



BASES DE DONNÉES

COURS:

ALGÈBRE RELATIONNELLE & "SQL"

Licence 2

Présenté par:
Dr.A BOUTORH

Informatique

L'ALGÈBRE RELATIONNELLE

- « L'algèbre relationnelle consiste en un ensemble d'opérations qui permettent de manipuler des *relations*, considérées comme des ensemble de tuples. »
- Une **Requête** est une **expression algébrique** qui s'applique à une ou un ensemble de relations en entrée (la base de données) et produit une relation finale en sortie (le résultat de la requête).
- On peut voir l'**algèbre relationnelle** comme un langage de programmation très simple qui permet d'exprimer des requêtes sur une base de données relationnelle.

ALGÈBRE RELATIONNELLE

LES OPÉRATIONS

- 1) La **Sélection** σ
- 2) La **Projection** Π
- 3) L'**Union** \cup
- 4) L'**Intersection** \cap
- 5) La **Différence** $-$
- 6) La **Division** \div
- 7) Le **Produit Cartésien** \times
- 8) La **Jointure** \bowtie

1- LA RESTRICTION « SÉLECTION »

- L'opération de la **Sélection** s'applique sur une relation **R1** et produit une relation **R2** de **même schéma** (même attributs) mais comportant seulement les tuples qui **vérifient une condition spécifique** (*supprimer des lignes*).
- La **Sélection** notée **$\sigma_F(R)$** extrait les tuples de la relation **R** qui satisfont le critère de sélection **F**.
- Le critère de sélection peut être :
 - La comparaison entre un **attribut 'A'** de la relation **R** et une **Constante 'c'**.
 - La comparaison entre deux attributs **'A1'** et **'A2'**.
- **L'opérateur de comparaison** peut appartenir à **$\{=, >, <, >=, <= \}$**
- La condition de sélection peut être une combinaison de critères avec les **opérateurs logiques** **$\{ET, OU, NON\}$**

SÉLECTION : EXEMPLE

Relation: **Emprunt**

Cote	Matricule	Date_Emprunt	Date_Remise
12	12345	05/09/2019	03/10/2019
5	53678	29/11/2019	01/01/2020
9	84302	20/12/2019	30/01/2020
17	53678	15/02/2020	15/03/2020

Les emprunts du matricule : **53678**

$\sigma_{(\text{Matricule}=53678)}$ (**Emprunt**)

Cote	Matricule	Date_Emprunt	Date_Remise
5	53678	29/11/2019	01/01/2020
17	53678	15/02/2020	15/03/2020

SÉLECTION :

ALGÈBRE RELATIONNELLE ET SQL

- Restriction en SQL :

SELECT *

FROM Table

WHERE Condition

Exemple: Donner les emprunts de la personne avec le matricule 53678

SELECT *

FROM Emprunt

WHERE Matricule = 53678

SÉLECTION :

ALGÈBRE RELATIONNELLE ET SQL

- La condition de la clause « **WHERE** » peut être exprimée avec:
 - Les opérateurs de Comparaison { =, >, <, >=, <= }
 - Les opérateurs Logiques { **AND**, **OR**, **NOT** }
 - Les Mots Clés:
- **BETWEEN** / **(NOT BETWEEN)** : pour tester si la valeur d'une expression est (n'est pas) comprise entre deux autres valeurs
- **IN** / **(NOT IN)** : pour tester si la valeur d'une expression appartient (n'appartient pas) à une liste de valeurs
- **LIKE** / **(NOT LIKE)** : pour tester si une expression de type chaîne de caractères contient (ne contient pas) une sous-chaîne

- **Table: Livre**

Cote	Titre	Auteur
17	Bases de Données	Auteur_1
5	Langage SQL	Auteur_2
9	SQL de Base	Auteur_3

- **Exemple :** Sélectionner les livres dont la cote est entre 9 et 17.

```
SELECT *  
FROM Livre  
WHERE Cote >= 9 AND Cote <= 17
```

- En utilisant « *BETWEEN* »

```
SELECT *  
FROM Livre  
WHERE Cote BETWEEN 9 AND 17
```


