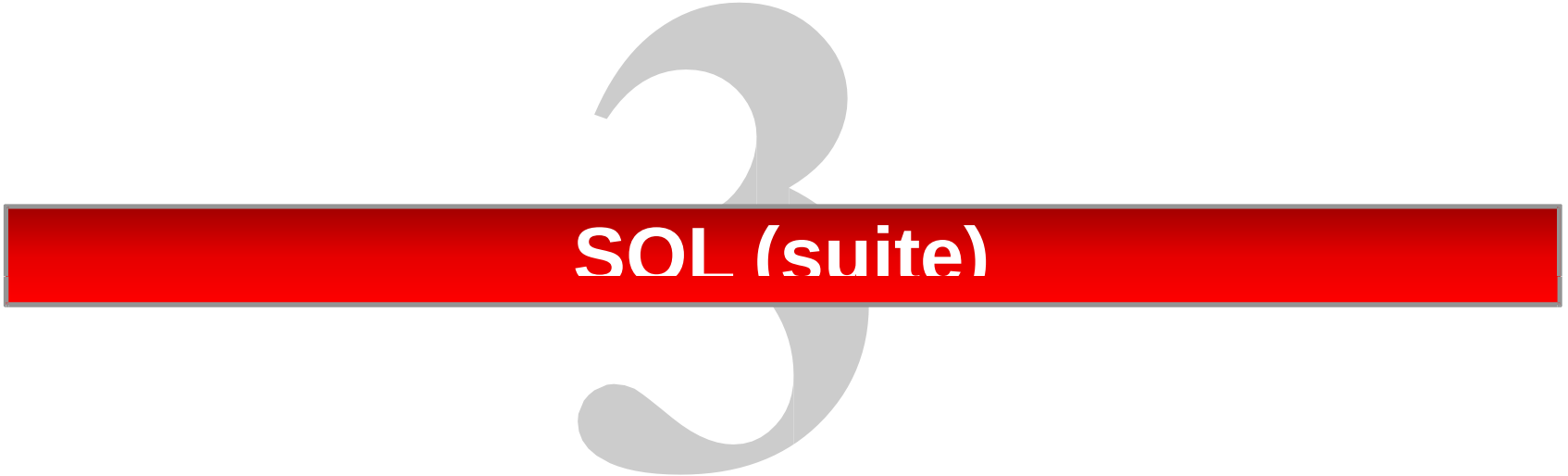
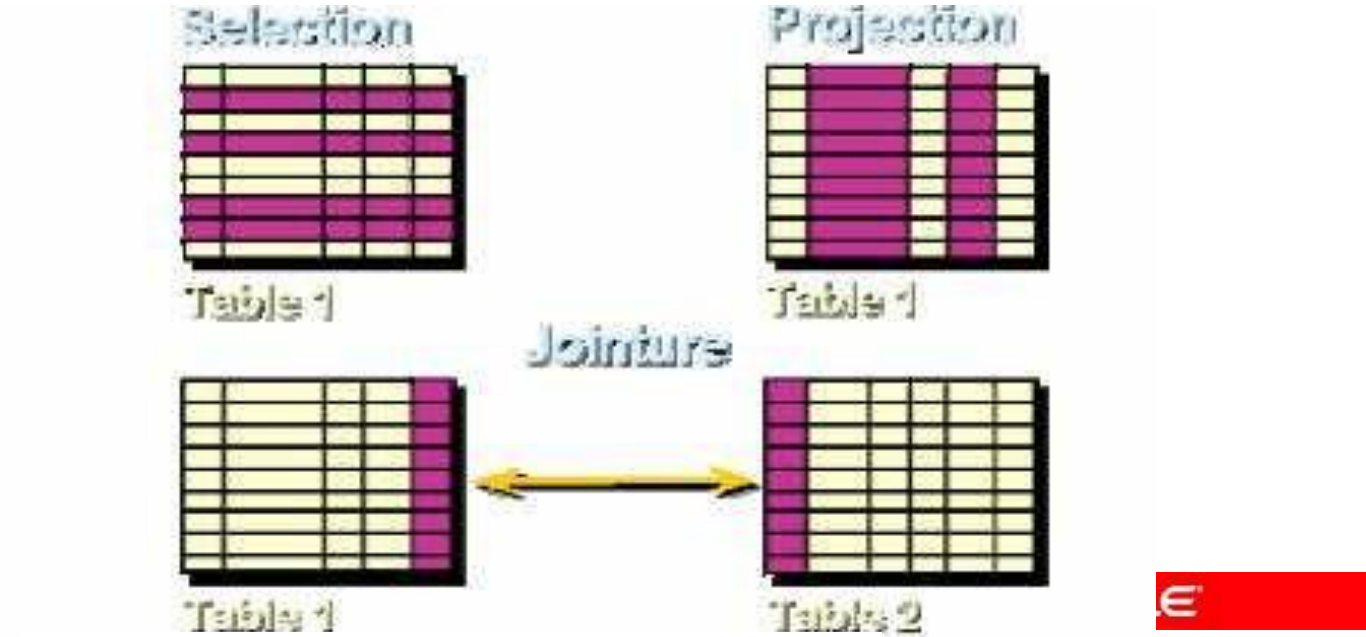

