Chapitre 15 - Base de données

Objectif :

Centraliser toutes les données à un unique endroit, la base de données, qui possède son propre système de gestion, un SGBD, avec un unique langage pour interagir avec les données, le SQL.

I. Conception d'une base de données

1. Paradigme logique

Une base de données est un ensemble de données, d'informations en rapport les unes avec les autres, stockés sur un support informatique avec lequel différents utilisateurs interagissent. Ces interactions se font avec des requêtes.

Le SGBD (système de gestion de bases de données) est un logiciel permettant de simplifier la gestion de la base pour l'utilisateur.

Missions du SGBD :

- Choisir les structures de données les plus pertinentes et efficaces pour stocker toutes les données.
- Garantir l'efficacité maximale possible pour toutes les opérations sur la base (lecture, écriture, modification …).
- Garantir la maintenance des données.
- Gérer des utilisateurs (droits, …)
- Gérer les accès concurrents.
- Réceptionner et traiter les requêtes en respectant les principes ACID
  - Atomicité : "Toute la requête ou rien".
  - Cohérence : "Respect des contraintes".
  - Isolation : "Traitement d'une requête comme si c'était la seule".
  - Durabilité : "Les requêtes sont permanentes".

Définition :

Le paradigme logique est un paradigme de programmation qui définit un programme comme une suite de propriétés que doit respecter le résultat, sans aucune indication sur la manière dont ce résultat va être calculé.

2. Modèle entité-association

→ Représentation graphique des données qu'on souhaite stocker et des liens entre ces ces données.

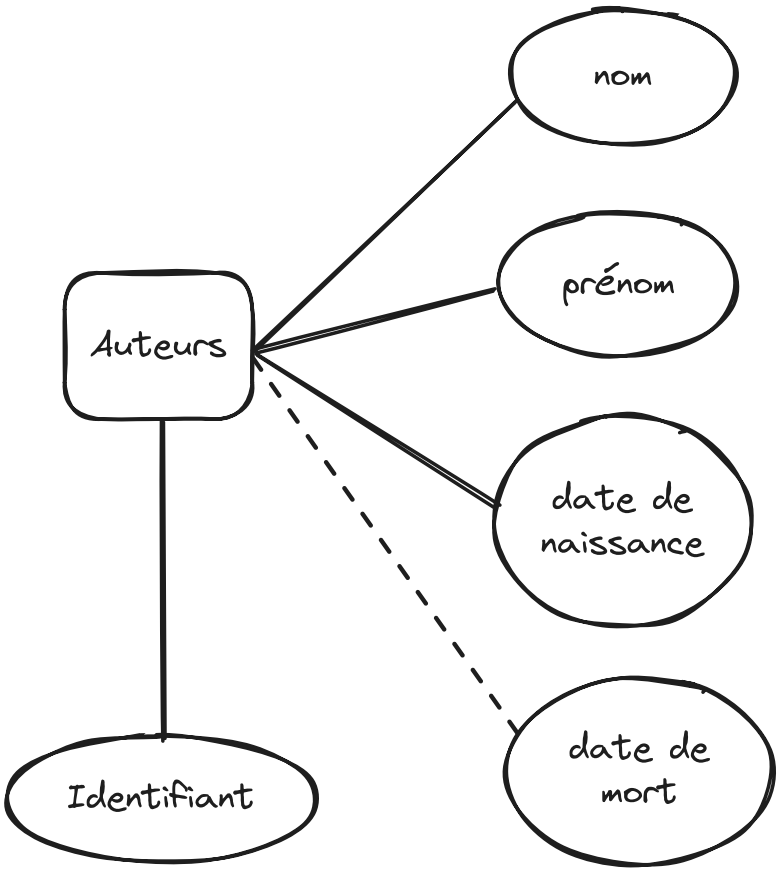
Définition :

Une entité est un objet (au sens large) du monde réel. Les informations liées à cet objet sont appelées ses propriétés.

Une entité doit toujours pouvoir être identifiée de manière unique. Parmi ses propriétés, une ou plusieurs la distingue des autres entités.

Un type d'entité regroupe toutes les entités ayant le même type de propriétés.

Représentation graphique d'un type d'entité :

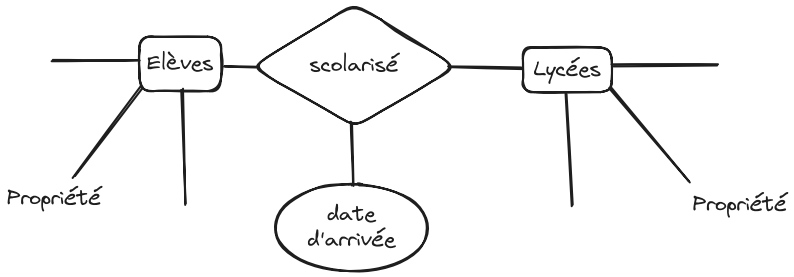


Définition :

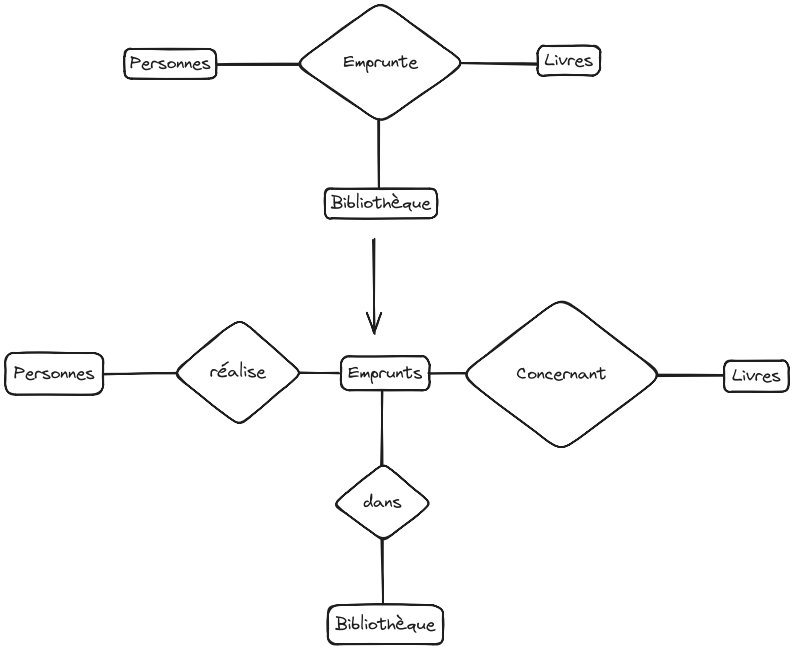
Une association représente un lien entre plusieurs entités. Les associations peuvent avoir des propriétés.

