30	M	

Table : Adresse

N:1

NumA	Ville	Quartier	Avenue	Numéro	NumP
301	Rabat	HayRiad	Nakhil	120	11
302	Casablanca	Maarif	Atlas	7	
303	Casablanca	Maarif	Biranzaran	3	
304	Marrakech	Gueliz	Mlyrachid	5	

Solution 1 :

Supprimer l'attribut adresse

Ajouter l'attribut personne

Attributs complexes monovalués (5)

Table : Personne

NumP	Nom	Age	Sexe
150	Elyoussefi	40	M
19	Dahbi	25	F
11	Marzouk	20	M
40	Ouchkir	30	M

Table : Adresse

N:1

NumA	Ville	Quartier	Avenue	Numéro	NumP
301	Rabat	HayRiad	Nakhil	120	11
302	Casablanca	Maarif	Atlas	7	
303	Casablanca	Maarif	Biranzaran	3	
304	Marrakech	Gueliz	Mlyrachid	5	

Problème de la solution 1 :

2 personnes ne peuvent pas avoir la même adresse

Attributs complexes multivalués (1)

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	
19	Dahbi	25	F	
11	Marzouk	20	M	301
40	Ouchkir	30	M	

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro	NumP
301	Rabat	HayRiad	Nakhil	120	11
302	Casablanca	Maarif	Atlas	7	
303	Casablanca	Maarif	Biranzaran	3	
304	Marrakech	Gueliz	Mlyrachid	5	

Solution 2 :

Pas d'attributs adresse et personne

Nouvelle table

Table : Pers-Adr

NumP	NumA
11	301
11	405
19	303

N:M

Attributs complexes multivalués (2)

Table : Personne

NumP	Nom	Age	Sexe
150	Elyoussefi	40	M
19	Dahbi	25	F
11	Marzouk	20	M
40	Ouchkir	30	M

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
303	Casablanca	Maarif	Biranzaran	3
304	Marrakech	Gueliz	Mlyrachid	5

Maintenant il est possible d'avoir :

- 2 personnes à la même adresse
- 2 adresses pour la même personne

Table : Pers-Adr

NumP	NumA
A343VX	CDG
A241BB	ONEE
C443AX	INWI

N:M

Résumé sur le modèle relationnel (1)

- Le système de gestion de base de données relationnelle est actuellement le logiciel de traitement de données le plus fréquemment utilisé
- Dans les mathématique, une relation est un sous-ensemble du produit cartésien de deux ensembles
- En termes de base de données, une relation est n'importe quel sous-ensemble du produit cartésien des domaines des attributs
- Les relations sont représentées de manière physique par des tables, les lignes correspondent aux tuples individuels et les colonnes aux attributs

Résumé sur le modèle relationnel (2)

- Une base de données a les propriétés suivantes :
  - Chaque cellule contient exactement une valeur atomique
  - Les noms d'attributs sont distincts les uns des autres
  - L'ordre des attributs est immatériel
  - L'ordre des tuples est immatériel
  - Il n'existe pas de tuples en double
- La valeur NULL représente une valeur d'un attribut inconnue à l'heure actuelle ou qui est impossible à s'appliquer

Résumé sur le modèle relationnel (3)

- Une clé primaire = la clé candidate choisie pour l'identification de tuples
- Une clé étrangère = une clé candidate d'une autre relation
- Une clé candidate = les attributs minimum pour identifier les tuples de façon unique
- Intégrité d'entité :
  Les attributs de la clé primaire ne peuvent pas être NULL
- Intégrité référentielle :
  Valeur d'une clé étrangère = valeur d'une clé candidate ou NULL

Cours 2 : Le langage SQL (partie 1) requêtes et opérateurs

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Principaux axes du cours

- Les types de données dans SQL
- Les fonctions prédéfinies
- Les requêtes simples
- Les clauses Where, Order by, Group by
- Les requêtes multi-tables (jointures)
- Les opérateurs Union, Intersect, Except

SQL (1)

SQL (**S**tructured **Q**uery **L**anguage)

- Langage déclaratif introduit par IBM milieu des années 70
- Implémenté dans plusieurs SGBD tel que MySQL
- Standardisé par l'ANSI (American National Standard Institute)
  - SQL 1 (v1) initial ANSI 1986
  - SQL 1 (v2) avec intégrité référentielle ANSI 1989
  - SQL 2 ANSI 1992
  - SQL 3 incorpore la notion d'objet ANSI 1999

SQL (2)

Fonctionnalités :

- Créer la structure de la base de données et de ses tables
- Exécuter les taches de base de la gestion des données, telle que :
  - l'insertion,
  - la modification,
  - la suppression de données dans les tables
- Effectuer des requêtes simples ou complexes

Langage orienté transformation

SQL (3)

Autres fonctionnalités :

- Contrôle des accès aux données
  - Gestion des utilisateurs
  - Gestion des habilitations
- Gestion des transactions

Principaux types d'utilisateurs :

- DBA = DataBase Administrator
- Développeurs
- Clients (utilisateurs finaux)

SQL (4)

Important : **MySQL**

- Avant d'introduire la syntaxe du langage SQL, il est important de noter que la syntaxe peut varier d'un SGBD à un autre
  - Toutefois, cette variation est minime
  - La migration d'un SGBD à un autre requiert une adaptation de la syntaxe
- Dans la suite de ce cours, on considèrera le SGBD MySQL
  - MySQL dérive directement de SQL
  - L'interface phpMyAdmin développé en PHP offre un outil graphique de manipulation du SGBD MySQL

Types de données

- Les entiers : Nombres entiers signés ou non (int)
- Les flottants : Nombres à virgule (float)
- Les chaines : Chaines de caractères (varchar)
- Les dates : Date et heure (date)
- Enumération
- Ensemble

Les entiers

nom	borne inférieure	borne supérieure
TINYINT	-128	127
TINYINT UNSIGNED	0	255
SMALLINT	-32768	32767
SMALLINT UNSIGNED	0	65535
MEDIUMINT	-8388608	8388607
MEDIUMINT UNSIGNED	0	16777215
INT*	-2147483648	2147483647
INT* UNSIGNED	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	0	18446744073709551615

(*) : **INTEGER** est un synonyme de **INT**.

UNSIGNED permet d'avoir un type non signé.

ZEROFILL : remplissage des zéros non significatifs.

Les flottants

nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
FLOAT	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(*) : **REAL** est un synonyme de **DOUBLE**.

Les chaines

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où 1<M<255, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où 1<M<255, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 caractères.
MEDIUMTEXT	Longueur maximale de 16777215 caractères.
LONGTEXT	Longueur maximale de 4294967295 caractères.
DECIMAL(M,D)*	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupe un caractère.

(*) : **NUMERIC** est un synonyme de **DECIMAL**.

Les dates

nom	description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de **TIMESTAMP**.
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
…	…

Si attribut de type TIMESTAMP

= *NULL* :

- prend la date et heure de l'insertion.
- suit le format indiqué

Enumérations

- Le type énumération ENUM restreint le choix de la valeur à une liste de chaines de caractères
- Exemple : **ENUM**

```
CREATE TABLE shirts (
   nom VARCHAR(40),
   taille ENUM('x-small', 'small', 'medium', 'large', 'x-large')
);
```

- Dans cette exemple la valeur de taille ne peut contenir qu'une des cinq valeurs prédéfinies

Ensembles

- Le type ensemble SET restreint le choix de la valeur à l'ensemble des combinaisons possibles d'une liste de chaines de caractères
- Exemple : **SET**

```
CREATE TABLE equipe (
   joueur SET('saad', 'jed')
);
```

- Dans cette exemple la valeur de joueur peut être l'une des quatre valeurs suivantes '', 'saad', 'jed', 'saad, jed'

Les fonctions prédéfinies (1)

- **SELECT** ABS (-32); 32
- **SELECT** FLOOR (1.23); 1
- **SELECT** MOD (234, 10); 4
- **SELECT 253** % 7; 1
- **SELECT** ROUND (1.298, 1); 1,3
- **SELECT** ROUND (1.298, 0); 1
- **SELECT** SIGN (234) SIGN (-32) SIGN (0); 1 / -1 / 0
- **SELECT** 3 / 5; 0,60

