

UNIVERSITE DE CORSE

Master DFS-DE 1^{ère} année

2025-2026

Bases de données et Optimisation

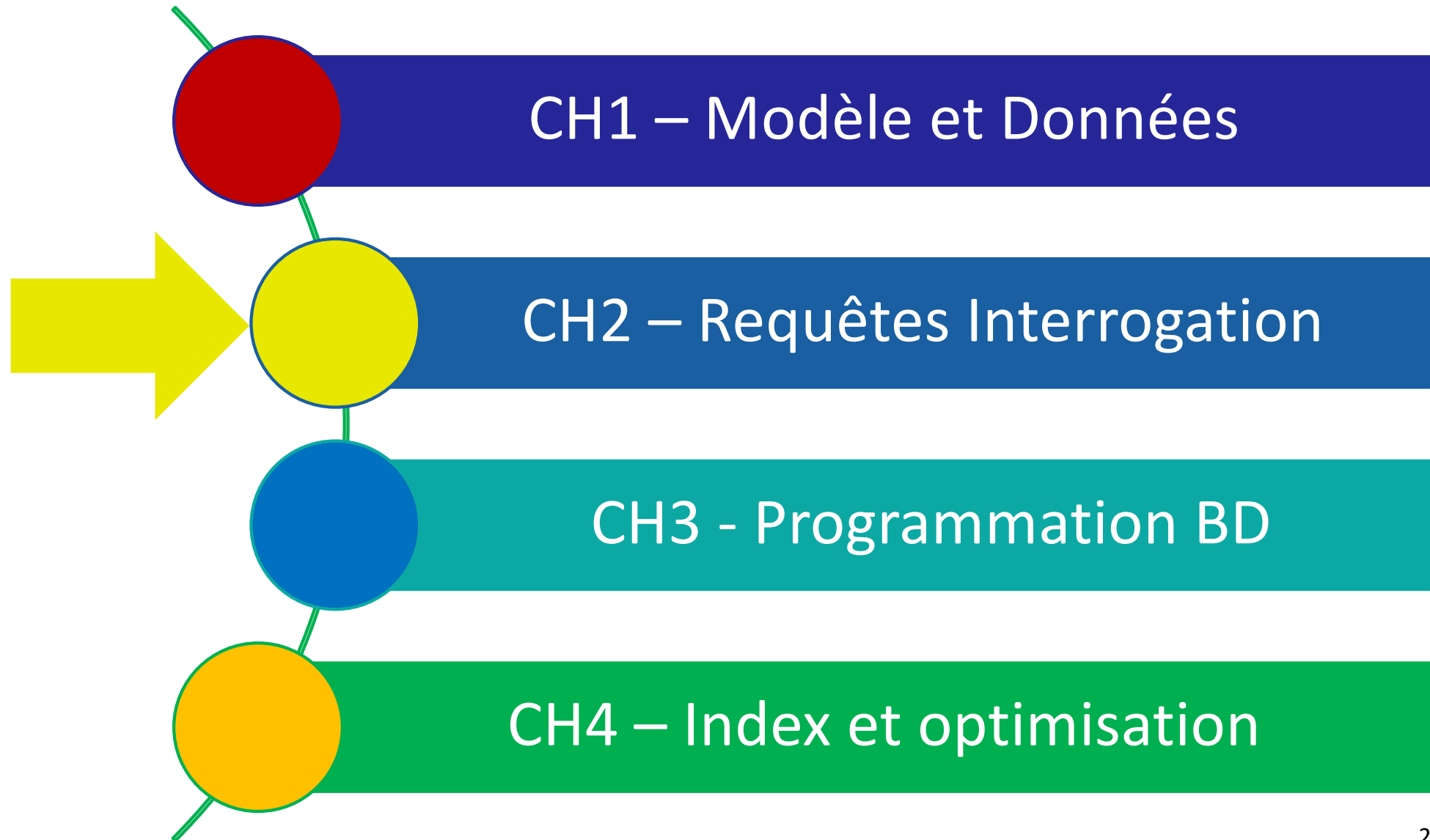
CH2 – Requêtes d'interrogation



Evelyne VITTORI
vittori@univ-corse.fr

[plan SQL](#)

Optimisation et Bases de données



CH2- Requêtes d'Interrogation

Objectifs

- Revoir et comprendre les opérateurs de l'algèbre relationnelle et leur rôle dans un SGBD
- Maîtriser la définition de requêtes d'interrogation SQL complexes



Questions + Exercices pour s'évaluer

Toutes les diapos ne seront pas présentées pendant les séances



1. Algèbre relationnelle
2. Interrogation en SQL

- Sélection-Projections
- Jointures
- Sous-Requêtes
- Partitionnement

- Clause CASE ...WHEN
- Vues
- Vues Matérialisées
- Common Table Expressions (CTE) (clause WITH)
- Manipulation Dictionnaire de données
- Fenêtres SQL



Rappels



Question

Langage de requête relationnel

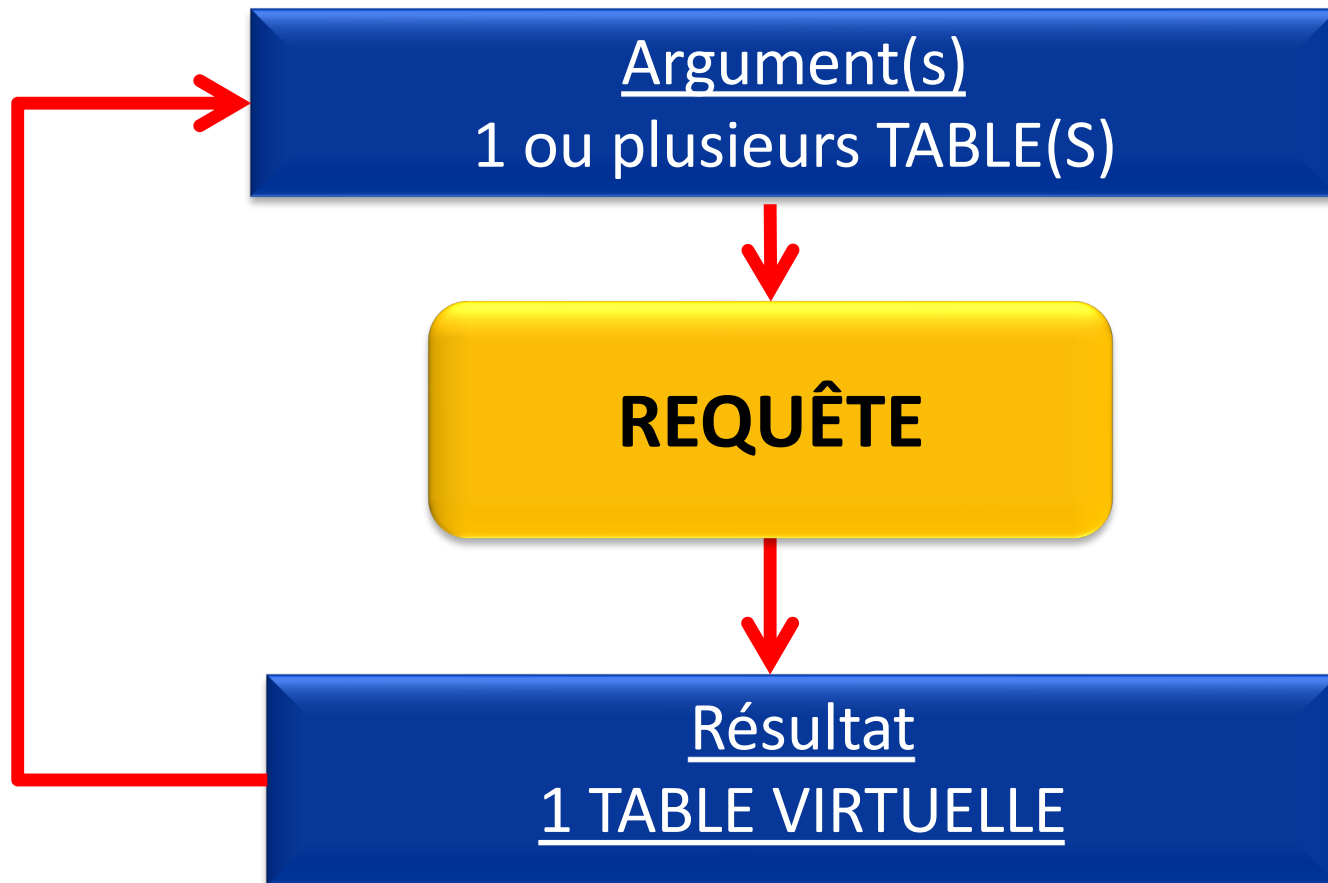


Le langage algébrique relationnel et le langage d'interrogation SQL sont des langages

- Déclaratifs
- Fermés

Qu'est-ce que cela signifie?

Langages de requête **fermés**



Langages de requêtes **déclaratifs**

- Le développeur spécifie uniquement les **résultats attendus** sans décrire la démarche de recherche
- Le développeur ne connaît pas les algorithmes d'exécution

L'optimisation est automatique et assurée par le SGBD

Optimiseur Cost-Based Optimizer
sous Oracle et postGres



Algèbre relationnelle



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

plan SQL

Algèbre relationnelle

- Opérateurs associés au modèle relationnel

Vocabulaire à connaître

- Base d'évaluation des langages relationnels

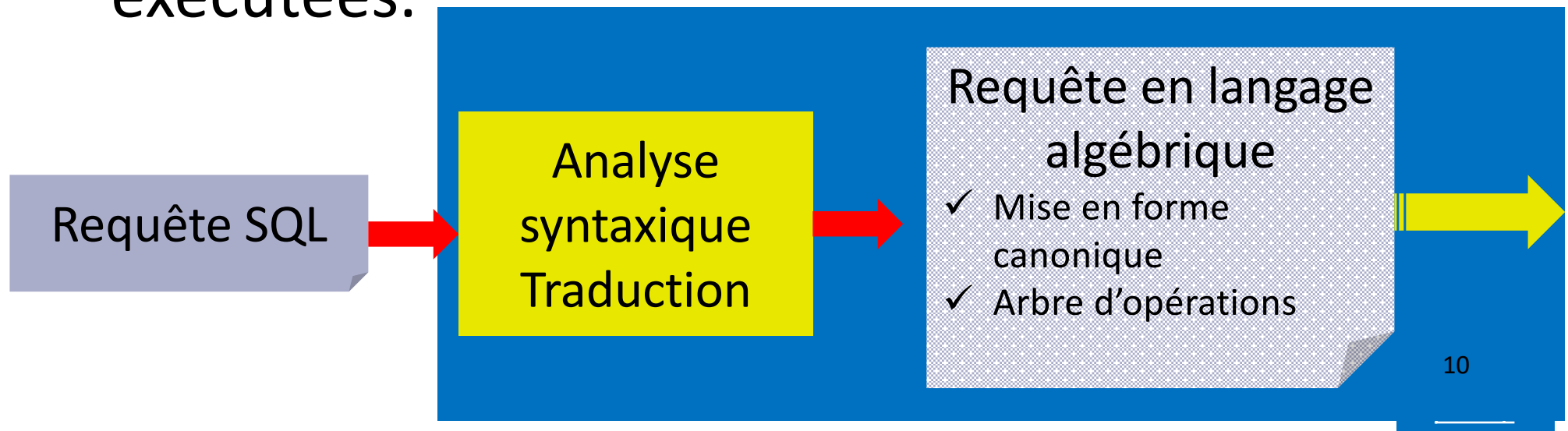
- Pourquoi l'étudier?

Pour mieux comprendre la construction des requêtes SQL et surtout leur exécution

Important pour optimiser

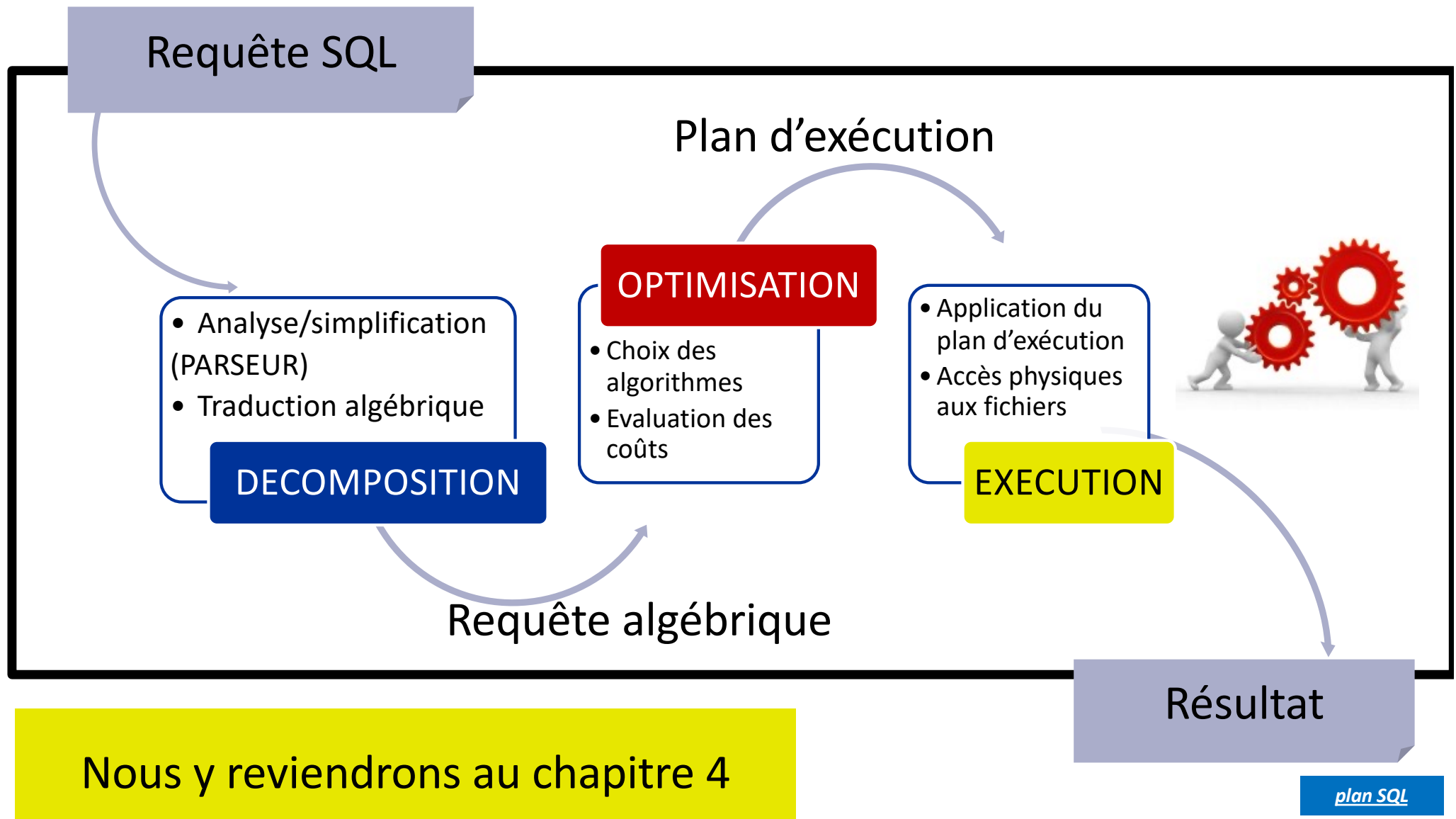
Algèbre relationnelle et SGBDr

- Les opérateurs de l'algèbre relationnelle sont au cœur du **fonctionnement interne** d'un SGBD relationnel.
- Les requêtes SQL sont **traduites en requêtes algébriques** avant d'être optimisées puis exécutées.