- **Exemple :** Sélectionner les livres dont la cote n'est pas entre 9 et 17

```
SELECT  *  
  
FROM    Livre  
  
WHERE    Cote >= 17 OR Cote <= 9
```

➤ En utilisant « *NOT* »

```
SELECT  *  
  
FROM    Livre  
  
WHERE    NOT (Cote >= 9 AND Cote <= 17)
```

- **Exemple :** Sélectionner les livres dont la cote est 9, 17, 5

```
SELECT *  
FROM Livre  
WHERE Cote IN (9, 17, 5)
```

- **Exemple :** Sélectionner les livres dont le titre contient « *BDD* »

```
SELECT *  
FROM Livre  
WHERE Titre LIKE “*BDD*”
```

- **Exemple :** Sélectionner les livres dont le titre **commence** par « SQL »

SELECT *

FROM Livre

WHERE Titre **LIKE** “SQL*”

- **Exemple :** Sélectionner les livres dont le titre **se termine** par « SQL »

SELECT *

FROM Livre

WHERE Titre **LIKE** “*SQL”

- **Exemple** : Sélectionner les livres dont le titre **commence** par un caractère suivie par « SQL »

```
SELECT *  
  
FROM Livre  
  
WHERE Titre LIKE “?SQL*”
```

- **Exemple** : Sélectionner les livres dont le titre contient **deux chiffres** qui se suivent

```
SELECT *  
  
FROM Livre  
  
WHERE Titre LIKE “*##*”
```

En **SQL** Standard: * <=> % , ? <=> _ , # <=> **Chiffre**

2- LA PROJECTION

- L'opération de la **Projection** s'applique sur une relation **R1** et produit une relation **R2** en éliminant de **R1** tous les attributs **non mentionnés** (*supprimer des colonnes*) et en gardant toutes les lignes.
- La **Projection** notée $\Pi_{A1, A2, A3, \dots, Ak}(R)$ ne garde que les attributs **A1, A2, A3, ..., Ak** sans redondance.
- **Exemple** : **Etudiant** (Matricule, Nom, Prénom, Filière, Année)

Donner les noms et les années des étudiants.

$$R2 = \Pi_{Nom, Année} (Etudiant)$$

- **R2 (Nom, Année)** On obtient le résultat suivant, après suppression des colonnes **Matricule, Prénom et Filière**

L'ALGÈBRE RELATIONNELLE:

2- LA PROJECTION

- Comme le résultat est une relation, en principe il ne peut pas y avoir deux lignes identiques, sachant que le nombre de lignes dans la relation résultante est le même que dans la relation initiale.
- Après une projection, on peut avoir deux lignes identiques qui étaient initialement distinctes, cela on raison de la suppression de l'attributs qui les distinguait.
- Dans ce cas, une seule des deux lignes identiques est conservée.

- **Exemple** : on souhaite connaître toutes les filières où sont inscrit des étudiants.

Table Etudiant

Matricule	Nom	Prénom	Filière	Année
1234	Nom_1	Prénom_1	MI	2
5378	Nom_2	Prénom_2	SNV	1
9264	Nom_1	Prénom_3	MI	5
4294	Nom_3	Prénom_2	SM	2

$\Pi_{\text{Filière}} (\text{Etudiant})$

Filière
MI
SNV
SM

La ligne **MI** était présente *deux fois* dans la table initiale « *Relation Etudiant* » et n'apparaît plus qu'en un seul exemplaire dans le résultat

PROJECTION :

ALGÈBRE RELATIONNELLE ET SQL

- Projection en SQL :

SELECT Attributs

FROM Table

Exemple: Donner le prénom et l'année de tous les étudiants

SELECT Prénom, Année
FROM Etudiant

Pour ne pas avoir des tuples en double, on utilise « **DISTINCT** »

SELECT **DISTINCT** Prénom
FROM Etudiant

COMBINER EN SQL

PROJECTION ET RESTRICTION

- **Sélectionner** d'une relation **R** certains **Attributs** des tuples vérifiant une **Condition**.