Les fonctions prédéfinies (2)

- **SELECT** CONCAT ('*My*', '*S*', '*QL*'); 'MySQL'
- **SELECT** CHAR_LENGTH ('*MySQL*'); 5
- **SELECT** LOCATE ('bar', 'foobarbar'); 4
- **SELECT** LOCATE ('bar', 'foobarbar', 5); 7
- **SELECT** INSERT ('Quadratic', 3, 4, 'What'); 'QuWhattic'
- **SELECT** LOWER ('*MySQL*'); 'mysql'
- **SELECT** SUBSTRING ('Quadratically',5,6); 'ratica'
- **SELECT** 'David!' LIKE 'David_'; 1
- **SELECT** 'David!' LIKE '%D%v%'; 1
- **SELECT** STRCMP (S1, S2); -1,0,1

_ et %
0 (false), 1 (true)

Les fonctions prédéfinies (3)

- La moyenne : **AVG**

```
SELECT AVG(prix)
```

- Le minimum : **MIN**

```
SELECT MIN(prix)
```

- Le maximum : **MAX**

```
SELECT MAX(prix)
```

- La somme : **SUM**

```
SELECT SUM(prix)
```

- Le décompte : **COUNT**

```
SELECT COUNT(*)                  -> compte toutes les lignes de la table
SELECT COUNT(DISTINCT prix)      -> compte toutes les lignes avec prix distincts
```

Les requêtes simples (1)

- Syntaxe d'une requête simple :

SELECT … FROM Table WHERE … GROUP BY … HAVING … ORDER BY …

- **SELECT** spécifie les colonnes qui doivent apparaitre dans les résultats
- **FROM** spécifie la table ou les tables à utiliser (requête simple = 1 table)
- **WHERE** filtre les lignes selon une condition donnée
- **GROUP BY** forme des groupes de lignes de même valeur de colonne
- **HAVING** filtre les groupes sujets à une certaine condition
- **ORDER BY** spécifie l'ordre d'apparition des données dans le résultat

Les requêtes simples (2)

Quelques exemples :

- Affichage de tous les noms et adresses des clients

```
SELECT nom, adresse FROM Client;
```

- Affichage de toutes les informations des clients

```
SELECT * FROM Client;
```

- Affichage de toutes les adresses sans doublons

```
SELECT DISTINCT adresse FROM Client;
```


La clause WHERE

- Objectif : Ajouter des contraintes à la sélection

Quelques exemples :

- Une comparaison `WHERE salaire>10000, ville='Casablanca'`
- Un intervalle `WHERE salaire BETWEEN 20000 and 30000`
- Un ensemble `WHERE couleur IN ('red', 'vert')`
- Une correspondance `WHERE adresse LIKE '%Rabat%'`
- Une valeur NULL `WHERE adresse IS NULL`

Les requêtes simples (3)

Quelques exemples :

- Affichage des produits dont le nom est XBOX

```
SELECT * FROM Produit WHERE nom = 'XBOX';
```

- Affichage des ventes réalisées il y'a plus de 30 jours

```
SELECT * FROM Vente WHERE date<CURRENT_DATE()-30;
```

- Affichage des ventes réalisées après le 01 Janvier 2017

```
SELECT * FROM Vente WHERE date>DATE('2017-01-01');
```

Les requêtes simples (4)

- Affichage des ventes dont le prix est entre 1000 et 3000 et dont le client n'est pas le numéro 23

```
SELECT * FROM Ventre WHERE (prix between 1000 and 3000) and (numero != 23);
```

- Affichage des clients dont le nom est soit saad ou jed ou karam

```
SELECT * FROM Client WHERE nom in ('saad', 'jed', 'karam');
```


Les requêtes simples (5)

- Affichage des clients dont le nom commence par P

```
SELECT * FROM Client WHERE nom LIKE 'P%';
```

- Affichage des clients dont le nom commence par P et à la 4eme lettre un S

```
SELECT * FROM Client WHERE nom LIKE 'P__S%';
```


Les requêtes simples (6)

- Affichage des ventes dont la date est inconnue

```
SELECT * FROM Vente WHERE date is NULL;
```

Attention : On n'utilise pas la notation `date = NULL`

Toute opération comparé à NULL donne comme résultat NULL

Pour une valeur non-nulle, on utilisera `date is not NULL`

La clause ORDER BY (1)

- Objectif : Trier selon les attributs

```
ORDER BY Attribut1 DESC, Attribut2 ASC
```

- Lorsqu'il s'agit d'un entier ORDER BY effectue un tri : ASC du plus petit au plus grand et DESC du plus grand au plus petit
- Lorsqu'il s'agit d'une chaine de caractère ORDER BY effectue un tri alphabétique : ASC de A à Z et DESC de Z à A

Attention : Dans cette exemple, le tri s'effectue d'abord pour Attribut1, puis une fois terminé, il s'effectue pour Attribut2

La clause ORDER BY (2)

SELECT Marque, Prix, CodeP FROM Produit ORDER BY Marque DESC, Prix ASC;

Table : Produit

Marque	Prix	CodeP
BMW	400000	BM8908
Renault	250000	RE3672
BMW	800000	BM1208
Volvo	600000	VO3412
Volvo	450000	VO3779
BMW	300000	BM3459
Renault	180000	RE1196

Table : Produit

Marque ↘	Prix ↗	CodeP
Volvo	450000	VO3779
Volvo	600000	VO3412
Renault	180000	RE1196
Renault	250000	RE3672
BMW	300000	BM3459
BMW	400000	BM8908
BMW	800000	BM1208

La clause GROUP BY (1)

- Objectif : Grouper selon les attributs

GROUP BY Attribut HAVING (Condition)

- La condition qui suit HAVING agit comme la clause WHERE, elle peut donc éliminer des tuples du résultat

Attention : Dans cette exemple, si l'on utilise une fonction statistique sur Attribut, cette fonction sera appliquée uniquement par groupe

La clause GROUP BY (2)

SELECT Marque, AVG(Prix) FROM Produit GROUP BY Marque HAVING AVG(prix)>300000;

Table : Produit

Marque	Prix	CodeP
BMW	400000	BM8908
Renault	250000	RE3672
BMW	800000	BM1208
Volvo	600000	VO3412
Volvo	450000	VO3779
BMW	300000	BM3459
Renault	180000	RE1196

Table : Produit

Marque	AVG(Prix)
BMW	500000
Volvo	525000

La clause GROUP BY calcule AVG(prix) par marque

Pour BMW : AVG(Prix) = 500000 ; Renault : AVG(Prix) = 215000 ; VOLVO : AVG(Prix) = 525000

Les requêtes multi-tables (1)

- Syntaxe d'une requête multi-tables :

SELECT … FROM Table1, Table2 … WHERE … GROUP BY … HAVING … ORDER BY …

- Le résultat regroupe toutes les occurrences qui satisfont la clause WHERE

Exemple :

SELECT Produit.nom, Vente.prix FROM Produit, Vente WHERE Vente.numProduit = Produit.numero ;

- La clause WHERE désigne le critère de jointure
- Le résultat implique plusieurs tables, il faut alors joindre les tables (Jointure)

Les requêtes multi-tables (2)

- Jointure sans critère de jointure

```
SELECT Table1.attribut5, Table2.attribut4 FROM Table1, Table2;
```

- Dans cette exemple, il n'y pas de clause WHERE donc pas de critère de jointure
- Le résultat est alors le produit cartésien des deux tables = l'ensemble des tuples composés de (Table1.attribut5, Table2.attribut4)
- Il n'y a aucun intérêt à effectuer ce genre de requêtes multi-tables

Les requêtes multi-tables (3)

- Les Jointures par défaut sont du type **INNER JOIN**

```
SELECT … FROM Table1, Table2 WHERE …
SELECT … FROM Table1 INNER JOIN Table2 WHERE …
```

- Ces deux requêtes sont équivalentes, c'est pour cette raison qu'en pratique, on ne spécifie pas INNER JOIN
- INNER JOIN = Les tuples qui ne satisfont pas la clause WHERE sont éliminés

Les requêtes multi-tables (4)

