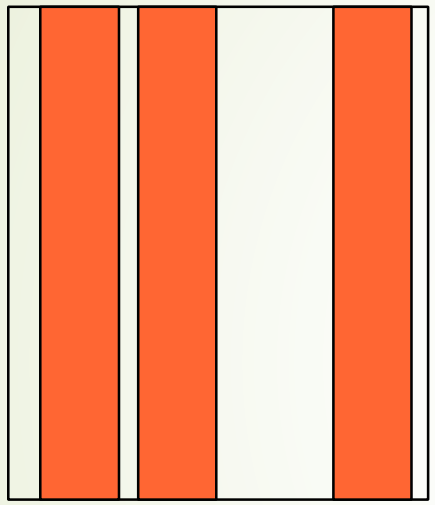
SMI-S5

SGBD ORACLE : LID-Partie 2

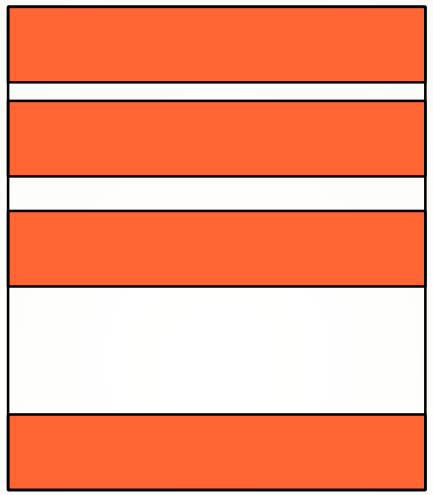
Pr. Issam QAFFOU

Laboratoire Ingénierie des Systèmes d'Information
Département d'Informatique
FSSM-UCA

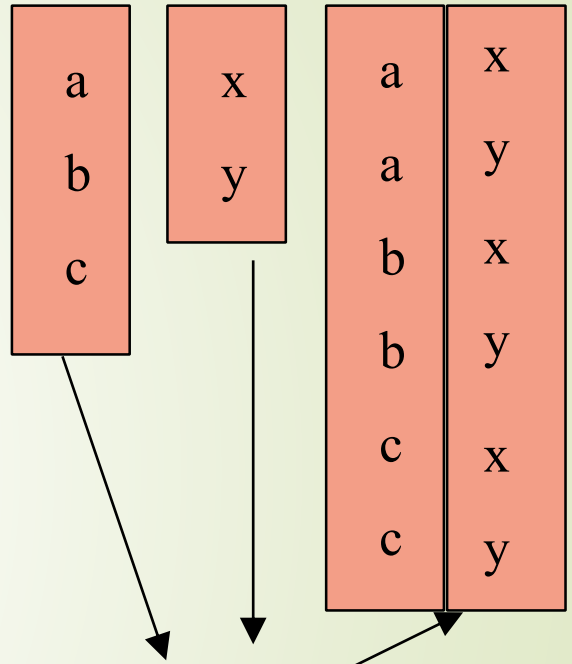
Différentes Opérations



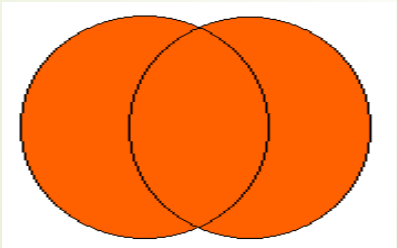
Projection



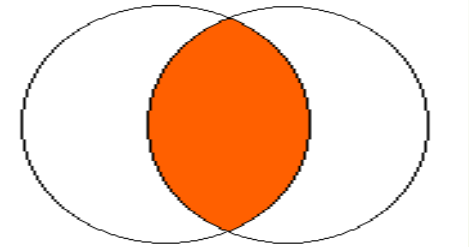
Selection



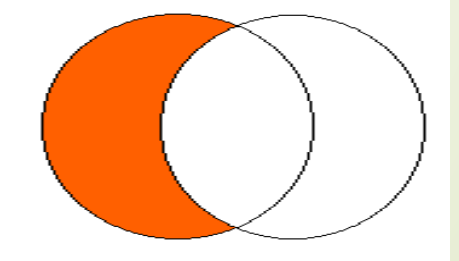
Produit cartésien



Union



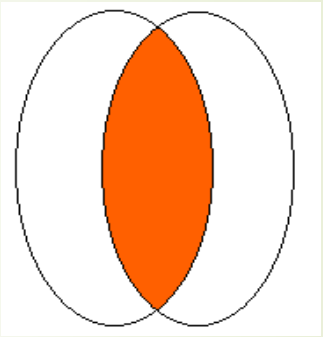
Intersection



Difference

INTERSECTION

Cet opérateur extrait des données présentes simultanément dans les deux tables.



