

Chapitre 3:

L'algèbre Relationnelle

Les types d'opérations

i. Opérations de base

- Projection, Sélection, Jointure

ii. Opérations ensemblistes

- Union, Intersection
- Différence, Produit

iii. Opérations d'agrégation

- Somme, Moyenne, Minimum, comptage...

Opérations de base. Projection

(1/2)

i. Définition

Notée (π) cet opération permet d'extraire des colonnes d'une relation. Cet opérateur ne porte que sur **1** relation.

Il permet de ne retenir que **certains attributs** spécifiés d'une relation.

L'opérateur de projection doit éliminer les *duplicata*

ii. Syntaxe

$R = \text{PROJECTION} (R_1, \text{liste des colonnes})$

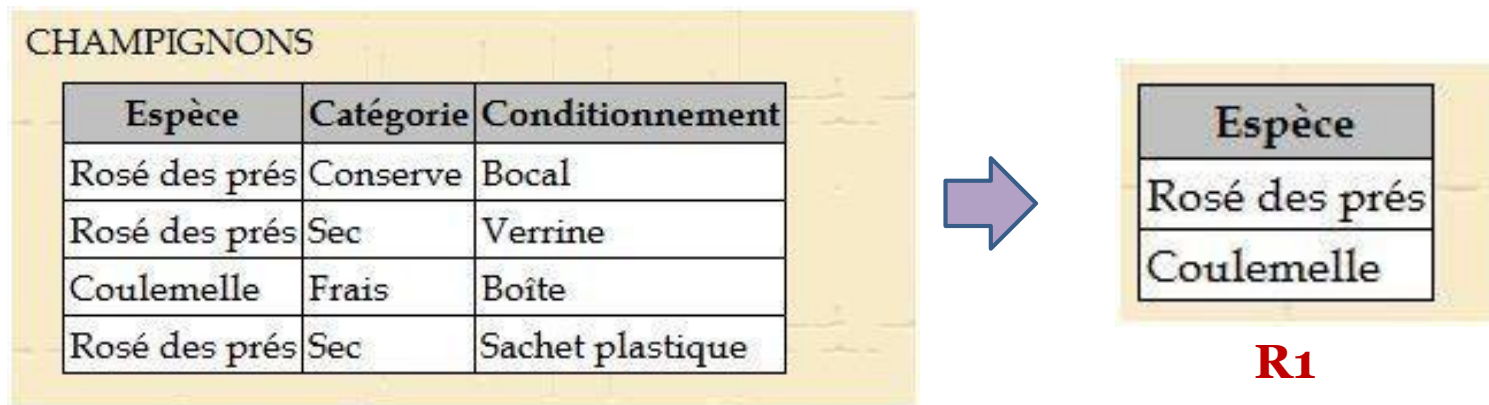
Équivalent à $\pi_{\text{colonnes}} (R_1)$

Opérations de base. Projection

(2/2)

iii. Exemple

Soit le schéma relationnel suivant :



Quelles sont les espèces enregistrées dans la tables CHAMPIGNONS?

Résultat :

R1 = PROJECTION (CHAMPIGNONS, Espèce)

Opérations de base. Sélection

(1/2)

i. Définition

Notée (σ) cet opération permet de **Sélectionner** un sous-ensemble de lignes d'une relation. (tout en conservant la totalité des colonnes)

ii. Syntaxe

$R = \text{SELECTION}(R_1, \text{condition})$

Équivalent à $\sigma_{\text{condition}}(R_1)$

Rq: une condition est exprimée à l'aide des opérateurs arithmétiques ($=$, $>$, $<$, $>=$, $<=$, $<>$) ou logiques de base (ET, OU, NON).

Opérations de base. Sélection

(2/2)

iii. Exemple

Soit le schéma relationnel suivant :

CHAMPIGNONS

Espèce	Catégorie	Conditionnement
Rosé des prés	Conserve	Bocal
Rosé des prés	Sec	Verrine
Coulemelle	Frais	Boîte
Rosé des prés	Sec	Sachet plastique



Espèce	Catégorie	Conditionnement
Rosé des prés	Sec	Verrine
Rosé des prés	Sec	Sachet plastique

R₃

Quelle est la liste des champignons dont la catégorie est 'Sec' ?

Résultat : R₃ = SELECTION (CHAMPIGNONS, Catégorie = "Sec")

Opérations de base. Jointure

(1/3)

i. Définition

Notée (\bowtie) cette opération permet d'extraire des lignes de deux relations différentes. Généralement, ce sont les Primary Key et Foreign Key qui sont utilisées.

ii. Syntaxe(s)

$$R = \text{JOINTURE} (R_1, R_2, \text{conditions})$$

Rq: - condition d'égalité entre attributs \rightarrow équijointure

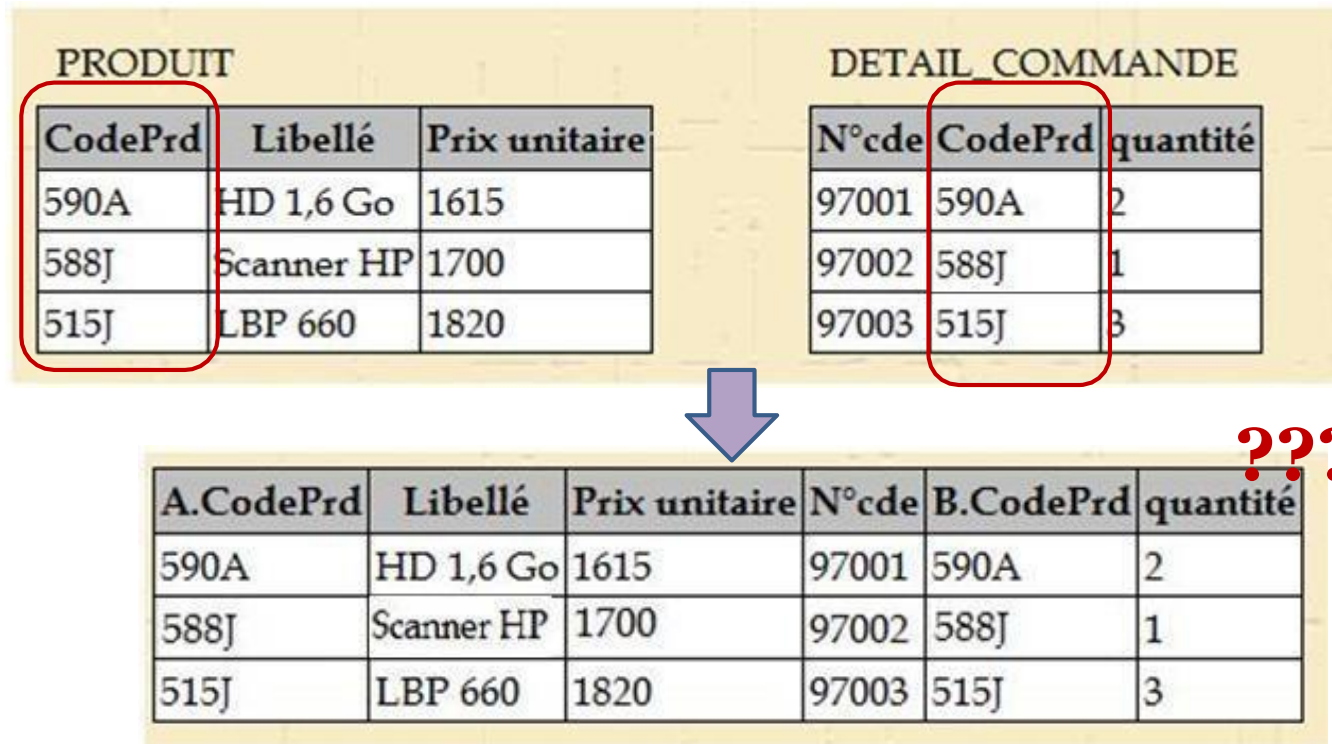
- condition d'inégalité ($<, >, <=, >=, <> \dots$) \rightarrow non équijointure