Algèbre relationnelle et SGBDr

Etapes d'exécution d'une requête





Question

Opérateurs

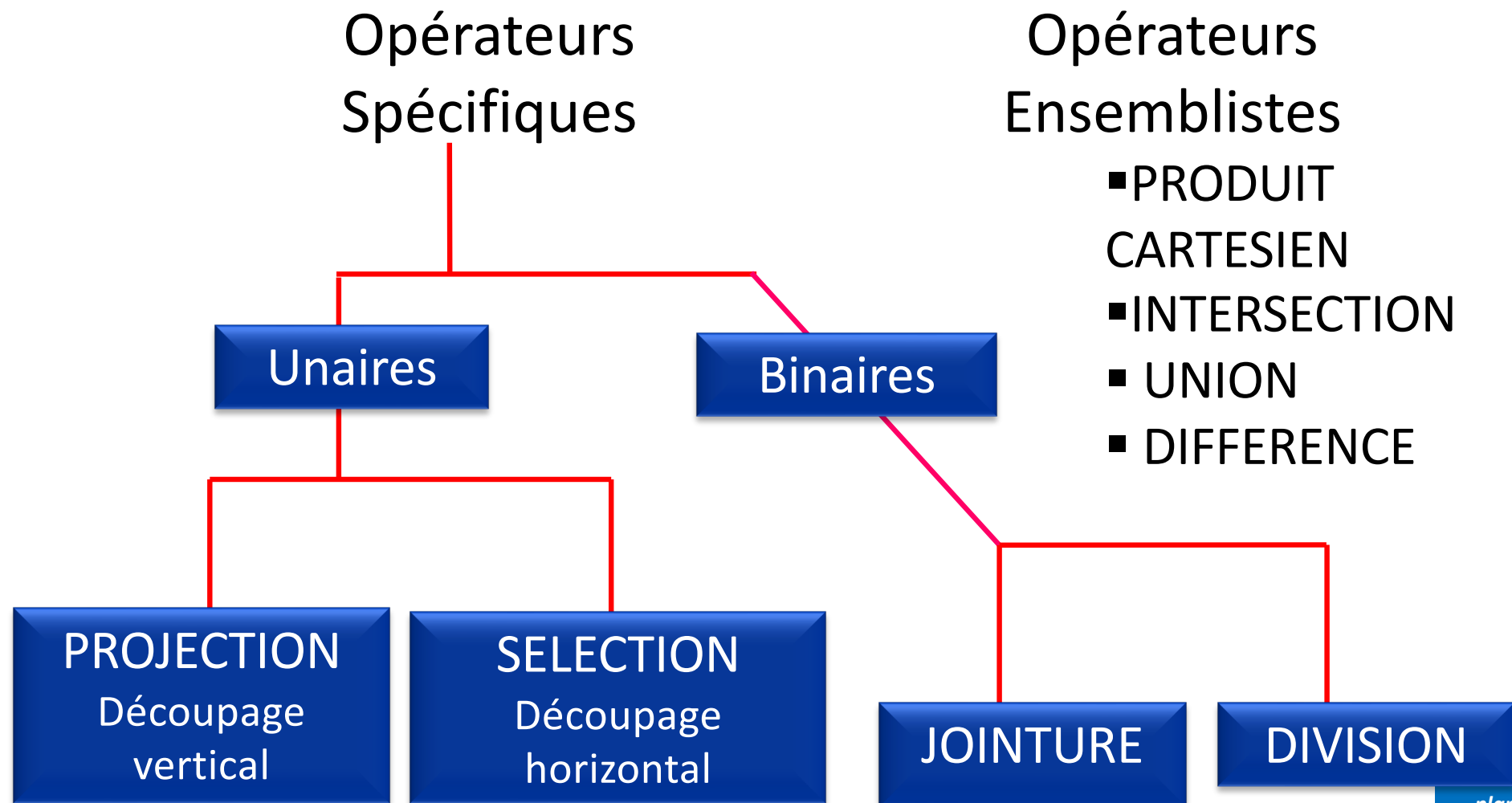
Algébriques



1. Quels sont les trois principaux opérateurs algébriques?
2. Quels sont les autres opérateurs algébriques?
3. Comment représenter une requête algébrique sous la forme d'un arbre?

Opérateurs de l'algèbre Relationnelle

Requête= Séquence ordonnée d'opérations



Exemples de requêtes

■ Sélection/Projection

Noms des bateaux de capacité supérieure ou égale à 10?

R1=SELECT BATEAUX (CAPACITE>=10)


R2=PROJECT R1 (BATNOM)

		R2			
		BATNUM	BATNOM	BATPORT	CAPACITE
R1	BATEAUX	B001	LIBERTE	ANTIBES	10
		B002	LOUISIANE	BASTIA	12
		B003	KALISTE	BASTIA	6
		B004	INDEPENDANCE	CALVI	6

JOINTURE approche intuitive


Quels sont les noms des skippers qui assurent une ou plusieurs croisières au départ de BASTIA?

1. Sélection dans CROISIERES (DEPPORT= "BASTIA")
2. Jointure entre CROISIERES et SKIPPERS avec SKNUM=SKNUM
3. Projection de SKNOM

 CROISIERES	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT
	C001	1	B002	BASTIA	CALVI
	C002	3	B002	ANTIBES	AJACCIO
	C003	1	B003	AJACCIO	BASTIA
	C004	3	B004	BASTIA	MARSEILLE

Jointure entre CROISIERES et SKIPPERS

Association des tuples des 2 relations en fonction d'un critère de comparaison entre un attribut de CROISIERES (SKNUM) et un attribut de SKIPPERS (SKNUM)

 SKIPPERS	SKNUM	SKNOM	SKPORT
	1	JEAN	AJACCIO
	2	PAUL	AJACCIO
	3	MARINE	ANTIBES
	4	PIERRE	BASTIA

SKNOM
JEAN
MARINE

15

JOINTURE approche formelle

Quels sont les noms et adresses des adhérents ayant emprunté un livre?

JOINTURE = PRODUIT CARTESIEN + SELECTION

ADHERENTS	NUM	NOM	AGE	ADRESSE
	1	DUPONT	31	AJACCIO
	2	DURAND	25	PARIS
	3	DUPUIS	20	GRENOBLE

EMPRUNTS	EMPNUM	ADHNUM	LIVNUM	DATE
	101	1	10	2003
	102	1	11	2005
	103	2	45	2007

PRODUIT CARTESIEN ADHERENTS X EMPRUNTS

NUM	NOM	AGE	ADRESSE	EMPNUM	ADHNUM	LIVNUM	DATE
1	DUPONT	31	AJACCIO	101	1	10	2003
1	DUPONT	31	AJACCIO	102	1	11	2005
1	DUPONT	31	AJACCIO	103	2	45	2007
2	DURAND	25	PARIS	101	1	10	2003
2	DURAND	25	PARIS	102	1	11	2005
2	DURAND	25	PARIS	103	2	45	2007
3	DUPUIS	20	GRENOBLE	101	1	10	2003
3	DUPUIS	20	GRENOBLE	102	1	11	2005
3	DUPUIS	20	GRENOBLE	103	2	45	2007

SELECTION des tuples vérifiant NUM= ADHNUM

Exemples de requêtes

■ Equi-Jointure

Noms des skippers qui assurent une ou plusieurs croisières?

R1=JOIN SKIPPERS (SKNUM=SKNUM) CROISIERES

R2=PROJECT R1 (SKIPPERS.SKNOM)

Jointure= sous-ensemble du produit cartésien

R2							
SKNUM	SKNOM	SKPORT	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
3	MARINE	ANTIBES	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	MARINE	ANTIBES	C004	3	B004	BASTIA	MARSEILLE

Exemple de requête

- Théta-Jointure

Quels sont les noms des skippers dont le salaire est supérieur à celui de Marine ?

R1= **SELECT** SKIPPERS(SKNOM= "Marine")

R2= **PROJECT** R1(SALAIRE)

R3= **JOIN** SKIPPERS(SALAIRE>SALAIRE) R2

R4= **PROJECT** R3(SKNOM)

DIVISION

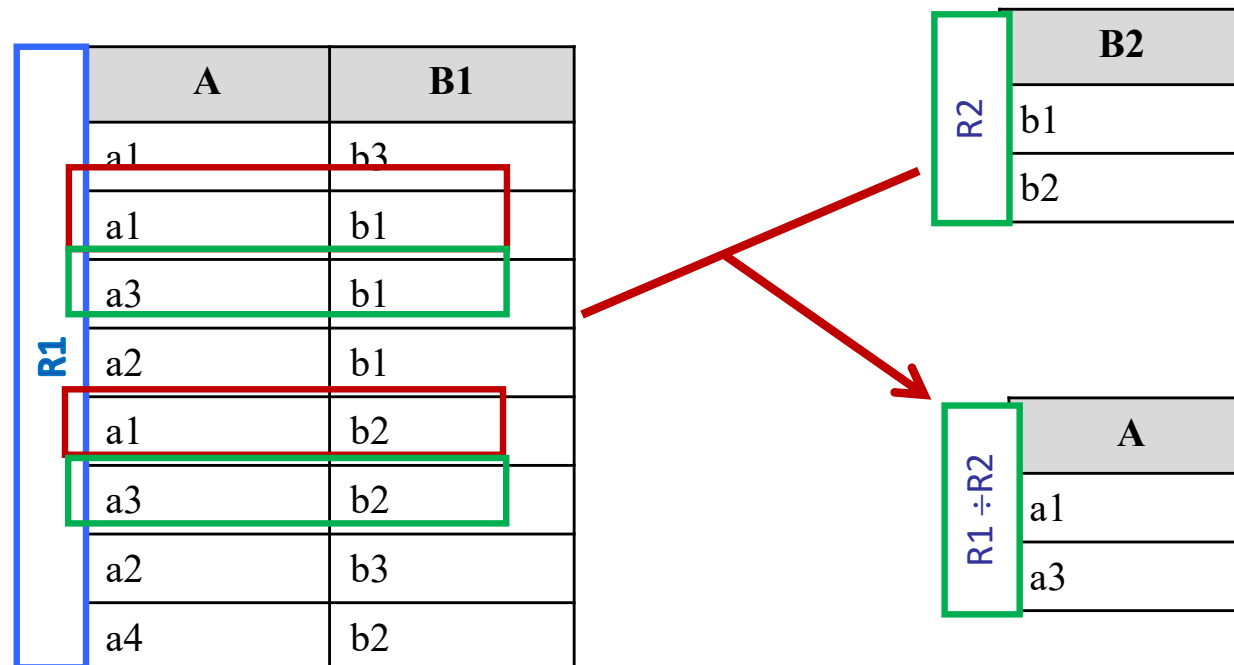
$$RES = R1(A, B1) \div R2(B2)$$

A et B1 sont des attributs de R1

B2 est un attribut de R2

B1 et B2 sont définis sur le même domaine

La relation résultat RES contient les valeurs de l'attribut A qui sont associées à toutes les valeurs de B2 dans la relation R1.



Exemple de requête

- Division

Quels sont les numéros des skippers qui assurent des croisières sur tous les bateaux localisés à AJACCIO?

