

A vertical decorative element on the left side of the slide, consisting of a cluster of blue, yellow, and dark blue hexagons of various sizes.

M1104

Introduction aux bases de données

MODULE : Algèbre & Modèle relationnel

A decorative element on the right side of the slide, featuring a cluster of blue, yellow, and dark blue hexagons of various sizes, mirroring the one on the left.

DE L'ALGÈBRE AU SQL



Introduction

- ❑ Le **SQL** (Structured Query Language) va nous permettre de réaliser des requêtes standard vers les SGBDs afin d'extraire (entre autre) des informations de la base de données
- ❑ Nous nous intéressons dans cette partie qu'aux requêtes d'interrogation de données.
- ❑ Les requêtes concernent le SGBD PostGres et MySQL

Opérations spécifiques : la projection

■ La projection

- Relation restreinte aux attributs spécifiés dans la projection
- Notation en algèbre relationnel :

$$R1 = \Pi_{(Col_1 \dots Col_i)} R$$

■ Notation en SQL :

```
SELECT Col1, Col2,...,Coli  
FROM R;
```

Relation R					
Col ₁	Col ₂	Col ₃	Col ₄	Col ₅	Col ₆

Opérations spécifiques : la projection

■ Exemple : Nom et nationalité des coureurs ?

■ Notation en algèbre relationnel :

$$R_1 = \Pi_{(\text{NomCoureur}, \text{CodePays})} \text{COUREUR}$$

■ Notation en SQL :

```
SELECT NomCoureur, CodePays  
FROM COUREUR;
```

Relation : COUREUR				Resultat	
NumCoureur	NomCoureur	CodeEquipe	CodePa	NomCoureur	CodePays
8	ULLRICH Jan	TEL	ALL	ULLRICH Jan	ALL
31	JALABERT Laurent	ONC	FRA	JALABERT Laurent	FRA
61	ROMINGER Tony	COF	SUI	ROMINGER Tony	SUI
91	BOARDMAN Chris	GAN	G-B	BOARDMAN Chris	G-B
114	CIPOLLINI Mario		ITA	CIPOLLINI Mario	ITA

Relation résultat

Opérations spécifiques : la projection

■ Le tri simple ou multiple :

■ Notation en SQL :

```
SELECT Col1,Col2  
FROM R  
ORDER BY Col1;
```

Tri croissant sur la colonne nommée Col1 par défaut ASC

```
SELECT Col1,Col2  
FROM R  
ORDER BY Col1 DESC;
```

Tri décroissant sur la colonne nommée (DESC)

Opérations spécifiques : la projection

■ Le tri simple ou multiple :

■ Exemple :

```
SELECT      CLI_NOM, CLI_PRENOM  
FROM        CLIENT  
ORDER BY    CLI_NOM, CLI_PRENOM DESC;
```

CLI_NOM	CLI_PRENOM
AIACH	Alexandre
ALBERT	Christian
AUZENAT	Michel
BACQUE	Michel
BAILLY	Jean-François
BAVEREL	Frédéric
BEAUNEE	Pierre
BENATTAR	Pierre
BENATTAR	Bernard
BENATTAR	Luc

Opérations spécifiques : la projection

■ L'élimination des doublons

■ Notation en SQL :

```
SELECT DISTINCT col  
FROM R;
```

■ Exemple

```
SELECT DISTINCT CLI_PRENOM  
FROM Personnes;
```

CLI_PRENOM
Alain
Alain
Alain
Alain
Alain
Alain
Alain
Alexandre
André
André

Utilisation du distinct

CLI_PRENOM
Alain
Alexandre
André
Arnaud
Arsène
Bernard

Opérations spécifiques : la projection

■ Renommer les colonnes de résultat

■ Notation en SQL :

```
SELECT Col1 AS "Nouveau_nom "  
FROM R1;
```

■ Exemple sous POSTGRES

```
SELECT Nom || ' ' || Prenom AS "Nom_complet "  
FROM table1;
```

|| caractère de concaténation sous Postgres différent sous les autres SGBD

■ Exemple sous Mysql

```
SELECT CONCAT( Nom , ' ' ,Prenom) AS "Nom_complet"  
FROM table1;
```

CONCAT est une fonction MySQL

	Nom_complet
▶	chose jules
	machin pierre
	truc patrick
*	

Opérations spécifiques : la sélection

■ La sélection

- La sélection (parfois appelée restriction) génère une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont l'expression logique E.
- Notation en algèbre relationnel :

$$R_1 = \sigma(\text{Expression})R$$

■ Notation en SQL :

```
SELECT *  
FROM R  
WHERE Expression;
```

Relation R		
Col ₁	Col ₂	Col ₃

Expression doit retourner un booléen (VRAI ou FAUX)

Opérations spécifiques : la sélection

■ Exemple : Quels sont les coureurs suisses ?