Opérations de base. Jointure

(2/3)

iii. Exemple

Soit le schéma relationnel suivant :



Opérations de base. Jointure

(3/3)

iii. Exemple – solution

A.CodePrd	Libellé	Prix unitaire	N°cde	B.CodePrd	quantité
590A	HD 1,6 Go	1615	97001	590A	2
588J	Scanner HP	1700	97002	588J	1
515J	LBP 660	1820	97003	515J	3



R = **JOINTURE** (PRODUIT, DETAIL_COMMANDE,
Produit.CodePrd=Détail_Commande.CodePrd)

Opérations ensemblistes. Union

(1/3)

i. Définition

Notée (\cup) cet opération permet de ramener toutes les lignes **distinctes** existante dans les deux relations..

ii. Syntaxe

$$R = R_1 \text{ UNION } R_2$$

Ou

$$R = \text{UNION} (R_1 , R_2)$$

Opérations ensemblistes. Union

(2/3)

iii. Exemple

Soit le schéma relationnel suivant :

E1 : Enseignants élus au CA

n° enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND

E2 : Enseignants représentants syndicaux

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN
6	MICHEL



Quelle est la liste des enseignantes qui sont élus au CA **OU** représentants syndicaux?

Opérations ensemblistes. Union

(3/3)

iii. Exemple – solution

n°enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND
6	MICHEL



$R = \text{UNION} (E1, E2)$

Rq: La relation résultat possède les attributs des relations d'origine et les n-uplets de chacune, avec élimination des doublons éventuels.

Opérations ensemblistes. Intersection (1/3)

i. Définition

Notée (\cap) cet opération permet de ramener toutes les lignes **communes** dans les deux relations..

ii. Syntaxe

$R = \text{INTERSECTION} (R_1, R_2)$

Opérations ensemblistes. Intersection

(2/3)

iii. Exemple

Soit le schéma relationnel suivant :

E1 : Enseignants élus au CA

n° enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND

E2 : Enseignants représentants syndicaux

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN
6	MICHEL



Quelle est la liste des enseignantes qui sont élus au CA **ET** représentants syndicaux?

Opérations ensemblistes. Intersection

(3/3)

iii. Exemple – solution

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN



$R = \text{INTERSECTION} (E_1, E_2)$

Remarque : La relation résultat possède les attributs des relations d'origine et les n-uplets communs à chacune.

Opérations ensemblistes. Différence

(1/3)

i. Définition

Notée (-) cet opération permet de ramener toutes les lignes de la première relation **sauf** les lignes existante dans la seconde relation.

ii. Syntaxe

$$R = \text{DIFFERENCE} (R_1, R_2)$$

Opérations ensemblistes. Différence

(2/3)

iii. Exemple

Soit le schéma relationnel suivant :

E1 : Enseignants élus au CA

n° enseignant	nom_enseignant
1	DUPONT
3	DURAND
4	MARTIN
5	BERTRAND

E2 : Enseignants représentants syndicaux

n°enseignant	nom_enseignant
1	DUPONT
4	MARTIN
6	MICHEL



Quelle est la liste des enseignantes du CA qui ne sont pas des représentants syndicaux?

Opérations ensemblistes. Différence

(3/3)

iii. Exemple – solution

n°enseignant	nom_enseignant
3	DURAND
5	BERTRAND



$$R = \text{DIFFERENCE} (E_1, E_2)$$

Rq: La relation résultat possède les attributs des relations d'origine et les n-uplets de la première relation qui n'appartiennent pas à la deuxième.

- DIFFERENCE (R1, R2) ne donne pas le même résultat que DIFFERENCE (R2, R1)

Opérations ensemblistes. **Produit** (1/3)

i. Définition

Notée (X) cet opération permet de ramener un **produit cartésien** formé par la concaténation de chaque n-uplet de la première relation avec l'ensemble des n-uplets de la deuxième relation.

ii. Syntaxe

$$R = \text{PRODUIT} (R_1, R_2)$$

Opérations ensemblistes. Produit

(2/3)

iii. Exemple

Soit le schéma relationnel suivant :

Etudiants		Epreuves	
n°étudiant	nom	libellé épreuve	coefficient
101	DUPONT	Informatique	2
102	MARTIN	Mathématiques	3
		Gestion financière	5



Examen??

n°étudiant	nom	libellé épreuve	coefficient
101	DUPONT	Informatique	2
101	DUPONT	Mathématiques	3
101	DUPONT	Gestion financière	5
102	MARTIN	Informatique	2
102	MARTIN	Mathématiques	3
102	MARTIN	Gestion financière	5

Opérations ensemblistes. Produit

(3/3)

iii. Exemple – solution

n°étudiant	nom	libellé épreuve	coefficient
101	DUPONT	Informatique	2
101	DUPONT	Mathématiques	3
101	DUPONT	Gestion financière	5
102	MARTIN	Informatique	2
102	MARTIN	Mathématiques	3
102	MARTIN	Gestion financière	5



Examen = **PRODUIT** (Etudiants, Epreuves)

Opérations d'agrégation. Définition (1/2)

- Elles sont utilisées dans les opérateurs **CALCULER** et **REGROUPER_ET_CALCULER**.
- Elles portent sur un ou plusieurs groupes de n-uplets et évidemment sur un attribut de type numérique.
 - i. Somme (attribut) : total des valeurs d'un attribut
 - ii. Moyenne (attribut) : moyenne des valeurs d'un attribut
 - iii. Minimum (attribut) : plus petite valeur d'un attribut
 - iv. Maximum (attribut) : plus grande valeur d'un attribut

Opérations d'agrégation. Définition (2/2)

v. La fonction de comptage : Comptage()

Définition:

La fonction de comptage donne le nombre de n-uplets ou enregistrements d'une relation.

$R = \text{CALCULER}(R_0, \text{fonction_agreg1}, \text{fonction_agreg2}, \dots)$ ou

$N = \text{CALCULER}(R_0, \text{fonction_agreg})$

$R = \text{REGROUPER_ET_CALCULER}(R_0, \text{att1}, \text{att2}, \dots, \text{fonction_agreg1}, \text{fonction_agreg2}, \dots)$

Chapitre 4:

SQL: Langage de définition des données

Objectifs du cours

(1/2)

1^{ère} Partie : Introduction du langage SQL

2^{ème} Partie : Présentation du langage LDD

1. Définir « LDD »
2. Les types de données

3^{ème} Partie : Création d'une table/des contraintes d'intégrités

1. Créer une table
2. Ajouter une contrainte clé primaire
3. Ajouter une contrainte clé étrangère
4. Autres contraintes (not null, check, unique)

Objectifs du cours

(2/2)

5. Activer / désactiver une contrainte

4^{ème} Partie : Manipulation de la structure de la table

1. Ajouter, modifier, supprimer une colonne/contrainte
2. Supprimer une table
3. Renommer une table

2ème Partie :

Présentation du langage LDD

Objectifs de la partie :

Définition et présentation du langage de définition de données ainsi que les différents types de données.