Un type d'association regroupe les associations qui ont des propriétés de type similaires, et qui font le lien entre des entités appartenant au même type d'entité.

Exemple :



On se limitera aux associations binaires (2 entités).

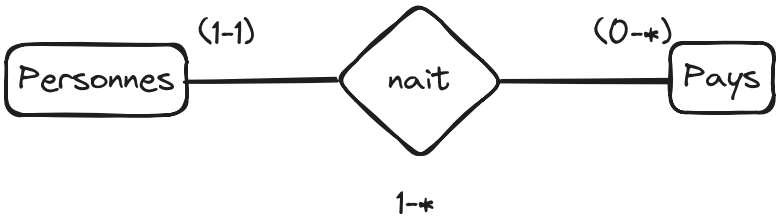


Cardinalité :

Une contrainte de cardinalité d'une entité E participant à une association A est (min-max) avec min le nombre minimal de fois où E participe à A et max le nombre maximal où E participe à A.

La cardinalité d'une association reliant 2 entités de contraintes respectives $(\min_1 - \max_1)$ et $(\min_2 - \max_2)$ est $\max_1 - \max_2$.

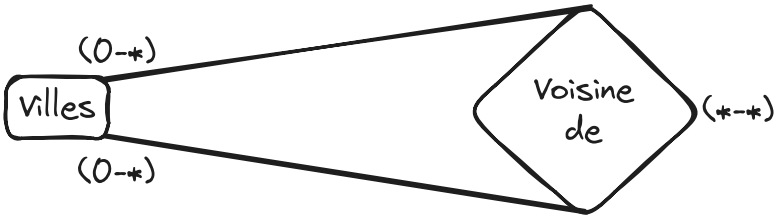
Exemple :



Cardinalités usuelles :

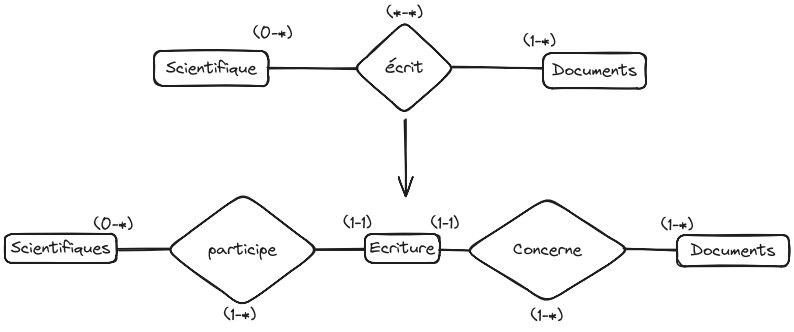
- $1-1$ (one to one)
- $1-*$ (one to many)
- $*-*$ (many to many)

Exemple :



Si on a dans le modèle entité-association des associations de cardinalités $*-*$, on doit avant de concevoir la base les transformer en deux associations de cardinalité $1-*$.

Exemple :



3. Modèle relationnel

= Organisation concrète des données dans la base

= interface avec l'utilisateur

Dans le modèle relationnel, une base de données est vue comme un ensemble de relations. Une relation (“table”) est un tableau à deux dimensions stockant des informations relatives à un même sujet. Ces informations correspondent aux colonnes de la relation, ils sont appelées attributs.

Définition :

Le domaine d’un attribut est l’ensemble des valeurs que peut prendre cet attribut.

(type + contraintes supplémentaires)

- chaines de caractères (TEXT)
- entiers (INTEGER)
- réels (REAL)

Schéma relationnel :

Relation R à n attributs $A_1, \ldots, A_n$ de domaines respectifs $D_1, \ldots, D_n$

$R(A_1 : D_1, \ldots, A_n : D_n)$

Définition :

Les lignes d’une relation R correspondant à des données, sont appelés enregistrements.

Clé primaire :

Chaque enregistrement d’une relation doit être unique, une clé primaire permet de faire respecter cette contrainte d’unicité.

Une clé primaire est un attribut ou ensemble d’attributs pour lequel la valeur de chaque enregistrement est distincte. On la souligne dans le schéma relationnel.

Exemple :

$R(\underline{A_1} : D_1, A_2 : D_2, \underline{A_3} : D_3, A_4 : D_4, \ldots)$

La clé première de R est (A_1, A_3). On choisit toujours le nombre minimum d’attributs possibles.

Clés étrangères :

Une clé étrangère d'une relation R_1 est un attribut ou ensemble d'attributs qui constituent la clé primaire d'une autre relation R_2. On dit que la clé étrangère de R_1 fait référence à R_2. Les clés étrangères font respecter la contrainte d'intégrité.

Schéma relationnel complet :

Exemple :

$R_1(\underline{A_{1,1}} : D_{1,1}, A_{1,2} : D_{1,2}, \ldots)$

$R_2(\underline{A_{2,1}} : D_{2,1} \to A_{2,2} : D_{2,2}, \ldots)$

$\ldots$

Conception d'une base de données = déterminer un schéma relationnel complet

- Dessiner le modèle $E - A$
- Transformer les cardinalités $* - *$ en deux $1 - *$
- Dans le schéma relationnel

entité $\to$ relation

association $1 - 1 \to$ fusionne relation

association $1 - * \to$ clé étrangère (dans "1 faisant référence à $*$)

Exemple :

Passage du modèle entité-association au modèle relationnel.