- Notation: $\Pi_{A1, A2, A3, \dots, Ak} (\sigma_{\text{Condition}} (R))$

- **Syntaxe en SQL:**

SELECT **Attributs**

FROM **Table**

WHERE **Condition**

- **Exemple :** Donner le matricule et l'année des étudiants de « MI »

SELECT **Matricule, Année**

FROM **Etudiant**

WHERE **Filière = "MI"**

- **Exemple :** Donner le nom, le prénom et la filière des étudiants dont le nom contient « Ben » et dont le prénom contient « ss »

SELECT **Nom, Prénom, Filière**

FROM **Etudiant**

WHERE **Nom LIKE "*Ben*" AND Prénom LIKE "*ss*"**

3- L'UNION

- L'**Union** est un opérateur binaire. L'expression **R U S** crée une relation comprenant tous les tuples existant dans l'une ou l'autre des deux relations **R** et **S**.
- Il existe une condition impérative : **les deux relations doivent avoir le même schéma**, c'est-à-dire :
 - même nombre d'attributs,
 - mêmes noms
 - mêmes types.

3- L'UNION

- L'union des relations **R(A,B)** et **S (C,D)** est interdit car **R** et **S** ne sont pas de même schéma (on ne saurait pas comment nommer les attributs dans le résultat).
- En modifiant les noms des attributs de la relation **S** par les noms de la relation **R** (opération de renommage)

$$S' = \rho_{C \rightarrow A, D \rightarrow B}(S)$$

il devient possible de calculer **R U S'** avec le résultat suivant :

A	B
a	b
x	y

R

A	B
c	d
u	v
a	b

S'

U

=

A	B
a	b
x	y
c	d
u	v

Comme pour la projection, il faut penser à éviter les doublons. Donc le tuple **(x, y)** qui existe à la fois dans **R** et dans **S'** ne figure qu'une seule fois dans le résultat.

- Exemple :

R1

Matricule	Depart
1234	MI
5378	SNV
9264	MI
3274	SM

R2

Matricule	Année-Insc
1234	2018
6358	2019
9264	2015
4294	2018

- Donner les matricules des étudiants du département « MI » ou ceux qui sont inscrits en 2018.

$\left[\Pi_{\text{Matricule}} \left(\sigma_{\text{Depart}=\text{"MI"}} (\mathbf{R1}) \right) \right]$

$\mathbf{U} \left[\Pi_{\text{Matricule}} \left(\sigma_{\text{Année-Insc}=2018} (\mathbf{R2}) \right) \right]$

4- INTERSECTION

L'**Intersection** de deux relations **R1** et **R2** de **même schéma** est une relation **R3** de **même schéma** dont les tuples sont ceux appartenant à la fois à **R1** et à **R2**. **$R3 = R1 \cap R2$**

R1

Matricule	Nom	Prénom	Filière	Année
1234	Nom_3	Prénom_3	SNV	2
6789	Nom_7	Nom_7	MI	2

\cap

R2

Matricule	Nom	Prénom	Filière	Année
6789	Nom_7	Nom_7	MI	2

=

R3

Matricule	Nom	Prénom	Filière	Année
6789	Nom_7	Nom_7	MI	2

- Exemple :

R1

Matricule	Depart
1234	MI
5378	SNV
9264	MI
3274	SM

R2

Matricule	Année-Insc
1234	2018
6358	2019
9264	2015
4294	2018

- Donner les matricules des étudiants du département « MI » et ceux qui sont inscrits en 2018.

$$\left[\Pi_{\text{matricule}} \left(\sigma_{\text{Depart}=\text{MI}} (\text{R1}) \right) \right]$$

$$\cap \left[\Pi_{\text{matricule}} \left(\sigma_{\text{Année-Insc}=2018} (\text{R2}) \right) \right]$$

5- LA DIFFÉRENCE

- Comme l'union et l'intersection, la **Différence** s'applique à deux relations qui ont le **même schéma**. L'expression **R – S** a alors pour résultat tous les tuples de **R** qui ne sont pas dans **S**
- La différence est le seul opérateur qui permet d'exprimer des requêtes comportant **une négation** (on veut '**rejeter**' des lignes ayant/ n'ayant pas une propriété).