LDD.

Le langage de définition de données, c'est un langage qui permet de **définir** et **manipuler** les structures de données et non pas les données.

Structure de données : tout objet de la BD destiné à contenir des données : Exemple : TABLE.

Les types de données.

Types de données	Description
CHAR [(size [BYTE CHAR])]	Taille fixe comprise entre 1 et 2000
VARCHAR2 (size)	Taille Variable comprise entre 1 et 4000
NUMBER[(precision [, scale])]	Nombre ayant une précision p et une échelle s. La précision est comprise entre 1 et 38. L'échelle varie de -84 à 127
BINARY_FLOAT	32-bit nombre avec virgule flottante. C etype nécessite 5 octets
BINARY_DOUBLE	64-bit nombre avec virgule flottante. C etype nécessite 9 octets
LONG	Données caractères ayant une taille <= 2GO
DATE	Date comprise entre 1/1/4712 AJC et 31/12/999 APJC
TIMESTAMP	Année, mois, jour , heure, minute et seconde, fraction de seconde
INTERVAL YEAR(year_precision) TO MONTH	Stocke une période de temps en année et mois, où year_precision est compris entre 0 et 9. La valeur par défaut est 2

3ème et 4ème Partie :

Création d'une table/contraintes d'intégrités -
Manipulation de la structure d'une table

Objectifs de la partie :

Comment créer une table et définir les contraintes d'intégrités.

Comment ajouter, modifier ou
supprimer des colonnes ou des
contraintes...

Créer une table. Syntaxe

Syntaxe :

```
CREATE TABLE <nom_de_la_table>  
  
    ( <nom_colonne1> <type_colonne1>,  
      <nom_colonne2> <type_colonne2>,    ...  
      <nom_colonne'> <type_colonne'>  
  
    );
```

Remarque : Pour créer une table il faut avoir :

- Le privilège CREATE TABLE
- Un espace de stockage

Ajouter une contrainte clé primaire - Syntaxe (1/2)

1^{er} cas : clé primaire simple

Syntaxe 1: lors de la création de la table:

```
CREATE TABLE <nom_de_la_table>
```

```
( <nom_col1 > <type_col1> CONSTRAINT <nom_contrainte> PRIMARY KEY,
```

```
<nom_col2 > <type_col2>, ...
```

```
<nom_coln > <type_coln> );
```

→ Contrainte niveau colonne.

Syntaxe 2: après la création de la table :

```
ALTER TABLE <nomtable> ADD Constraint <nom_contrainte> Primary key  
(nom Col)
```

Ajouter une contrainte clé primaire - Syntaxe (2/2)

2ème cas : clé primaire composée

Syntaxe 1: lors de la création de la table:

```
CREATE TABLE <nom_de_la_table>  
  
( <nom_col1 > <type_col1>,  
  <nom_col2> <type_col2>,  
  ...  
  <nom_coln> <type_coln>,  
  Constraint <nom_contrainte> Primary key (nomCol1, nomCol2...) );
```

→ Contrainte niveau Table

Syntaxe 2: après la création de la table :

```
ALTER TABLE <nom_de_la_table> ADD constraint <nom_contr> Primary key  
(nomCol1,nomcol2...)
```

ALTER TABLE <nom_de_la_table> **ADD Constraint**
 <nom_contrainte> **PRIMARY KEY** (nom_colonne(s))

```
ALTER TABLE Etudiants ADD Constraint pk_nom_prenom PRIMARY KEY
(nom,prenom) ;
```

Type d'objet TABLE Objet ETUDIANTS

Table	Column	Type De Donnees	Longueur	Précision	Echelle	Clé Primaire	Valeur Nullable	Valeur Par Défaut	Commentaire
ETUDIANTS	NETUDIANT	Number	-	-	-	-	✓	-	-
	NOM	Varchar2	10	-	-	1	-	-	-
	PRENOM	Varchar2	10	-	-	2	-	-	-

[illegible]

Ajouter une contrainte clé étrangère.

Syntaxe :

```
ALTER TABLE <nom_de_la_table> ADD constraint <nom_contrainte>  
FOREIGN KEY (nom_cols) references <table_référencée>  
(nom_cols) ;
```

Create table Etud as select * from Etudiants;

Alter table Etud ADD constraint pk_nom_prenom PRIMARY KEY (nom,prenom) ;

```
ALTER TABLE Etud ADD constraint fk_etud_etudiants FOREIGN KEY  
(nom,prenom) references Etudiants (nom,prenom) ;
```

Ajouter une contrainte <NOT NULL>.

Cette contrainte est définie :

- Soit lors de la création de la table, syntaxe :

```
<nom_colonne> <type_colonne> constraint <nom_de_contrainte> NOT NULL
```

- Soit après la création de la table, syntaxe :

```
Alter table <nom_tab> modify <nom_col> constraint <nom_de_contr> NOT NULL
```

La contrainte NOT NULL ne peut être définie qu'au niveau de la colonne, pas au niveau de la table

Ajouter une contrainte <UNIQUE> - Définition

Une **contrainte d'intégrité** de type clé **unique** exige que chaque valeur dans une colonne ou dans un ensemble de colonnes constituant une clé soit unique.

Remarque : Une contrainte unique autorise la valeur NULL à moins que vous définissiez des contraintes NOT NULL

Ajouter une contrainte <UNIQUE> - Syntaxe

Cette contrainte est définie :

- Soit lors de la création de la table, syntaxe :

```
<nom_colonne> <type_colonne> constraint <nom_de_contrainte> UNIQUE
```

- Soit après la création de la table, syntaxe :

```
Alter table <nom_table> add constraint <nom_de_contrainte> UNIQUE  
(<nom_colonne>)
```


Ajouter une contrainte <CHECK>.

La contrainte Check définit une condition que chaque ligne doit vérifier

Cette contrainte est définie :

- Soit lors de la création de la table, syntaxe :

```
<nom_colonne> <type_col> CONSTRAINT <nom_de_contr> CHECK (condition)
```

- Soit après la création de la table, syntaxe :

```
Alter table <nom_table> ADD CONSTRAINT <nom_de_contr> CHECK (condition)
```

Gérer des colonnes et/ou des contraintes - Syntaxes

Syntaxe générale :

ALTER TABLE <nom_table>

- + ADD
- + MODIFY
- + DROP

<Nouvelle définition de la colonne>

<Nom_colonne>

+ ADD CONSTRAINT

<Nouvelle définition de la contrainte>

- + DROP CONSTRAINT
- + ENABLE CONSTRAINT
- + DISABLE CONSTRAINT

<Nom_contrainte>

Ajouter une/plusieurs colonne(s). Exemple

Ajouter 2 colonnes à la table « Etudiants »

```
ALTER TABLE Etudiants ADD (telephone number ,email varchar2(50) );
```