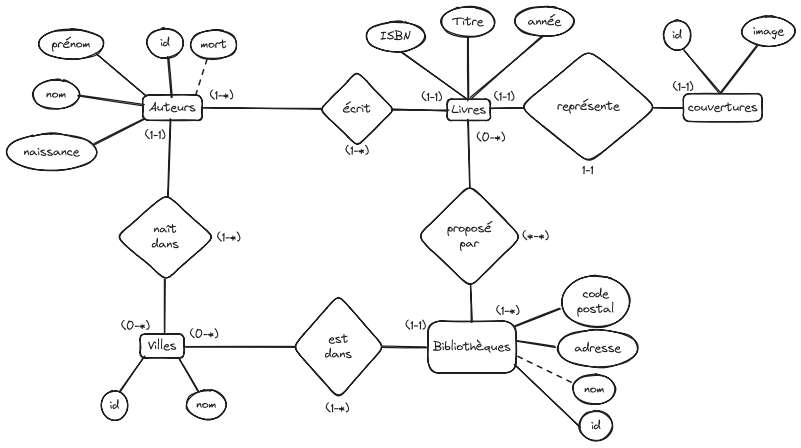


Schéma relationnel complet :

Auteurs($\underline{id}$: INT, prénom : TEXT, nom : TEXT, naissance : TEXT, mort : TEXT, ville_naissance $\rightarrow$ Villes.id INTEGRER)

Villes($\underline{id}$: INT, nom : TEXT)

Bibliothèques($\underline{id}$: INT, nom : TEXT, adresse : TEXT, code postal : INT, ville $\rightarrow$ Villes.id : INTEGRER)

Livres($\underline{isbn}$: TEXT, titre : TEXT, année : INT, image : TEXT, auteur $\rightarrow$ Auteurs.id :INTEGRER)

Inventaire($\underline{biblio} \rightarrow$ Bibliothèques.id : INTEGRER, $\underline{livre} \rightarrow$ Livres.isbn : TEXT, nb_exemplaire : INTEGRER)

II. Requêtes SQL

SQL signifie Structured Query Language.

Quelques règles :

- MOTS-CLES en majuscule
- Relations en minuscule sauf l'initiale
- Attributs en minuscule
- retours à la ligne fréquents (chaque mot-clé)
- éviter accents, caractères spéciaux
- Seules les requêtes de lecture de données sont au programme

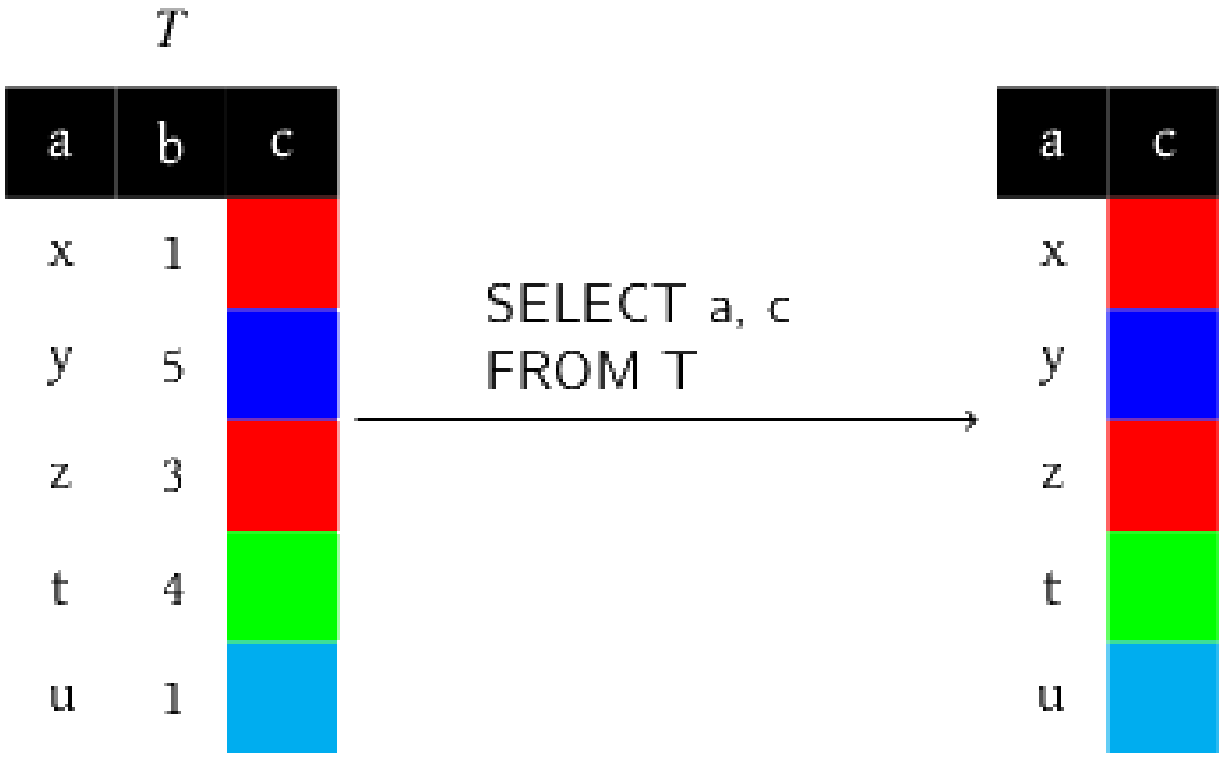
1. Projection

Mots-clefs : SELECT, FROM, DISTINCT.

On sélectionne tous les attributs avec $*$.

Les opérateurs sont : $(+, -, *, /$,%,ABS, LOWER,UPPER,LENGTH,||).