R

Matricule	Nom	Prénom	Filière	Année
1234	Nom_3	Prénom_3	SNV	2
6789	Nom_7	Nom_7	MI	2

S

Matricule	Nom	Prénom	Filière	Année
6789	Nom_7	Nom_7	MI	2

R - S =

Matricule	Nom	Prénom	Filière	Année
1234	Nom_3	Prénom_3	SNV	2

- Exemple :

Filière	
Code	Nom_Fil
1	MI
2	SM
3	SNV

Filière	
Code	Nom_Fil
1	MI
2	SM
3	SNV

Etudiant				
Matricule	Nom	Prénom	Année	Code_F
1234	Nom_3	Prénom_3	2	1
9876	Nom_9	Prénom_9	5	3

Etudiant				
Matricule	Nom	Prénom	Année	Code_F
1234	Nom_3	Prénom_3	2	1
9876	Nom_9	Prénom_9	5	3

- Donner les filières dont lesquelles aucun étudiant n'est inscrit

$[\pi_{\text{Code}} (\text{Filière})] - [\pi_{\text{code_F}} (\text{Etudiant})]$

➤ SQL :

SELECT Code

FROM Filière, Etudiant

WHERE NOT (Code = Code_F)

6- LA DIVISION

- La **Division** de la relation $R1(A1, A2, A3, \dots, An)$ par la sous-relation $R2(Ap+1, \dots, An)$ est la relation $R3(A1, A2, A3, \dots, Ap)$ formées de tous les tuples qui **concaténés** à chaque tuple de **R2** donne toujours un tuple de **R1**. **$R3 = R1 \div R2$**

Nom	Filière	Année
Nom_3	SNV	3
Nom_7	MI	2
Nom_5	MI	3
Nom_7	SNV	5
Nom_3	MI	2

Filière	Année
MI	2
SNV	5

Nom
Nom_7

R1

\div

R2

=

R3

- Exemple :

Etudiant

Matricule	Nom	Dep
5432	Yacine	3
2345	Ahmed	2
6789	Sarah	2
45677	Amel	2
3412	Yacine	2
9534	Sarah	3
341289	ILyes	3

Département

No_Dep	Nom_Dep
2	MI
3	SNV

- Donner les noms des étudiants qui existent dans tous les départements

$$[\Pi_{\text{Nom, Dep}} (\text{Etudiant})] \div [\Pi_{\text{No_Dep}} (\text{Département})]$$

Nom
Yacine
Sarah

7- LE PRODUIT CARTÉSIEN

- L'opération du **Produit Cartésien** s'applique sur deux relations **R1** et **R2** et produit une troisième relation **R3** ayant pour:
 - **Schéma** : la concaténation de ces deux relations,
 - **Tuples**: toutes les combinaisons des tuples des relations **R1** et **R2**.
- Le **Produit Cartésien** est noté **R1 x R2**

7- LE PRODUIT CARTÉSIEN

R

A	B
a	b
x	y

S

C	D
c	d
u	v
x	y

**R
X
S**

A	B	C	D
a	b	c	d
a	b	u	v
a	b	x	y
x	y	c	d
x	y	u	v
x	y	x	y

Le nombre de
lignes dans le
résultat est
exactement

$$|R| \times |S|$$

7- LE PRODUIT CARTÉSIEN

- **Conflits de noms d'attributs**
 - Si les schémas des relations **R** et **S** sont complètement **distincts**, il n'y aura pas d'ambiguïté sur la provenance des colonnes dans le résultat.
 - Par exemple on sait que les valeurs de la colonne **A** dans **R x S** vient de la relation **R**.
 - On peut tomber dans le cas où les deux relations aient des attributs qui ont le **même nom**. On doit alors **distinguer l'origine** des attributs dans la table résultat en donnant un **nom distinct** à chaque attribut.