Type d'objet	TABLE	Objet	ETUDIANTS			
Table	Column	Type De Données	Longueur	Précision	Echelle	Clé Primaire
<u>ETUDIANTS</u>	<u>NETUDIANT</u>	Number	-	-	-	-
	<u>NOM</u>	Varchar2	10	-	-	1
	<u>PRENOM</u>	Varchar2	10	-	-	2
	<u>TELEPHONE</u>	Number	-	-	-	-
	<u>EMAIL</u>	Varchar2	50	-	-	-

Modifier une colonne. Exemple

Syntaxe :

ALTER TABLE <nom_de_la_table> **MODIFY** <nouvelle définition de la colonne>

ALTER TABLE Etudiants **MODIFY** (email varchar2(100));

Type d'objet	TABLE	Objet	ETUDIANTS			
Table	Column	Type De Données	Longueur	Précision	Echelle	Clé Primaire
<u>ETUDIANTS</u>	<u>NETUDIANT</u>	Number	-	-	-	-
	<u>NOM</u>	Varchar2	10	-	-	1
	<u>PRENOM</u>	Varchar2	10	-	-	2
	<u>TELEPHONE</u>	Number	-	-	-	-
	<u>EMAIL</u>	Varchar2	100	-	-	-

Activer/Désactiver une contrainte.

Syntaxe :

```
ALTER TABLE <nom_de_la_table>  
ENABLE | DISABLE constraint <nom_de_la_contrainte>;
```

Supprimer une table. DROP / TRUNCATE

Syntaxe :

```
DROP TABLE <nom_de_la_table> ;
```

Remarque:

On a aussi la commande « TRUNCATE » qui permet de vider la table

```
TRUNCATE TABLE <nom_de_la_table> ;
```

Renommer une table.

Syntaxe :

```
RENAME <ancien_nom_table> TO <nouveau_nom_table> ;
```


Chapitre 5:

SQL : Langage de Manipulation des données

Objectifs du cours

1. Définir le langage « LMD »
2. Insertion des données
3. Mise à jour des données
4. Suppression des données

Définir le langage LMD

LMD : **L**angage de **M**anipulation des **D**onnées (DML, *Data Manipulation Language*)

- i. Il Permet la manipulation des tables, soit l'insertion, la mise à jour et la suppression des données.
- ii. Il est composé de 3 Requêtes :

UPDATE, INSERT, DELETE.

Insertion des données - Syntaxe IMPLICITE

(1/3)

1^{er} cas: Insertion dans toutes les colonnes

INSERT INTO <nom_de_la_table> **VALUES**

(<valeur_colonne1>,
 <valeur_colonne2>, ...
 <valeur_colonne'>
);

Remarque: le nombre de valeurs de colonnes doit être égal au nombre de colonne de la table et conformément à leurs types

Insertion des données - Syntaxe EXPLICITE

(2/3)

2ème cas: Insertion dans quelques colonnes

INSERT INTO <nom_de_la_table>

(<colonne1> ,

<colonne2> ,

<colonne3>)

VALUES

(<valeur_colonne1> ,

<valeur_colonne2> ,

<valeur_colonne3>);

Insertion des données - Remarques

(3/3)

- i. Les valeurs à ajouter doivent vérifier **les contraintes** définies au moment de la définition des données. Tout enregistrement ne vérifiant pas les contraintes sera rejeté.
- ii. Les champs ayant été créés avec la contrainte **Not Null** devront, obligatoirement, avoir des valeurs.
- iii. Insertion à partir d'une autre table :

INSERT INTO nom_table [(les_champs_de_la_table)] Requête;

Insertion des données - Exercice - Enoncé

(1/2)

Soit le schéma relationnel suivant :

CLIENTS (CodeClient, NomClient, AdrClient, TelClient)

COMMANDES (NumCommande, Date, CodeClient#)

Questions:

1. Créer les 2 tables avec les contraintes d'intégrité (clés primaires et étrangères)

Insertion des données - Exercice - Enoncé (2/2)

2. Insérer les lignes suivantes:

CLIENTS

CodeClient	NomClient	AdrClient	TelClient
Co1	Bensalem Ali	3 rue tunis 2080 tunis	71123800
Co2	Toumi salma	-	71129870

COMMANDES

NumCommande	Date	CodeClient
110	03/04/2011	Co1
111	09/07/2011	Co2

Mise à jour des données

(1/2)

Syntaxe:

```
UPDATE <nom_table> SET <colonne_a_modifié> = <nouvelle_valeur>  
WHERE ( <condition_de_mise_à_jour> );
```

Exemple:

Le client Bensalem Ali vient de changer d'adresse et habite avec le client Toumi salma à « 15 rue Basra Bizerte 7000 », veuillez mettre à jour les deux adresses.

Mise à jour des données - Remarques (2/2)

- i. Il n'est pas possible de mettre à jour plus qu'une table à la fois.
- ii. La modification des données n'est pas autorisée que si les contraintes sont toujours **valides**.
- iii. Les valeurs peuvent être des **constantes**, des **expressions**, des **résultats** de sous-requêtes ou **NULL** (pour supprimer la valeur initiale du champ).
- iv. Si la clause WHERE n'apparaît pas dans la commande, il s'agit de mettre à jour tous les enregistrements de la table avec **la même valeur**.

Suppression des données

Syntaxe:

```
DELETE FROM <nom_table> WHERE ( <conditions> );
```

Chapitre 6:

SQL : Langage d'interrogation des données

Objectifs du cours.

1. Définir le LID
2. Savoir formuler des Requêtes simples « SELECT »
3. Distinguer les fonctions mono lignes/multi lignes
4. Définir les Jointures, les Opérateurs ensemblistes et les sous-interrogations

1ère Partie :

Les Requêtes simples « la clause SELECT »

Objectifs de la partie :

Définir «LID», savoir sélectionner des colonnes avec ou sans doublons, avec ou sans des conditions connaître les Expressions arithmétiques, la Valeur « NULL », les Alias de colonne, les opérateurs logique et ceux de comparaison.

LID.

(1/2)

Le langage LID permet, comme son nom l'indique, d'interroger une BD. Il sert à **rechercher**, **extraire**, **trier**, mettre en forme des données et **calculer** d'autres données à partir des données existantes.

La structure de base d'une interrogation est formée des 3 clauses suivantes :

SELECT *|<liste champ(s)> **FROM** <nom_table>
WHERE <condition(s)> ;

LID.

(2/2)




- La clause SELECT : indique les colonnes à récupérer. (ou encore des champs projetés).
- La clause FROM indique le nom de la ou des table(s) impliquée(s) dans l'interrogation.
- La clause WHERE correspond aux conditions de sélection des champs.
- Chaque nom de champ ou de table est séparé par une virgule.

Sélection des colonnes. Toutes les colonnes

SELECT * → Affiche tous les champs de la table

FROM employees;

Résultat:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
100	Steven	King	SKING	 515.123.4567 	17/06/87	AD_PRES
101	Neena	Kochhar	NKOCHHAR	 515.123.4568 	21/09/89	AD_VP
102	Lex	De Haan	LDEHAAN	 515.123.4569 	13/01/93	AD_VP
103	Alexander	Hunold	AHUNOLD	590.423.4567	03/01/90	IT_PROG
104	Bruce	Ernst	BERNST	590.423.4568	21/05/91	IT_PROG
105	David	Austin	DAUSTIN	590.423.4569	25/06/97	IT_PROG

Sélection des colonnes. Une/plusieurs colonnes

```
SELECT first_name, last_name  
FROM employees;
```

Affiche les champs :
« last_name » et
« first_name »

Résultat:

FIRST_NAME	LAST_NAME
Ellen	Abel
Sundar	Ande
Mozhe	Atkinson
David	Austin
Hermann	Baer
Shelli	Baida
Amit	Banda
Elizabeth	Bates

Expressions arithmétiques. Définition

- i. Une expression contient des données de type **NUMBER**, **DATE** et des **opérateurs arithmétiques**.