Syntaxe :

requête1 INTERSECT requête2

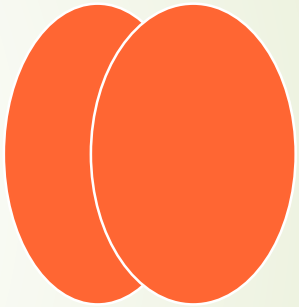
Exemple :

SELECT indip FROM segment **INTERSECT** SELECT indip FROM salle;

Cet opérateur élimine les duplicatas entre les deux tables avant d'opérer l'intersection.

UNION

L’opérateur UNION fusionne les deux tables et affiche les données des deux tables en éliminant les doublons



Syntaxe :

requête1 UNION requête2

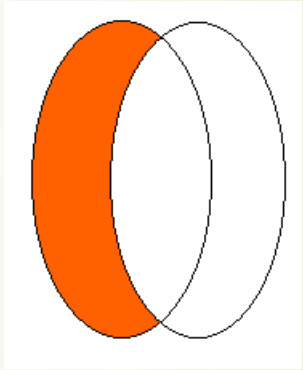
Exemple :

SELECT indip FROM segment **UNION** SELECT indip FROM salle;

RQ : Si on souhaite conserver les doublons on met **UNION ALL** au lieu de UNION

MINUS

MINUS extrait des données présentes dans une table sans être présentes dans la deuxième table.



Syntaxe :

requête1 MINUS requête2

Exemple :

SELECT indip FROM segment **MINUS** SELECT indip FROM salle;

RQ : L'opérateur MINUS est le seul opérateur ensembliste qui ne soit pas commutatif.

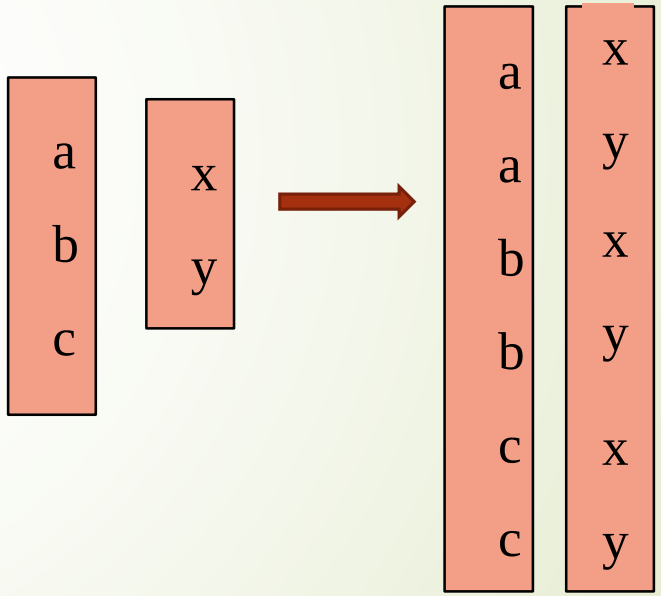
Les opérateurs ensemblistes

- Seules des colonnes de **même type** doivent être comparées avec des opérateurs ensemblistes.
- La clause ORDER BY est utilisable à la fin de la **deuxième** requête. Cette clause accepte les noms des colonnes de la **première** requête ou la **position** de ces colonnes.
- Il es possible de relier **n** requêtes avec **n-1** opérateurs ensemblistes.

Produit cartésien

Le produit cartésien entre deux tables T1 et T2 se programme sous SQL en positionnant les deux tables dans la clause FROM

Résultat : combinaison des données



Exemple :

```
SELECT NomSeg, NSalle FROM Segment , Salle;
```

Jointures (Produit cartésien)

- Les **jointures** permettent d'extraire des données issues de plusieurs tables.

- Une **jointure** met en relation deux tables sur la base d'une clause de jointure (comparaison de colonnes). Généralement, cette comparaison fait intervenir une clé étrangère d'une table avec une clé primaire d'une autre table.

Jointure relationnelle

Les jointure dites relationnelles sont les plus courantes.

Elles sont caractérisées par une seule clause FROM contenant les tables et alias à mettre en jointure deux à deux.

Syntaxe :

SELECT [*alias1.]col1, [alias2.]col2…*
FROM *nomTable1 [alias1], nomTable2 [alias2]…*
WHERE (*conditionsDeJointure);*

Jointures SQL2

Syntaxe conforme à la norme SQL2.