R1=SELECT BATEAUX (BATPORT= "AJACCIO")

R2=PROJECT R1 (BATNUM)

R3=PROJECT CROISIERES (SKNUM,BATNUM)

R4= R3(SKNUM,BATNUM) \div R2(BATNUM)

Algèbre relationnelle

OPERATEUR	REQUETE	Structure de RES	Tuples de RES
SELECTION	RES=SELECT R(condition)	Attributs de R	Tuples de R vérifiant la condition
PROJECTION	RES=PROJECT R(A1, A2, ..)	A1, A2, ...	Tuples de R restreints à A1, A2, ...
Elimination des redondances			
JOINTURE	RES=JOIN R1(A1 op A2)R2	Attributs de R1 + attributs de R2	Tuples du produit cartésien R1xR2 vérifiant (A1 op A2)
DIVISION	RES=R1 (A,B1) ÷ R2(B2)	A	Valeurs de A associées à toutes les valeurs de B2 dans R1

Algèbre relationnelle

OPERATEUR	REQUETE	Structure de RES	Tuples de RES
UNION	$RES=R1 \cup R2$	Même structure que R1 (et R2)	Tous les tuples de R1 + tous les tuples de R2
INTERSECTION	$RES=R1 \cap R2$	Même structure que R1 (et R2)	Tuples présents à la fois dans R1 et dans R2
DIFFERENCE	$RES=R1 - R2$	Même structure que R1 (et R2)	Tuples présents dans R1 mais pas dans R2

R1 et R2 doivent être Union-Compatibles

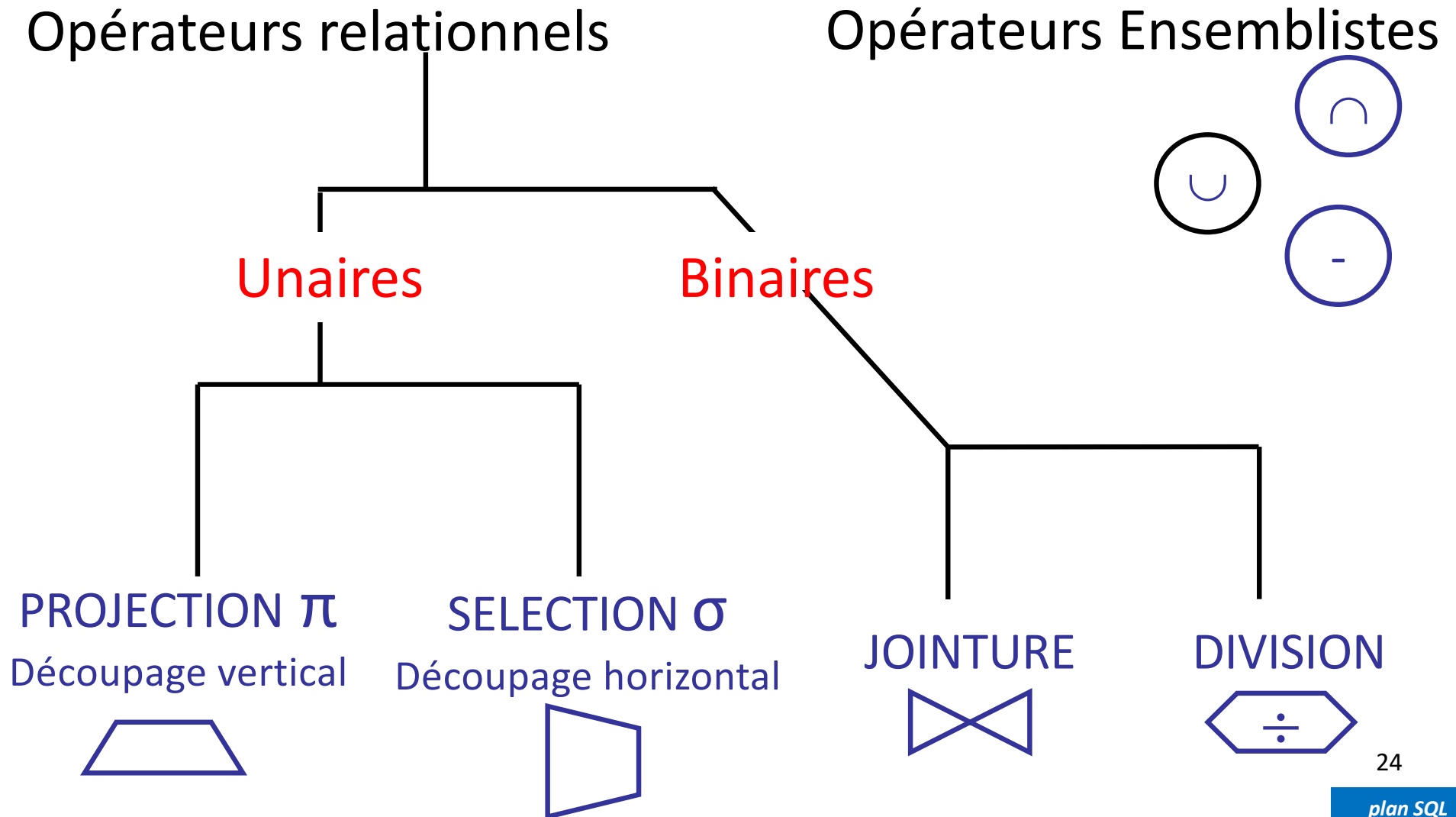
- Même degré (même nombre d'attributs)
- Les attributs de même rang dans chaque relation sont définis sur le même domaine

PRODUIT CARTESIEN	$RES= R1 \times R2$	Attributs de R1 + attributs de R2	
-------------------	---------------------	-----------------------------------	--

Algèbre relationnelle: Les points à retenir

- Notion d'attributs comparables (même domaine)
- Plusieurs types de jointures
 - Equi-jointure: égalité
 - Théta-Jointure: autres opérateurs ($<$, $>$, ...)
- Notion de relation union-compatible
 - Opérateurs ensemblistes

Représentation graphique arborescente



Arbre de représentation d'une requête

Quels sont les noms des skippers qui gagnent moins de 2000 euros et qui assurent des croisières un ou plusieurs bateaux localisés à AJACCIO?

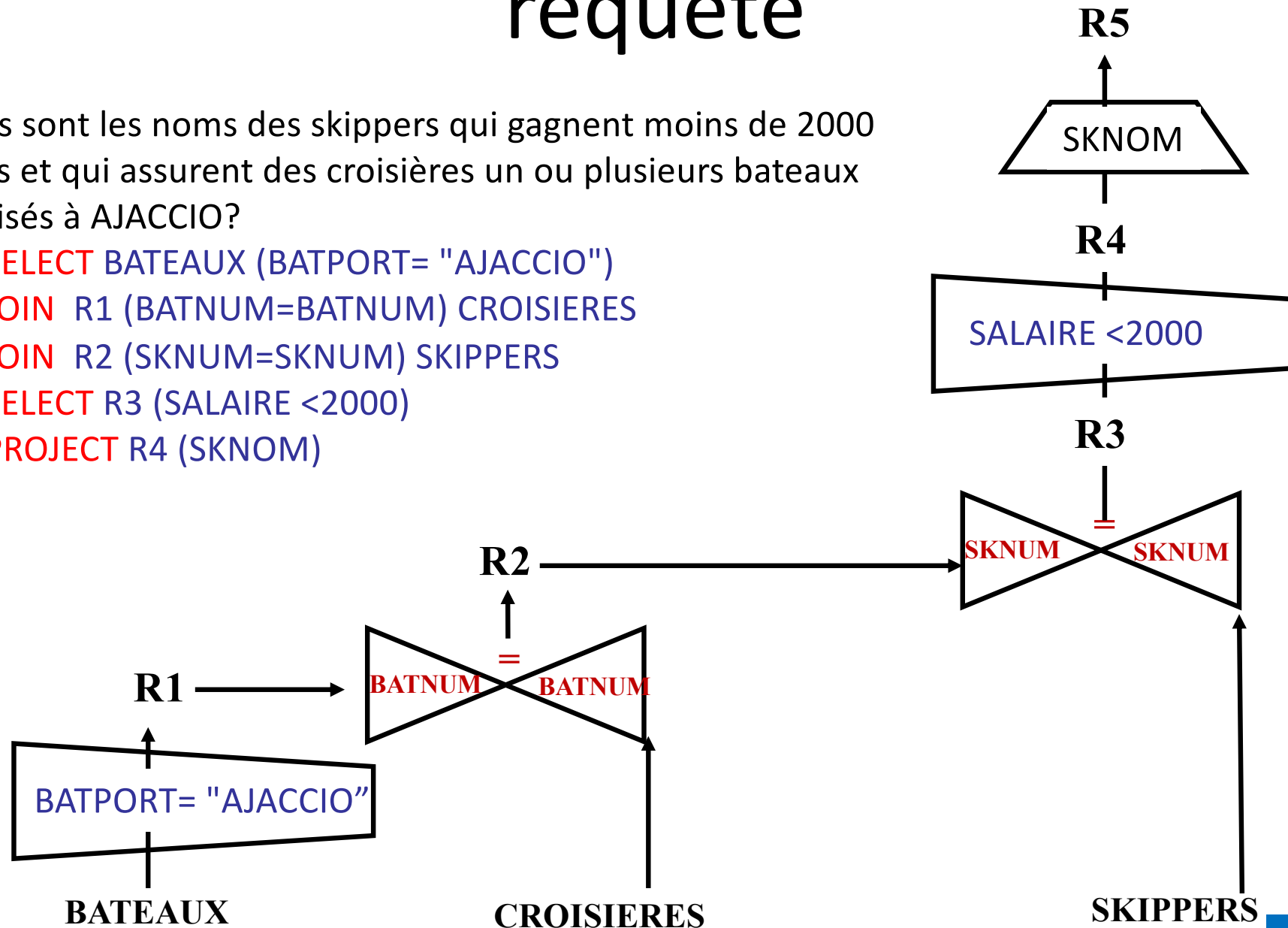
R1=SELECT BATEAUX (BATPORT= "AJACCIO")

R2=JOIN R1 (BATNUM=BATNUM) CROISIERES

R3=JOIN R2 (SKNUM=SKNUM) SKIPPERS

R4=SELECT R3 (SALAIRE <2000)

R5=PROJECT R4 (SKNOM)

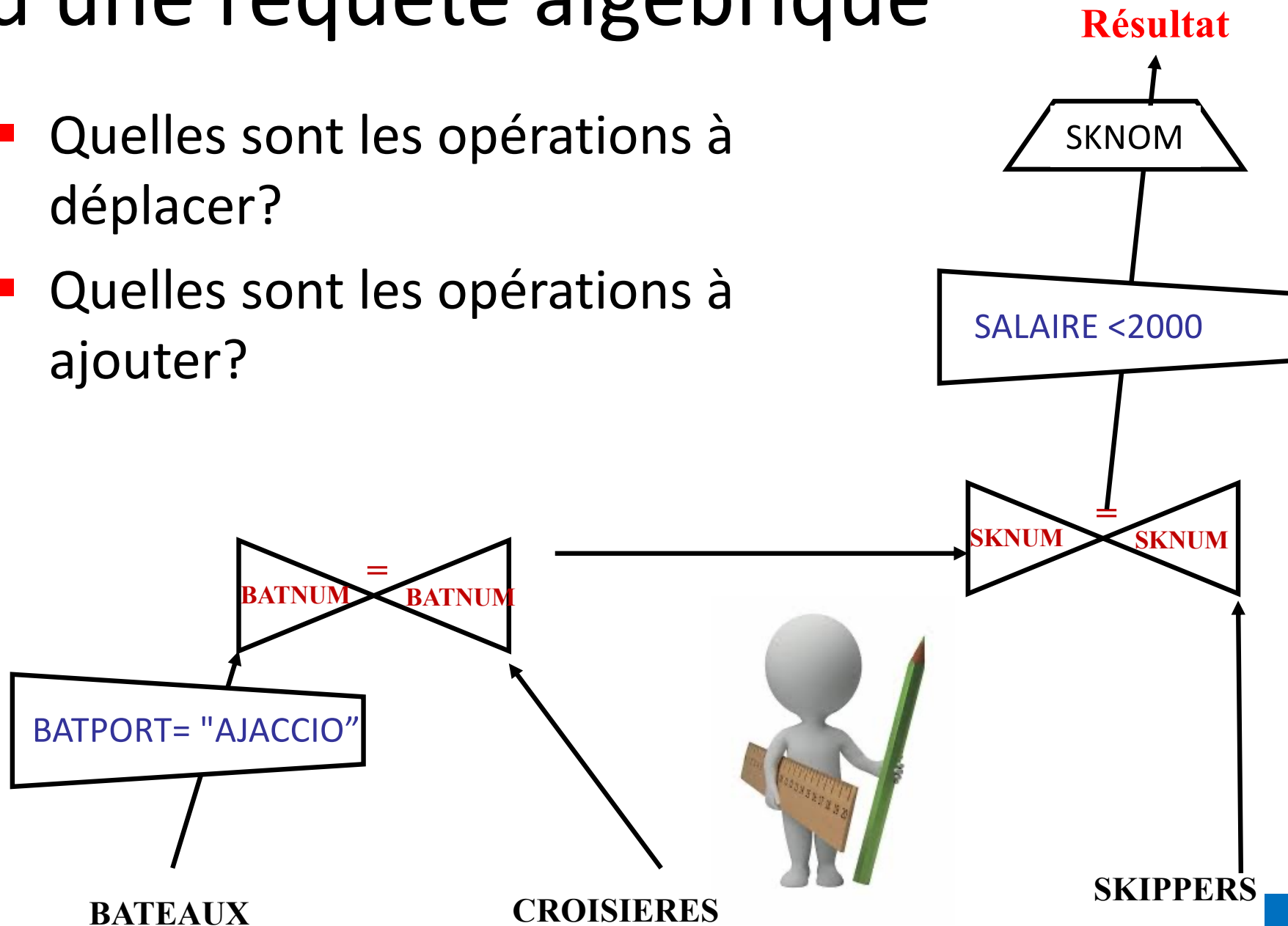


Algèbre relationnelle et SGBDr

- Un principe **d'optimisation** utilisé par les optimiseurs intégrés dans les SGBD:
Restreindre au maximum la taille des tables avant de réaliser une jointure
- Concrètement: Placer les **sélections** et les **projections** **avant les jointures**
 - Restriction horizontale: selections
 - Restriction verticale: projections

Exercice : Optimisation d'une requête algébrique

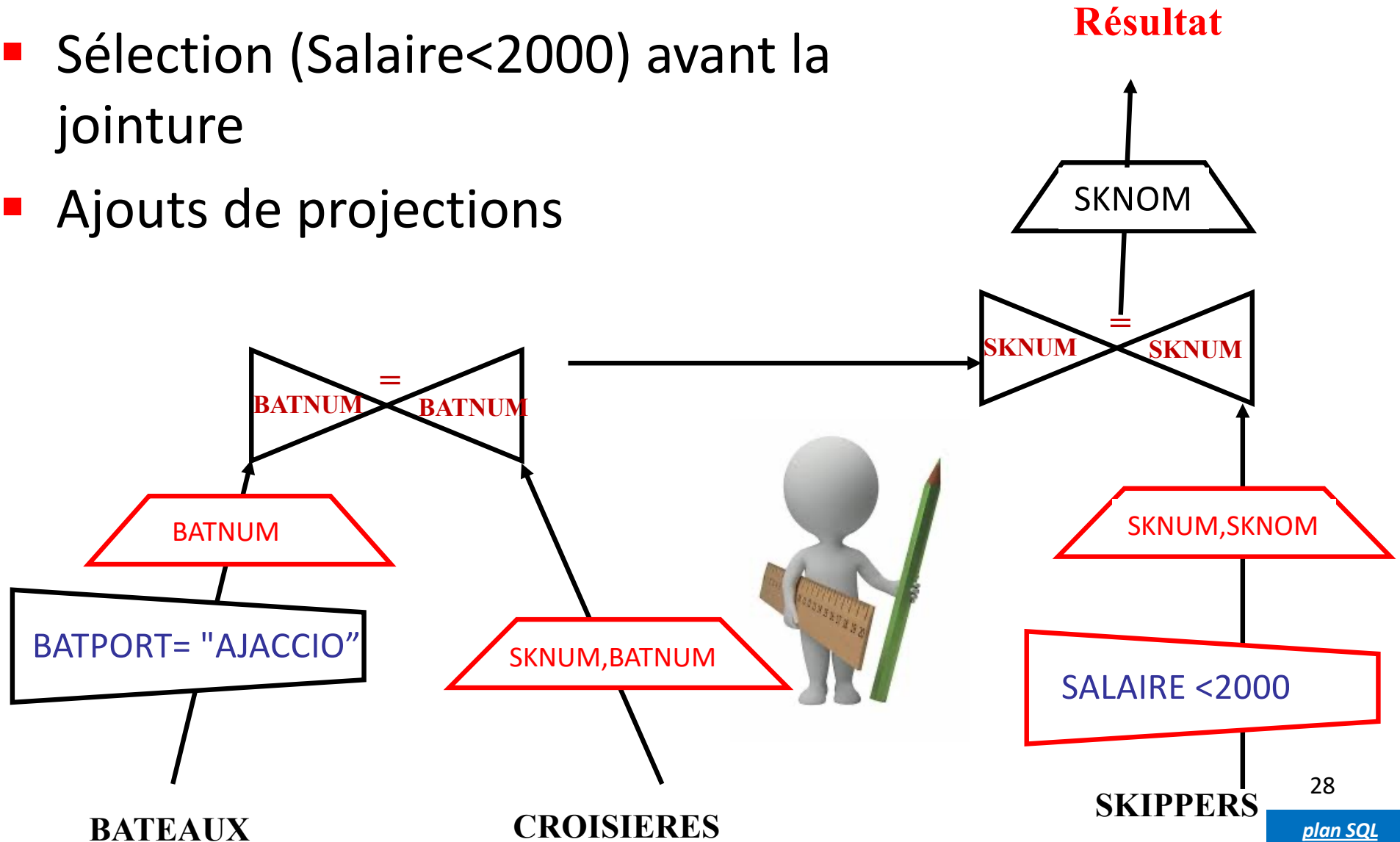
- Quelles sont les opérations à déplacer?
- Quelles sont les opérations à ajouter?