- **Exemple** : La table **T** a les mêmes noms d'attributs que la table **R**

R

A	B
a	b
x	y

A	B
m	n
o	p

La table *T*

T

- Le résultat du **Produit Cartésien** **R x T** a pour **Schéma (A, B, A, B)** et présente donc une **ambiguïté**, avec les colonnes **A** et **B** en double.
- La première solution pour lever l'ambiguïté est de **préfixer un attribut par le nom** Ou la **1ère lettre** de la table d'où il provient.

Exemple: Relation **Etudiant « E.A »**

Ou bien : **« Etudiant.A »**

Le résultat de R x T devient:

R.A	R.B	T.A	T.B
a	b	m	n
a	b	n	p
x	y	m	n
x	y	n	p

R x T

- Exemple :

Filière

Code	Nom_Fil
1	MI
2	SM
3	SNV

Etudiant

Matricule	Nom	Prénom	Année	Code_F
1234	Nom_3	Prénom_3	2	1
9876	Nom_9	Prénom_9	5	3

Filière x Etudiant

Code	Nom_Fil	Matricule	Nom	Prénom	Année	Code_F
1	MI	1234	Nom_3	Prénom_3	2	1
1	MI	9876	Nom_9	Prénom_9	5	3
2	SM	1234	Nom_3	Prénom_3	2	1
2	SM	9876	Nom_9	Prénom_9	5	3
3	SNV	1234	Nom_3	Prénom_3	2	1
3	SNV	9876	Nom_9	Prénom_9	5	3

PRODUIT CARTÉSIEN :

ALGÈBRE RELATIONNELLE ET SQL

- Produit cartésien en SQL :

```
SELECT      *  
  
FROM        Table_1, Table_2, Table_3, ..., Table_n
```

Exemple:

```
SELECT      *  
  
FROM        Filière, Etudiant
```

8- LA JOINTURE

- En pratique, il existe d'autres opérations, très couramment utilisées, qui peuvent se construire par **composition** des opérations de base. La plus importante est la **Jointure**
- La **Jointure** notée **R1** ⋈ **R2** consiste à combiner deux relations **R1** et **R2** (**Tuple à Tuple**, **Lignes à Ligne**) en vérifiant la **concordance** entre certains **attributs** (**colonnes**) des deux relations (en générale une **clé primaire** avec une **clé étrangère**)
.

8- LA JOINTURE

- La **Jointure** est une combinaison d'opérations, c'est une **Projection** d'une **Restriction** sur un **Produit Cartésien** entre **plusieurs relations** afin de **sélectionner** certains attributs des tuples de **produit cartésien** vérifiant une **condition** donnée.
- Afin de comprendre l'intérêt de cet opérateur, regardons le résultat du **Produit cartésien**.
- Le résultat est énorme et comprend manifestement un grand nombre de lignes qui ne nous intéressent pas.

8- LA JOINTURE

- Si, on considère le produit cartésien comme un *résultat intermédiaire*, il génère avec une simple sélection des informations plus intéressantes.

$\sigma_{F.Code=E.Code_F}$ (Filière x Etudiant)

- On a donc effectué une *composition* de deux opérations (un **Produit Cartésien** et une **Sélection**) afin de rapprocher des informations réparties dans *plusieurs tables*, mais ayant des *liens entre elles*.
- Cette opération est **Jointure**

Filière



Etudiant

F.Code = E.Code_F

Filière

Code	Nom_Fil
1	MI
2	SM
3	SNV

Etudiant

Matricule	Nom	Prénom	Année	Code_F
1234	Nom_3	Prénom_3	2	1
9876	Nom_9	Prénom_9	5	3

Filière



Etudiant

F.Code = E.Code_F

Code	Nom_Fil	Matricule	Nom	Prénom	Année	Code_F
1	MI	1234	Nom_3	Prénom_3	2	1
3	SNV	9876	Nom_9	Prénom_9	5	3

Résultat :

- ✓ Autant d'attributs que le produit cartésien
- ✓ Moins de tuples

Critères de Sélection :

{ = , > , < , >= , <= , <> }

- Exemple :

Client

Numéro	Nom	Adresse	Tel
12	Ahmed	Alger	057391
100	ILyes	Oran	NULL
82	Amel	Alger	038143
34	Youcef	Setif	NULL

Vente

Numéro	Ref_Produit	No_Client	Date
6	P123	34	5/01/20
20	P38	100	3/02/20
15	P20	12	9/01/20
3	P259	100	7/02/20

Produit

Référence	Marque	Prix
P234	Audi	4000
P20	Toyota	3000
P259	Renault	2000
P123	BMW	5000
P38	Nissan	2000

- Exemple :

➤ Q1 - Donner les noms des clients avec les dates de leurs achats.