Les enregistrements dans le résultat d'une projection sont dans un ordre arbitraire.



2. Formater une projection

Mots-clefs : AS, ORDER BY (ASC / DESC), LIMIT, OFFSET.

Le mot-clef OFFSET doit toujours être précédé de LIMIT.

ORDER BY utilise l'ordre lexicographique.

Exemple :

SELECT b AS "autre nom"
FROM R_1

SELECT b
FROM R_1
ORDER BY b – ou ASC

SELECT b
FROM R_1
ORDER BY b DESC
LIMIT 2

SELECT b
FROM R_1
ORDER BY b DESC
LIMIT 2
OFFSET 1


T
a
b
c
x
1
y
5
z
3
t
4
u
1
SELECT a AS d, c
FROM T
d
c
x
y
z
t
u

T

a	b	c
x	1	
y	5	
z	3	
t	4	
u	1	

SELECT *
FROM T
ORDER BY b ASC

→

a	b	c
x	1	
u	1	
z	3	
t	4	
y	5	

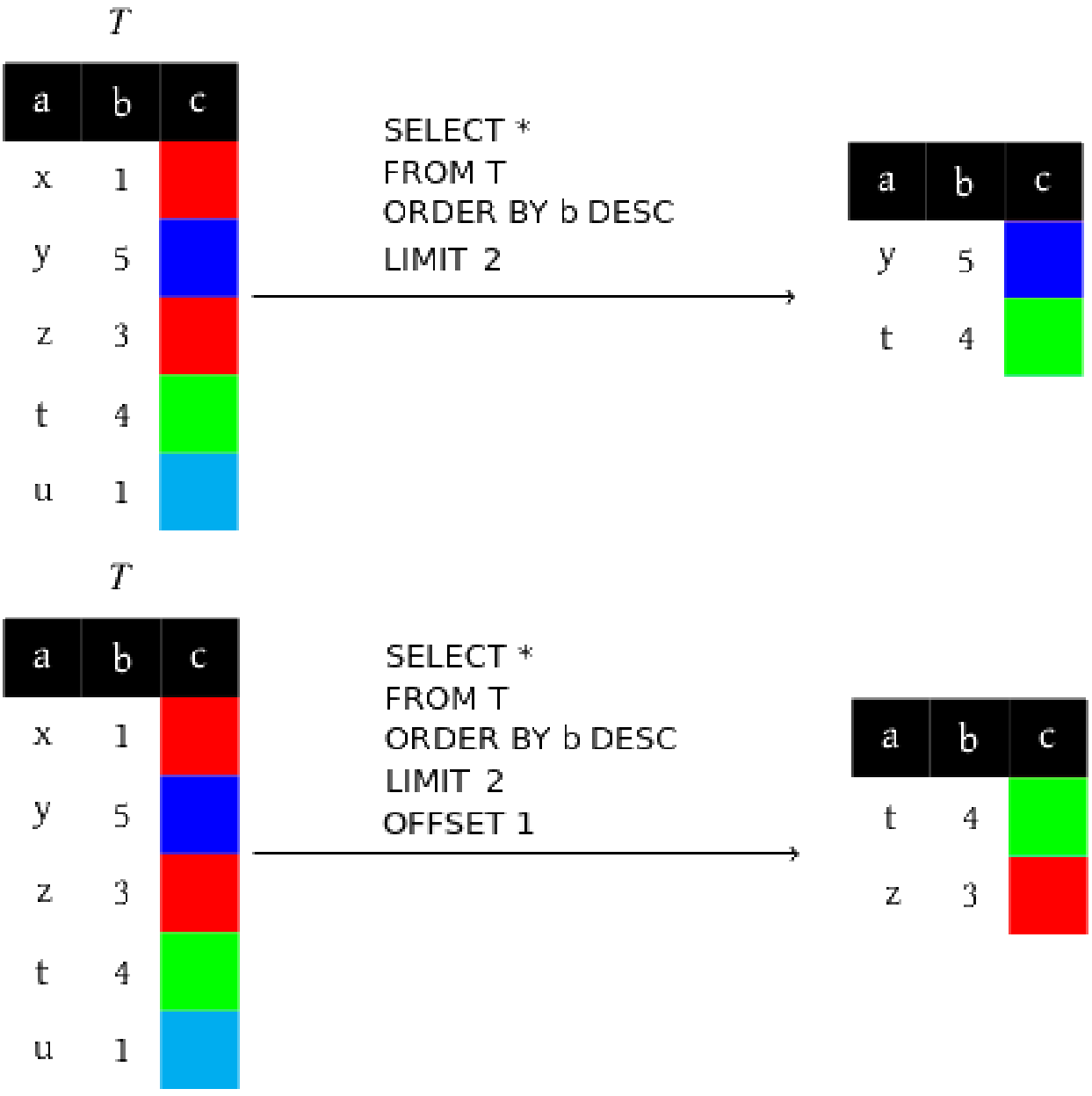
T

a	b	c
x	1	
y	5	
z	3	
t	4	
u	1	

SELECT *
FROM T
ORDER BY b DESC

→

a	b	c
y	5	
t	4	
z	3	
x	1	
u	1	



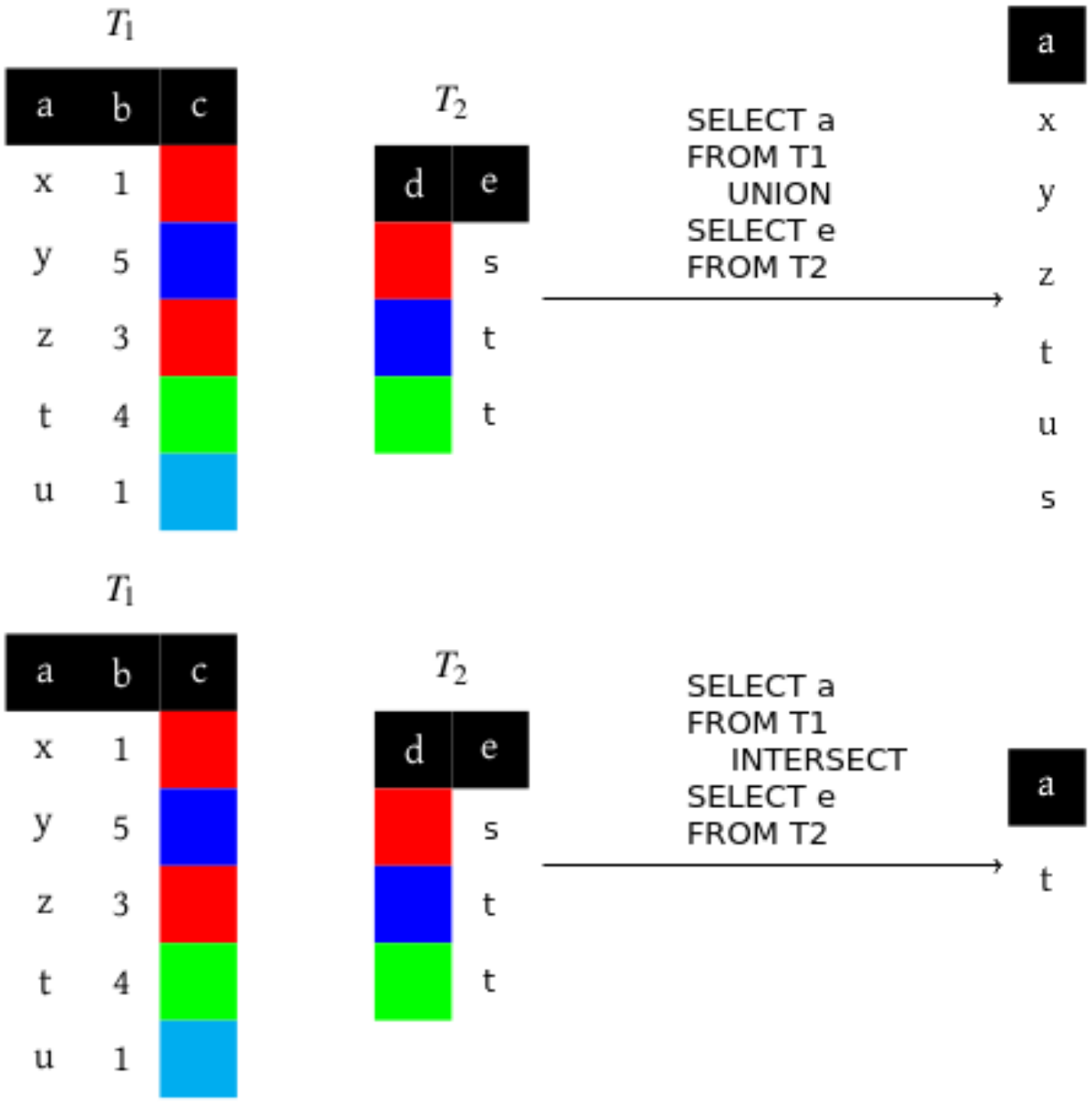
3. Opérations ensemblistes

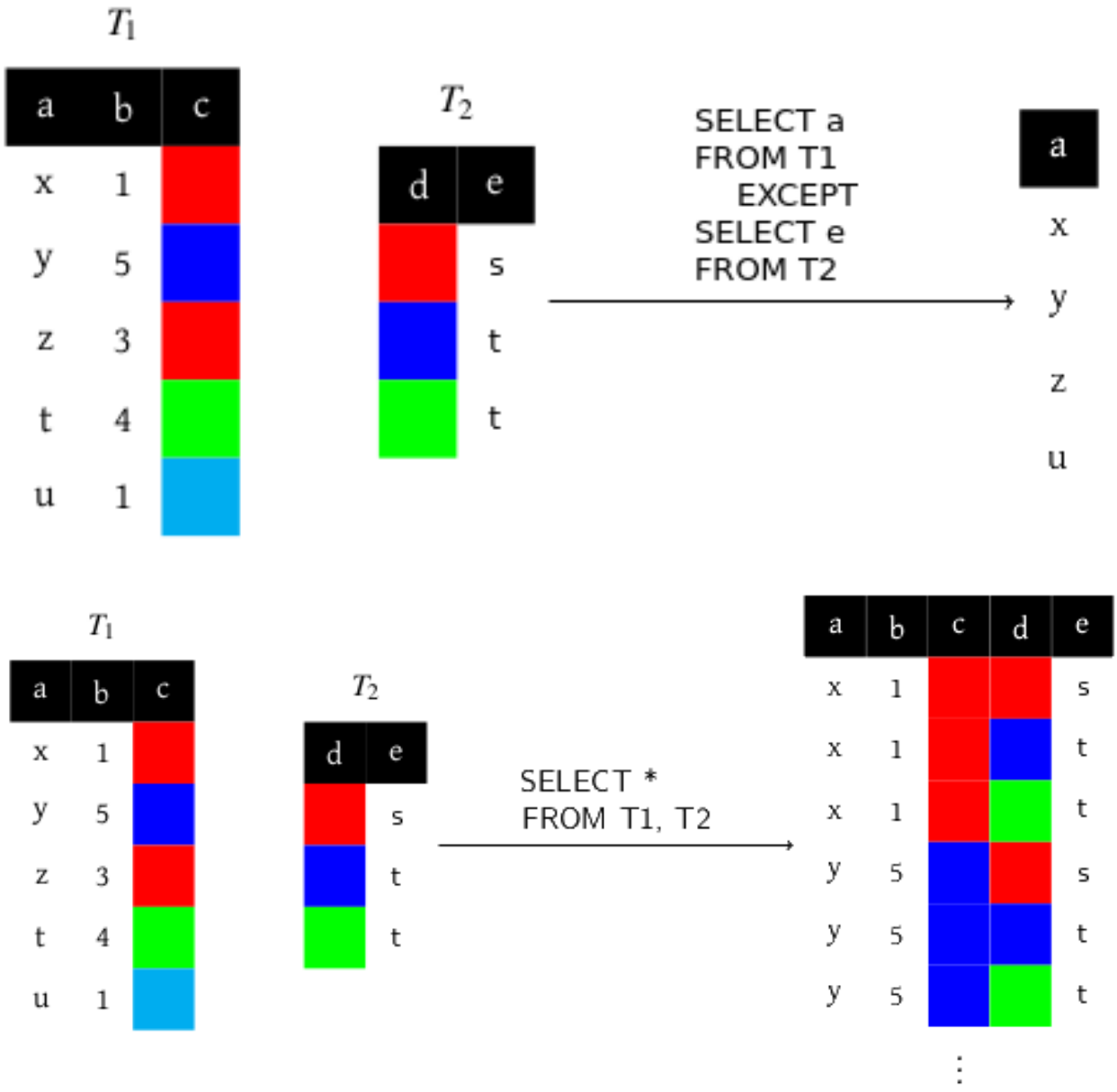
Mots-clef : UNION, INTERSECT, EXCEPT.