Syntaxe :

SELECT *listeColonnes*

FROM **nomTable1** [{ LEFT | RIGHT | FULL } [OUTER]]

JOIN *nomTable2 ON condition (colonne1 [, colonne2] ...)*

JOIN *nomTable3 ON ...*

[WHERE *condition];*

Types de jointures

- **Équijointure** : WHERE comp = compa;
- **Autojointure** : WHERE **alias1**.chefPil = **alias2**.brevet;
- **Jointure externe** : WHERE comp= compa (+) ;

Équijointure

- Une équijointure utilise l’opérateur **d’égalité** dans la clause de jointure et compare généralement des clés primaires avec des clés étrangères.

Équijointure

- Afficher la date des commandes et le nom du client qui a passé la commande.

Écriture « relationnelle »

```
SELECT a.id_clt, a.date, b.nom
FROM commande a , client b
WHERE a.id_clt= b.id_clt;
```

Écriture « SQL2 »

```
SELECT a.id_clt, a.date, b.nom
FROM commande a
JOIN client b ON (a.id_clt = b.id_clt);
```

Équijointure

Afficher les noms des produits ainsi que leurs prix et la quantité commandée pour le client « BENAZIZ »

Écriture « relationnelle »

```
SELECT c.nom, c.prix, b.quantite
FROM commande a , lignes_commande b , produit c, client d
WHERE a.num_cmd=b.commande
AND b.article=c.num_prd
AND a.client=d.num_client
AND d.nom = 'BENAZIZ' ;
```

Écriture « SQL2 »

```
SELECT c.nom, c.prix, b.quantite
FROM commande a
JOIN lignes_commande b ON a.num-cmd=b.commande
JOIN produit c ON b.article=c.num_prd
JOIN client d ON a.client=d.num_client
WHERE d.nom = 'BENAZIZ' ;
```

Remarques

- Si on oublie la clause WHERE pour une jointure on aura le produit cartésien.
- Les alias sont **obligatoires** pour des colonnes qui portent le **même nom**

Autojointure

Afficher les numéro des clients qui ont commandé en même temps ?

Solution : faire appel à la même table commande **deux fois.**

Possibilité : grâce aux **alias**

```
SELECT a.client , b.client
FROM commande a , commande b
WHERE a.client < b.client
AND a.date=c.date;
```

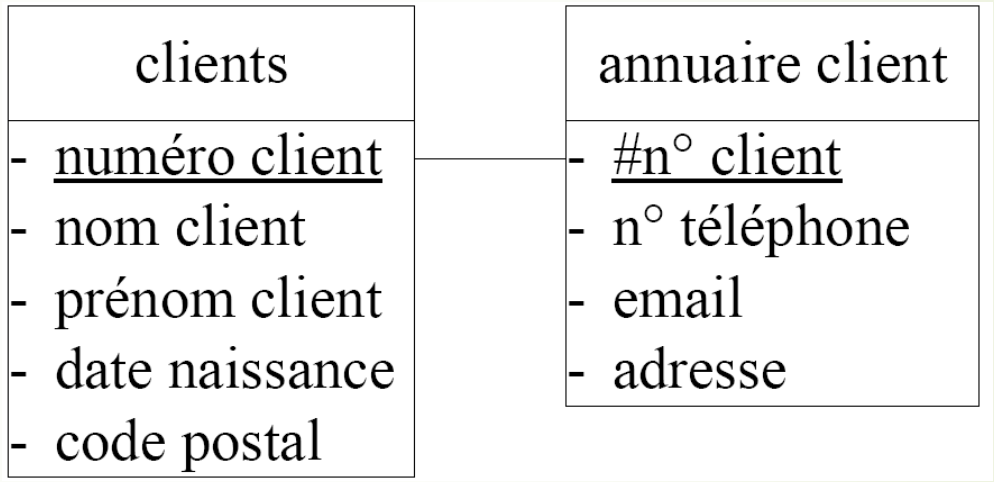
-- le signe < permet de ne pas afficher chaque couple deux fois.

Notes

- C’est un cas particulier de l’équijointure, **l’autojointure** relie une table à elle-même.
- Les **alias** sont **obligatoires** pour les autojointures.
- Lorsque la condition de jointure utilise un opérateur de type (<>, >, <, >=, <=, BETWEEN, LIKE, IN) on parle de **Inéquijointure**

Jointures externes

Problème : On souhaite afficher toutes les informations de tous les clients ?