Exercice : Optimisation d'une requête algébrique

correction

- Sélection (Salaire<2000) avant la jointure
- Ajouts de projections





L'interrogation des données en Langage SQL

Requêtes SFW

SELECT FROM WHERE

DML
Data
Manipulation
language



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

plan SQL

Le langage SQL

Principales Commandes

DDL



Administrateur

Commandes de Définition des
structures de données
(Tables, Index, Vues)

- CREATE création
- ALTER modification
- DROP suppression



Administrateur

Commandes de Gestion du
contrôle des données

Droits d'accès, reprise sur
panne

DCL

Commandes de Manipulation
des données

DML

INTERROGATION

SELECT FROM WHERE
recherche

MISES A JOUR

- INSERT insertion
- UPDATE modification
- DELETE suppression



Utilisateurs



Cours SQL et Exercices



■ Rappels

- Sélection-Projections
 - Jointures
 - Sous-Requêtes
 - Requêtes de Partitionnement
- Exercice 1 – Jointures et résultats
 - Exercice 2 - LEFT-RIGHT
 - Exercice 3 – Sous-Requetes
 - Exercice 4 – Group BY
 - Exercice 5 – Requetes Bilan
 - Exercice 6 – CASE –WHEN
 - Exercice 7 – Vues
 - Exercice 8 – Vues matérialisées
 - Exercice 9 – CTE
 - Exercice 10 - DD
 - Exercice 11 - Fenêtres
- Requêtes avec CASE ...WHEN
 - Vues
 - Vues matérialisées
 - Common Table Expressions (CTE)
 - Requêtes sur le DD
 - Fenêtres SQL

Base de données Croisieres



SKIPERS	SKNUM	SKNOM	SKPORT
	1	Nivet	Bastia
	2	Delhom	Ajaccio
	3	Poggi	Calvi
	4	Santucci	Ajaccio

BATEAUX	BATNUM	BATNOM	BATPORT	CAPACITE
	B1	LIBERTE	Ajaccio	10
	B2	LOUISIANE	Bastia	12
	B3	KALISTE	Calvi	6
	B4	INDEPENDANCE	Ajaccio	6

CROISIERS	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT
	C001	1	B1	Ajaccio	Bastia
	C002	2	B2	Bastia	Nice
	C003	4	B1	Calvi	Paris
	C004	1	B4	Ajaccio	Nice



Requêtes d'interrogation des données

Sélections/Projections

DML
Data
Manipulation
language

Requêtes de sélection/Projection

Liste des attributs projetés

** Pour tous les attributs*

distinct pour éliminer les répétitions

```
SELECT A1, A2, ...  
FROM T  
WHERE condition ;
```

Table (*Relation*) concernée

Condition de sélection

- opérateurs de comparaison = < > <= > >=
- connecteurs logiques AND, OR, NOT
- IN (ex: Ville IN ("AJACCIO", "BASTIA"))
- BETWEEN (ex: Salaire BETWEEN 1000 and 4000)
- LIKE (comparaison de chaînes)

Requêtes de projections simples

- Quels sont les noms et ports d'attache des bateaux de l'entreprise?

```
SELECT BATNOM, BATPORT  
FROM BATEAUX ;
```

- Quels sont les ports d'attache des bateaux de l'entreprise?

```
SELECT distinct BATPORT  
FROM BATEAUX ;
```

- Quels sont toutes les caractéristiques des bateaux?

```
SELECT *  
FROM BATEAUX;
```

Requêtes de sélection simples

- Quels sont les bateaux dont le port d'attache est AJACCIO?

```
SELECT *  
FROM BATEAUX  
WHERE BATPORT='AJACCIO' ;
```

- Quels sont les noms des bateaux localisés au port d'AJACCIO pouvant accueillir au moins 10 passagers?

```
SELECT BATNOM  
FROM BATEAUX  
WHERE BATPORT='AJACCIO' AND CAPACITE>=10 ;
```

- Quels sont les numéros des croisières au départ de AJACCIO ou de BASTIA?

```
SELECT CROISNUM  
FROM CROISIERES  
WHERE DEPPORT='AJACCIO' OR DEPPORT='BASTIA' ;
```

Requêtes de sélection avec opérateurs ensemblistes

BETWEEN permet la définition d'intervalles de sélection

- Quels sont les skippers dont le nom est différent de Marine et dont le salaire est compris entre 1500 et 2500 euros?

```
SELECT      *  
FROM        SKIPPERS  
WHERE       SKNOM < > 'Marine' AND  
            (SALAIRE BETWEEN 1500 AND 2500) ;
```

IN permet d'exprimer l'appartenance à un ensemble de valeurs

- Quels sont les numéros des croisières dont le port de départ appartient à la liste suivante (AJACCIO, MARSEILLE, MENTON, BASTIA)?

```
SELECT      CROISNUM  
FROM        CROISIERES  
WHERE       DEPPORT IN ('AJACCIO','MARSEILLE', 'MENTON ', 'BASTIA');
```

Requêtes de sélection avec opérateur LIKE

LIKE permet de comparer des chaînes de caractères avec des modèles incluant des symboles spéciaux:

- le symbole **_** (SQL Standard) ou **?** (SQL Access)
remplace un caractère quelconque
- le symbole **%** (SQL Standard) ou ***** (SQL Access)
remplace 0 ou plusieurs caractères

Exemples de conditions (Oracle, MySQL, Postgres):

Nom **LIKE** 'B%' *'Noms commençant par B*

Nom **LIKE** 'C_N%' *'Noms commençant par C
avec un N en 3ème lettre*

Nom **LIKE** '%T' *'Noms se terminant par un T*

Sous Postgresql,
opérateur ILIKE
non sensible à la
casse

Requêtes de sélection avec opérateur LIKE

- Quels sont les skippers habitant AJACCIO et dont le nom comporte la lettre A en troisième position?

```
SELECT      *  
FROM        SKIPPERS  
WHERE       SKPORT='AJACCIO' AND SKNOM LIKE ' __A%';
```

- Quels sont les numéros des bateaux dont le nom ne commence pas par la lettre L?

```
SELECT      BATNUM  
FROM        BATEAUX  
WHERE       BATNUM NOT LIKE 'L%';
```



Requêtes de sélection avec valeurs nulles



- Quels sont les numéros des croisières dont la date d'arrivée n'est pas définie?

```
SELECT    CROISNUM
FROM      CROISIERES
WHERE     ARRDATE IS NULL;
```

null ne signifie
pas toujours
vide

- Quels sont les numéros des bateaux localisés au port de AJACCIO et dont la capacité est indéterminée?

```
SELECT    BATNUM
FROM      BATEAUX
WHERE     BATPORT='AJACCIO' AND CAPACITE IS NULL ;
```


Requêtes de tri

```
SELECT liste attributs  
FROM liste tables  
WHERE condition(s)  
ORDER BY A1 DESC, A2 ASC, ... ;
```

Tri du résultat par ordre
Décroissant sur l'attribut A1

Tri du résultat par ordre
Croissant sur l'attribut A2

- Quels sont les caractéristiques des croisières classées par port de départ et pour un même port de départ par ordre décroissant de date de départ?

```
SELECT *  
FROM CROISIERES  
ORDER BY DEPPORT, DEPDATE DESC ;
```

Requêtes avec Calculs et Fonctions statistiques

SELECT attribut(s) ← **Calculs, Fonctions statistiques**
FROM table(s)
WHERE condition(s) sur les tuples;

Calculs

Expression sur des attributs: **SELECT** SALAIRE * 0,1

Fonctions Statistiques

- **SUM**(Attribut) : *somme des valeurs de la colonne*
- **AVG**(Attribut) : *moyenne des valeurs de la colonne*
- **MAX**(Attribut) : *valeur maximale de la colonne*
- **MIN**(Attribut) : *valeur minimale de la colonne*
- **COUNT**(Attribut) ou **COUNT**(*)
: *nombre de lignes* de la colonne
(ou de la table)

Requêtes avec Calculs et Fonctions statistiques

Donner la somme des salaires des Skippers

```
SELECT SUM(SALAIRE) AS [Somme Salaires]  
FROM SKIPPERS;
```

Nom de la nouvelle colonne



Donner le nombre de bateaux ayant une capacité supérieure à 8.

```
SELECT COUNT( * ) AS NombreBateaux  
FROM BATEAUX  
WHERE CAPACITE > 8 ;
```





Requêtes d'interrogation des données

Jointures

DML
Data
Manipulation
language



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

plan SQL

Requêtes de jointure

SELECT liste attributs

FROM T1, T2 , ...

WHERE T1.A1 op. T2.A2 ;

Tables cibles

Condition de jointure

- Quels sont les noms des skippers qui assurent une ou plusieurs croisières ?

SELECT distinct sknom

FROM skipper, croisiere

WHERE skipper.sknum=croisiere.sknum ;



Autre écriture: InnerJoin (ou join)

- Jointure interne (notation introduite en SQL2)

```
SELECT liste attributs  
FROM T1 INNER JOIN T2 ON T1.A1=T2.A2
```

ou

```
SELECT liste attributs  
FROM T1 INNER JOIN T2 USING (A1)
```

Si les colonnes
de jointure ont
le même nom
A1 dans les deux
tables

- Quels sont les noms des skippers qui assurent une
ou plusieurs croisières ?

```
SELECT distinct sknom  
FROM skipper INNER JOIN croisiere  
ON skipper.sknum=croisiere.sknum;
```

```
SELECT distinct sknom  
FROM skipper INNER  
JOIN croisiere  
USING(sknum);
```

Requêtes de jointure

- Quel est le résultat de cette requête?

SELECT sknom

FROM skipper, croisiere;

WHERE skipper.sknum=croisiere.sknum

ATTENTION à ne pas oublier
la condition de jointure

croisn u m	sknum	batnum	depport	arrport	sknum	sknom	skport
C001	1	B002	BASTIA	CALVI	1	JEAN	AJACCIO
C002	3	B002	ANTIBES	AJACCIO	3	LAURA	ANTIBES
C003	1	B003	AJACCIO	BASTIA	1	JEAN	AJACCIO
C003	1	B003	AJACCIO	BASTIA	2	PAUL	AJACCIO
C003	1	B003	AJACCIO	BASTIA	3	LAURA	ANTIBES
C003	1	B003	AJACCIO	BASTIA	4	PIERRE	BASTIA
C004	3	B004	BASTIA	MARSEILLE	1	JEAN	AJACCIO
C004	3	B004	BASTIA	MARSEILLE	2	PAUL	AJACCIO
C004	3	B004	BASTIA	MARSEILLE	3	LAURA	ANTIBES
C004	3	B004	BASTIA	MARSEILLE	4	PIERRE	BASTIA

Jointure de 3 tables

Quels sont les noms des bateaux barrés par Jean?

```
SELECT batnom  
FROM skipper, bateau, croisiere  
WHERE skipper.sknum=croisiere.sknum  
AND bateau.batnum=croisiere.batnum  
AND sknom="jean"
```

Equi-jointures:
jointures avec
condition =

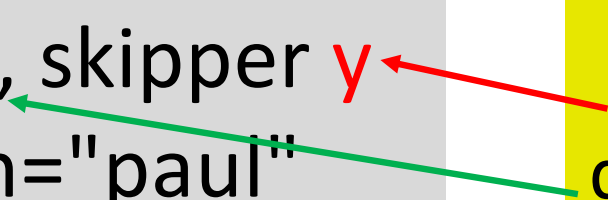
3 tables : 2 jointures
Donc 2 conditions de jointure

Auto-Jointure

Quels sont les noms des skippers qui ont le même salaire que Paul?

```
SELECT y.sknom  
FROM skipper x, skipper y  
WHERE x.sknom="paul"  
AND x.salaire=y.salaire  
AND y.sknom<>"paul"
```

Variables
de parcours
des tables

A yellow rectangular box containing the text 'Variables de parcours des tables'. Two arrows originate from this box: a red arrow points to the variable 'y' in the 'FROM skipper x, skipper y' line of the SQL query, and a green arrow points to the variable 'x' in the 'WHERE x.sknom="paul"' line.