Le résultat des requêtes reliées par un opérateur ensembliste doivent avoir le même nombre de colonnes, et les colonnes aux mêmes positions doivent avoir le même domaine. Les colonnes du résultat porteront le même nom que ceux de la première requête. Le résultat ne comportera aucun doublon.

Produit cartésien : FROM Relation1, Relation2, ...

Exemple :


T_1
a b c
x 1
y 5
z 3
t 4
u 1
T_2
d e
s
t
t
SELECT a
FROM T1
UNION
SELECT e
FROM T2
a
x
y
z
t
u
s
T_1
a b c
x 1
y 5
z 3
t 4
u 1
T_2
d e
s
t
t
SELECT a
FROM T1
INTERSECT
SELECT e
FROM T2
a
t



Le nombre d'enregistrements du résultat est égal au nombre d'enregistrements de $R_1 \times$ le nombre d'enregistrements de R_2.

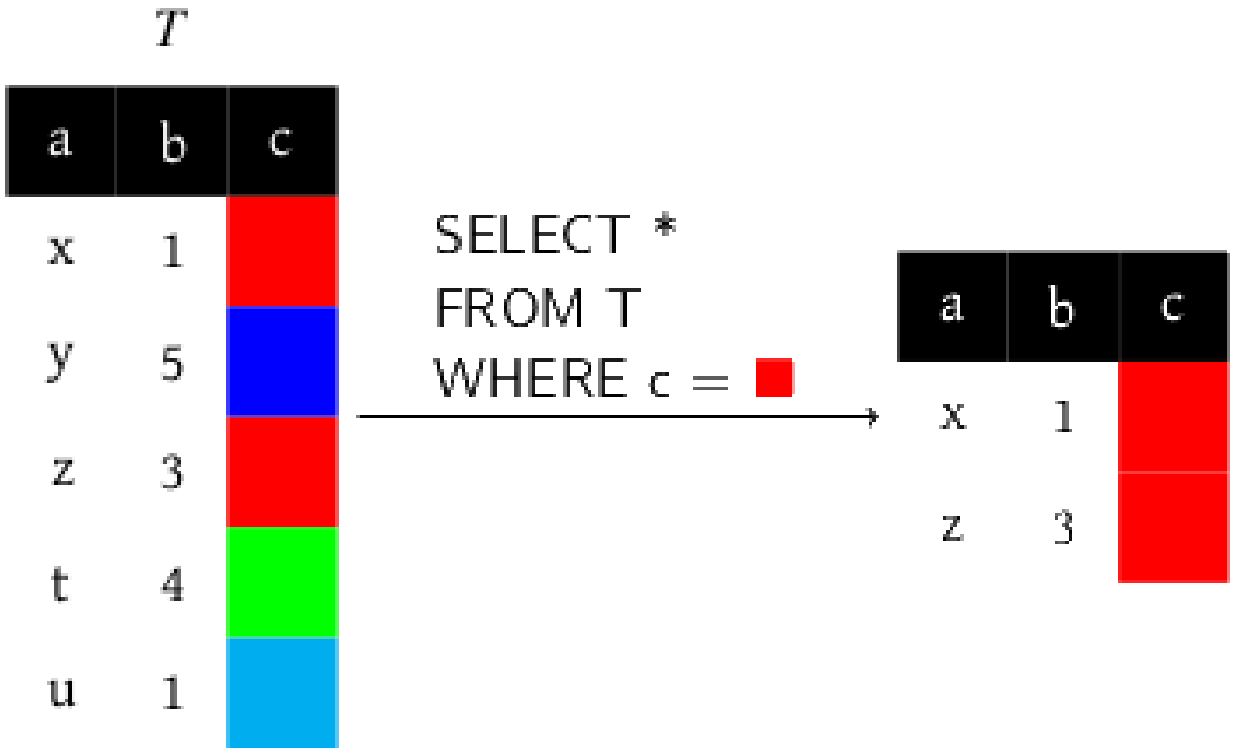
C'est généralisable à autant de relation que nécessaire.

4. Sélection

Mots-clefs : WHERE, AND, OR, NOT, IS (NOT) NULL, LIKE, %, _.

Opérateurs booléens : $=, <>, \leq, <, \geq, >$.

Exemple :



5. Jointures

Mots-clefs (jointures internes): JOIN ... ON ...