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

```
SELECT min_salary + max_salary , min_salary - max_salary,  
        min_salary * max_salary, min_salary / max_salary  
  
FROM jobs;
```

[illegible]

Expressions arithmétiques. Exemple 2

Ecrire la requête qui permet d'afficher le nom, prénom de tous les employés ainsi que leur salaire annuel avec une augmentation de 50 dinars par mois.

FIRST_NAME	LAST_NAME	Salaire Annuel
Steven	King	288600
Neena	Kochhar	204600
Lex	De Haan	204600
Alexander	Hunold	108600
Bruce	Ernst	72600
David	Austin	58200
Valli	Pataballa	58200
Diana	Lorentz	51000

Valeur «NULL». Définition

- NULL est une valeur qui n'est pas **disponible, non affectée ou inconnue.**
- NULL est différent de **zéro, espace ou chaîne vide**

Exemple 1:

```
SELECT employee_id, commission_pct  
FROM employees ;
```

Résultat:

EMPLOYEE_ID	COMMISSION_PCT
100	-
101	-
102	-
103	-
104	-
105	-

Valeur «NULL». Dans les expressions arithmétique.

Exemple 2 :

```
SELECT Employee_id, (1+commission_pct)*salary  
FROM employees;
```

Résultat:

EMPLOYEE_ID	(1+COMMISSION_PCT)*SALARY
100	-
101	-
102	-
103	-
104	-
105	-

Alias de colonne.

(1/2)

- Renomme un **en-tête** de colonne
- Suit le nom de colonne
- Peut utiliser le mot clé « **as** »
- Doit **obligatoirement** être inclus entre **guillemets** s'il contient des espaces, des caractères spéciaux ou s'il y a des majuscules et des minuscules

Alias de colonne. Exemple

(2/2)

```
SELECT first_name as nom, last_name prenom  
FROM employees;
```

Résultat:

NOM	PRENOM
Ellen	Abel
Sundar	Ande
Mozhe	Atkinson
David	Austin
Hermann	Baer

```
SELECT first_name as "nom de l'employee", last_name "prenom de l'employee"  
FROM employees;
```

Résultat:

Nom De L'employee	Prénom De L'employee
Ellen	Abel
Sundar	Ande
Mozhe	Atkinson
David	Austin
Hermann	Baer

Opérateur de concaténation.

- Concatène des colonnes **et/ou** des chaînes de caractères
- Est représenté par le symbole **||**

Exemple:

```
SELECT 'le nom est ' || first_name || ', le prénom est ' || last_name as "nom et  
    prenom de l'employe"  
FROM employees;
```

Résultat:

Nom Et Prénom De L'employé
le nom est Ellen, le prénom est Abel
le nom est Sundar, le prénom est Ande
le nom est Mozhe, le prénom est Atkinson
le nom est David, le prénom est Austin

Éliminer les doublons

- le mot-clé **DISTINCT** élimine les doublons.

Exemple:

```
SELECT DISTINCT(commission_pct)  
FROM employees;
```

Résultat:

COMMISSION_PCT
-
,15
,35
,4
,3
,2
,25
,1

Restriction avec la clause « WHERE »

Syntaxe :

```
SELECT <nom_des_colonnes> FROM <nom_de_la_table >  
WHERE <conditions>;
```

Exemple: liste des employés du département 20

```
SELECT last_name , first_name FROM employees  
WHERE department_id=20;
```

Résultat:

LAST_NAME	FIRST_NAME
Hartstein	Michael
Fay	Pat

2 lignes renvoyées en 0,05 secondes

Opérateurs de comparaison.

(1/2)

OPERATEUR	DESCRIPTION
=	Egal à
<	Inférieur à
<=	Inférieur à ou égal
>	Supérieur à
>=	Supérieur à ou égale à
<> Ou !=	Différent
BETWEEN val1 AND val2	val1 <= val <= val2
In (liste de valeurs)	Valeur de la liste
LIKE (_ :un caractère, %=plusieurs caractères)	Comme
IS NULL	Correspond à une valeur NULL

Opérateurs logiques.

(2/2)

OPERATEUR	DESCRIPTION
AND	Retourne TRUE si les deux conditions sont VRAIES
OR	Retourne TRUE si au moins une des conditions est VRAIE
NOT	Inverse la valeur de la condition TRUE si la condition est FAUSSE FALSE si la condition est VRAIE

Trier avec « ORDER by ». Définition

Triez les lignes selon un ordre bien précis avec la clause ORDER BY:

- **ASC** : ordre croissant (par défaut)
- **DESC** : ordre décroissant

Remarque: La clause ORDER BY vient en dernier dans l'instruction SELECT

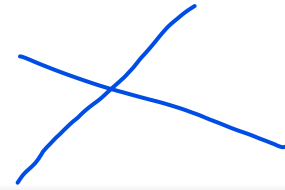
Trier avec « ORDER by ». Exemple

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date DESC;
```

Résultat:

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Kumar	SA_REP	80	21/04/00
Banda	SA_REP	80	21/04/00
Ande	SA_REP	80	24/03/00
Markle	ST_CLERK	50	08/03/00
Lee	SA_REP	80	23/02/00
Philtanker	ST_CLERK	50	06/02/00

Exercices d'applications.



(1/2)

(nous allons utilisé la table « employees » sous le schéma HR)

- Afficher la liste de tous les employés.
- Afficher le nom, le prénom, la date d'embauche de tous les employés et renommer les colonnes comme suit: « nom de l'employe », « prenom de l'employe » et « date dembauche »
- Afficher le nom et le prénom des employés triés par date d'embauche.
- Afficher les employés qui ont été embauché au cours de l'année 1990.
- Afficher la liste des employés dont le salaire est > 2800.
- Afficher la liste des employés des départements 10, 30 et 50.

Exercices d'applications.

(2/2)

- Afficher la liste des employés dont le salaire entre 4000 et 6000 du département 20
- Afficher la liste des employés dont la commission_pct est NULL.
- Afficher la liste des employés dont le nom commence par la lettre 'A'
- Afficher la liste des employés dont le prénom comporte la lettre 's'
- Afficher la liste des employés dont la deuxième lettre du nom est 'i'
- Afficher la liste des employés embauchés pendant les années entre 1986 et 1990
- Afficher la liste des employés dont le job est différent de 'CLERK' ou 'MANAGER'

2ème Partie : Les Fonctions Mono lignes

Objectifs de la partie:

Définir et savoir utiliser les :

- i. Fonctions de caractères
- ii. Fonctions de manipulation des caractères
- iii. Fonctions de manipulation des dates
- iv. Fonctions numériques
- v. Fonctions de conversions
- vi. Autres fonctions

Fonctions de caractères.

Conversion Minuscule/Majuscule(1/2)

Il existe trois fonctions principales:

- **LOWER** : convertir les caractères « M » en « m »
- **UPPER** : convertir les caractères « m » en « M »
- **INITCAP** : convertir l'initiale de chaque mot en « M » et les caractères suivants en « m »
- **Remarque:** Une fonction **mono ligne** est une fonction qui s'applique enregistrement par enregistrement.

Fonctions de caractères. Conversion Minuscule/Majuscule(2/2)

SELECT LOWER(last_name) as "prénom" FROM employees;

Prénom
abel
ande

SELECT UPPER(last_name) as "prénom" FROM employees;

Prénom
ABEL
ANDE

SELECT INITCAP(last_name) as "prénom" FROM employees;

Prénom
Abel
Ande

Fonctions de manipulation des caractères. Définitions (1/2)

- **CONCAT**_(chaine1,chaine2) : concatène 2 chaînes = ||
- **SUBSTR**_(chaine, pos, taille) : extrait une sous chaîne d'une autre chaîne
- **LENGTH**_(ch) : taille d'une chaîne en caractères
- **INSTR**_(ch,sch) : position d'une chaîne de caractères dans une autre chaîne
- **TRIM**_(ch) : élimine les espaces à gauche et à droite
- **LTRIM**_(ch) : élimine les espaces à gauche
- **RTRIM**_(ch) : élimine les espaces à droite

Fonctions de manipulation des caractères. Définitions (2/2)

- **LPAD** (chaîne, nbr, caract) : complète une chaîne de caractères sur la gauche avec une autre chaîne pour avoir n caractères.
- **RPAD** (chaîne, nbr, caract) : complète une chaîne de caractères sur la droite avec une autre chaîne pour avoir n caractères.
- **ASCII** (chaîne) : retourne le code ascii du premier caractère de la chaîne.
- **CHR** (nbr) : retourne le caractère (inverse de ascii).

Fonctions de manipulation des caractères. Exemples

```
SELECT  'station' ch1, 'agil' ch2,  
        Concat ('station 1','agil') "la station",  
        Substr ('station 1',9,1) "numéro station",  
        Length ('agil') "longueur ch1",  
        Instr ('agil','i') "position de i",  
        Length (Trim(' agil ')) "trim",  
        Length (Rtrim(' agil ')) "Rtrim",  
        Length (Ltrim(' agil ')) "Ltrim",  
        Rpad ('agil',8,'*') "RPAD", Lpad ('agil',8,'-') "LPAD",  
        ASCII ('test') " code ascii ", CHR(75)  
  
FROM  Dual;
```


Fonctions numériques. Round

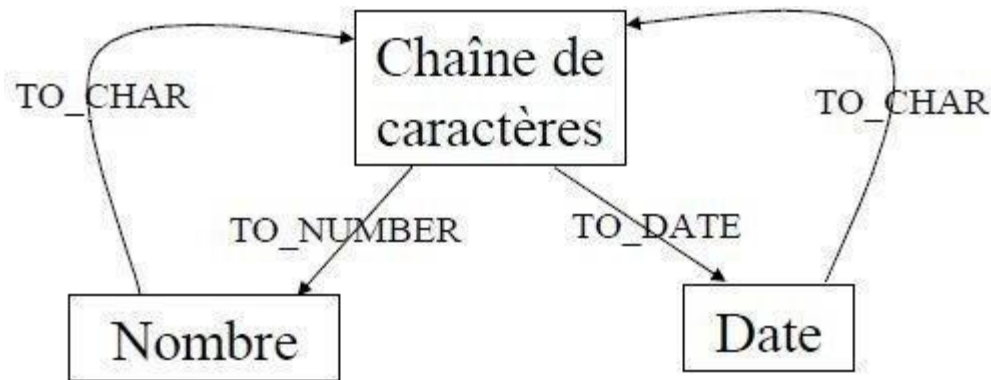
(1/2)

```
SELECT round(123.646, 0) "0 chiffres" , round(123.646, 1) "1 chiffres",  
        round(123.646, 2) "2 chiffres", round(123.646, 3) "3 chiffres"  
FROM Dual;
```

Résultat:

0 Chiffres	1 Chiffres	2 Chiffres	3 Chiffres
124	123.6	123.65	123.646

Fonctions de conversions.



- **TO_DATE :** Convertir une Chaîne en format Date
- **TO_NUMBER :** Convertir une Chaîne en format Numérique
- **TO_CHAR :** Convertir une Date /un nombre en une Chaîne

Fonctions de conversions.

Options avec To_char

OPTION	DESCRIPTION
yyyy	l'année (quatre chiffres)
month	nom complet du mois en minuscule
mon	abréviation du nom du mois en minuscule (trois caractères)
day	nom complet du jour en minuscule
DD	jour du mois (01-31)
D	jour de la semaine (de 1 à 7, dimanche étant le 1)
dy	abréviation du nom du jour en minuscule (3 caractères)
ddd	jour de l'année (001-366)
q	trimestre
w	numéro de semaine du mois (de 1 à 5) (la première semaine commence le premier jour du mois.)
ww	numéro de semaine dans l'année (de 1 à 53) (la première semaine commence le premier jour de l'année.)

Autres Fonctions.

(1/5)

- **NVL (nom_col, valeur) :** remplace une valeur nulle
- **NVL2 (expr, val1, val2) :** si expr n'est pas nulle alors elle est remplacée par val1 sinon par val2.
- **NULLIF (val1, val2) :** si val1= val2 la valeur NULL est retournée sinon val1
- **Case :** évalue une liste de conditions et retourne un résultat parmi les cas possibles

Autres Fonctions.

(2/5)

- Ecrire les requêtes SELECT qui permettent de:
 - Remplacer « `commission_pct` » par 0 au niveau de la table « employees » si elle est null
 - Remplacer « `commission_pct` » par 0 au niveau de la table « employees » si elle est null sinon par 1 ajoutant une colonne « nom_departement » qui affiche :
- Afficher la liste des employés en
 - Si `department_id=10` , `nom_departement='depart_10'`
 - Si `department_id=20` , `nom_departement='depart_20'`
 - Si `department_id=30` , `nom_departement='depart_30'`
 - ...

Autres Fonctions.

(3/5)

- **ROW_NUMBER** : retourne le numéro séquentiel d'une ligne d'une partition d'un ensemble de résultats, en commençant à 1 pour la première ligne de chaque partition. **Ne prend pas en considération les doublons**
- **RANK**: retourne le rang de chaque ligne au sein de la partition d'un ensemble de résultats. **Compte les doublons mais laisse des trous**
- **DENSE_RANK** : retourne le rang des lignes à l'intérieur de la partition d'un ensemble de résultats, sans aucun vide dans le classement.
Compte les doublons mais ne laisse pas des trous

Autres Fonctions.

(4/5)

SELECT

**Row_number() over (partition by <col x> order by <col_y>
DESC/ASC), Col1,...colN**

FROM <nom_table> ;

A retenir:

- La clause **<order by>** est obligatoire.
- La clause **<partition by>** est facultative, est utilisée pour faire un ordre par ensemble de lignes selon **<colx>**.
- **Rank** et **dense_rank** ont la même syntaxe.

Autres Fonctions.

(5/5)

Ecrire les requêtes SELECT qui permettent de :

- Afficher la liste des employees numérotés par « department_id ».
- Afficher la liste des employees numérotés par département et selon le salaire **décroissant** :
 - Utiliser « row_number »
 - Utiliser « rank »
 - Utiliser « dense_rank »

3ème Partie :

Les Fonctions Multi lignes (ou fonctions de groupe)

Objectif du cours :

Définir la clause « GROUP BY »,
la clause « HAVING »

Fonctions Multi lignes. Définitions

(1/3)

Une fonction **mono ligne** est une fonction qui s'applique enregistrement par enregistrement.

Une fonction **multi ligne** ou fonction de groupe s'applique sur un groupe d'enregistrements et donne un résultat par groupe.

- Clause **GROUP BY** : Définit le critère de groupement pour la fonction
- Clause **HAVING** : Permet de mettre des conditions sur les groupes d'enregistrements

Fonctions Multi lignes. Définitions

(2/3)

- **COUNT** : Nombre de lignes,
- **SUM** : Somme des valeurs,
- **AVG** : Moyenne des valeurs,
- **MIN** : Minimum,
- **MAX** : Maximum,
- **STDDEV** : Ecart type,
- **VARIANCE** : Variance

Fonctions Multi lignes. Exemple1