SELECT * FROM client a, annauie_client b
WHERE a.numero=b.numero;
→ Affiche **seuls** les clients dont on connait des informations supplémentaires.

Jointures externes

Solution : On utilise les jointure Externe. Il suffit d'ajouter **le signe (+)** dans la condition de jointure du coté de la table facultative.

Écriture relationnelle

```
SELECT * FROM client a, annauie_client b
WHERE a.numero=b.numero (+) ;
```

--Affiche toutes les information relatives à tous les client et remplis les informations non fournis par des NULL

Jointures externes

Écriture SQL2 :

```
SELECT * FROM client a
LEFT OUTER JOIN annauie_client b
ON a.numero=b.numero ;
```

Autre écriture SQL2 :

```
SELECT * FROM annauie_client b
RIGHT OUTER JOIN client a
ON a.numero=b.numero ;
```

Fonctions d’agrégation

Introduction

- Les fonctions d'agrégations servent à faire des calculs arithmétiques sur les colonnes d'une table.
- Les fonctions d'agrégations qui existent pour SQL-Oracle sont:
  - **Count():** calcule combien de lignes existe pour l'attribut désigné;
  - **Avg():** calcule la moyenne des valeurs de l'attribut désigné.
  - **Max():** calcule la valeur maximale des valeurs de l'attribut désigné.
  - **Min():** calcule la valeur minimale des valeurs de l'attribut désigné.
  - **Sum():** calcule la somme des valeurs de l'attribut désigné.

N.B: Les attributs pour lesquels on applique ces fonctions (sauf count()) doivent être de type numérique: integer, number.

La fonction count()

- Cette fonction peut être appliquée à un attribut de n'importe quel type puisqu'elle rend le nombre d'éléments quoi que se soit leur type.
- Cette fonction peut être appliquée sur toute une table en écrivant: select count(*) from <nom_table>;
- **Exemple:**

Etudiant

N°	Nom	Prénom	Age
1	KACHLOUL	Hassan	17
2	MRABET	Salwa	19
3	ZAHOURI	Ali	18
4	SEBTI		18

select count(*) from Etudiant;
/*donne en résultat 4, c.à.d dans la table Etudiant, il y a 4 lignes. */

select count(Prénom) from Etudiant;
/*donne en résultat 3, c.à.d dans la table Etudiant, l'attribut Prénom contient 3 lignes. */

La fonction avg()

- Cette fonction ne peut être appliquée que si l'attribut concerné est d'un type numérique.
- Exemple:

Etudiant

N°	Nom	Prénom	Age
1	KACHLOUL	Hassan	17
2	MRABET	Salwa	19
3	ZAHOURI	Ali	18
4	SEBTI		18

select avg(age) from Etudiant;
/*donne en résultat 18 */

On peut ajouter des conditions dans la clause where:

select avg(age) from Etudiant
where nom='ZAHOURI' OR
nom='SEBTI';

La fonction max()

- Les conditions d'application de cette fonction sont identiques à celle de avg().
- Exemple

Etudiant

N°	Nom	Prénom	Note
1	KACHLOUL	Hassan	17
2	MRABET	Salwa	19
3	ZAHOURI	Ali	18
4	SEBTI		18

select max(note) from Etudiant;
/*donne en résultat 19 */

On peut ajouter des conditions dans la clause where:

select MAX(note) from Etudiant
where nom='ZAHOURI' OR
nom='KACHLOUL';

La fonction min()

- Cette fonction retourne la valeur minimale des valeurs d'un attribut numérique.
- Exemple

Etudiant

N°	Nom	Prénom	Note
1	KACHLOUL	Hassan	17
2	MRABET	Salwa	19
3	ZAHOURI	Ali	18
4	SEBTI		18

select min(note) from Etudiant;
/*donne en résultat 17 */

On peut ajouter des conditions dans la clause where:

select min(note) from Etudiant where nom='ZAHOURI' OR nom='KACHLOUL';

La fonction sum()

- Cette fonction retourne la somme des valeurs d'un attribut numérique.
- Exemple

Etudiant

N°	Nom	Prénom	Note
1	KACHLOUL	Hassan	17
2	MRABET	Salwa	19
3	ZAHOURI	Ali	18
4	SEBTI		18

select sum(note) from Etudiant;
/*donne en résultat 72 */

On peut ajouter des conditions dans la clause where:

select sum(note) from Etudiant
where nom='ZAHOURI' OR
nom='KACHLOUL';