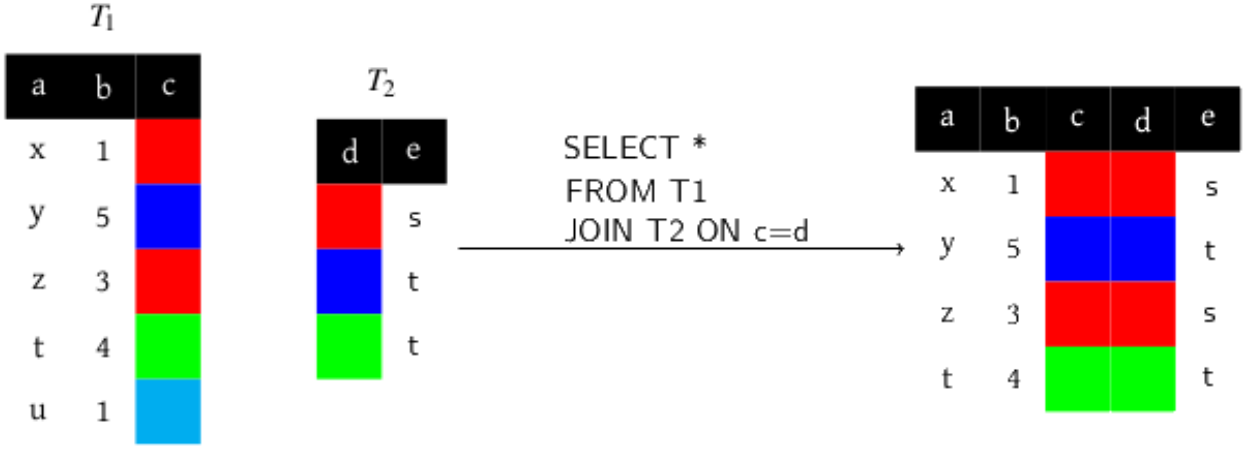
Mots-clefs (jointures externes): LEFT JOIN ... ON ....

On peut relier deux relations selon un attribut en commun.

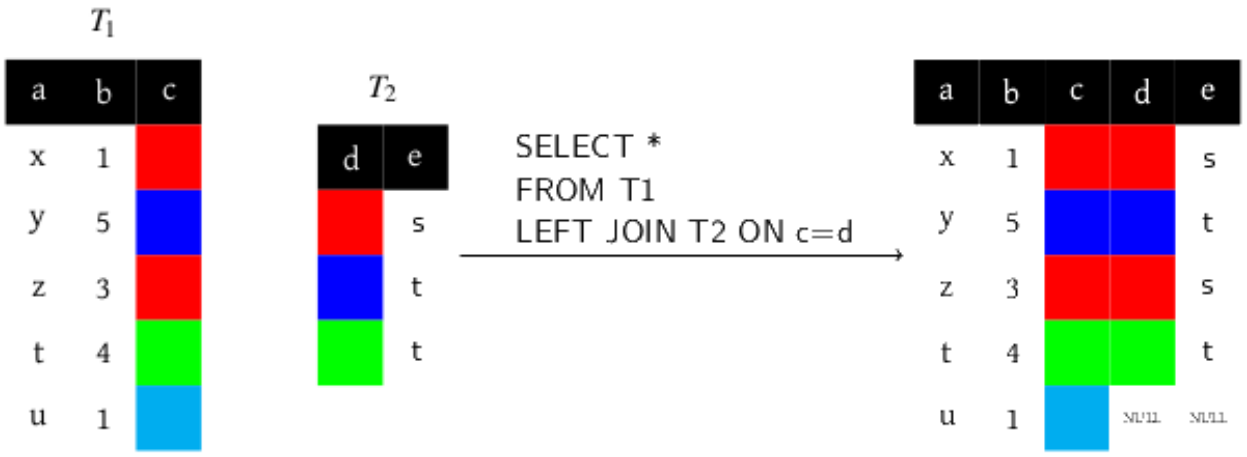
On peut enchaîner plusieurs jointures si on veut relier plusieurs tables.

Si deux relations à relier ne sont pas liées par une clé étrangère, on peut éventuellement passer par des relations intermédiaires.

Jointure interne :



Jointure externe :



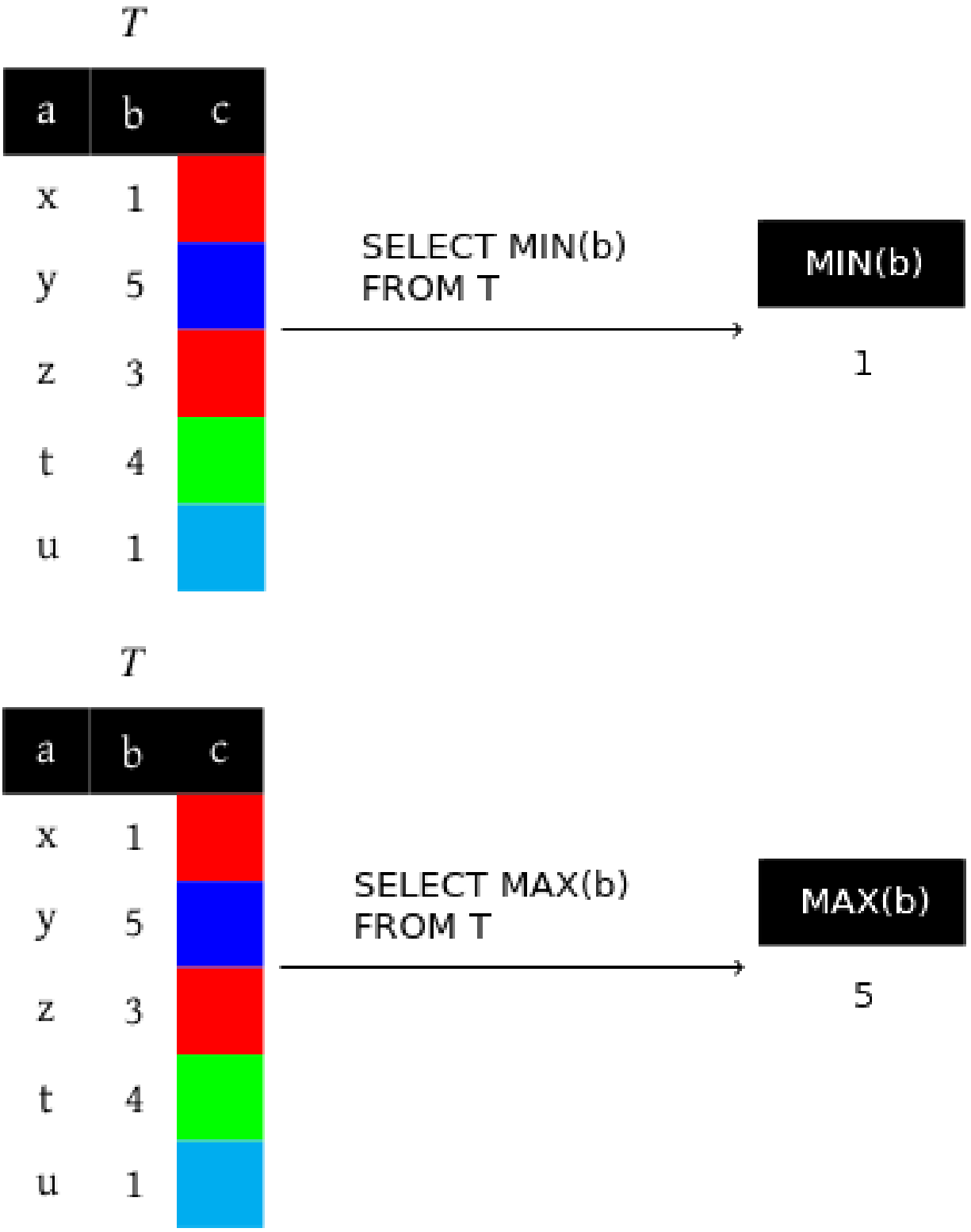
Autojointures :