Q1 : π (Client \bowtie Vente)
Client.nom, Vente.date Client.numéro = Vente.no_client

➤ Q2- Donner, pour le client numéro 12, le numéro de vente et la marque des produits achetés

Q2 : $V1 = \sigma$ (Vente)
Vente.no_client = 12

$R1 = V1 \bowtie$ Produit
 $V1$.ref_produit = Produit.référence

Res = π (R1)
R1.numéro, | R1.marque

- Exemple :

➤ Q3 - Donner les références des produits dont le prix est supérieur au produit qui a pour référence P20

Q3 P1 = ρ (Produit) *opérateur de renommage*

P2 = σ (P1)

P1.référence = P20

Res = π (Produit \bowtie P2)
Produit.référence Produit.prix > P2.prix

OPÉRATEUR

EQUIJOINTURE / JOINTURE NATURELLE

- **Equijointure** : la condition fait appel à l'opérateur « = »
 - **Jointure Naturelle**: notée aussi « * » , c'est une **Equijointure** dont la condition porte sur des attributs identiques (de même domaine et même nom)
- ✓ **Un seul** des deux attributs est conservé dans le résultat

Numéro	Nom	Adresse	Tel	Numéro	Ref_Produit	No_Client	Date
12	Ahmed	Alger	057391	6	P123	12	5/01/20
100	ILyes	Oran	NULL	20	P38	100	3/02/20
82	Amel	Alger	038143	15	P20	82	9/01/20
34	Youcef	Setif	NULL	3	P259	34	7/02/20

JOINTURE :

ALGÈBRE RELATIONNELLE ET SQL

- Jointure en SQL :

SELECT **Attributs**

FROM *Table_1, Table_2, Table_3, ..., Table_n*

WHERE *Conditions_de_Jointure_et_de_Sélection*

- Jointure en SQL :

SELECT **Attributs**

FROM *Table_i JOIN Table_j ON (Condition_Jointure)*

WHERE *Conditions_de_Sélection*

- **Exemple :**
- **Client** (Numéro, Nom, Adresse, Tel)
- **Vente** (Numéro, Ref_Produit, No_Client, Date)
- **Requête:** Donner le numéro de vente, la référence du produit et l'adresse du client Ahmed

➤ Algèbre Relationnelle:

$\Pi_{V.\text{Numéro}, V.\text{Ref_Produit}, C.\text{Adresse}} (\sigma_{NOM='Ahmed'} (Clien \bowtie_{C.\text{Numéro}=V.\text{No_Client}} Vente))$

➤ SQL:

SELECT Vente. Numéro, Ref_Produit, Adresse

FROM Client, Vente

WHERE Client.Numéro = Vente.No_Client **AND** NOM = 'Ahmed'

- **Exemple :**
- **Client** (Numéro, Nom, Adresse, Tel)
- **Vente** (Numéro, Ref_Produit, #No_Client, date)
- **Requête:** Donner le numéro de vente, la référence du produit et l'adresse du client Ahmed

➤ Algèbre Relationnelle:

$\Pi_{V.\text{Numéro}, V.\text{Ref_Produit}, C.\text{Adresse}} (\sigma_{\text{NOM} = \text{'Ahmed'}} (\text{Clien} \bowtie_{C.\text{Numéro} = V.\text{No_Client}} \text{Vente}))$

➤ SQL:

```
SELECT Vente. Numéro, Ref_Produit, Adresse
FROM Client JOIN Vente ON (Client.Numéro = Vente.No_Client )
WHERE NOM = 'Ahmed'
```


JOINTURE EXTERNE

- La jointure externe entre les relations **R1** et **R2** notée **R1** ⋈ **R2** :