- Les Jointures du type **OUTER JOIN**

```
SELECT … FROM Table1 FULL OUTER JOIN Table2 WHERE …
SELECT … FROM Table1 LEFT OUTER JOIN Table2 WHERE …
SELECT … FROM Table1 RIGHT OUTER JOIN Table2 WHERE …
```

- **LEFT OUTER JOIN** : Les tuples de la table de gauche (LEFT) sont conservés même si la clause WHERE n'est pas vérifiée
- **RIGHT OUTER JOIN** : Les tuples de la table de droite (RIGHT) sont conservés même si la clause WHERE n'est pas vérifiée
- **FULL OUTER JOIN** : Les tuples de la table de droite et gauche sont conservés même si la clause WHERE n'est pas vérifiée

INNER JOIN

```
SELECT Personne.Nom, Adresse.Ville FROM Personne, Adresse
WHERE Personne.NumA = Adresse.NumA;
```

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat

LEFT OUTER JOIN

SELECT Personne.Nom, Adresse.Ville FROM Personne LEFT OUTER JOIN Adresse
WHERE Personne.NumA = Adresse.NumA;

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat
Elyoussefi	NULL
Dahbi	NULL

RIGHT OUTER JOIN

SELECT Personne.Nom, Adresse.Ville FROM Personne RIGHT OUTER JOIN Adresse
WHERE Personne.NumA = Adresse.NumA;

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat
NULL	Casablanca
NULL	Marrakech

FULL OUTER JOIN

SELECT Personne.Nom, Adresse.Ville FROM Personne FULL OUTER JOIN Adresse
WHERE Personne.NumA = Adresse.NumA;

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat
NULL	Casablanca
NULL	Marrakech
Elyoussefi	NULL
Dahbi	NULL

L'opérateur UNION (1)

- L'union de deux tables rassemble les tuples de la première table et les tuples de la seconde table
- Les tuples en communs ne sont conservées qu'en un seul exemplaire, c'est à dire que l'opération d'union élimine les doublons

Exemple :

SELECT Nom, Prenom FROM Professeur UNION SELECT Nom, Prenom FROM Etudiant;

- L'opération retourne une table avec nom et prénom de tous les professeurs et étudiants

Remarque : L'opérateur UNION ne peut pas être reproduit avec des jointures

L'opérateur UNION (2)

SELECT Nom, Prenom FROM Professeur UNION SELECT Nom, Prenom FROM Etudiant;

Table : Professeur

Nom	Prenom	Ville
Boukili	Driss	Casablanca
Rziza	Sara	Marrakech
El ghorfi	Rabii	Rabat

Table : Etudiant

Nom	Prenom	Age
Elyoussefi	Abdelali	40
Rziza	Sara	22
Mohtaram	Saad	21

Table : Résultat

Nom	Prenom
Boukili	Driss
Rziza	Sara
El ghorfi	Rabii
Elyoussefi	Abdelali
Mohtaram	Saad

L'opérateur INTERSECT (1)

- L'intersection de deux tables revoie les tuples communs aux deux tables

Exemple :

SELECT Nom, Prenom FROM Etudiant INTERSECT SELECT Nom, Prenom FROM Entrepreneur;

- L'opération retourne les noms et prénoms qui se trouvent à la fois dans la table Etudiant et la table Entrepreneur

Remarque : L'opérateur INTERSECT peut être reproduit avec des jointures

L'opérateur INTERSECT (2)

SELECT Nom, Prenom FROM Etudiant INTERSECT SELECT Nom, Prenom FROM Entrepreneur;

Table : Etudiant

Nom	Prenom	Age
Elyoussefi	Abdelali	40
Rziza	Sara	22
Mohtaram	Saad	21

Table : Entrepreneur

Nom	Prenom	Age
Marzouk	Reda	35
Dhiman	Alae	30
Mohtaram	Saad	21

Table : Résultat

Nom	Prenom
Mohtaram	Saad

L'opérateur EXCEPT (1)

- La différence de deux tables renvoie les tuples de la première table qu'on ne retrouve pas dans la seconde

Exemple :

SELECT Nom, Prenom FROM Etudiant EXCEPT SELECT Nom, Prenom FROM Entrepreneur;

- L'opération retourne les noms et prénoms qui se trouvent dans la table Etudiant et qui ne se trouvent pas dans la table Entrepreneur

Remarque : L'opérateur EXCEPT peut être reproduit avec des jointures

L'opérateur EXCEPT (2)

SELECT Nom, Prenom FROM Etudiant EXCEPT SELECT Nom, Prenom FROM Entrepreneur;

Table : Etudiant

Nom	Prenom	Age
Elyoussefi	Abdelali	40
Rziza	Sara	22
Mohtaram	Saad	21

Table : Entrepreneur

Nom	Prenom	Age
Marzouk	Reda	35
Dhiman	Alae	30
Mohtaram	Saad	21

Table : Résultat

Nom	Prenom
Elyoussefi	Abdelali
Rziza	Sara

L'opérateur EXCEPT (3)

SELECT Nom, Prenom FROM Etudiant EXCEPT SELECT Nom, Prenom FROM Entrepreneur;

- LE SGBD mySQL ne supporte pas l'utilisation de l'opérateur EXCEPT
- L'opérateur EXCEPT est très couteux en mémoire

Remarque : Toutefois, on peut trouver un équivalent de l'opérateur EXCEPT avec les requêtes imbriquées (Voir Cour 3)

SELECT Nom, Prenom FROM Etudiant WHERE NOT EXISTS (SELECT * FROM Entrepreneur WHERE Etudiant.nom = Entrepreneur.nom AND Etudiant.nom = Entrepreneur.nom)

L'opérateur EXCEPT (4)

De manière générale, on a les équivalences suivantes :

- Pour un seul attribut

SELECT A FROM T1 EXCEPT SELECT A FROM T2
⇔
SELECT A FROM T1 WHERE A NOT IN (SELECT * FROM T2 WHERE T1.A = T2.A)

- Pour plusieurs attribut

SELECT A, B FROM T1 EXCEPT SELECT A, B FROM T2
⇔
SELECT A, B FROM T1 WHERE NOT EXISTS (SELECT * FROM T2 WHERE T1.A = T2.A AND T1.B = T2.B)

Cours 3 : Le langage SQL (partie 2) sous-requêtes et structure

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Principaux axes du cours

- Connecteurs In, Exists, Any, All
- Sous-requêtes indépendantes
- Sous-requêtes corrélées
- Opérateur division
- Modification des tables et des tuples
- Gestion des privilèges
- Compléments les vues et les index

Sous-requêtes

Syntaxe d'une requête contenant une sous-requête R

SELECT … FROM … WHERE Bloc contenant (R)

- Une sous-requête (aussi appelé requête imbriquée) est une requête SQL encapsulée à l'intérieur d'une autre requête

Exemple :

SELECT * FROM Etudiant WHERE AgeE = (R) avec R : SELECT AVG(AgeJ) FROM JTennis

SELECT * FROM Etudiant WHERE AgeE = (SELECT AVG(AgeJ) FROM JTennis)

- Donne les étudiants dont l'âge correspond à l'âge moyen des joueurs de tennis

Connecteurs (1)

Principaux connecteurs : =, !=, <, >

- Comparaison avec une seule valeur

Exemple : SELECT * FROM table WHERE Attribut1 = 'valeur'

Les connecteurs : IN, EXISTS, ANY, ALL

- Comparaison avec plusieurs valeurs
- Ces valeurs sont issues du résultat d'une requête R = ('valeur1', 'valeur2', …)

Exemple : **IN**

SELECT * FROM table WHERE Attribut1 IN (R)

- La condition WHERE est vérifiée si Attribut1 se trouve dans R

Connecteurs (2)

Exemple : **EXISTS**

SELECT * FROM table WHERE EXISTS (R)

- La condition WHERE est vérifiée si R n'est pas vide

Exemple : **ANY**

SELECT * FROM table WHERE Attribut1 > ANY (R)

- La condition WHERE est vérifiée si Attribut1 est supérieur à un élément de R

Exemple : **ALL**

SELECT * FROM table WHERE Attribut1 > ALL (R)

- La condition WHERE est vérifiée si Attribut1 est supérieur à tout élément de R

Sous-requêtes indépendantes (1)

Sous-requêtes indépendantes = blocs indépendants

SELECT … FROM … WHERE Bloc contenant (R)