```

SELECT    COUNT (salary) as count,
          Trunc (SUM (salary),2) as sum,
          MIN (salary) as min,
          MAX (salary) as max,
          Trunc (AVG (salary),2) as avg,
          Trunc (VARIANCE (salary),2) as variance,
          Trunc (STDDEV(salary),2) as ecart
FROM employees;

```

Résultat:

COUNT	SUM	MIN	MAX	AVG	VARIANCE	ECART
107	691400	2100	24000	6461,682	15283140,539	3909,365

Fonctions Multi lignes. Syntaxe de la clause GROUP BY

SELECT <colonnes>, <fonction de groupe>

FROM table

WHERE <conditions>

GROUP BY <col> | <expr>

HAVING <conditions>

ORDER BY <col> | <expr>

Fonctions ~~Multi lignes.~~ Exemple2

jusqu'a ici

```
SELECT department_id,
COUNT(employee_id) "nbr employees"
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	Nbr Employees
100	6
30	6
-	1
90	3
20	2
70	1
110	2
50	45
80	34
40	1
60	5
10	1

12
Départements

```
SELECT department_id, count(employee_id) "nbr employees"
FROM employees
GROUP BY department_id
HAVING department_id < 40;
```

DEPARTMENT_ID	Nbr Employees
30	6
20	2
10	1

4ème Partie : Les Jointures

Objectifs de cette partie :

- i. Equijointure et jointure interne, Jointure naturelle
- ii. Jointure externe
- iii. Non équijointure
- iv. Produit cartésien

Les Jointures. Définition.

- Une jointure permet **d'extraire** des données de **plusieurs tables** à la fois.
- La condition de jointure peut être exprimée dans la clause « WHERE » ou « ON ».
- **Précédez** le nom de la colonne par le nom de la table lorsque celui-ci figure dans plusieurs tables

Types de Jointures.

- Equijointure ou jointure interne
- Jointure naturelle
- Jointure externe
- Non équijointure
- Produit cartésien

Equijointure. Définition

- Une jointure **interne** ou **équijointure** est une jointure avec une condition de jointure contenant **un opérateur d'égalité**.
- C'est la plus répandue, elle combine les lignes qui ont des valeurs **équivalentes** pour les colonnes de la jointure.
- Généralement dans ce type de jointure, ce sont les **Primary Key** et **Foreign Key** qui sont utilisées.

Equijointure. Syntaxe

(1/2)

```
SELECT table1.column, table2.column
```

```
FROM table1
```

```
INNER JOIN table2
```

```
ON (table1.column_name = table2.column_name);
```

Equijointure. Syntaxe

(2/2)

```
SELECT      T1.Colonne1, ...T1.ColonneN,  
            T2.Colonne1, ...T2.ColonneN  
FROM table1 T1 INNER JOIN table2 T2  
ON T1.C1=T2.C1  
WHERE <Condition(s)>
```

- ✓ **T1** : Alias de la table : table1
- ✓ **T2** : Alias de la table : table2
- ✓ **C1** : colonne permettant la jointure entre les 2 tables
(primary key + foreign key)

Équivalent à :

```
SELECT      T1.Colonne1, ...T1.ColonneN,  
            T2.Colonne1, ...T2.ColonneN  
FROM T1 ,T2  
WHERE T1.C1=T2.C1 AND <Condition(s)>
```

Equijointure. Exemple

Exemple1:

```
SELECT nomE, e.numdept, nomdept  
FROM emp e INNER JOIN dept d  
ON e.numdept=d.numdept;
```

Exemple2:

```
SELECT nomE, e.numdept, nomdept  
FROM emp e INNER JOIN dept d  
ON e.numdept=d.numdept  
WHERE e.numdept=11;
```

Jointure naturelle. Définition

- La jointure naturelle est une **équijointure**.
- La clause **NATURAL JOIN** est basée sur toutes les colonnes des deux tables portant **le même nom**.
- Elle sélectionne les lignes des deux tables dont les valeurs sont **identiques** dans toutes les colonnes qui correspondent.
- Si les colonnes portant le même nom présentent des types de données **différents**, une erreur est renvoyée.

Jointure naturelle. Syntaxe

```
SELECT colonne1, colonne2...  
FROM Table1  
NATURAL JOIN Table2  
WHERE Condition(s) ;
```



Conditions autres que la condition
de jointure

Jointure naturelle. **Exemple**

Exemple 1:

```
SELECT nomE, numdept,nomdept  
FROM emp NATURAL JOIN dept;
```

Exemple 2:

```
SELECT nomE, numdept,nomdept  
FROM emp NATURAL JOIN dept  
WHERE numdept=11;
```


Non équijointure. Définition

- Il s'agit là d'utiliser n'importe quelle condition de jointure entre deux tables, exceptée la stricte égalité. Ce peuvent être les conditions suivantes :

>	supérieur
>=	supérieur ou égal
<	inférieur
<=	inférieur ou égal
<>	différent de
IN	dans un ensemble
LIKE	correspondance partielle
BETWEEN ... AND ...	entre deux valeurs
EXISTS	dans une table

Jointure externe. Définition

- JOINTURES EXTERNES = **OUTER JOINS**.
- Une jointure externe **élargie** le résultat d'une jointure interne (INNER JOINS) et permet d'extraire des enregistrements qui ne répondent pas aux critères de jointure.
- Une jointure externe renvoie **toutes les lignes** qui satisfont la condition de jointure et retourne également une partie de ces lignes de la table pour laquelle aucune des lignes de l'autre ne satisfait pas la condition de jointure.