➤ est une jointure entre **R1** ⋈ **R2**

➤ contient les tuples de **R1** et de **R2** ne participant pas à la jointure

Client ⋈ **Vente**

Numéro	Nom	Adresse	Tel	Numéro	Ref_Produit	Date
12	Ahmed	Alger	057391	15	P20	9/01/20
100	ILyes	Oran	NULL	20	P38	3/02/20
100	ILyes	Oran	NULL	3	P259	7/02/20
82	Amel	Alger	038143	NULL	NULL	NULL
34	Youcef	Setif	NULL	6	P123	5/01/20

JOINTURE EXTERNE

- La **jointure externe** permet de retourner toutes les lignes d'une table, même si celle-ci ne possède pas de correspondances dans la table jointe
- Trois types de jointures externes :
 - Jointure Externe **Gauche** (**LEFT OUTER JOIN**)
 - Jointure Externe **Droite** (**RIGHT OUTER JOIN**)
 - Jointure Externe **Gauche-Droite** (**FULL OUTER JOIN**)

➤ Syntaxe SQL:

SELECT Attributs

FROM Tab_1 [**LEFT|RIGHT|FULL**] **OUTER JOIN** Tab_2

ON Tab_1.col = Tab_2.col



Num	Nom	Adresse
12	Ahmed	Alger
100	Ilyes	Oran
45	Ahmed	Setif

R1


Matr	Nom	Tel
23	Ahmed	057391
6	Amel	038143
90	Amel	046723

R2

R1 ⋈ R2

Num	Nom	Adresse	Matr	Tel
12	Ahmed	Alger	23	057391
100	Ilyes	Oran	NULL	NULL
45	Ahmed	Setif	23	057391
NULL	Amel	NULL	6	038143
NULL	Amel	NULL	90	046723

SEMI-JOINTURE

- La **Semi-Jointure** de deux relations **R1** et **R2** notée **R1**  **R2** est une relation **R3** dont le **schéma** est celui de **R1** et les **tuples** sont ceux de **R1** appartenant à la **jointure** entre **R1** et **R2**
- Les tuples de **R2** **ne participent** pas à la semi-jointure.

R1

Num	Nom	Adresse
12	Ahmed	Alger
100	Ilyes	Oran
45	Ahmed	Setif

R2

Matr	Nom	Tel
23	Ahmed	057391
6	Amel	038143
90	Amel	046723

R1



R2

R2



R1

Num	Nom	Adresse
12	Ahmed	Alger
45	Ahmed	Setif

R3

COMPLEXITÉ DES OPÉRATIONS

Sélection : σ [condition] R

- Au plus: balayer la relation + tester la condition sur chaque tuple.
- Complexité = $\text{card}(R)$.
- Taille du résultat : $[0 : \text{card}(R)]$.

Projection : π [$A_i, A_k \dots$] R

- Balayer la relation + élimination doublons
- Complexité = $\text{card}(R)$. *0 si inclut dans une sélection*
- Taille du résultat : $[1 : \text{card}(R)]$.

Jointure (naturelle ou θ) 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 = $\text{card}(R) \times \text{card}(S)$.
- Taille du résultat : $[0 : \text{card}(R) \times \text{card}(S)]$.

LIENS UTILES

➤ **Algèbre Relationnelle: Division**

<https://www.youtube.com/watch?v=5kBUpHvYOZA>

➤ **Algèbre Relationnelle: Jointure avec boucles imbriquées**

https://www.youtube.com/watch?v=4x5uFA_FqSA

➤ **Algèbre Relationnelle: Jointure avec tri fusion**

<https://www.youtube.com/watch?v=nu61l8ikpBE>

BIBLIOGRAPHIE

- GARDARIN, Georges. *Bases de données*. Editions Eyrolles, 2003.
- Sébastien Choplin, Chapitre 5: L'algèbre relationnelle.
Enseignements à l'Université de Picardie Jules Verne.
- EL FADDOULI NOUR-EDDINE, Initiation aux Bases de Données.
almohandiss.com
- Philippe LAHIRE – Cours Base de Données L2I