Bloc principal

- Dans une sous-requête **indépendante**, le bloc contenant R peut être évalué séparément du bloc principal
- La sous-requête R ne contient pas d'attribut provenant du bloc principal

Exemple :

SELECT * FROM Etudiant WHERE AgeE = (SELECT AVG(AgeJ) FROM JTennis);

Sous-requêtes indépendantes (2)

Les sous-requêtes indépendantes peuvent jouer le rôle d'une jointure

Exemple : Le nom des étudiants qui habitent Rabat

- En jointure :

SELECT Etudiant.nom FROM Etudiant, Adressse WHERE Etudiant.Adresse = Adresse.Num and Adresse.Ville = 'Rabat';

- En sous-requêtes indépendantes :

SELECT Etudiant.nom FROM Etudiant WHERE Etudiant.Adresse IN (SELECT Adresse.Num FROM Adresse WHERE Ville = 'Rabat';

Sous-requêtes indépendantes (3)

Les sous-requêtes indépendantes peuvent jouer le rôle d'une jointure

Exemple : Le nom des produits vendus le 01/01/2018

- En jointure :

SELECT Produit.nom FROM Produit, Vente WHERE Vente.RefP = Produit.Reference and Vente.date = DATE(01/01/2018);

- En sous-requêtes indépendantes :

SELECT Produit.nom FROM Produit WHERE Produit.Reference IN (SELECT Vente.RefP FROM Vente WHERE date = DATE(01/01/2018);

Sous-requêtes indépendantes (4)

Duplication d'une table pour préserver l'indépendance

- L'idée est de dupliquer la table en lui donnant un nouveau nom dans la sous-requête

Exemple : Les produits dont le prix est supérieur au prix moyen des produits

SELECT * FROM Produit, WHERE Produit.Prix > (SELECT AVG(P.Prix) FROM Produit P);

Sous-requêtes indépendantes (5)

Duplication d'une table pour préserver l'indépendance

Exemple : Les produits dont le prix est le plus élevé

SELECT * FROM Produit, WHERE Produit.Prix >= ALL (SELECT P.Prix FROM Produit P);

Exemple : Les produits dont le prix n'est pas le plus élevé

SELECT * FROM Produit, WHERE Produit.Prix < ANY (SELECT P.Prix FROM Produit P);

Sous-requêtes corrélées (1)

Sous-requêtes corrélées = blocs dépendants

SELECT … FROM … WHERE Bloc contenant (R)

Bloc principal

- Dans une sous-requête **corrélées**, le bloc contenant R ne peut pas être évalué séparément du bloc principal
- La sous-requête R contient des attributs provenant du bloc principal

Exemple :

SELECT * FROM Professeur WHERE Age > (SELECT AVG(P.Age) FROM Professeur P WHERE Professeur.Ville = P.Ville);

Sous-requêtes corrélées (2)

Exemple : Les professeurs dont l'âge est supérieur à la moyenne d'âge de leurs villes

SELECT * FROM Professeur WHERE Age > (SELECT AVG(P.Age) FROM Professeur P WHERE Professeur.Ville = P.Ville);

Exemple : Les produits dont le prix est supérieur à la moyenne des prix de leurs marques

SELECT * FROM Produit WHERE Prix > (SELECT AVG(P.Prix) FROM Produit P WHERE Produit.Marque = P.Marque);

Sous-requêtes corrélées (3)

Sous-requêtes corrélées et jointure

Exemple : Les villes dont aucun professeur n'est responsable du module base de données SL2IBD

SELECT Ville FROM Professeur WHERE NOT EXISTS (SELECT * FROM Matiere, Professeur P WHERE Matiere.Responsable = P.Numero and Matiere.Code = 'SL2IBD' and Professeur.Ville = P.Ville);

Sous-requêtes corrélées (4)

Sous-requêtes corrélées et jointure

Exemple : Les marques dont aucun produit n'a été vendu le 01/01/2018

SELECT Marque FROM Produit WHERE NOT EXISTS (SELECT * FROM Vente, Produit P WHERE Vente.RefP = P.Reference and Vente.date = Date (01/01/2018) and Produit.Marque = P.Marque);

Opérateur division (1)

Définition : C'est une relation composée des tuples tels que le produit cartésien avec le diviseur soit un sous-ensemble de la relation dividende

R = A / B

Table : A

attr1	attr2
a	1
a	2
a	3
b	1
c	2

Table : B

attr2
1
2

Table : R

attr1
a

Opérateur division (2)

La division : c'est les valeurs de A.attr1 pour lesquelles :
Il n'existe pas de valeur B.attr2 tel que (A.attr1, B.attr2) n'appartienne pas à A

R = A / B

Table : A

attr1	attr2
a	1
a	2
a	3
b	1
c	2

Table : B

attr2
1
2

Table : R

attr1
a

Opérateur division (3)

La division : c'est les valeurs de A.attr1 pour lesquelles :
Il n'existe pas de valeur B.attr2 tel que (A.attr1, B.attr2) n'appartienne pas à A

R = A / B

Requête SQL :

SELECT DISTINCT A.attr1 FROM A AS A1WHERE
NOT EXISTS (SELECT * FROM B AS B0 WHERE NOT EXISTS
(SELECT * FROM A AS A2 WHERE A1.attr1 = A2.attr1 and A2.attr2 = B0.attr2))

Opérations sur les tables (1)

Principales opérations sur les tables

- Création

CREATE TABLE

- Modification

ALTER TABLE

- Suppression

DROP TABLE

Opérations sur les tables (2)

Création des tables Produit, Client et Vente

- Table Produit

CREATE TABLE Produit (ref CHAR(4), nom VARCHAR(20), marque VARCHAR(20));

- Table Client

CREATE TABLE Client (num INT(10), nom VARCHAR(20), adresse VARCHAR(30));

- Table Vente

CREATE TABLE Vente (num INT(10), refP CHAR(4), refC INT(10), date DATE);

Opérations sur les tables (3)

Spécification des attributs non nuls et uniques et de la clé primaire

- Table Produit

```
CREATE TABLE Produit ( ref CHAR(4) NOT NULL,
                       nom VARCHAR(20) NOT NULL,
                       marque VARCHAR(20),
                       PRIMARY KEY (ref),
                       UNIQUE (nom)
                       );
```


Opérations sur les tables (4)

Spécification des clés étrangères

- Table Vente

```
CREATE TABLE Vente ( num INT(10) NOT NULL,
                     refP CHAR(4) NOT NULL,
                     refC INT(10) NOT NULL,
                     date DATE,
                     PRIMARY KEY (num),
                     FOREIGN KEY (refP) REFERENCES Produit (ref),
                     FOREIGN KEY (refC) REFERENCES Client (num),
                     );
```

Opérations sur les tables (5)

Gestion de l'intégrité référentielle

- Table Vente

```
CREATE TABLE Vente ( num INT(10) NOT NULL,
                     refP CHAR(4) NOT NULL, refC INT(10) NOT NULL,
                     date DATE, PRIMARY KEY (num),
                     FOREIGN KEY (refP) REFERENCES Produit (ref),
                     ON DELETE SET NULL ON UPDATE CASCADE
                     FOREIGN KEY (refC) REFERENCES Client (num),
                     ON DELETE SET NULL ON UPDATE CASCADE
                     );
```


Opérations sur les tables (6)

Gestion de l'intégrité référentielle

- ON UPDATE (mise à jour), ON DELETE (suppression)

```
FOREIGN KEY (refP) REFERENCES Produit (ref),
ON DELETE SET NULL ON UPDATE CASCADE
```

- En cas de mise à jour de la table Produit -> Mise à jour de refP
- En cas de suppression de la table Produit -> refP = NULL

Attention : Dans cette exemple, refP et refC peuvent être NULL

Opérations sur les tables (7)

Modification de la structure

- **ALTER TABLE** Produit **MODIFY** nom **VARCHAR(40);**
- **ALTER TABLE** Produit **ADD** stock **VARCHAR(20);**
- **ALTER TABLE** Client **ALTER** adresse **SET DEFAULT** 'Rabat'**;**
- **ALTER TABLE** Client **DROP** adresse**;**
- **ALTER TABLE** Client **ADD CONSTRAINT** CT1 **UNIQUE**(nom);
- **ALTER TABLE** Client **ADD CONSTRAINT** CT2 **PRIMARY KEY**(num)**;**
- **ALTER TABLE** Vente **ADD CONSTRAINT** CT3 **FOREIGN KEY**(refC) **REFERENCES** Client (num)**;**