Autre écriture: Jointure naturelle

- Jointure naturelle (attributs ayant le même nom dans les deux tables)

```
SELECT liste attributs  
FROM T1 NATURAL JOIN T2
```

- Quels sont les noms des skippers qui assurent une ou plusieurs croisières ?

```
SELECT distinct sknom  
FROM skipper NATURAL JOIN croisiere;
```

Notation la plus simple mais **attention** dangereuse parce que uniquement basée sur les noms des colonnes

Exercice 1 – Requêtes et résultats

Q0 – On considère la requête suivante:

```
SELECT *  
FROM  SKIPPERS, CROISIERES;
```



1. Définissez la table résultat de cette requête
 - Combien a-t-elle de colonnes?
 - Combien a-t-elle de lignes.
2. S'agit-il d'une requête de jointure?

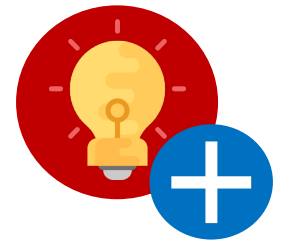
Exercice 1 – Requêtes et résultats

Q1 – On considère la requête suivante:

```
SELECT *  
FROM  BATEAUX, CROISIERES  
WHERE CROISIERES.BATNUM=  
      BATEAUX.BATNUM ;
```



1. Définissez la table résultat de cette requête
2. Identifiez la question à laquelle répond cette requête
3. Donnez une écriture équivalente en utilisant la clause INNER JOIN
4. Est-il possible d'utiliser la clause NATURAL JOIN?



Jointures

53

Exercice 1 – Requêtes et résultats

Q2 – On considère la requête suivante:

```
SELECT *  
FROM  BATEAUX, CROISIERES  
WHERE BATPORT=DEPPORT;
```



1. Définissez la table résultat de cette requête
2. Que devient la table résultat si l'on remplace la première ligne par
SELECT BATEAUX.BATNUM
puis par SELECT **DISTINCT** BATEAUX.BATNUM
3. Identifiez la question à laquelle répond cette requête
4. Donnez une écriture équivalente en utilisant la clause INNER JOIN
5. Est-il possible d'utiliser la clause NATURAL JOIN?



Jointures

54

Exercice 1 : Requêtes et résultats



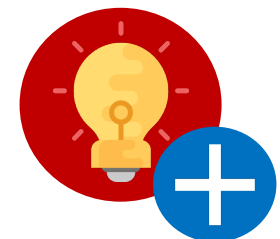
Q3 – On considère la requête suivante:

```
SELECT CROISNUM
```

```
FROM SKIPPERS, CROISIERES
```

```
WHERE SKPORT=DEPPORT AND SKNOM="Poggi";
```

1. Définissez la table résultat de cette requête
2. Identifiez la question à laquelle répond cette requête



Jointures externes:

Left Right Outer Join

Permet de préciser que l'on garde tous les tuples d'une des deux tables (même ceux qui ne vérifient pas la condition)

LEFT OUTER JOIN :

on garde tous les tuples de la table de gauche

RIGHT OUTER JOIN :

on garde tous les tuples de la table de droite

SELECT liste attributs

FROM T1 LEFT JOIN T2 ON T1.A1=T2.A2

ou

FROM T1 RIGHT JOIN T2 ON T1.A1=T2.A2

Le mot clé OUTER est facultatif

Différences entre INNER JOIN et LEFT JOIN

Quels sont les noms des skippers avec la liste des numéros de croisières qu'ils assurent?

croisiere	croisnum	sknum	batnum	depport	arrport
	C001	1	B002	BASTIA	CALVI
	C002	3	B002	ANTIBES	AJACCIO
	C003	1	B003	AJACCIO	BASTIA
	C004	3	B004	BASTIA	MARSEILLE

skipper	sknum	sknom	skport
	1	JEAN	AJACCIO
	2	PAUL	AJACCIO
	3	LAURA	ANTIBES
	4	PIERRE	BASTIA

❑ Avec une jointure classique

```
SELECT sknom, croisnum  
FROM skipper INNER JOIN croisiere  
ON skipper.sknum=croisiere.sknum;
```

❑ Avec une left join

```
SELECT sknom, croisnum  
FROM skipper LEFT JOIN croisiere  
ON skipper.sknum=croisiere.sknum;
```

Quelle différences
dans le résultat?

INNER JOIN

SELECT sknom, croisnum
FROM skipper **INNER JOIN** croisiere
ON skipper.sknum=croisiere.sknum;

sknum	sknom	skport	croisnum	sknum	batnum	depport	arrport
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
2	PAUL	AJACCIO	C001	1	B002	BASTIA	CALVI
3	LAURA	ANTIBES	C001	1	B002	BASTIA	CALVI
4	PIERRE	BASTIA	C001	1	B002	BASTIA	CALVI
1	JEAN	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
2	PAUL	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
3	LAURA	ANTIBES	C002	3	B002	ANTIBES	AJACCIO
4	PIERRE	BASTIA	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
2	PAUL	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	LAURA	ANTIBES	C003	1	B003	AJACCIO	BASTIA
4	PIERRE	BASTIA	C003	1	B003	AJACCIO	BASTIA
1	JEAN	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
2	PAUL	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
3	LAURA	ANTIBES	C004	3	B004	BASTIA	MARSEILLE
4	PIERRE	BASTIA	C004	3	B004	BASTIA	MARSEILLE

sknom	croisnum
JEAN	C001
LAURA	C002
JEAN	C003
LAURA	C004

LEFT JOIN

SELECT sknom, croisnum
FROM skipper LEFT JOIN croisiere
ON skipper.sknum=croisiere.sknum;

sknum	sknom	skport	croisnum	sknum	batnum	deport	arrport
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
2	PAUL	AJACCIO	C001	1	B002	BASTIA	CALVI
3	LAURA	ANTIBES	C001	1	B002	BASTIA	CALVI
4	PIERRE	BASTIA	C001	1	B002	BASTIA	CALVI
1	JEAN	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
2	PAUL	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
3	LAURA	ANTIBES	C002	3	B002	ANTIBES	AJACCIO
4	PIERRE	BASTIA	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
2	PAUL	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	LAURA	ANTIBES	C003	1	B003	AJACCIO	BASTIA
4	PIERRE	BASTIA	C003	1	B003	AJACCIO	BASTIA
1	JEAN	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
2	PAUL	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
3	LAURA	ANTIBES	C004	3	B004	BASTIA	MARSEILLE
4	PIERRE	BASTIA	C004	3	B004	BASTIA	MARSEILLE

sknom	croisnum
JEAN	C001
LAURA	C002
JEAN	C003
LAURA	C004
PIERRE	NULL
PAUL	NULL

PIERRE et PAUL
apparaissent dans le
résultat alors qu'ils
n'effectuent aucune
croisière

RIGHT JOIN

```
SELECT sknom, croisnum  
FROM skipper RIGHT JOIN croisiere  
ON skipper.sknum=croisiere.sknum;
```

Quel est le résultat de cette
requête ?
Comment l'expliquez vous?

croisnum	sknom
C001	JEAN
C002	LAURA
C003	JEAN
C004	LAURA



*Le résultat est identique à la requête avec une
INNER JOIN car il n'y a pas de tuples de la table
croisiere qui ont une valeur de sknum non
présente dans la table skipper (par définition
car sknum est une clé étrangère dans croisiere)*

Left Join

- Quels sont les noms des skipper qui n'assurent aucune croisière ?

```
SELECT sknom  
FROM skipper LEFT JOIN croisiere  
ON skipper.sknum=croisiere.sknum  
WHERE croisiere.sknum IS NULL;
```



sknom
PIERRE
PAUL

Exercice 2 – INNER, LEFT, RIGHT et FULL JOIN

croisiere	croisnum	sknum	batnum	depport	arrport
	C001	1	B002	BASTIA	CALVI
	C002	3	B002	ANTIBES	AJACCIO
	C003	1	B003	AJACCIO	BASTIA
	C004	3	B004	BASTIA	MARSEILLE

skipper	sknum	sknom	skport
	1	JEAN	AJACCIO
	2	PAUL	AJACCIO
	3	LAURA	NICE
	4	PIERRE	BASTIA

1. A quelle question répond la requête suivante :

```
SELECT sknom, croisnum  
FROM skipper INNER JOIN croisiere  
ON skipper.skport=croisiere.depport;
```

2. Quel est son résultat ?

3. Quel serait son résultat

☐ Avec une LEFT JOIN ?

☐ Avec une RIGHT JOIN?

☐ Avec une FULL JOIN?



Une FULL (OUTER) JOIN permet de garder tous les tuples des deux tables même ceux qui ne vérifient pas la condition.

Exercice 2 – Réponses

```
SELECT sknom, croisnum
FROM skipper INNER JOIN croisiere
ON skipper.skport=croisiere.deport;
```

1- Quels sont les noms des skippers avec les numéros des croisières qui partent de leur port d'attache?

sknum	sknom	skport	croisnum	sknum	batnum	deport	arrport
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
2	PAUL	AJACCIO	C001	1	B002	BASTIA	CALVI
3	LAURA	NICE	C001	1	B002	BASTIA	CALVI
4	PIERRE	BASTIA	C001	1	B002	BASTIA	CALVI
1	JEAN	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
2	PAUL	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
3	LAURA	NICE	C002	3	B002	ANTIBES	AJACCIO
4	PIERRE	BASTIA	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
2	PAUL	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	LAURA	NICE	C003	1	B003	AJACCIO	BASTIA
4	PIERRE	BASTIA	C003	1	B003	AJACCIO	BASTIA
1	JEAN	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
2	PAUL	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
3	LAURA	NICE	C004	3	B004	BASTIA	MARSEILLE
4	PIERRE	BASTIA	C004	3	B004	BASTIA	MARSEILLE

2- Résultat

sknom	croisnum
PIERRE	C001
JEAN	C003
PAUL	C003
PIERRE	C004

Exercice 2 – Réponses

3 – Résultat requête avec LEFT JOIN

```
SELECT sknom, croisnum
FROM skipper LEFT JOIN croisiere
ON skipper.skport=croisiere.depport;
```

sknum	sknom	skport	croisnum	sknum	batnum	depport	arrport
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
2	PAUL	AJACCIO	C001	1	B002	BASTIA	CALVI
3	LAURA	NICE	C001	1	B002	BASTIA	CALVI
4	PIERRE	BASTIA	C001	1	B002	BASTIA	CALVI
1	JEAN	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
2	PAUL	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
3	LAURA	NICE	C002	3	B002	ANTIBES	AJACCIO
4	PIERRE	BASTIA	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
2	PAUL	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	LAURA	NICE	C003	1	B003	AJACCIO	BASTIA
4	PIERRE	BASTIA	C003	1	B003	AJACCIO	BASTIA
1	JEAN	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
2	PAUL	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
3	LAURA	NICE	C004	3	B004	BASTIA	MARSEILLE
4	PIERRE	BASTIA	C004	3	B004	BASTIA	MARSEILLE

2- Résultat

sknom	croisnum
PIERRE	C001
JEAN	C003
PAUL	C003
PIERRE	C004
LAURA	NULL

Exercice 2 – Réponses

3 – Résultat requête avec RIGHT JOIN

```
SELECT sknom, croisnum
FROM skipper RIGHT JOIN croisiere
ON skipper.skport=croisiere.depport;
```

sknum	sknom	skport	croisnum	sknum	batnum	depport	arrport
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
2	PAUL	AJACCIO	C001	1	B002	BASTIA	CALVI
3	LAURA	NICE	C001	1	B002	BASTIA	CALVI
	PIERRE	BASTIA	C001	1	B002	BASTIA	CALVI
1	JEAN	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
2	PAUL	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
3	LAURA	NICE	C002	3	B002	ANTIBES	AJACCIO
4	PIERRE	BASTIA	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
2	PAUL	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	LAURA	NICE	C003	1	B003	AJACCIO	BASTIA
4	PIERRE	BASTIA	C003	1	B003	AJACCIO	BASTIA
1	JEAN	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
2	PAUL	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
3	LAURA	NICE	C004	3	B004	BASTIA	MARSEILLE
4	PIERRE	BASTIA	C004	3	B004	BASTIA	MARSEILLE

sknom	croisnum
PIERRE	C001
JEAN	C003
PAUL	C003
PIERRE	C004
NULL	C002

Exercice 2 – Réponses

3 – Résultat requête avec FULL JOIN

```
SELECT sknom, croisnum
FROM skipper FULL JOIN croisiere
ON skipper.skport=croisiere.depport;
```

sknum	sknom	skport	croisnum	sknum	batnum	depport	arrport
1	JEAN	AJACCIO	C001	1	B002	BASTIA	CALVI
2	PAUL	AJACCIO	C001	1	B002	BASTIA	CALVI
3	LAURA	NICE	C001	1	B002	BASTIA	CALVI
4	PIERRE	BASTIA	C001	1	B002	BASTIA	CALVI
1	JEAN	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
2	PAUL	AJACCIO	C002	3	B002	ANTIBES	AJACCIO
3	LAURA	NICE	C002	3	B002	ANTIBES	AJACCIO
4	PIERRE	BASTIA	C002	3	B002	ANTIBES	AJACCIO
1	JEAN	AJACCIO	C003	1	B003	AJACCIO	BASTIA
2	PAUL	AJACCIO	C003	1	B003	AJACCIO	BASTIA
3	LAURA	NICE	C003	1	B003	AJACCIO	BASTIA
4	PIERRE	BASTIA	C003	1	B003	AJACCIO	BASTIA
1	JEAN	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
2	PAUL	AJACCIO	C004	3	B004	BASTIA	MARSEILLE
3	LAURA	NICE	C004	3	B004	BASTIA	MARSEILLE
4	PIERRE	BASTIA	C004	3	B004	BASTIA	MARSEILLE

sknom	croisnum
PIERRE	C001
JEAN	C003
PAUL	C003
PIERRE	C004
LAURA	NULL
NULL	C002



Requêtes d'interrogation des données

Sous-requêtes

DML
Data
Manipulation
language



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

plan SQL

Requêtes SQL imbriquées

- Sous-requêtes avec résultat unique

SELECT liste attributs

FROM liste tables

Opérateur de comparaison

= <> < = < > = >

WHERE Attribut **Θ** Bloc SFW avec résultat valeur unique;

- Sous-requêtes avec résultat ensemble

SELECT liste attributs

FROM liste tables

WHERE Attribut **Co** Bloc SFW avec résultat ensemble;

Connecteurs

IN - EXISTS / NOT EXISTS - ANY/ ALL

Requêtes SQL imbriquées

- Quels sont les numéros des skippers dont le salaire est supérieur à au moins un des salaires des skippers localisés le port de « BASTIA »?