- Notation en algèbre relationnel :

$R1 = \sigma_{\text{CodePays} = \text{"SUI"}}(\text{COUREUR})$

- Notation en SQL :

SELECT * FROM COUREUR WHERE CodePays = 'SUI';

Relation : COUREUR			
NumCoureur	NomCoureur	CodeEquipe	CodePays
8	ULLRICH Jan	TEL	ALL
31	JALABERT Laurent	ONC	FRA
61	ROMINGER Tony	COF	SUI
91	BOARDMAN Chris	GAN	GBR
114	CIPOLLINI		

Relation : R1			
NumCoureur	NomCoureur	CodeEquipe	CodePays
61	ROMINGER Tony	COF	SUI

Relation résultat

Opérations spécifiques : la sélection

■ Les expressions de la clause WHERE

OPÉRATEUR(S)	RENOI "TRUE" SI...	Exemple
<> ou !=	les deux valeurs ne sont pas égales	... WHERE prix != 100
<	strictement inférieure	... WHERE prix < 100
>	strictement supérieure	... WHERE prix > 100
<=	inférieure ou égale	... WHERE prix <= 100
>=	supérieure ou égale	... WHERE prix >= 100
Val BETWEEN x AND y	(val >= x AND val<=y)	... WHERE prix BETWEEN 100 AND 200
IN	la valeur testée se situe dans une liste valeurs données	... WHERE prix IN (100,200,300)
NOT IN	la valeur testée ne se situe pas dans une liste de valeurs données	... WHERE prix NOT IN (50, 150,205)
LIKE	la valeur de gauche correspond à celle de droite (celle de droite peut utiliser le caractère % pour simuler n'importe quel nombre de caractère, et _ pour un seul caractère)	... WHERE prenom like 'j%'
NOT LIKE	les deux valeurs ne correspondent pas	... WHERE prenom NOT like 'Jules'

Opérations spécifiques : la sélection

■ Quelques exemples de test dans la clause WHERE

- Nom, prénom, date des personnes embauché après 2001

```
SELECT nom, prenom, date
FROM personnes
WHERE date > '2001-12-31';
```

- Nom, prénom, date des personnes embauché en 2002

```
SELECT nom, prenom, date
FROM personnes
WHERE extract(year from date)=2002;
```

Extract est une fonction postgres et MySQL qui permet d'extraire des informations d'une date telles que : l'année, le mois , le jour

Opérations spécifiques : la sélection

■ Quelques exemples de test dans la clause WHERE

- Nom, prénom des personnes dont le nom contient un B en troisième lettre, un R un cinquième et quelque soit ce qui suit

```
SELECT nom, prenom  
FROM personnes  
WHERE nom like '___B__R%';
```

Le like permet d'utiliser les joker _ et % qui permettent respectivement de remplacer un caractère (_) ou 1 ou plusieurs (%)



Opérations spécifiques : la sélection

■ fonctions agrégats :

Elles s'appliquent à l'ensemble des valeurs d'une colonne. Elles ignorent les valeurs "NULL".

- **MAX** : valeur maximale des éléments d'une colonne
- **MIN** : valeur minimum des éléments d'une colonne
- **AVG** : moyenne arithmétique des éléments d'une colonne
- **SUM** : somme des éléments d'une colonne
- **COUNT** : nombre d'occurrences des éléments d'une colonne

Opérations spécifiques : la sélection

■ fonctions agrégats

■ Quelques Exemples :

- Trouver le nombre de clients de la ville de Paris

```
SELECT COUNT(*) AS "Nombre Client"  
FROM CLIENT  
WHERE Ville = 'Paris'
```

	"Nombre Client"
▶	3

- Trouver la moyenne des prix des produits

```
SELECT AVG(Prix) AS "Moyenne"  
FROM PRODUIT
```

	Moyenne
▶	166,66666667

- Trouver LE PRIX Maximum des produits

```
SELECT MAX(Prix) AS "Max"  
FROM PRODUIT
```

	Moyenne
▶	350

Opérations spécifiques : la sélection

■ Remarque :

On ne peut pas combiner dans le même SELECT une requête avec une fonction agrégat (qui renvoie une valeur) et une colonne (qui renvoie un ensemble de valeurs).