Opérations sur les tuples (1)

Principales opérations sur les tables

- Insertion

INSERT INTO

- Modification

UPDATE

- Suppression

DELETE FROM

Opérations sur les tuples (2)

Insertion de données

- Avec désignation des colonnes

INSERT INTO Produit (ref, nom, marque) **VALUES** ('MB23', 'AMG C63', 'Mercedes');

- Selon l'ordre par défaut des colonnes

INSERT INTO Produit **VALUES** (**DEFAULT**, 'MB23', 'AMG C63', 'Mercedes');

- A partir d'une autre table

INSERT INTO Produit (ref, nom, marque) **SELECT** P.ref, P.nom, P.marque **FROM** OldProduit P **WHERE** P.marque = 'BMW';

Opérations sur les tuples (3)

Modification de données

- Modification d'un tuple

UPDATE Produit **SET** nom = '300 SLS' **WHERE** ref = 'MB75' ;

- Modification d'une collection de tuples

UPDATE Produit **SET** prix = prix * 0,9 **WHERE** ref **NOT IN**
(**SELECT** refP **FROM** Vente **WHERE** quantite > 10) ;

Opérations sur les tuples (4)

Suppression de données

- Suppression de tous les tuples

DELETE FROM Produit;

- Suppression avec condition

DELETE FROM Produit **WHERE** ref **IN**
(**SELECT** refP **FROM** Vente **WHERE** date = DATE (01/01/2018) ;

Gestion des privilèges (1)

Ajout et suppression de droits

- Ajouter des droits

GRANT privileges ON database.table TO user

- Enlever des droits

REVOKE privileges ON database.table FROM user

ALL [PRIVILEGES]	Tous les droits
ALTER	Autorise l'utilisation de ALTER TABLE
CREATE	Autorise l'utilisation de CREATE TABLE
DELETE	Autorise l'utilisation de DELETE
DROP	Autorise l'utilisation de DROP TABLE
...	...

Gestion des privilèges (2)

Création d'un nouvel utilisateur et affectation des droits

- Création d'un utilisateur

```
CREATE USER 'saad'@'localhost' IDENTIFIED BY 'password';
```

- Lui donner tous les droits sur toutes les tables de db1

```
GRANT ALL ON db1.* TO 'saad'@'localhost';
```


Les vues

Une vue est une table virtuelle dérivée d'une ou plusieurs tables de bases

- Création d'une vue

```
CREATE VIEW ProduitStar (ref, prix, nom, marque) AS SELECT FROM Vente V, Produit P
WHERE V.refP =P.ref and V.quantite > 100;
```

- Plusieurs représentations d'une même table
- Affichage des données selon les besoins de l'utilisateur
- Restreindre l'accès aux données en fonction de l'utilisateur

Les index (1)

Un index permet au SGBD de retrouver rapidement les données, il existe 3 types d'index : PRIMARY KEY, UNIQUE, ou INDEX

- Une clé primaire est strictement unique, les NULL ne sont pas autorisés
- Un index de type UNIQUE est comparable à une clé primaire, avec les valeurs NULL autorisées et potentiellement en plusieurs occurrences
- Un index de type INDEX signifie que l'on souhaite indexer une colonne susceptible de contenir des doublons

Les index (2)

La syntaxe pour créer un index est la suivante

```
CREATE INDEX index1 ON Client(nom, prenom)
```

- Lors des requêtes, l'index est en permanence comparé, il est donc utile de sélectionner des index peu volumineux (exemple : TINY INT)
- Les index sont à choisir parmi les champs concernés par une clause WHERE, ORDER BY, GROUP BY, MIN(), MAX(), ainsi que les champs qui permettent de relier des tables entre elles

Cours 4 : Le langage algébrique

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Principaux axes du cours

- Opérateurs unaires
- Opérateurs ensemblistes
- Jointure naturelle et thêta jointure
- Opérateur division
- Lien entre les opérateurs
- Complexité des opérateurs

L'algèbre relationnel

- C'est le support mathématique sur lequel repose le modèle relationnel
- Il permet de représenter les requêtes sur la base de données
- Il propose un ensemble d'opérations élémentaires formelles sur les relations
- Le résultat de toute opération est une nouvelle relation (fermeture)

Remarque : Les notations de l'algèbre relationnelle ne sont pas standardisées

Classification des opérateurs (1)

Opérateurs unaires : ⟶ Prend une table en entrée

- Sélection (σ)
- Projection (π)

Opérateurs binaires : ⟶ Prend 2 tables en entrée

- Union (∪)
- Intersection (∩)
- Différence (−)
- Produit cartésien (x)
- Jointures (|X|)
- Division (/)

Classification des opérateurs (2)

Opérateurs unaires :

- Sélection (σ)
- Projection (π)

Opérateurs binaires :

Opérateurs ensemblistes :

- Union (∪)
- Intersection (∩)
- Différence (−)
- Produit cartésien (x)

Opérateurs spécifiques :

- Jointures (|X|)
- Division (/)

Syntaxe d'une relation (1)

Syntaxe :

Table(attr1, attr2, attr3, …, attrn)

- Clé primaire (_)
- Clé étrangère (#)

Exemple :

Table(attr1, attr2#, attr3#, attr4)

Dans cette exemple, attr1 est une clé primaire, attr2 et attr3 sont des clés étrangères

Syntaxe d'une relation (2)

- Client(num, nom, ville, telephone)
- Produit(ref, prix, marque)
- Vente(num, refP#, numC#, date)

Table : Produit

ref	prix	marque
BM8908	400000	BMW
RE3672	250000	Renault
BM1208	800000	BMW
VO3412	600000	Volvo
VO3779	450000	Volvo
BM3459	300000	BMW
RE1196	180000	Renault

Table : Client

num	nom	ville	telephone
2	Elyoussefi	Rabat	0613...
17	Ouadou	Rabat	NULL
23	Elmohadi	Casablanca	NULL

Table : Vente

num	refP	numC	date
1003	VO3412	2	01/01/2018
1427	VO3779	2	15/01/2018
1499	BM3459	17	20/01/2018

Opérateurs unaires

Opérateurs unaires :

- Sélection (σ)
- Projection (π)

Autre opérateur unaire :

- Renommage (α)

La signature de ces opérateurs correspond à :

relation * paramètres → relation

Exemple : σ ville=Rabat (Professeur)

Sélection des tuples de la table Professeur où la ville est Rabat

Sélection (1)

- La sélection travaille sur une relation R et définit une nouvelle relation qui ne contient que des tuples de R qui satisfont une condition donnée

Syntaxe :

σ condition (Table)

Exemple : σ ville=Rabat (Professeur)

Table : Professeur

nom	prenom	ville
Boukili	Driss	Casablanca
Rziza	Sara	Marrakech
El ghorfi	Rabii	Rabat

Sélection (2)

- Afficher les professeurs qui habitent Rabat ou Casablanca

σ ville=Rabat or ville=Casablanca (Professeur)

- Afficher les professeurs qui n'habitent pas Casablanca

σ ville≠Casablanca (Table)

- Afficher les ventes du client numéro 120 effectuées le 01/01/2018

σ num=120 and date=01/01/2018 (Vente)

Projection (1)

- La projection travaille sur une relation R et définit une nouvelle relation en extrayant les valeurs des attributs spécifiés et en supprimant les doublons

Syntaxe :

π attributs (Table)

Exemple : π nom, prenom (Professeur)

Table : Professeur

nom	prenom	ville
Boukili	Driss	Casablanca
Rziza	Sara	Marrakech
El ghorfi	Rabii	Rabat

Projection (2)

- Afficher le nom et le prénom des professeurs

π nom, prenom (Professeur)

- Afficher le nom et prénom des professeurs de Casablanca

π nom, prenom (σ ville=Casablanca (Professeur))

- Afficher la référence du produit et le numéro client pour chaque vente

π refP, numC (Vente)

Renommage (1)

- La renommage travaille sur une relation R et définit une nouvelle relation en renommant les attributs spécifiés