3
SQL (suite)

Ordre Select

L'ordre SELECT possède trois capacités :

- **Sélection :** Sélection d'une ou plusieurs ligne(s).
- **Projection :** Sélection d'une ou plusieurs colonne(s).
- **Jointure :** Sélection de deux colonnes dans deux tables différentes, créant ainsi une relation entre les données des deux colonnes.



Jointures de tables

❖ Produit cartésien

Se produit lorsque :

- ✓ Une condition de jointure est omise.
- ✓ Une condition de jointure est invalide.
- ✓ tous les enregistrements de la première table sont lié à tous les enregistrements de la seconde table.

Syntaxe:

SELECT * FROM *table*$_1$ [*Alias*$_1$], ..., *table*$_n$ [*Alias*$_n$],

Exemple; étudiant(30 enregistrements)

matière(10 enregistrements)

Produit cartésien=30 * 10.

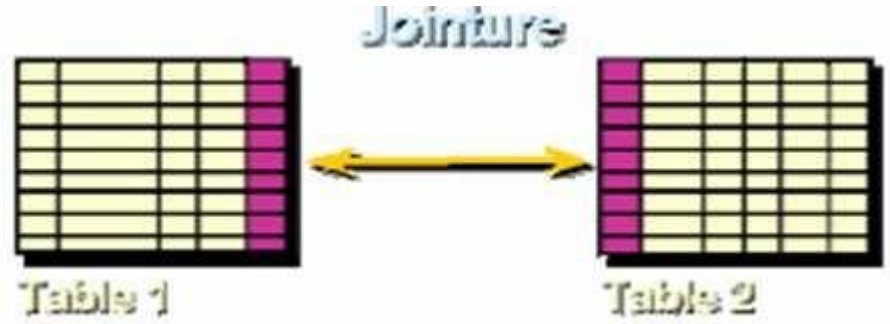
Jointures de tables

❖ Objectif

- Afficher des données issues de plusieurs tables.

  Utiliser une condition appeléejointure.

- Une condition de jointure spécifie une relation existante entre les données d'une colonne dans une table avec les données d'une autre colonne dans une autre table.
- Cette relation est souvent établie entre des colonnes définies comme clé primaire et clé étrangère.



Jointures de tables

Exemple:

Etudiant

Id_etudiant	Nom	Id_formation
E1	Karimi	F1
E2	Hachimi	F1
E3	Ahmadi	F2

Formation

Id_formation	intitulé
F1	Telecom
F2	Informatique
F3	Réseau

Id_etudiant	Nom	Id_formation	intitulé
E1	Karimi	F1	Telecom
E2	Hachimi	F1	Telecom
E3	Ahmadi	F2	Informatique

Jointures de tables

❖ **Types de jointures**

Les jointure propriétaires d'Oracle (8i et plus)	Les jointures complémentaires
Equijoin	Cross joins
Non-Equijoin	Natural joins
Outer join	Using clause
Self join	Full or two sided outer joins
	Arbitrary join conditions for outer joins

Equi-jointure

- Utilisée pour afficher des données provenant de plusieurs tables lorsqu'une valeur dans une colonne d'une table correspond directement à une valeur d'une autre colonne dans une autre table.

- Les noms des colonnes doivent être qualifiés avec le nom de la table ou l'alias de la table à laquelle elles appartiennent afin d'éviter toute ambiguïté.

Equi-jointure

➢ **Syntaxe**:

La condition de jointure doit être réalisée dans la clause **WHERE**.

SELECT *table1.colonne, table2.colonne, table3.colonne*
FROM *table1*, *table2, table3*
WHERE *table1.colonne1 = table2.colonne2*
[**AND** *table2.colonne2=table3.colonne*3**];**

➢ **Remarques:**

Pour joindre les tables dans la clause **SELECT**

- ✓ On place les noms des tables avant les noms des colonnes, suivi d'un point (.)
- ✓ Vous devez précéder les noms de colonnes par les noms de tables si les mêmes colonnes apparaissent dans les différentes tables.
- ✓ Pour joindre n tables vous devez utiliser la condition de jointure n-1.