■ Mauvaise requête :

```
SELECT Libelle, AVG(Prix)
FROM PRODUIT
```

Remarque : impossible d'afficher d'autres champs sans réaliser des groupements que nous verrons plus tard !!

Opérations spécifiques : la sélection

■ Opérateurs Mathématiques

Opérateur	Description	Exemple	Résultat
+	addition	$2 + 3$	5
-	soustraction	$2 - 3$	-1
*	multiplication	$2 * 3$	6
/	division (la division entière tronque les résultats)	$4 / 2$	2
%	modulo (reste)	$5 \% 4$	1
^	exponentiel	$2.0 ^ 3.0$	8
/	racine carrée	$ / 25.0$	5
/	racine cubique	$ / 27.0$	3
!	factoriel	$5 !$	120
!!	factoriel (opérateur préfixe)	$!! 5$	120
@	valeur absolue	$@ -5.0$	5
&	AND bit à bit	$91 \& 15$	11
	OR bit à bit	$32 3$	35
#	XOR bit à bit	$17 \# 5$	20
~	NOT bit à bit	~ 1	-2
<<	décalage gauche	$1 << 4$	16
>>	décalage droit	$8 >> 2$	2

Opérations spécifiques : la jointure

■ Les jointures :

■ *theta-jointure*

- c'est une jointure dans laquelle l'expression logique E est une simple comparaison entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2.

- Notation en algèbre relationnel : $R1 \bowtie_{\text{Expression}} R2$

■ Notation en SQL :

```
SELECT *  
FROM R1 inner join R2 ON Expression;
```

Opérations spécifiques : la jointure

■ Exemple de theta-jointure

■ Notation en algèbre relationnel :

$R = \text{Famille} \bowtie_{((\text{Age} \leq \text{AgeC}) \text{ ET } (\text{Prix} < 50))} \text{Cadeau}$

□ Notation en SQL :

SELECT *

FROM famille INNER JOIN cadeau ON Age <= AgeC

AND Prix < 50;

Relation Famille		
Nom	Prénom	Age
Fourt	Lisa	6
Juny	Carole	42
Fidus	Laure	16

Relation Cadeau		
AgeC	Article	Prix
99	livre	30
6	poupée	60
20	baladeur	45
10	déguisement	15

Relation résultat					
Nom	Prénom	Age	AgeC	Article	Prix
Fourt	Lisa	6	99	livre	30
Fourt	Lisa	6	20	baladeur	45
Fourt	Lisa	6	10	déguisement	15
Juny	Carole	42	99	livre	30
Fidus	Laure	16	99	livre	30
Fidus	Laure	16	20	baladeur	45

Opérations spécifiques : la jointure

■ *equi-jointure*

- *Une equi-jointure est une theta-jointure dans laquelle l'expression logique E est un test d'égalité entre un attribut A_1 de la relation R_1 et un attribut A_2 de la relation R_2 .*

- *Notation en algèbre relationnel : $R_1 \bowtie_{A_1=A_2} R_2$.*

■ *Notation en SQL :*

```
SELECT *  
FROM R1 inner join R2 ON A1=A2;
```

Opérations spécifiques : la jointure

■ Exemple d'equi-jointure

Notation en algèbre relationnel

$R = \text{coureur} \bowtie_{\text{CodePays, CodePays}} \text{pays}$

□ Notation en SQL :

SELECT *

FROM coureur C INNER JOIN pays P

ON C.CodePays= P.CodePays;

Relation : COUREUR			
NumCoureur	NomCoureur	CodeEquipe	CodePays
8	ULLRICH Jan	TEL	ALL
31	JALABERT Laurent	ONC	FRA
61	ROMINGER Tony	COF	SUI
91	BOARDMAN Chris	GAN	G-B