Syntaxe :

α nomAttr : nouveauNomAttr (Table)

Exemple : α ville : adresse (Professeur)

Table : Professeur

nom	prenom	ville
Boukili	Driss	Casablanca
Rziza	Sara	Marrakech
El ghorfi	Rabii	Rabat

Renommage (2)

- Renommage de l'attribut ref de la table Produit

α ref : refP (Produit)

- Renommage de toute la table Produit en P1

P1 = α (Produit)

Remarques :

Le renommage des attributs sert à faire des jointures naturelles
Le renommage d'une table sert à utiliser plusieurs fois une même table, par exemple le produit cartésien d'une même table
En pratique, le renommage d'une table se fait à la suite d'une sélection ou une projection

Opérateurs ensemblistes

Opérateurs ensemblistes :

- Union (∪)
- Intersection (∩),
- Différence (−)
- Produit cartésien (x)

La signature de ces opérateurs correspond à :

relation * relation → relation

Exemple : π nom, prenom (Professeur) ∪ π nom, prenom (Etudiant)

Fusion des tables Professeur et Etudiant

Union (1)

- L'union travaille sur deux relations R et S et définit une nouvelle relation qui contient tous les tuples de R et S avec suppression des doublons

Syntaxe :

Table1 ∪ Table2

Exemple :

Res = Professeur ∪ Etudiant

Table : Professeur

Nom	Prenom
Boukili	Driss
Rziza	Sara
El ghorfi	Rabii

Table : Etudiant

Nom	Prenom
Elyoussefi	Abdelali
Rziza	Sara
Mohtaram	Saad

Table : Res

Nom	Prenom
Boukili	Driss
Rziza	Sara
El ghorfi	Rabii
Elyoussefi	Abdelali
Mohtaram	Saad

Union (2)

- Commutativité

$[R1 \cup R2] = [R2 \cup R1]$

- Associativité

$[(R1 \cup R2) \cup R3] = [R2 \cup (R1 \cup R3)]$

Attention :

L'union nécessite de travailler avec des tables de même degré

$degré(R1 \cup R2) = degré(R1) = degré(R2)$

Intersection (1)

- L'union travaille sur deux relations R et S et définit une nouvelle relation qui contient tous les tuples présents à la fois dans R et S

Syntaxe :

Table1 ∩ Table2

Exemple :

Res = Professeur ∩ Etudiant

Table : Professeur

Nom	Prenom
Boukili	Driss
Rziza	Sara
El ghorfi	Rabii

Table : Etudiant

Nom	Prenom
Elyoussefi	Abdelali
Rziza	Sara
Mohtaram	Saad

Table : Res

Nom	Prenom
Rziza	Sara

Intersection (2)

- Commutativité

$[R1 \cap R2] = [R2 \cap R1]$

- Associativité

$[(R1 \cap R2) \cap R3] = [R2 \cap (R1 \cap R3)]$

Attention :

L'intersection nécessite de travailler avec des tables de même degré

$degré(R1 \cap R2) = degré(R1) = degré(R2)$

Différence (1)

- La différence travaille sur deux relations R et S et définit une nouvelle relation qui contient les tuples existants dans R et non dans S

Syntaxe :

Table1 – Table2

Exemple :

Res = Professeur – Etudiant

Table : Professeur

Nom	Prenom
Boukili	Driss
Rziza	Sara
El ghorfi	Rabii

Table : Etudiant

Nom	Prenom
Elyoussefi	Abdelali
Rziza	Sara
Mohtaram	Saad

Table : Res

Nom	Prenom
Boukili	Driss
El ghorfi	Rabii

Différence (2)

- Pas de commutativité

$[R1 - R2] \neq [R2 - R1]$

- Pas d'associativité

$[(R1 - R2) - R3] \neq [R2 - (R1 - R3)]$

Attention :

La différence nécessite de travailler avec des tables de même degré

degré(R1 – R2) = degré(R1) = degré(R2)

Produit cartésien (1)

- La produit cartésien travaille sur deux relations R et S et définit une nouvelle relation qui contient tous les tuples de R avec tous ceux de S

Syntaxe :

Table1 x Table2

Exemple :

Res = Professeur x Etudiant

Table : Tab1

attr1
a
b
c

Table : Tab2

attr2
x
a

Table : Res

attr1	attr2
a	x
a	a
b	x
b	a
c	x
c	a

Produit cartésien (2)

- Commutativité

$[R1 \times R2] = [R2 \times R1]$

- Associativité

$[(R1 \times R2) \times R3] = [R2 \times (R1 \times R3)]$

Attention :

Le produit cartésien ne nécessite pas des tables de même degré

degré(R1 x R2) = degré(R1) + degré(R2)

Jointure (1)

Opérateur jointure :

- Jointures (|X|)

La signature de cet opérateur correspond à :

relation * relation * paramètres → relation

Exemple : Client $|X|_{numero=numC}$ Vente

$Client \bowtie_{numero = numC} Vente$

Jointure entre les relations Client et Vente

Jointure (2)

- La jointure travaille sur deux relations R et S et définit une nouvelle relation qui contient le produit cartésien de R et S avec la condition p

Syntaxe :

Table1 |X|$_{p}$ Table2

- Thêta jointure : la condition p contient un opérateur de comparaison quelconque (<,>,=<,=>, =, ≠)
- Equijointure : la condition p contient un opérateur égalité (=)
- Jointure naturelle : la condition p porte sur un opérateur égalité (=) entre des attributs identiques (de même type et nom)

Jointure (3)

- La jointure naturelle retire les occurrences du même attribut

Exemple : C1 = α numero : numC (Client) // Renommage de numero en numC

Res = C1 |X|$_{numC}$ Vente

La table Res contiendra une seule occurrence de l'attribut numC

Attention :

Pour une jointure naturelle : degré(R1 |X| R2) = degré(R1) + degré(R2) - n
attr1, attr2, ..., attrn

Pour une Equijointure ou une théta jointure : degré(R1 |X|$_{p}$ R2) = degré(R1) + degré(R2)

Jointure (4)

- Afficher le nom des clients avec les dates de leurs achats

π Client.nom, Vente.date (Client |X|$_{Client.numero = Vente.numC}$ Vente)

- Afficher, pour le client numéro 125, le numéro de vente et la marque des produits achetés

π Vente.num, Produit.marque (Produit |X|$_{Produit.ref = Vente.refP}$ σ numC=125 (Vente))

Peut aussi s'obtenir en 3 étapes :

1. V1 = σ numC=125 (Vente)
2. R1 = (Produit |X|$_{Produit.ref = Vente.refP}$ V1)
3. Res = π Vente.num, Produit.marque (R1)

Jointure (5)

- Afficher la référence des produits dont le prix est supérieur au produit qui a pour référence 153

P1 = σ ref=153 (Produit) // Etape du renommage obligatoire

π Produit.ref (Produit |X|$_{Produit.prix > P1.prix}$ P1)

Peut aussi s'obtenir en 3 étapes :

1. P1 = σ ref=153 (Produit)
2. R1 = (Produit |X|$_{Produit.prix > P1.prix}$ P1)
3. Res = π Produit.ref (R1)

Division (1)

Opérateur jointure :

- Division (/)

La signature de cet opérateur correspond à :

relation * relation → relation

Attention :

La division nécessite des tables tels que les attributs de B sont contenus parmi les attributs de A

degré(A / B) = degré(A) - degré(B)

Division (2)

- La division travaille sur deux relations A et B et définit une nouvelle relation qui contient tous les tuples tels que le produit cartésien avec le diviseur soit un sous-ensemble de la relation dividende

Syntaxe :

Table1 / Table2

Exemple :

R = A / B

Table : A

attr1	attr2
a	1
a	2
a	3
b	1
c	2

Table : B

attr2
1
2

Table : R

attr1
a

Division (3)

- La division est utilisée pour répondre à des requêtes du type :

"Quels sont les références des produits achetés par tous les clients?"