Jointures de tables

Exercice:

Soient les relations:

Produit (prod, nomProd, fournisseur, pu**)**

Commande (cmd, prod, pu, qte, remise**)**

Dépôt(dep,prod,adr,qte**)**

1. Quels sont les numéros de commande correspondant à l'achat d'une vase?
2. Même question en utilisant des aléas de table.

Non equi-jointure

- Une condition de non équi-jointure est utilisée lorsque deux tables n'ont pas de colonnes qui correspondent directement.

Employé

Id_employé	Nom	salaire
E1	karimi	4500
E2	ahmadi	4800
E3	hachimi	3100

Grade

id_grade	Salaire_inf	Salaire_sup
1	3000	4000
2	4100	5000
3	5100	6000

Le salaire est compris entre la limite inférieure (Salaire_inf) et la limite supérieure (Salaire_sup)

Non equi-jointure

Exemple:

```
SELECT e.nom, e.salaire, g.id_grade
  FROM employe e, grade g
 WHERE e.salaire BETWEEN  g.salaire_inf AND
g.salaire_sup;
```

N.B: Les opérateurs tels que <= et >= peuvent être utilisés.

nom	salaire	Id_grade
karimi	4500	2
ahmadi	4800	2
hachimi	3100	1

Jointure externe

- La condition (outer join) est utilisée pour afficher tous les enregistrements incluant ceux qui ne respectent **pas la** condition de jointure .
- L'opérateur de jointure externe est le signe plus (+) :

❖ **Syntaxe:**

SELECT table1.column, table2.column FROM table1, table2

WHERE table1. column(+) = table2.column ;

Explication; Cette requête affiche tous les enregistrements de la colonne de table1 même s'ils ne respectent pas la condition de jointure.

Jointure externe (2)

Exemple:

```
SELECT e.enom, e.iddep, d.dnom FROM Employe e, Département d
WHERE e.iddep = d.iddep(+);
```

N.B:

- L'opérateur(+) ne peut apparaître que d'un seul côté de l'expression, le côté où il manque de l'information.
- Une condition de jointure externe ne peut pas utiliser l'opérateur **IN** et ne peut pas être liée à une autre condition par l'opérateur **OR**.

Employé

enom	iddep
karimi	1
ahmadi	2
hachimi	3

Département

iddep	dnom
1	informatique
2	vente
3	Comptabilité
4	Etudes

Aucun employé dans le département des études

Auto-Jointure

- Une condition d'auto-jointure permet de faire une jointure sur deux colonnes liées appartenant à la même table.

❖ **Syntaxe:**

```
SELECT alias1.column, alias2.column
FROM table1 alias1, table1 alias2
WHERE alais1.column = alias2.column ;
```

✓ Pour simuler deux tables dans la clause FROM on utilise les alias.

Jointure CROSS JOIN

- Une condition de jointure croisée permet de retourner chaque ligne d'une table avec chaque ligne d'une autre table.
- Identique au produit cartésien de deux tables.

❖ **Syntaxe:**

```
SELECT table1.colonne, table2.colonne
FROM table1
CROSS JOIN table2;
```

Exemple 1:

```
SELECT e.enom, d.dnom FROM employé e
CROSS JOIN département d;
```

Jointure CROSS JOIN

Exemple 2: (Tables legume et fruit)

l_id	l_nom_fr_fr	l_nom_en_gb
45	Carotte	Carott
46	Oignon	Onion
47	Poireau	Leek

f_id	f_nom_fr_fr	f_nom_en_gb
87	Banane	Banana
88	Kiwi	Kiwi
89	Poire	Pear

SELECT l_id, l_nom_fr_fr, f_id, f_nom_fr_fr FROM legume CROSS JOIN fruit

l_id	l_nom_fr_fr	f_id	f_nom_fr_fr
45	Carotte	87	Banane
45	Carotte	88	Kiwi
45	Carotte	89	Poire
46	Oignon	87	Banane
46	Oignon	88	Kiwi
46	Oignon	89	Poire
47	Poireau	87	Banane
47	Poireau	88	Kiwi
47	Poireau	89	Poire

Jointure NATURAL JOIN

- La jointure naturelle se base sur toutes les colonnes de deux tables qui possèdent le même nom et sélectionne les lignes qui possèdent les mêmes valeurs.