Relation : PAYS	
CodePays	NomPays
ALL	Allemagne
FRA	France
SUI	Suisse

Relation résultat

NumCoureur	NomCoureur	CodeEquipe	CodePays	CodePays	NomPays
8	ULLRICH Jan	TEL	ALL	ALL	Allemagne
31	JALABERT Laurent	ONC	FRA	FRA	France
61	ROMINGER Tony	COF	SUI	SUI	Suisse
91	BOARDMAN Chris	GAN	G-B	G-B	Grande -Bretagne

Opérations ensemblistes : Produit cartésien

■ Produit cartésien

- *Le produit cartésien est une opération portant sur deux relations $R1$ et $R2$ et qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations $R1$ et $R2$.*
- Notation en algèbre relationnel : $R3 : R1 \times R2$
- Notation en SQL :

```
SELECT *  
FROM R1, R2;
```

ou

```
SELECT *  
FROM R1 CROSS JOIN R2;
```

Opérations ensemblistes : Produit cartésien

■ **Exemple : SELECT ***
FROM coureur cross join pays;

Relation : COUREUR			
NumCoureur	NomCoureur	CodeEquipe	CodePays
8	ULLRICH Jan	TEL	ALL
31	JALABERT Laurent	ONC	FRA
61	ROMINGER Tony	COF	SUI
91	BOARDMAN Chris	GAN	G-B
114	CIPOLLINI Mario	SAE	ITA

Relation : PAYS	
CodePays	NomPays
ALL	Allemagne
FRA	France
SUI	Suisse
G-B	Grande -Bretagne

Relation
résultat

NumCoureur	NomCoureur	CodeEquipe	CodePays	CodePays	NomPays
8	ULLRICH Jan	TEL	ALL	ALL	Allemagne
8	ULLRICH Jan	TEL	ALL	FRA	France
8	ULLRICH Jan	TEL	ALL	SUI	Suisse
8	ULLRICH Jan	TEL	ALL	G-B	Grande -Bretagne
31	JALABERT Laurent	ONC	FRA	ALL	Allemagne
31	JALABERT Laurent	ONC	FRA	FRA	France
31	JALABERT Laurent	ONC	FRA	SUI	Suisse
31	JALABERT Laurent	ONC	FRA	G-B	Grande -Bretagne

Opérations ensemblistes : **Différence**

■ Notation en SQL:

■ **La syntaxe normalisée :**

- Pour faire une différence, il suffit de disposer de deux ensembles de données compatibles et d'utiliser le mot clef **EXCEPT** . La syntaxe est alors :

SELECT ... EXCEPT SELECT ...

- Bien entendu il est indispensable que les deux ordres **SELECT** :
 - produisent un même nombre de colonnes
 - que les types de données de chaque paires ordonnées de colonnes soient de même type (ou d'un type équivalent)

Opérations ensemblistes : **Différence**

■ **Notation en SQL:**

■ **Avec la syntaxe SQL**

- **Exemple : Les étudiants dont le nom contient un A mais pas ceux dont le nom commence par un B**

En algèbre relationnel :

$$R1 = \sigma_{(\text{nom} = '%A\%')} \text{Etudiants} - \sigma_{(\text{nom} = 'B\%')} \text{Etudiants}$$

En SQL :

```
SELECT * FROM etudiants
WHERE nom LIKE '%A%'
EXCEPT
SELECT * from etudiants
WHERE nom LIKE 'B%'
```

Les lignes dupliquées sont éliminées sauf si EXCEPT ALL est utilisé.

Opérations ensemblistes : l'intersection

■ Notation en SQL:

■ La syntaxe normalisée :

- Pour faire une intersection, il suffit de disposer de deux ensembles de données compatibles et d'utiliser le mot clef **INTERSECT** . La syntaxe est alors :

SELECT ... INTERSECT SELECT ...

- Bien entendu il est indispensable que les deux ordres **SELECT** :
 - produisent un même nombre de colonnes
 - que les types de données de chaque paires ordonnées de colonnes soient de même type (ou d'un type équivalent)

Opérations ensemblistes : l'intersection

■ Notation en SQL:

■ Avec la syntaxe SQL

- **Exemple : Les étudiants dont le nom contient un A ET commence par un B**

En algèbre relationnel :

$$R1 = \sigma_{(\text{nom} = '%A\%')} \text{Etudiants} \cap \sigma_{(\text{nom} = 'B\%')} \text{Etudiants}$$

En SQL :

```
SELECT nom FROM etudiants  
WHERE nom LIKE '%A%'  
INTERSECT  
SELECT nom FROM etudiants  
WHERE nom LIKE 'B%'
```

Les lignes dupliquées sont éliminées sauf si INTERSECT ALL est utilisé.