SELECT sknum

FROM skipper

WHERE salaire >

ANY (SELECT salaire FROM skipper
WHERE skport= 'BASTIA');

sous-requête
indépendante



Requêtes SQL imbriquées

- Quels sont les numéros des skippers dont le salaire est supérieur à tous les salaires des skippers localisés le port de « BASTIA »?

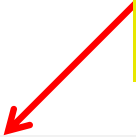
SELECT sknum

FROM skipper

WHERE salaire >

ALL (SELECT salaire FROM skipper
WHERE skport= 'BASTIA');

sous-requête
indépendante



Requêtes SQL imbriquées

- Quels sont les numéros des skippers qui n'effectuent aucune croisière?


SELECT sknum

FROM skipper

WHERE NOT EXISTS

(SELECT * FROM croisiere
WHERE sknum=skipper.sknum) ;

sous-requête
dépendante




Autre solution: LEFT JOIN !!

Requêtes SQL corrélées

- Quels sont les numéros des skippers dont le salaire est supérieur à au moins un des salaires des skippers localisés dans leur port d'attache?

```
SELECT sknum  
FROM skipper AS X  
WHERE salaire >  
    ANY (SELECT salaire FROM skipper  
        WHERE X.skport= skport ) ;
```

sous-requête
dépendante du
bloc SFW externe



Exercice 3 : Requêtes et résultats



Q1 – On considère la requête suivante:

```
SELECT BATNUM
```

```
FROM BATEAUX
```

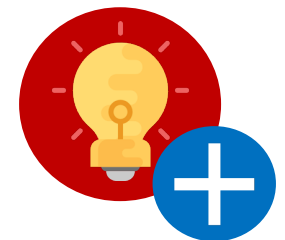
```
WHERE BATNUM NOT IN
```

```
(SELECT BATNUM
```

```
FROM CROISIERES JOIN SKIPPERES
```

```
ON CROISIERES.SKNUM= SKIPPERES.SKNUM AND
```

```
SKNOM="Nivet");
```



Sous-requêtes

1. Définissez la table résultat de cette requête
2. Identifiez la question à laquelle répond cette requête

Exercice 3 : Requêtes et résultats



Q2 – On considère la requête suivante:

```
SELECT SKNUM, SKNOM
```

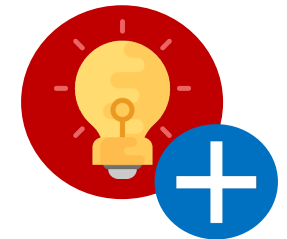
```
FROM SKIPPERS
```

```
WHERE NOT EXISTS
```

```
( SELECT *
```

```
FROM CROISIERES
```

```
WHERE CROISIERES.SKNUM= SKIPPERS.SKNUM)
```



Sous-requêtes

1. Définissez la table résultat de cette requête
2. Identifiez la question à laquelle répond cette requête
3. Proposez une solution équivalente sans utiliser de sous-requête



Requêtes d'interrogation des données

Requêtes avec partitionnement

DML
Data
Manipulation
language



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

plan SQL

Requêtes de partitionnement

fonctions statistiques

```
SELECT attribut de partitionnement, liste attributs
FROM liste tables cibles
WHERE condition(s) sur les tuples
GROUP BY attribut de partitionnement
HAVING condition sur les groupes ;
```

Quels sont les salaires maximaux des skippers pour chaque port de résidence?

```
SELECT skport , MAX(salaire) AS SalaireMax
FROM skipper
GROUP BY skport ;
```

attribut de partitionnement

Requêtes de partitionnement

```
SELECT  skport , MAX(salaire) AS SalaireMax  
FROM    skipper  
GROUP BY skport ;
```

skipper	sknum	sknom	skport	salaire
	1	JEAN	AJACCIO	2000
	2	PAUL	AJACCIO	2500
	3	LAURA	ANTIBES	1400
	5	JULES	ANTIBES	1550
	4	PIERRE	BASTIA	1800

skport	SalaireMax
AJACCIO	2500
ANTIBES	1550
BASTIA	1800

Requêtes de partitionnement

- Pour chaque bateau effectuant au moins deux croisières, donner son numéro et le nombre de croisières qu'il effectue?

```
SELECT    batnum, COUNT(croisnum)
FROM      croisiere
GROUP BY  batnum
HAVING    COUNT(croisnum) >=2 ;
```

Condition de sélection sur les groupes de tuples

Requêtes de partitionnement

- Pour chaque port de départ d'au moins trois croisières après le 10/10/2018, donner le nombre de croisières partant après le 10/10/2018?

SELECT depport, **COUNT**(croisnum)

FROM croisiere

WHERE DEPDATE > '10/10/2018'

GROUP BY depport

HAVING **COUNT**(croisnum) >=3 ;

1 - Elimination des tuples de croisiere pour lesquels DEPDATE > '10/10/2018'

2 - Création d'une sous-table (groupe) pour chaque valeur de depport.

3 - Elimination des sous-tables (groupes) ayant un nombre de lignes strictement inférieur à 3.

Requêtes avec clauses ensemblistes

Résultats « union compatibles »

Bloc SFW

UNION

Bloc SFW;

INTERSECT
EXCEPT

Donner la liste des ports dans lesquels sont domiciliés des skippers et/ou des bateaux?

```
(SELECT  BATPORT AS Ports
FROM    BATEAUX )
```

UNION

```
(SELECT  SKPORT
FROM    SKIPPERS ) ;
```

PORTS
AJACCIO
ANTIBES
CALVI

Requête complexe

- Quels sont les numéros et noms des skippers gagnant moins que tous les skippers localisés à BASTIA et effectuant plus de 2 croisières?

```
SELECT  SKIPPER.SKNUM, SKNOM  
FROM    SKIPPER, CROISIERES
```

```
WHERE  SALAIRE < ALL (SELECT SALAIRE  
                        FROM    SKIPPER  
                        WHERE    SKPORT= 'BASTIA')
```

```
AND SKIPPER.SKNUM=CROISIERES.SKNUM
```

```
GROUP BY SKIPPER.SKNUM, SKNOM  
HAVING COUNT(CROISNUM)>2 ;
```


Requête complexe (division)

- Quels sont les numéros des skippers qui assurent des croisières sur tous les bateaux localisés à AJACCIO?

```
SELECT SKNUM  
FROM SKIPPERS  
WHERE NOT EXISTS
```

```
(SELECT *  
FROM BATEAUX  
WHERE SKPORT= 'AJACCIO' AND  
NOT EXISTS
```

```
(SELECT *  
FROM CROISIERES  
WHERE CROISIERES.BATNUM=BATEAUX.BATNUM  
AND CROISIERES.SKNUM=SKIPPERS.SKNUM ) ) ;
```

Exercice 4 : Group BY et résultats



On considère la requête suivante:

```
SELECT CROISNUM, count(SKIPPERS.SKNUM)
FROM SKIPPERS, CROISIERES
WHERE SKPORT=DEPPORT;
GROUP BY CROISNUM;
```



1. Définissez la table résultat de cette requête
2. Identifiez la question à laquelle répond cette requête

Exercice 5 – Requêtes

0 - Définissez les requêtes suivantes :

1. *Quels sont les noms des skippers qui assurent une ou plusieurs croisières au départ d'Ajaccio?*
2. *Quels sont les numéros des croisières assurées par des bateaux de capacité supérieure à 10?*
3. *Donner la liste des numéros des croisières avec le nom du skipper et le nom du bateau impliqués?*
4. *Quels sont les numéros des croisières dont le port d'arrivée est le port d'attache du skipper de nom "Poggi"?
(remplacer "Poggi" par Paul ou autre en fonction des données dans vos tables)*

Exercice 5 – Requêtes

1. Définissez trois versions différentes de la requête suivante :

Quels sont les noms des skippers qui assurent une ou plusieurs croisières sur un bateau ayant une capacité inférieure à celle de tous les bateaux localisés à Ajaccio?

- version avec une sous-requete utilisant la clause IN
- version avec une sous-requete utilisant la clause Not exists
- version avec Group by et Having

Exercice 5 – Requêtes


Définissez les requêtes suivantes :

2. Quels sont les noms des skippers qui assurent au moins 2 croisières?
3. Pour chaque bateau ayant une capacité de plus de 5 et qui effectue au moins 3 croisières, donner le nombre de croisières effectuées?
4. Quels sont les skippers qui assurent des croisières avec tous les bateaux localisés à Ajaccio?

Exercice 5 – Requêtes

5. Quels sont les noms des bateaux qui ont effectué les croisières les plus longues en termes de jours.
6. Quels sont les noms des skippers qui habitent dans un port dans lequel sont localisés des bateaux qui ne font aucune croisière





Requêtes d'interrogation des données

Requêtes avec clause CASE ...WHEN

DML
Data
Manipulation
language

Requêtes avec clause CASE...WHEN

- Pour renvoyer une valeur selon la valeur d'une colonne

```
SELECT immat,  
       CASE COULEUR  
         WHEN 1 THEN 'Rouge'  
         WHEN 2 THEN 'Bleu'  
         WHEN 3 THEN 'Vert'  
         ELSE 'Indéterminé'  
       END  
FROM VOITURES
```

VOITURES

IMMAT	COULEUR
FA235FG	1
DF456CS	2
CX672SI	4



IMMAT	CASE
FA235FG	Rouge
DF456CS	Bleu
CX672SI	Indéterminé

Requêtes avec clause CASE...WHEN

- Pour renvoyer une valeur selon des conditions booléennes

```
SELECT SKNUM, SKNOM, SALAIRE,  
CASE  
  WHEN salaire<=1500 THEN 150  
  WHEN salaire>1500 AND <=2500 THEN 100  
  ELSE 0  
END AS PRIME  
FROM SKIPPERS
```

SKNUM	SKNOM	SALAIRE	PRIME
1	JEAN	3000	0
2	PAUL	2500	100
3	LAURA	2500	100
4	PIERRE	1500	150

Requêtes UPDATE avec clause CASE...WHEN

- Pour effectuer des mises à jour conditionnelles

```
UPDATE SKIPPERS
SET SALAIRE = (
    CASE
        WHEN salaire<=1500 THEN SALAIRE + 150
        WHEN salaire>1500 AND salaire<=2500
            THEN SALAIRE + 100
    END );
```

CASE .. WHEN peut être utilisée dans une clause SELECT, UPDATE, DELETE, WHERE, ORDER BY ou HAVING.

Exercice 6 – Requete avec CASE WHEN

Définissez une requête qui retourne liste des numéros de croisières avec leur catégorie (courte, moyenne ou longue), basée sur leur durée :



- '**Courte**' : Si la durée est inférieure ou égale à 2 jours.
- '**Moyenne**' : Si la durée est comprise entre 3 et 7 jours.
- '**Longue**' : Si la durée dépasse 7 jours.



SQL

Notion de Vue

- Définition
- Vues et indépendance logique



VUES et Requêtes administrateur

VUE = Table virtuelle définie à partir d'autres tables grâce à une requête

Définition d'une vue

```
CREATE VIEW Nom-Vue AS Requete SFW;
```

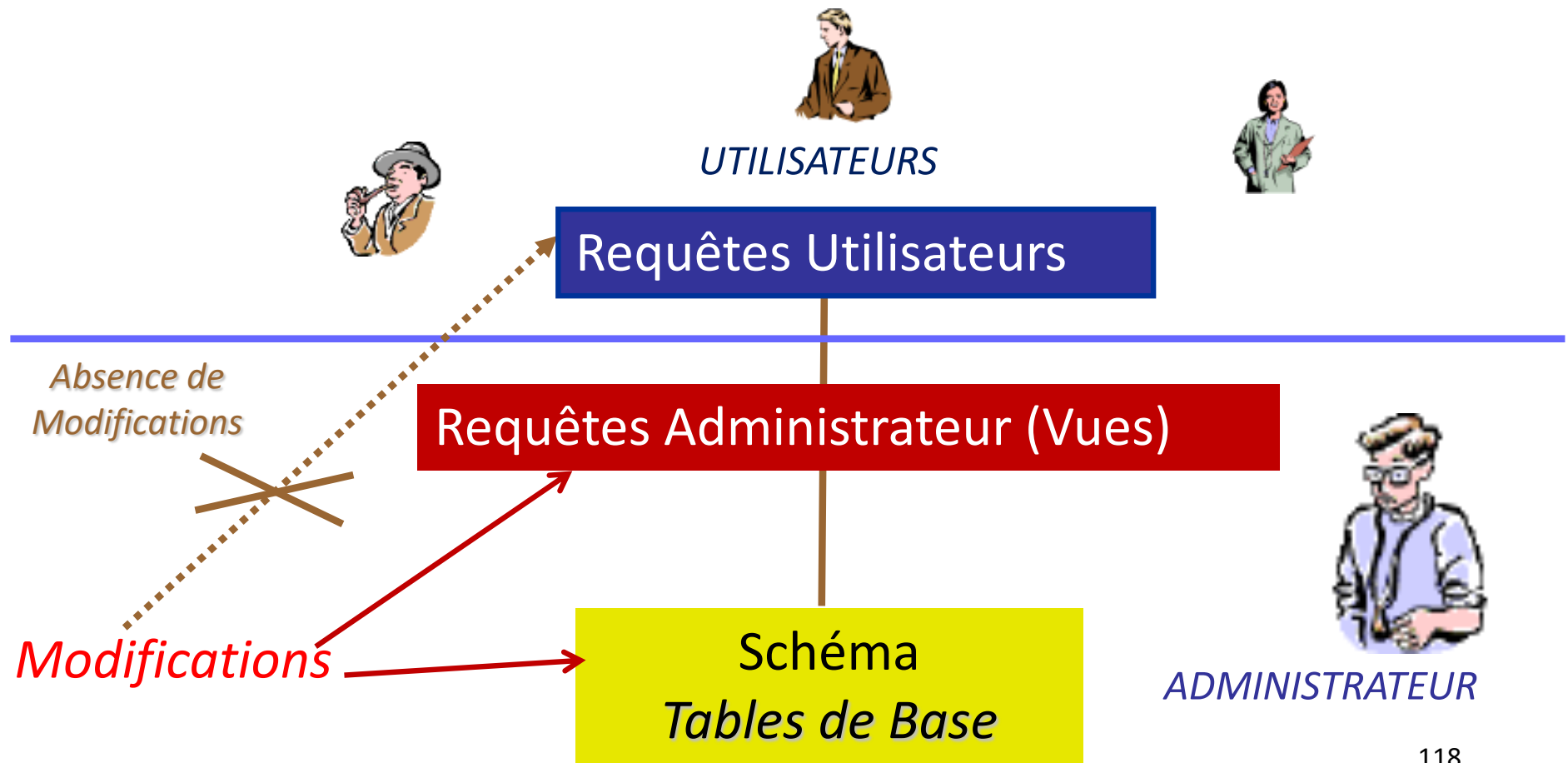
- Créer une vue contenant tous les skippers localisés à Nice

```
CREATE VIEW SKIPNICE AS  
SELECT *  
FROM SKIPPERS  
WHERE SKPORT= 'NICE' ;
```

VUES et Requêtes administrateur

Indépendance Logique

Les utilisateurs ne manipulent les tables de base qu'à travers les requêtes (ou vues) définies par l'administrateur.