Jointure externe. Types de jointures externes

```
SELECT t1.column, t2.column..
```

```
FROM table1 t1
```

```
LEFT | RIGHT | FULL OUTER JOIN table2 t2 ON  
(t1.column_name = t2.column_name) ;
```

- Le sens de la jointure externe **LEFT** ou **RIGHT** de la clause **OUTER JOIN** désigne la table **dominante**.
- **FULL = INNER + LEFT + RIGHT**

Jointure externe. Gauche

Exemple:

```
SELECT nomE, e.numdept, nomdept  
FROM dept d LEFT JOIN emp e  
ON e.numdept=d.numdept
```

NOME	NUMDEPT	NOMDEPT
afia tarek	11	informatique
benflen salah	11	informatique
bensalah salma	11	informatique
hamid ali	11	informatique
benafia ahmed	11	informatique
benflen ali	21	télécommunication
benafia tarek	21	télécommunication
benaf salma	31	électromécanique
-	-	genie logiciel

Jointure externe. Droite

Exemple:

```
SELECT nomE, e.numdept, nomdept  
FROM dept d RIGHT JOIN emp e  
ON e.numdept=d.numdept
```

NOME	NUMDEPT	NOMDEPT
test test	-	-
benafia ahmed	11	informatique
hamid ali	11	informatique
bensalah salma	11	informatique
afia tarek	11	informatique
benflen salah	11	informatique
benflen ali	21	télécommunication
benafia salma	31	électromécanique
benafia tarek	21	télécommunication

Jointure externe. complète

Exemple:

```
SELECT nomE, e.numdept,nomdept
FROM dept d FULL JOIN emp e
ON e.numdept=d.numdept
```

NOME	NUMDEPT	NOMDEPT
afia terek	11	informatique
benflen salah	11	informatique
bensalah salma	11	informatique
hamid ali	11	informatique
benafia ahmed	11	informatique
benflen ali	21	télécommunication
benafia terek	21	télécommunication
bena salma	31	électromécanique
-	-	genie logiciel
test test	-	-

Produit Cartésien. Définition

On obtient un produit cartésien lorsque :

- Une condition de jointure est omise
- Une condition de jointure est incorrecte
- A chaque ligne de la table 1 sont jointes toutes les lignes de la table 2.
- Le nombre de lignes renvoyés est égal $n1 * n2$ où
 - ✓ $n1$ est le nombre de lignes de la table 1
 - ✓ $n2$ est le nombre de lignes de la table 2

Produit Cartésien. **Syntaxe**

```
SELECT « nom_colonne1 », « nom_colonne2 »...  
FROM table1 t1  
CROSS JOIN table2 t2
```

Exemple:

```
Select * from employees CROSS JOIN departments
```

→ On va avoir **107*27** lignes = 2889 lignes

Produit Cartésien. Exemple

SELECT * From emp e

INNER JOIN dept d

ON e.numdept=e.numdept

▪ OU

SELECT * From emp, dept

→ On va avoir **9*4** lignes

Jointure. Récapitulatif

Jointure interne	<pre> SELECT FROM <table gauche> [INNER]JOIN <table droite> ON <condition de jointure> </pre>
Jointure externe	<pre> SELECT FROM <table gauche> LEFT RIGHT FULL [OUTER]JOIN <table droite> ON <condition de jointure> </pre>
Jointure naturelle	<pre> SELECT FROM <table gauche> NATURAL JOIN <table droite> (USING <noms de colonnes>) </pre>
Jointure croisée	<pre> SELECT FROM <table gauche> CROSS JOIN <table droite> </pre>

5ème Partie : Les Opérateurs ensemblistes

- i. UNION
- ii. UNION ALL
- iii. INTERSECT
- iv. MINUS

Les opérateurs ensemblistes.

OPERATEUR	DESCRIPTION
INTERSECT	Ramène toutes les lignes communes aux deux requêtes
UNION	Toutes les lignes distinctes ramenées par les deux requêtes
UNION ALL	Toutes les lignes ramenées par les deux requêtes y compris les doublons
MINUS	Toutes les lignes ramenées par la première requête sauf les lignes ramenées par la seconde requête

Union(ALL). Exemple

- Créer une table d'archivage qui contient les informations des employés qui ont été embauchés après l'année 2009.
- Ecrire une requête qui va afficher tous les employés:

```
SELECT * from emp
```

```
Union (ALL)
```

```
SELECT * from emp2
```

Intersect/Minus. Exemple

Ecrire les requêtes permettant de :

- Afficher les employés qui **ont été archivés**.
- Afficher les employés qui **n'ont pas été archivés**.

```
SELECT * from emp
```

Intersect/minus

```
SELECT * from emp2 ;
```

6ème Partie :

Les Sous interrogations

- i. Sous interrogations mono lignes
- ii. Sous interrogations multi lignes

Sous interrogations. Définition

- La sous-interrogation (**requête interne**) est exécutée une **seule** fois avant la requête principale
- Le résultat de la sous-interrogation est utilisé par la requête principale (**requête externe**)
- Une sous-interrogation est utilisée dans les clauses suivantes :
 - **WHERE**
 - **HAVING**
 - **FROM**

Sous interrogations mono lignes. Définition

Une sous-interrogation peut ramener **un seul résultat** on l'appelle sous interrogation mono ligne.

Les opérateurs de comparaison **mono ligne** sont :

=, !=, <, >, <=, >=

Sous interrogations mono lignes. Exemple1

Exemple de Mise à jour avec une sous interrogation :

l'employé numéro **112** est affecté au même poste et même département que l'employé numéro **111**

```
UPDATE employee_id SET (job_id, department_id)
= ( SELECT job_id, department_id
FROM employees
WHERE employee_id=111 )
WHERE employee_id=112 ;
```

Sous interrogations mono lignes. Exemple2

Exemple de Suppression avec une sous interrogation :

Supprimer tous les employés du département 'SALES'

```
DELETE FROM employees
WHERE Department_id =
( SELECT department_id
  FROM departments
 WHERE department_name='SALES' );
```

Sous interrogations multi lignes. Définition (1/3)

- Une sous-interrogation peut ramener **plusieurs** lignes à condition que **l'opérateur de comparaison** admette à sa droite un **ensemble** de valeurs.
- Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :
 - l'opérateur **IN** et
 - les opérateurs obtenus en ajoutant **ANY** ou **ALL** à la suite des opérateurs de comparaison classique :
=, !=, <, >, <=, >=.

Sous interrogations multi lignes. Définition (2/3)

- **ANY** : la comparaison sera vraie si elle est vraie pour au moins un élément de l'ensemble (elle est donc fausse si l'ensemble est vide).
- **ALL** : la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble (elle est vraie si l'ensemble est vide).
- L'opérateur **IN** est équivalent à **= ANY**,
- L'opérateur **NOT IN** est équivalent à **!= ALL**.

Sous interrogations multi lignes. Définition (3/3)

Opérateur de comparaison	Résultat
WHERE salaire IN (1200,1600,2000,2900)	SALAIRE doit être égale à une des valeurs dans la liste
WHERE salaire NOT IN (1200,1600,2000,2900)	SALAIRE ne doit pas être égale à une des valeurs dans la liste
WHERE salaire >ANY (1200,1600,2000,2900)	SALAIRE doit être supérieur à au moins une des valeurs. Donc plus que le minimum (+ de 1200).
WHERE salaire <ANY (1200,1600,2000,2900)	SALAIRE doit être inférieur à au moins une des valeurs. Donc moins que le maximum (- de 2900).
WHERE salaire >ALL (1200,1600,2000,2900)	SALAIRE doit être supérieur au maximum de toutes les valeurs. Donc plus que le maximum (+ de 2900).
WHERE salaire <ALL (1200,1600,2000,2900)	SALAIRE doit être inférieur au minimum de toutes les valeurs. Donc moins que le minimum (- de 1200).