R1 = π refP, numC (Vente)

R2 = π numero (Client)

Res = R1 / R2

Equivalences

- Ordre des conditions dans une sélection

σ c1 and c2 (T) ⇔ σ c1 (σ c2 (T)) ⇔ σ c2 (σ c1 (T))

- Ordre des opérations : sélection et jointure

R1 = σ c (T1)

Res = R1 |X|$_{p}$ T2

⇔

R1 = T1 |X|$_{p}$ T2

Res = σ c (R1)

Liens entre les opérateurs (1)

Opérateurs fondamentales :

- Sélection (σ)
- Projection (π)
- Union (∪)
- Différence (−)
- Produit cartésien (x)

Opérateurs déduits :

- Intersection (∩)
- Jointures (|X|)
- Division (/)

Liens entre les opérateurs (2)

- Intersection :

$R \cap S = R - (R - S) = S - (S - R)$ ou $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$

- Jointure naturelle :

Soient R (A, B#) et S (B, C) : $R \,|X|\, S = \pi_{A,B,C} (\sigma_{B = B'} (R \times \alpha_{[B : B']} (S)))$

- Thêta jointure :

Soient R (A, B) et S (C, D) : $R \,|X|_{[p]}\, S = \sigma_{[p]} (R \times S)$

- Division :

Soient R (A, B) et S (B) : $R / S = \pi_A (R) - \pi_A (((\pi_A (R)) \times S) - R)$

Complexité des opérateurs

- Sélection : σ [condition] R

Balayer la relation et tester la condition sur chaque tuple

Complexité = card (R) Taille du résultat : [0 : card (R)]

- Projection : π [Ai, Ak…] R

Balayer la relation + élimination doublons

Complexité = card (R) Taille du résultat : [0 : card (R)]

- Jointure (naturelle ou thêta) entre R et S

Balayer R et pour chaque tuple de R faire Balayer S et comparer chaque tuple de S avec celui de R

Complexité = card (R) x card (S) Taille du résultat : [0 : card (R) x card (S)]

Cours 5 : La normalisation

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

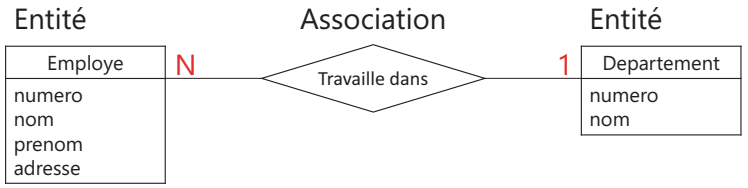
EHTP 2017-2018

Principaux axes du cours

- Passage des diagrammes E-A aux tables
- Dépendances fonctionnelles (DF)
- Fermeture d'un ensemble de DF
- Normalisation : 1NF, 2NF, 3NF, BCNF
- Normalisation : Avantages et inconvénients
- Dénormalisation

Diagrammes E-A → tables (1)

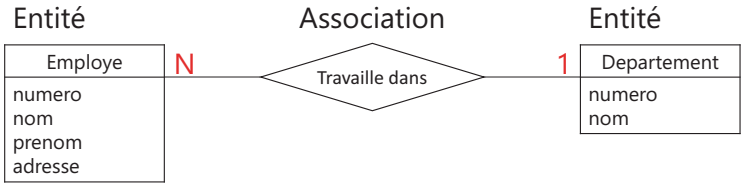
- Le modèle E-A (Entité-Association) fournit une description graphique des données



- Dans ce modèle, il existe différents types d'association entre les entités
  - L'association N-1
  - L'association N-N
  - L'association 1-1

Diagrammes E-A → tables (2)

- L'association N-1 :



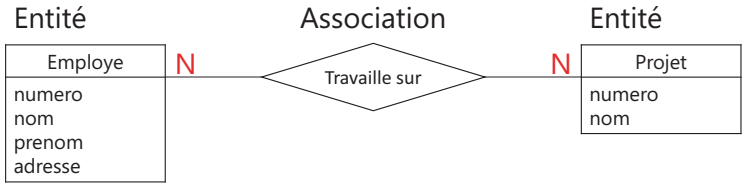
Règle de passage : Inclure la clé primaire de l'entité de cardinalité 1 dans l'autre entité de cardinalité n

Employe(numE, nom, prenom, adresse#, numD#)

Departement(numD, nom)

Diagrammes E-A → tables (3)

- L'association N-N :



Règle de passage : Créer un nouvelle table et inclure la clé primaire de chaque entité dans cette table

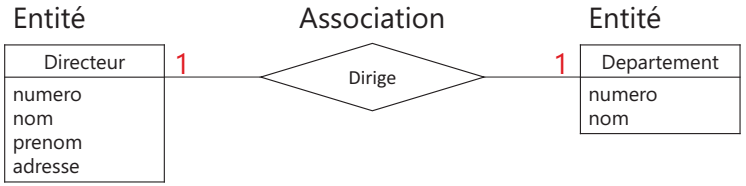
Employe(numE, nom, prenom, adresse#)

Projet(numP, nom)

EmployeProj(numE#, numP#)

Diagrammes E-A → tables (4)

- L'association 1-1 :



Règle de passage : inclure la clé primaire d'une entité dans l'autre entité **ou** combiner les deux entités

Solution 1 : Directeur(numE, nom, prenom, adresse#, numD#)
Departement(numD, nomDep)

Solution 2 : Directeur(numero, nom, prenom, adresse#, numD, nomDep)

Dépendances fonctionnelles (1)

- Une dépendance fonctionnelle concerne deux attributs ou deux groupes d'attributs notés X et Y

Syntaxe : X → Y

Notation : X détermine Y ou Y a une dépendance fonctionnelle de X

Signification : Si on connaît la valeur de X, on peut connaître celle de Y

- Exemple : K est une clé candidate de la table R, Si on a

∀ X attribut de R : K → X

Dépendances fonctionnelles (2)

- On constate la dépendance fonctionnelle :

ville → pays

Table : Adresse

numero	numRue	nomRue	ville	pays
201	120	Nakhil	Rabat	Maroc
202	7	Anfa	Casablanca	Maroc
801	3	Lapaix	Paris	France
203	8	Alfath	Rabat	Maroc

- Il est inutile de répéter le couple (Rabat, Maroc)
- L'information Ville est suffisante pour déduire Pays

Dépendances fonctionnelles (3)

- A partir des données présentes dans la table R déduire quelques dépendances fonctionnelles

Table : R

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b3	c2	d3
a3	b3	c2	d4

A → A : triviale
A → C : vrai
C → A : pas vrai
AC → D : pas vrai
AB → D : vrai (pas certain, val différentes)

Attention : l'ajout de nouveaux tuples peut tout changer

Dépendances fonctionnelles (4)

- Propriétés de base

Réflexivité :
Si Y est inclus dans X c.à.d. X = (Y, Z, …) alors X → Y

Augmentation :
Si X → Y alors WX → WY

Transitivité :
Si X → Y et Y → Z, alors X → Z

Attention : W, X, Y, Z représentent soit un attribut soit un groupe d'attributs

Dépendances fonctionnelles (5)

- Propriétés déduites

Pseudo transitivité :
Si X → Y et YW → Z, alors XW → Z

Union :
Si X → Y et X → Z, alors X → YZ

Décomposition :
Si X → YZ, alors X → Y et X → Z

Remarque : Les propriétés déduites découlent des propriétés de base

Fermeture d'un ensemble de DF (1)

- Une fermeture F+ est l'ensemble de toutes les dépendances fonctionnelles logiquement impliquées par F

- Exemple :

R = (A,B,C,D,E)

F : A → BC
CD → E
B → D
E → A

- Quelques éléments de la fermeture :

A → BC => A → B et A → C (décomposition)
A → B et B → D => A → D (transitivité)
A → C et A → D => A → CD (union)
{A → B, A → C, A → D, A → CD} ⊂ F+

- En pratique, on utilise un algorithme pour trouver tous les éléments de F+

Fermeture d'un ensemble de DF (2)

- Les clés candidates X sont les éléments de la fermeture F+ tels que X → R

Dans notre exemple : X clé candidate si X → ABCDE

- Exemple :

R = (A,B,C,D,E)

F : A → BC
CD → E
B → D
E → A

A → CD et CD → E => A → E (transitivité)
A → A, A → B, A → C, A → D et A → E => A → ABCDE (union)
E → A et A → ABCDE => E → ABCDE (transitivité)
CD → E et E → ABCDE => CD → ABCDE (transitivité)
B → D => BC → CD (augmentation)
BC → CD et CD → ABCDE => BC → ABCDE (transitivité)