Limites des VUES

- Les possibilités de mises à jour sur les vues sont souvent très limitées :
 - Vue mono-table uniquement
 - Vue ne comportant pas de clauses GROUP BY
 - Vue ne comportant pas d'attribut calculé
 - Vue ne comportant pas de clause ensembliste (UNION, MINUS, INTERSECT)



Exercice 7 (1) : Vues et indépendance logique

bdcroisieresCours.sql

- 1) Définissez une vue SKIPMOY contenant les numéros, noms et ports d'attache des skippers ayant un salaire supérieur à la moyenne des salaires des skippers.
- 2) Définissez une requête basée sur la vue SKIPMOY et renvoyant la liste des ports d'attache des skippers dont le salaire est supérieur à la moyenne.



Exercice 7 (2) : Vues et indépendance logique

Suite à un changement dans l'organisation de l'entreprise, un Skipper peut à présent être attaché à plusieurs ports de localisation.

- Ainsi la table SKIPPERS initiale ne permet pas d'insérer les informations suivantes:
 - Le skipper N° 1, JEAN a pour ports d'attache AJACCIO et PROPRIANO
 - Le skipper N°3, LAURA a pour ports d'attache ANTIBES et NICE.


sknum	skno	skport
1	JEAN	AJACCIO
2	PAUL	AJACCIO
3	LAURA	ANTIBES
4	PIERRE	BASTIA

Comment modifier le schéma pour prendre en compte ce changement?



Exercice 7 (3) : Vues et indépendance logique

- 3) Proposez une solution et définissez les requêtes nécessaires à la prise en compte de vos modifications (ajout et modification de tables, mises à jour de données, vue)
- 4) Pouvez-vous réexécuter la requête de la question 2 dans le nouveau schéma ? Que pouvez-vous en conclure sur l'intérêt de l'utilisation des vues ?



Requêtes d'interrogation des données

Vues Matérialisées

DML
Data
Manipulation
language

VUES MATERIALISEES

- Vues dont le résultat est stocké physiquement dans la BD

Contrairement à une vue classique

- **Avantage** : Performance

les requêtes sur une vue matérialisée sont souvent plus rapides car les données sont précalculées

- **Contraintes**

- Nécessité d'un rafraichissement
 - manuel ou automatique(possible uniquement sous oracle)
- Cout d'occupation en termes d'espace

Disponible sous Postgresql uniquement depuis la version 9.3

VUES MATERIALISEES POSTGRESQL

```
CREATE MATERIALIZED VIEW nom_vue_materialisee  
AS  
requete SFW
```

- En PostgreSQL, toutes les vues matérialisées sont construites immédiatement. Il n'y a pas d'option pour différer la construction comme en Oracle.

Le Rafraichissement
automatique n'est
pas possible



VUES MATERIALISEES



- Rafraichissement manuel

```
REFRESH MATERIALIZED VIEW nom_vue_materialisee;
```

2 options possibles :

- **WITH DATA** (par défaut si rien n'est précisé)
 - rafraîchit la vue matérialisée et remplit ses données en exécutant à nouveau la requête sous-jacente.
- **WITH NO DATA**
 - vide la vue matérialisée en attendant un rafraichissement ultérieur.

VUES MATERIALISEES :exemple

- Vue matérialisée pour afficher le nombre total de croisières par bateau.
- Cette vue doit se rafraîchir automatiquement lors de chaque mise à jour de la table croisiere.

```
CREATE MATERIALIZED VIEW mv_croisieres_par_bateau  
BUILD IMMEDIATE  
REFRESH FORCE ON COMMIT  
AS  
SELECT BATNUM, COUNT(CROISNUM) as NB_CROISIERES  
FROM CROISIERES  
GROUP BY BATNUM;
```

VUES MATERIALISEES ORACLE

```
CREATE MATERIALIZED VIEW nom_vue_materialisee  
BUILD [IMMEDIATE | DEFERRED]  
REFRESH [FAST | COMPLETE | FORCE]  
[START WITH date]  
[NEXT expr]  
[ON COMMIT | ON DEMAND]
```

Rafraichissement
synchrone ou
asynchrone (sur
demande)

AS

requete SFW



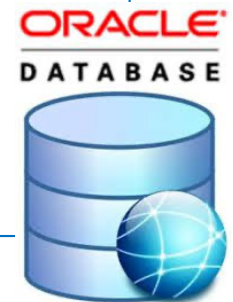
- FAST : rafraîchissement rapide si possible
- COMPLETE : Effectue toujours un rafraîchissement complet.
- FORCE : Oracle décide du type de rafraîchissement (rapide ou complet) en fonction de la disponibilité des journaux.

VUES MATERIALISEES

(sous ORACLE)

■ Rafraîchissement manuel

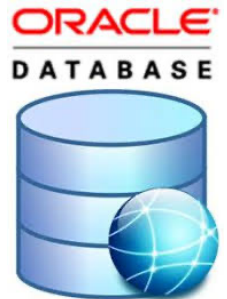
```
BEGIN  
    DBMS_MVIEW.REFRESH(list => nomVue,  
    method => 'C');  
END; /
```



- 'C': Rafraîchissement complet.
- 'F': Rafraîchissement rapide (si possible). Seules les modifications (insertions, mises à jour, suppressions) depuis le dernier rafraîchissement sont prises en compte.
- '?': Oracle décide de la meilleure méthode (complet ou rapide).

JOURNAL DE VUES MATERIALISEES

- Pour pouvoir effectuer des rafraichissements rapides (seules les lignes modifiées depuis le dernier rafraîchissement sont mises à jour) **Fast**
- Il faut définir un journal de vue matérialisée sur la table et les colonnes concernées.



```
CREATE MATERIALIZED VIEW LOG ON TABLE  
WITH ROWID,  
SEQUENCE(nomCol1, nomCol2,...)  
INCLUDING NEW VALUES;
```

Non
disponible
sous
postgres

VUES CLASSIQUES et VM


Caractéristiques	Vue classique	Vue Matérialisée
Stockage des données	Aucun	Stockage physique
Performance	Recalcul à chaque exécution	Plus rapide car exécution unique
Mise à jour des données	Automatique	Nécessite un rafraîchissement
Utilisation d'espace	Aucun espace de stockage	Requiert de l'espace de stockage
Flexibilité et performance	Peut représenter n'importe quelle requête SQL sans coûts supplémentaires en termes de stockage ou de performance.	Idéale pour les requêtes fréquemment utilisées et coûteuses, mais moins flexible

Exercice 8 : Vue Matérialisée



- Définissez une vue matérialisée qui donne la liste des numéros et noms de bateaux avec le nombre de skippers qui les barrent et le total de leurs salaires.

batnum character varying (5) 🔒	batnom character varying (50) 🔒	nb_skippers bigint 🔒	total_salaire bigint 🔒
B004	INDEPENDANCE	2	5000
B001	LIBERTE	1	2500
B003	KALISTE	1	3000
B002	LOUISIANE	4	11000



Requêtes d'interrogation des données

Common Table Expressions (CTE) (clause WITH)

DML
Data
Manipulation
language

Requêtes avec CTE ... WITH

- Une Common Table Expression (CTE) est une table virtuelle qui n'existe que pendant l'exécution d'une requête.

```
WITH nomCTE (nomCol1, nomCol2, ...) AS (  
    Requete SFW )  
)  
SELECT ...  
FROM nomCTE ...
```

- ✓ Amélioration de la lisibilité des requêtes complexes
- ✓ Maintenance facilitée
- ✓ Parfois amélioration des performances (à tester cf. CH4)

Requêtes avec CTE ... WITH

- Exemple : *Nombre de croisières par bateau (en incluant les bateaux sans aucune croisière)*

```
WITH CroisieresParBateau AS (  
    SELECT BATNUM, COUNT(CROISNUM) AS NbCroisieres  
    FROM CROISIERES  
    GROUP BY BATNUM  
)
```

```
SELECT B.BATNOM, COALESCE(C.NbCroisieres, 0) AS  
    NbCroisieres  
FROM BATEAUX B  
LEFT JOIN CroisieresParBateau C ON B.BATNUM = C.BATNUM;
```

COALESCE(val1, val2, ..valn) renvoie la première valeur non nulle parmi les arguments : ici 0 si NbCroisieres est NULL

CTE et Vue: Quelles différences?

Caractéristiques	CTE	VUES
Définition et Durée de vie	Temporaire: seulement pour la requête où elle est définie	Persistante: reste dans la BD jusqu'à sa suppression
Syntaxe	clause WITH	CREATE VIEW
Usage et Portée	Décompose les requêtes complexes. Portée limitée à la requête individuelle	Masque la complexité des données. Disponible pour toutes les requêtes après création
Modification des données	impossible	Possible sur certaines vues (avec restrictions)
Réutilisabilité	Non réutilisable; spécifique à une requête	Réutilisable dans n'importe quelle requête comme une table

Exercice 9 – Requete avec CTE (WITH)

Utiliser une Common Table Expression (CTE) pour calculer la durée totale cumulée des croisières pour chaque bateau.

- 1 – Définissez une CTE pour calculer la durée en jours de chaque croisière
- 2 – Définissez une requête basée sur cette CTE pour faire les calculs.





SQL

Notion de dictionnaire de données

- Principales vues
- Requêtes sur le DD



Notion de Dictionnaire de données

- **Méta-données** = informations sur les objets de la base de données (tables, index, ...)
- **Dictionnaire de données** (DD) = ensemble de tables définissant les métadonnées d'une BD
- Les tables du DD sont en général manipulées par l'intermédiaire d'un ensemble de vues






Existe dans tous les SGBD avec des noms de tables différents

Pourquoi utiliser les DD?

- Interroger la structure de la base de données
 - Savoir quelles tables, vues, colonnes, et types de données existent.
- Surveiller les performances et l'activité
 - Obtenir des statistiques sur l'utilisation des tables, des index, et des connexions.
- Gérer les utilisateurs et les privilèges
 - Vérifier quels rôles ont accès à quels objets de la base de données.
- Diagnostiquer et résoudre les problèmes
 - Identifier les verrous bloquants ou les requêtes longues.

Principales vues du DD sous PostgreSQL

4 catégories:

- Schéma pg_catalog
 - tables et vue systemes principales de PostgreSQL
- Vues de schéma d'information (information_schema) 
 - conforme au standard SQL
- Vues de statistiques et de surveillance (pg_stat_*) 
- Vues de description des objets (pg_description)

Principales vues du DD

Nom de la vue	Objets décrits	Colonnes
information_schema.tables	Tables	table_schema, table_name, table_type
information_schema.columns	Colonnes des tables	table_schema, table_name, column_name, data_type
information_schema.views	Vues	table_schema, table_name, view_definition
information_schema.table_constraints	Contraintes	constraint_schema, constraint_name, table_name, constraint_type
information_schema.key_column_usage	Utilisation des colonnes clés	constraint_name, table_name, column_name
information_schema.routines	Fonctions/Procédures	routine_schema, routine_name, routine_type
pg_roles	roles	
information_schema.role_table_grants	utilisateurs	

Noms standards communs à d'autres SGBD

Requêtes sur le DD

- Donner la liste des noms des tables de la base

```
SELECT TABLE_NAME  
FROM information_schema.tables  
WHERE table_schema = 'public' AND  
       table_type = 'BASE TABLE';
```

table_name	name
bateaux	
croisieres	
skippers	

- Donner la liste des noms de colonnes de la table BATEAUX et leur type


```
SELECT COLUMN_NAME, DATA_TYPE  
FROM information_schema.columns  
WHERE TABLE_NAME='bateaux';
```

column_name	data_type
capacite	integer
batnum	character varying
batnom	character varying
batport	character varying

Exercice 10 : Manipulation du DC



- 1) Définissez une requête qui donne la liste des noms de colonnes de la table CROISIERES
- 2) Définissez une requête qui donne le type de la colonne SKPORT de la table SKIPPERs
- 3) Définissez une requête qui donne pour chaque table, son nombre de colonnes
- 4) Définissez une requête qui donne le nombre moyen de colonnes dans les tables de la base de données Croisieres



Requêtes d'interrogation des données

Fonctions de Fenêtre SQL

DML
Data
Manipulation
language

Fonctions de fenêtres SQL

- Fonctions de fenêtres ou **Window functions**
- Apparus dans la norme SQL3
- Calculs sur un ensemble de lignes liées à la ligne courante, tout en conservant toutes les lignes du résultat
- Intéressant pour les analyses d'évolution chronologique des données

Nous y reviendrons dans le cours
sur les datawarehouses

Structure Générale des Fonctions de Fenêtre

*ROW_NUMBER()
RANK()
LAG() et LEAD()
SUM() AVG() ...*

*Divise les lignes en groupes
(partitions) sur lesquels la fonction
sera appliquée indépendamment*

<FONCTION_DE_FENETRE>() OVER (
 [PARTITION BY <colonne(s)>]
 [ORDER BY <colonne(s)>]
 [<frame_clause>]
)

*Définit l'ordre dans
lequel les lignes
sont traitées au
sein de chaque
partition*

*Détermine la plage de lignes à considérer pour le calcul
(par exemple, les lignes précédentes et suivantes).
ex: **ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING***