❖ **Syntaxe:**

```
SELECT table1.colonne1, table2.colonne2
FROM table1
NATURAL JOIN table 2;
```

Jointure NATURAL JOIN (2)

Exemple 1:

```
SELECT e.enom, d.dnom
FROM Employé e
NATURAL JOIN Département d;
```

Employé

enom	Id_dep
karimi	2
ahmadi	3
hachimi	4

Département

Id_dep	dnom
1	informatique
2	vente
3	Comptabilité
4	Etudes

enom	dnom
karimi	vente
ahmadi	Comptabilité
hachimi	Etudes

Jointure NATURAL JOIN (3)

Exemple 2: tables **utilisateur** et **pays**

user_id	user_prenom	user_ville	pays_id
1	Jérémie	Paris	1
2	Damien	Lyon	2
3	Sophie	Marseille	NULL
4	Yann	Lille	9999
5	Léa	Paris	1

pays_id	pays_nom
1	France
2	Canada
3	Belgique
4	Suisse

SELECT * FROM utilisateur NATURAL JOIN pays

pays_id	user_id	user_prenom	user_ville	pays_nom
1	1	Jérémie	Paris	France
2	2	Damien	Lyon	Canada
NULL	3	Sophie	Marseille	NULL
9999	4	Yann	Lille	NULL
1	5	Léa	Paris	France

Jointure avec la clause USING

- Certaines colonnes possèdent les mêmes noms, mais les types de données ne correspondent pas;
- la clause **USING** est utilisée pour spécifier explicitement une colonne qui sera utilisé pour faire la jointure.

NB: Pour cette clause les alias des tables ne doivent pas être utilisés;

❖ Syntaxe:

```
SELECT table1.col, table2.col
FROM table1 JOIN table 2;
USING (col_similaire);
```

Jointure avec la clause USING (2)

Exemple:

```
SELECT id_emp, enom, dnom
FROM employé
JOIN département
USING (id_dep);
```

id_emp	enom	id_dep
E1	karimi	2
E2	ahmadi	3
E3	hachimi	4

id_dep	dnom
1	informatique
2	vente
3	Comptabilité
4	Etudes

id_emp	enom	dnom
E1	karimi	vente
E2	ahmadi	Comptabilité
E3	hachimi	Etudes

Jointure avec la clause ON

- Avec la clause ON, on spécifie explicitement la condition de la jointure;

❖ **Syntaxe:**

```
SELECT table1.col, table2.col
FROM table1 JOIN table 2
On (table1.col_similaire= table2.col_similaire);
```

Jointure avec la clause ON (2)

Exemple:

```
SELECT e.id_emp, e.enom, d.dnom
FROM employé e
JOIN département d
ON (e.id_dep=d.id_dep);
```

id_nom	enom	id_dep
E1	karimi	2
E2	ahmadi	3
E3	hachimi	4

id_dep	dnom
1	informatique
2	vente
3	Comptabilité
4	Etudes

id_emp	enom	dnom
E1	karimi	vente
E2	ahmadi	Comptabilité
E3	hachimi	Etudes

Triple jointure avec la clause ON (3)

- Permet de relier trois tables.
- L'utilisation de cette clause est identique à l'utilisation des clauses WHERE et AND lorsqu'on relie plusieurs tables.

❖ **Syntaxe:**

```
SELECT table1.col1, table2.col2, table3.col3
FROM table1
JOIN table 2
On (table1.col_similaire= table2.col_similaire)
JOIN table 3
On (table2.col_similaire= table3.col_similaire);
```

Jointure: RIGHT OUTER JOIN

- Alternative de la jointure externe avec le signe plus (+) placé à droite dans la clause WHERE.

❖ **Syntaxe:**

```
SELECT table1.col1, table2.col2
 FROM table1 RIGHT OUTER JOIN table2
ON (table1.col_similaire= table2.col_similaire);
```

```
SELECT table1.col1, table2.col2 FROM table1, table2
 WHERE table1. col1 = table2.col2 (+) ;
```

Jointure: RIGHT OUTER JOIN (2)

Exemple: tables utilisateur et commande

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture FROM utilisateur **RIGHT JOIN commande** ON (utilisateur.id = commande.utilisateur_id)

id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
NULL	NULL	NULL	5	2013-03-02	A00107

Jointure: LEFT OUTER JOIN

Exemple: tables utilisateur et commande

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

SELECT id, prenom, nom,date_achat, num_facture , prix_total FROM utilisateur **LEFT JOIN** commande ON (utilisateur.id = commande.utilisateur_id)

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35
3	Marine	Prevost	NULL	NULL	NULL
4	Luc	Rolland	NULL	NULL	NULL

Jointure FULL OUTER JOIN

- Retrouve toutes les lignes de la table1 même s'il n'y a pas de correspondance dans la table2, et vice versa.

❖ **Syntaxe:**

```
SELECT table1.col1, table2.col2
FROM table1 FULL OUTER JOIN table2
ON (table1.col_similaire= table2.col_similaire);
```

Jointure FULL OUTER JOIN (2)

Exemple: tables **utilisateur** et **commande**

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture FROM utilisateur **FULL JOIN** commande ON utilisateur.id = commande.utilisateur_id

id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
4	Luc	Rolland	NULL	NULL	NULL
NULL	NULL		5	2013-03-02	A00107