- Les clés candidates trouvées sont : A, E, CD, BC

Normalisation (1)

- Processus de décomposition d'une table R en plusieurs tables R1, R2, …, Rn en se basant sur la notion de dépendance fonctionnelle
- La décomposition se fait sans perte d'information
- La jointure des tables obtenues après la normalisation est équivalente à la table décomposée $R = R1 \,|X|\, R2 \,|X|\, \ldots \,|X|\, Rn$

Objectifs :

- Éviter de stocker des données redondantes
- Simplifier la mise à jour des données
- Éviter les problèmes d'incohérences

Normalisation (2)

- Codd (mathématicien et chercheur à IBM) a proposé en 1972 trois formes normales
  - 1NF : première forme normale
  - 2NF : deuxième forme normale
  - 3NF : troisième forme normale
- Puis en 1974, Codd a proposé en collaboration avec Boyce une nouvelle forme normale
  - BCNF : forme normale de Boyce-Codd (3.5NF)

Normalisation (3)

- Avec le temps, d'autres formes normales sont apparues
  - 4NF : quatrième forme normale
  - 5NF : cinquième forme normale
  - PJNF : forme normale à projections jointives
  - DKN : forme normale à domaines-clés
- Ces formes normales traitent de cas particuliers non résolus par les autres formes normales

Normalisation 1NF (1)

- Une relation est en 1NF si tous les attributs sont atomiques

Table : Personne

numero	nom	prenom	age	adresse
1001	Loudad	Yassine	20	120, Nakhil, Rabat, Maroc
1002	Labrini	Fayrouz	30	7, Anfa, Casa, Maroc
1003	Tarek	Laamiri	32	8, Alfath, Rabat, Maroc

adresse n'est pas atomique

Table : Personne

numero	nom	prenom	age	numA
1001	Loudad	Yassine	20	201
1002	Labrini	Fayrouz	30	202
1003	Tarek	Laamiri	32	203

Table : Adresse

numA	numRue	nomRue	ville	pays
201	120	Nakhil	Rabat	Maroc
202	7	Anfa	Casa	Maroc
203	8	Alfath	Rabat	Maroc

Normalisation 1NF (2)

- Une relation est en 1NF si tous les attributs sont atomiques

Avant la normalisation :

Personne (numero, nom, prenom, age, adresse)

Résultat de la normalisation 1NF :

Personne (numero, nom, prenom, age, numA#)

Adresse (numA, numRue, nomRue, ville, pays)

Normalisation 2NF (1)

- Une relation est en 2NF si elle est en 1NF **et** si il n'y a pas de DF entre un sous-ensemble de la clé et des attributs non-clés

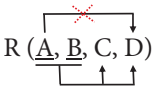


Table : AdressePersonne

numP	numA	taxeHab	numId
1001	201	4000	AE33633
1002	202	5000	AA11334
1003	203	3000	FJ45373

numP → numId

DF entre un sous ensemble de la clé et un attribut

Table : AdressePersonne

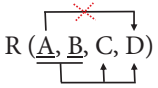
numP	numA	taxeHab
1001	201	4000
1002	202	5000
1003	203	3000

Table : Personne

numP	nom	prenom	age	numId
1001	Loudad	Yassine	20	AE33633
1002	Labrini	Fayrouz	30	AA11334
1003	Tarek	Laamiri	32	FJ45373

Normalisation 2NF (2)

- Une relation est en 2NF si elle est en 1NF **et** si il n'y a pas de DF entre un sous-ensemble de la clé et des attributs non-clés



Avant la normalisation :

AdressePersonne (numP#, numA#, taxeHab, numId)

Résultat de la normalisation 2NF :

AdressePersonne (numP#, numA#, taxeHab)

Personne (numP, nom, prenom, age, numId)

Normalisation 3NF (1)

- Une relation est en 3NF si elle est en 2NF **et** si il n'y a pas de DF entre un ensemble non inclus dans la clé et des attributs non-clés

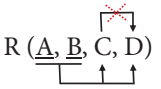


Table : Adresse

numA	numRue	nomRue	ville	pays
201	120	Nakhil	Rabat	Maroc
202	7	Anfa	Casa	Maroc
203	8	Alfath	Rabat	Maroc

ville → pays

DF entre deux attributs non-clés

Table : AdressePersonne

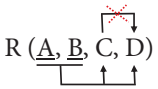
numA	numRue	nomRue	ville
201	120	Nakhil	Rabat
202	7	Anfa	Casa
203	8	Alfath	Rabat

Table : Ville

ville	pays
Rabat	Maroc
Casa	Maroc

Normalisation 3NF (2)

- Une relation est en 3NF si elle est en 2NF **et** si il n'y a pas de DF entre un ensemble non inclus dans la clé et des attributs non-clés



Avant la normalisation :

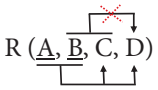
Adresse (numA, numRue, nomRue, ville, pays)

Résultat de la normalisation 3NF :

Adresse (numA, numRue, nomRue, ville#)

Ville (ville, pays)

- Remarque : En 3NF, la DF suivante n'est pas permise



Normalisation BCNF (1)

- Une relation est en BCNF si les seules DF sont celles de la clé qui détermine les attributs non-clés, par conséquent 3NF ⊂ BCNF

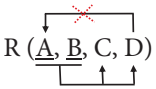


Table : CoupeDuMonde

paysOrg	annee	vainceur	villeFinale
Allemagne	1974	Allemagne	Munich
France	1998	France	Paris
Allemagne	2006	Italie	Berlin

villeFinale → paysOrg

DF différente de
clé → attribut non-clé

Table : CoupeDuMonde

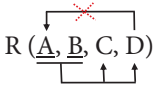
villeFinale	annee	vainceur
Munich	1974	Allemagne
Paris	1998	France
Berlin	2006	Italie

Table : VilleFinale

villeFinale	paysOrg
Munich	Allemagne
Paris	France
Berlin	Allemagne

Normalisation BCNF (2)

- Une relation est en BCNF si les seules DF sont celles de la clé qui détermine les attributs non-clés, par conséquent 3NF ⊂ BCNF



Avant la normalisation :

CoupeDuMonde (paysOrg, annee, vainqueur, villeFinale)

Résultat de la normalisation BCNF :

CoupeDuMonde (villeFinale#, annee, vainqueur)

VilleFinale (villeFinale, paysOrg)

- Remarque : En BCNF, on peut être amené à changer la clé de la relation

Avantages de la normalisation

- **Redondance** réduite
  - moins de données dans la BD (base de données)
  - plus petite est la BD et plus rapides sont les E/S (Entrées/Sorties)
- Petites relations et **petits tuples**
  - plus de tuples peuvent être affichés ou imprimés sur une page
  - plus de tuples peuvent être traités en une même E/S
  - plus de tuples peuvent être mis en mémoire cache
- **Consistance** de la base de donnée
  - Lors de la modification d'une donnée, toutes les réplications de cette donnée sont mises à jour automatiquement

Inconvénients de la normalisation

- Nombre **élevé** de relations de petites tailles
  - la performance n'est pas optimisée
  - plusieurs jointures pour retrouver l'information
  - les jointures peuvent être coûteuses en temps CPU (E/S)
- Pour cette raison, on a recours à la dénormalisation

Dénormalisation (1)

- La dénormalisation
  - s'effectue en général après la normalisation
  - vise à s'éloigner intentionnellement de la normalisation
  - a pour objectif d'améliorer les **performances** de la BD
  - peut être appliquée sur des attributs ou des relations entières
  - requiert de connaître comment les données seront utilisées
- La dénormalisation peut aussi être utilisé pour ajuster la BD à des applications particulières

Dénormalisation (2)

- Avantages de la dénormalisation
  - minimise le besoin de jointures
  - réduit le nombre de clés étrangères (Foreign Key)
  - réduit le nombre d'index
  - réduit le nombre de relations
- Le choix d'un bon schéma sera toujours un compromis entre la performance de la BD (moins de jointures) et les anomalies de mises à jour (redondance de données)
  - Le concepteur doit faire une étude sur la fréquence des mises à jours de certaines données pour justifier la redondance des données