Opérations ensemblistes : L'UNION

- Notation en algèbre relationnel :

- $R_3 : R_1 \cup R_2$

- Notation en SQL

- $R_1 \text{ union } R_2$

Opérations ensemblistes : L'UNION

■ Notation en SQL:

■ Avec la syntaxe SQL

- **Exemple : Les étudiants dont le nom contient un A OU les étudiants dont le nom commence par un B**

- En algèbre relationnel :

$$R1 = \sigma(\text{nom} = '%A\%') \text{Etudiants} \cup \sigma(\text{nom} = 'B\%') \text{Etudiants}$$

- En SQL

```
SELECT nom FROM etudiants  
WHERE nom LIKE '%A%'  
UNION ALL  
SELECT nom FROM etudiants  
WHERE nom LIKE 'B%'
```

Les lignes dupliquées sont éliminées sauf si UNION ALL est utilisé.

A decorative graphic on the left side of the slide, featuring a vertical arrangement of colored dots (blue, yellow, black) and hexagons (blue, yellow) of various sizes.

Exercices

■ Soit les relations :

Journal(code_j, titre, prix, type, periodicite)

Depot(no-depot, nom-depot, adresse, ville, cp)

Livraison(no-depot, code-j, date-liv, quantite-livree)

■ Requêtes :

1. **Quel est le prix des journaux ?**
2. **Donnez tous les renseignements connus sur les journaux hebdomadaires ?**
3. **Donner les codes des journaux livrés à Bordeaux.**
4. **Donner les numéros des dépôts qui reçoivent des journaux**



Exercices

■ Soit les relations :

Journal(code_j, titre, prix, type, periodicite)

Depot(no-depot, nom-depot, adresse, ville, cp)

Livraison(no-depot, code-j, date-liv, quantite-livree)

■ Solutions des requêtes :

1. Quel est le prix des journaux ?

■ Algèbre

$R = \Pi_{(\text{titre}, \text{prix})} \text{Journal}$

■ SQL

```
SELECT titre, prix  
From journal ;
```




Exercices

■ Soit les relations :

Journal(code_j, titre, prix, type, periodicite)

Depot(no-depot, nom-depot, adresse, ville, cp)

Livraison(no-depot, code-j, date-liv, quantite-livree)

■ Requêtes :

2. Donnez tous les renseignements connus sur les journaux hebdomadaires ?

■ Algèbre

R : $\Pi_{(\text{titre}, \text{prix}, \text{type})} \sigma_{(\text{periodicite} = \text{'hebdomadaires'})} \text{journal}$

■ SQL

```
SELECT *  
FROM journal  
WHERE periodicite='hebdomadaires';
```



Exercices

■ Soit les relations :

Journal(code_j, titre, prix, type, periodicite)

Depot(no-depot, nom-depot, adresse, ville, cp)

Livraison(no-depot, code-j, date-liv, quantite-livree)

■ Requêtes :

3. Donner les codes des journaux livrés à Bordeaux.

■ Algèbre

R : $\Pi_{(\text{code_j})} \sigma_{(\text{ville}='Bordeaux')} (\text{depot} \bowtie_{\text{no-depot, no-depot}} \text{livraison})$

■ SQL

```
SELECT code_j
FROM Livraison L INNER JOIN Depot D
ON L.no-depot=D.no-depot
WHERE ville Like 'Bordeaux';
```



Exercices

■ Soit les relations :

Journal(code_j, titre, prix, type, periodicite)

Depot(no-depot, nom-depot, adresse, ville, cp)

Livraison(no-depot, code-j, date-liv, quantite-livree)

■ Requêtes :

4. Donner les numéros des dépôts qui reçoivent des journaux

■ Algèbre

R : $\Pi_{(\text{no_depot})} \text{ depot}$

■ SQL

```
SELECT distinct (no_depot)
FROM Livraison;
```