Fonctions de fenêtres

- **Numérotage des lignes** : ROW_NUMBER
- **Classement** : RANK
- **Références** aux lignes précédentes ou suivantes :
 - LAG (Look-Ahead Gap) :décalage en arrière
 - LEAD (Look-Ahead) : décalage en avant
- **Calculs cumulatifs** : Agrégations cumulatives
- Restrictions des lignes pour les calculs :**Frames**

EXEMPLE 1 : ROW_NUMBER

Pour chaque nom de skipper, donner la liste de ses croisières classées par ordre de date de départ et numérotées

```
SELECT
  SKIPPER.SKNOM,
  CROISIERES.CROISNUM,
  CROISIERES.DEPPATE,
  ROW_NUMBER() OVER (PARTITION BY CROISIERES.SKNUM
                     ORDER BY CROISIERES.DEPPATE) AS Num_Croisiere
FROM
  CROISIERES
JOIN
  SKIPPER ON CROISIERES.SKNUM = SKIPPER.SKNUM
ORDER BY
  SKIPPER.SKNOM, CROISIERES.DEPPATE;
```

numéro unique
créé par la requête

sknom character varying (50)	croisnum character varying (5)	deppate date	num_croisiere bigint
JEAN	C001	2024-07-10	1
JEAN	C003	2024-08-21	2
JEAN	C005	2024-10-10	3
LAURA	C007	2024-08-02	1
LAURA	C009	2024-08-02	2
LAURA	C004	2024-09-02	3
LAURA	C008	2024-10-02	4
PAUL	C002	2024-07-15	1

Exercice 11.1 – Requete avec ROW_NUMBER

Définissez une requête qui donne pour chaque bateau, la liste numérotée des croisières qu'il effectue au départ de BASTIA



Résultat attendu (avec variations selon vos données)

batnum character varying (5)	croisnum [PK] character varying (5)	depdate date	arrdate date	deport character varying (50)	rang_croisiere bigint
B002	C001	2024-07-10	2024-07-11	BASTIA	1
B002	C007	2024-08-02	2024-08-03	BASTIA	2
B004	C008	2024-10-02	2024-10-03	BASTIA	1

EXEMPLE 2 : RANK

Classer les skippers par ordre de leur salaire en leur attribuant un numéro dans ce classement

```
SELECT
    SKNUM,
    SKNOM,
    SALAIRE,
    RANK() OVER (ORDER BY SALAIRE DESC) AS Rang_Salaire
FROM
    SKIPPERS
ORDER BY
    Rang_Salaire;
```

numéro de rang
créé par la requête

sknum [PK] character varying (5)	sknom character varying (50)	salaire integer	rang_salaire bigint
1	JEAN	3000	1
4	PIERRE	2535	2
2	PAUL	2500	3
3	LAURA	2500	3

EXEMPLE 2 (suite) : RANK

Donner les skippers qui font partie du top 2 des salaires

```
SELECT *  
FROM (  
    SELECT SKNUM, SKNOM, SALAIRE,  
           RANK() OVER (ORDER BY SALAIRE DESC) AS Rang_Salaire  
    FROM SKIPPERS ) AS t  
WHERE Rang_salaire<=2 ;
```

alias obligatoire

sknum [PK] character varying (5)	sknom character varying (50)	salaire integer	rang_salaire bigint
1	JEAN	3000	1
4	PIERRE	2535	2

EXEMPLE 2 (suite variante) : RANK
Donner les skippers qui font partie du top 2 des salaires

```
WITH Classement AS (  
    SELECT SKNUM, SKNOM, SALAIRE,  
           RANK() OVER (ORDER BY SALAIRE DESC) AS Rang_Salaire  
    FROM SKIPPERS )
```

variante
avec CTE

```
SELECT SKNUM, SKNOM, SALAIRE, Rang_Salaire  
FROM Classement  
WHERE Rang_salaire<=2 ;
```

sknum [PK] character varying (5)	sknom character varying (50)	salaire integer	rang_salaire bigint
1	JEAN	3000	1
4	PIERRE	2535	2

EXEMPLE 3 : LAG

Pour chaque croisière donner la date de la croisière, celle de la croisière précédente et la différence en nombre de jours entre les deux.

SELECT

CROISNUM,

DEPDATE,

LAG(DEPDATE) **OVER** (ORDER BY DEPDATE) AS **Date_Dep_Precedente**,

DEPDATE - **LAG**(DEPDATE) **OVER** (ORDER BY DEPDATE) AS **Delta_Jours**

FROM

CROISIERES C

ORDER BY DEPDATE;

LAG (Look-Ahead Gap) :décalage en arrière

croisnum [PK] character varying (5)	depdate date	date_dep_precedente date	delta_jours integer
C001	2024-07-10	[null]	[null]
C002	2024-07-15	2024-07-10	5
C009	2024-08-02	2024-07-15	18
C007	2024-08-02	2024-08-02	0
C003	2024-08-21	2024-08-02	19
C004	2024-09-02	2024-08-21	12
C008	2024-10-02	2024-09-02	30
C005	2024-10-10	2024-10-02	8

EXEMPLE 4 : LEAD

Pour chaque numéro de bateau donner pour chacune de ses croisières, la croisière suivante par ordre de date.

SELECT

BATNUM, CROISNUM, DEPDATE,

LEAD (Look-Ahead) :décalage en avant

LEAD(DEPDATE) OVER (PARTITION BY BATNUM ORDER BY DEPDATE)

AS Date_Suivante,

LEAD(DEPDATE) OVER (PARTITION BY BATNUM ORDER BY DEPDATE)

- DEPDATE AS Delta_Jours

FROM CROISIERES

ORDER BY BATNUM, DEPDATE;

batnum character varying (5)	croisnum [PK] character varying (5)	depdate date	date_suivante date	delta_jours integer
B002	C001	2024-07-10	2024-07-15	5
B002	C002	2024-07-15	2024-08-02	18
B002	C007	2024-08-02	2024-10-10	69
B002	C005	2024-10-10	[null]	[null]
B003	C003	2024-08-21	[null]	[null]
B004	C009	2024-08-02	2024-09-02	31
B004	C004	2024-09-02	2024-10-02	30
B004	C008	2024-10-02	[null]	[null]

EXEMPLE 4 (version améliorée) : LEAD
Pour chaque numéro de bateau donner pour chacune de
ses croisières, la croisière suivante par ordre de date.

```
WITH complete AS (  
    SELECT BATNUM, CROISNUM, DEPDATE,  
        LEAD(DEPDATE) OVER (PARTITION BY BATNUM ORDER BY DEPDATE)  
        AS date_suiv,  
    FROM CROISIERES)  
SELECT  
    BATNUM, CROISNUM,  
    TO_CHAR(DEPDATE, 'YYYY-MM-DD') AS deptime,  
    COALESCE(TO_CHAR(date_suiv, 'YYYY-MM-DD'), 'Aucune') AS date_suivante,  
    COALESCE(date_suiv - DEPDATE, 0) AS delta_jours  
FROM complete  
ORDER BY BATNUM, DEPDATE;;
```

COALESCE(expr1, ..., exprN) retourne la
première valeur non nulle de gauche à droite

Exercice 11.2 – Requete avec LAG

Définissez une requête qui donne pour chaque bateau, la date de sa dernière croisière, la date de la précédente et la différence entre les deux en nombre de jours



Résultat attendu (avec variations selon vos données)

batnom character varying (50) 🔒	croisnum character varying (5) 🔒	derniere_croisiere date 🔒	date_precedente date 🔒	delta_jours integer 🔒
INDEPENDANCE	C008	2024-10-02	2024-09-02	30
KALISTE	C003	2024-08-21	[null]	[null]
LOUISIANE	C005	2024-10-10	2024-08-02	69

Vous pouvez procéder en 2 étapes avec une CTE

Notion d'agrégation cumulative

SUM, AVG, COUNT

Agrégation classique

- Le calcul porte sur **toutes les lignes d'un groupe (GROUP BY)** ou d'une requête(si aucun **GROUP BY**)
- Résultat = une seule valeur par groupe (GROUP BY) ou une seule valeur (aucun group by)

Réduit les lignes en une seule par groupe et effectue un seul calcul

Agrégation cumulative


- Le calcul **s'effectue ligne par ligne**, en tenant compte de toutes les lignes précédentes jusqu'à la ligne courante.
- Résultat = plusieurs valeurs calculées : une pour chaque ligne

Conserve toutes les lignes et calcule un cumul progressif.

Comptage classique

- Exemple de comptage classique
 - calcul du nombre de croisières par bateau

```
SELECT  
  BATNUM,  
  COUNT(CROISNUM) AS Total_Croisieres  
FROM  
  CROISIERES  
GROUP BY  
  BATNUM;
```




batnum character varying (5) 🔒	total_croisieres bigint 🔒
B004	3
B003	1
B002	4

Comptage cumulatif

- Exemple de comptage cumulatif
 - Suivi de l'évolution du nombre de croisières cumulées au fil du temps

```
SELECT  
  BATNUM,CROISNUM, DEPDATE,  
  COUNT(CROISNUM) OVER (PARTITION BY BATNUM ORDER BY DEPDATE) AS  
    croisières_cumulees
```

```
FROM  
  CROISIERES  
ORDER BY  
  BATNUM, DEPDATE;
```



batnum character varying (5)	croisnum [PK] character varying (5)	deptime date	croisières_cumulees bigint
B002	C001	2024-07-10	1
B002	C002	2024-07-15	2
B002	C007	2024-08-02	3
B002	C005	2024-10-10	4
B003	C003	2024-08-21	1
B004	C009	2024-08-02	1
B004	C004	2024-09-02	2
B004	C008	2024-10-02	3

Intérêt des Agrégations cumulatives

- Sommes, moyennes (AVG) ou comptages (count)
- Suivi et analyse de l'évolution de données au fil du temps
- Ex:
 - Suivi des ventes cumulées dans une entreprise.
 - Suivi des croisières réalisées pour un bateau à chaque date de départ.
 - Suivi de la performance financière d'une organisation au fil des mois ou des trimestres.

Clause Frame

- Permet de définir une plage de lignes (ou "fenêtre") sur lesquelles une fonction de fenêtre effectue des calculs.

Par défaut sur toutes les lignes si aucune frame n'est définie

- **ROWS ou RANGE** : Indique si la fenêtre est basée sur le nombre de lignes (ROWS) ou sur une plage de valeurs (RANGE)
- **BETWEEN ... AND ...** : Définit les limites de la fenêtre.
- **UNBOUNDED PRECEDING** : Inclut toutes les lignes avant la ligne courante.
- **CURRENT ROW** : Se réfère à la ligne courante.
- **UNBOUNDED FOLLOWING** : Inclut toutes les lignes après la ligne courante.
- **N PRECEDING ou N FOLLOWING** : Définit un nombre de lignes avant ou après la ligne courante.

EXEMPLE 5 : FRAME

Pour chaque bateau, donne pour chacune de ses croisières, la durée moyenne des **deux dernières (croisière courante et précédente)**

```
SELECT
  BATNUM, CROISNUM,
  DEPDATE, ARRDATE,
  AVG(ARRDATE - DEPDATE) OVER (
    PARTITION BY BATNUM
    ORDER BY DEPDATE
    ROWS BETWEEN 1 PRECEDING AND CURRENT ROW
  ) AS Moyenne_Duree_Deux_Dernieres
FROM CROISIERES ORDER BY BATNUM, DEPDATE;
```

batnum character varying (5)	croisnum [PK] character varying (5)	depdate date	arrdate date	moyenne_duree_deux_dernieres numeric
B002	C001	2024-07-10	2024-07-11	1
B002	C002	2024-07-15	2024-07-16	1
B002	C007	2024-08-02	2024-08-03	1
B002	C005	2024-10-10	2024-10-11	1
B003	C003	2024-08-21	2024-08-22	1
B004	C009	2024-08-02	2024-08-03	1
B004	C004	2024-09-02	2024-09-22	11
B004	C008	2024-10-02	2024-10-03	7

Exercice 11.3 – Requete avec calcul cumulatif

Calculer la somme cumulative des jours en mer pour chaque skipper jusqu'à la croisière courante.



Résultat attendu (avec variations selon vos données)

sknum character varying (5)	croisnum [PK] character varying (5)	deptime date	arrdate date	jours_cumules_en_mer bigint
1	C001	2024-07-10	2024-07-11	1
1	C003	2024-08-21	2024-08-22	2
1	C005	2024-10-10	2024-10-11	3
2	C002	2024-07-15	2024-07-16	1
3	C007	2024-08-02	2024-08-03	2
3	C009	2024-08-02	2024-08-03	2
3	C004	2024-09-02	2024-09-22	22
3	C008	2024-10-02	2024-10-03	23

Exercice 11.4 –Requete avec calcul cumulatif

Calculer pour chaque bateau, pour chacune de ses croisières, la différence de durée (en jours) entre celle-ci et la croisière précédente



Résultat attendu (avec variations selon vos données)







batnum character varying (5)	croisnum [PK] character varying (5)	depdate date	arrdate date	duree_croisiere integer	difference_duree integer
B002	C001	2024-07-10	2024-07-11	1	[null]
B002	C002	2024-07-15	2024-07-16	1	0
B002	C007	2024-08-02	2024-08-03	1	0
B002	C005	2024-10-10	2024-10-11	1	0
B003	C003	2024-08-21	2024-08-22	1	[null]
B004	C009	2024-08-02	2024-08-03	1	[null]
B004	C004	2024-09-02	2024-09-22	20	19
B004	C008	2024-10-02	2024-10-03	1	-19

Exercice 11.5 –Requete avec calcul cumulatif et frame

Calculer pour chaque skipper et chacune de ses croisières, la différence de durée (en jours) entre chaque croisière et la moyenne des durées des croisières précédentes

utiliser une frame pour calculer la moyenne en incluant toutes les lignes avant la ligne courante

Résultat attendu (avec variations selon vos données)

sknum character varying (5) 	croisnum [PK] character varying (5) 	depdate date 	arrdate date 	duree_croisiere integer 	diff_moyenne_duree_precedente numeric 
1	C001	2024-07-10	2024-07-11	1	[null]
1	C003	2024-08-21	2024-08-22	1	0
1	C005	2024-10-10	2024-10-11	1	0
2	C002	2024-07-15	2024-07-16	1	[null]
3	C007	2024-08-02	2024-08-03	1	[null]
3	C009	2024-08-02	2024-08-03	1	0
3	C004	2024-09-02	2024-09-22	20	19
3	C008	2024-10-02	2024-10-03	1	-6