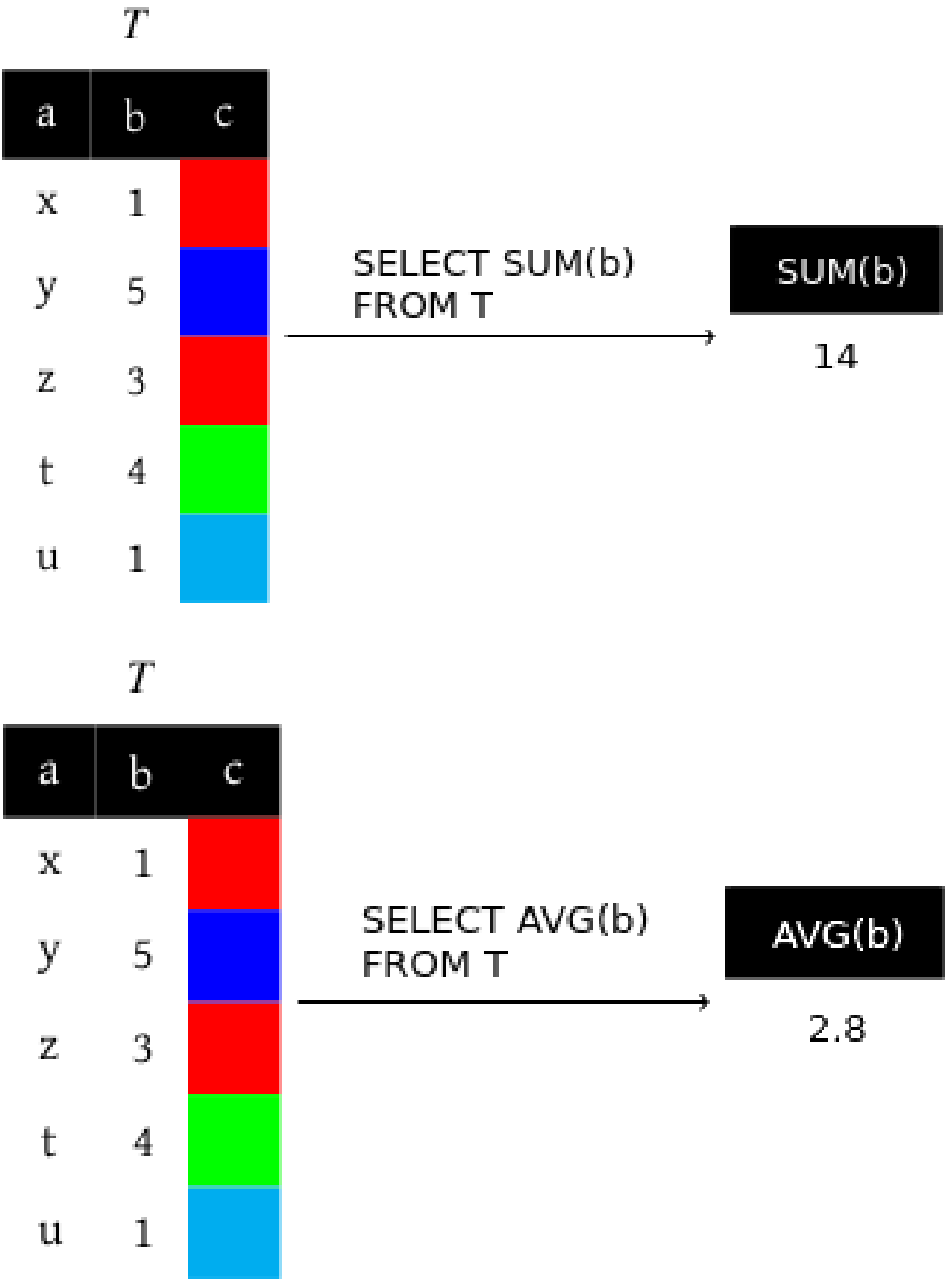
SELECT $X.a, X.a, X.c$
FROM R_1 AS X
JOIN R_1 AS Y ON $X.c = Y.c$

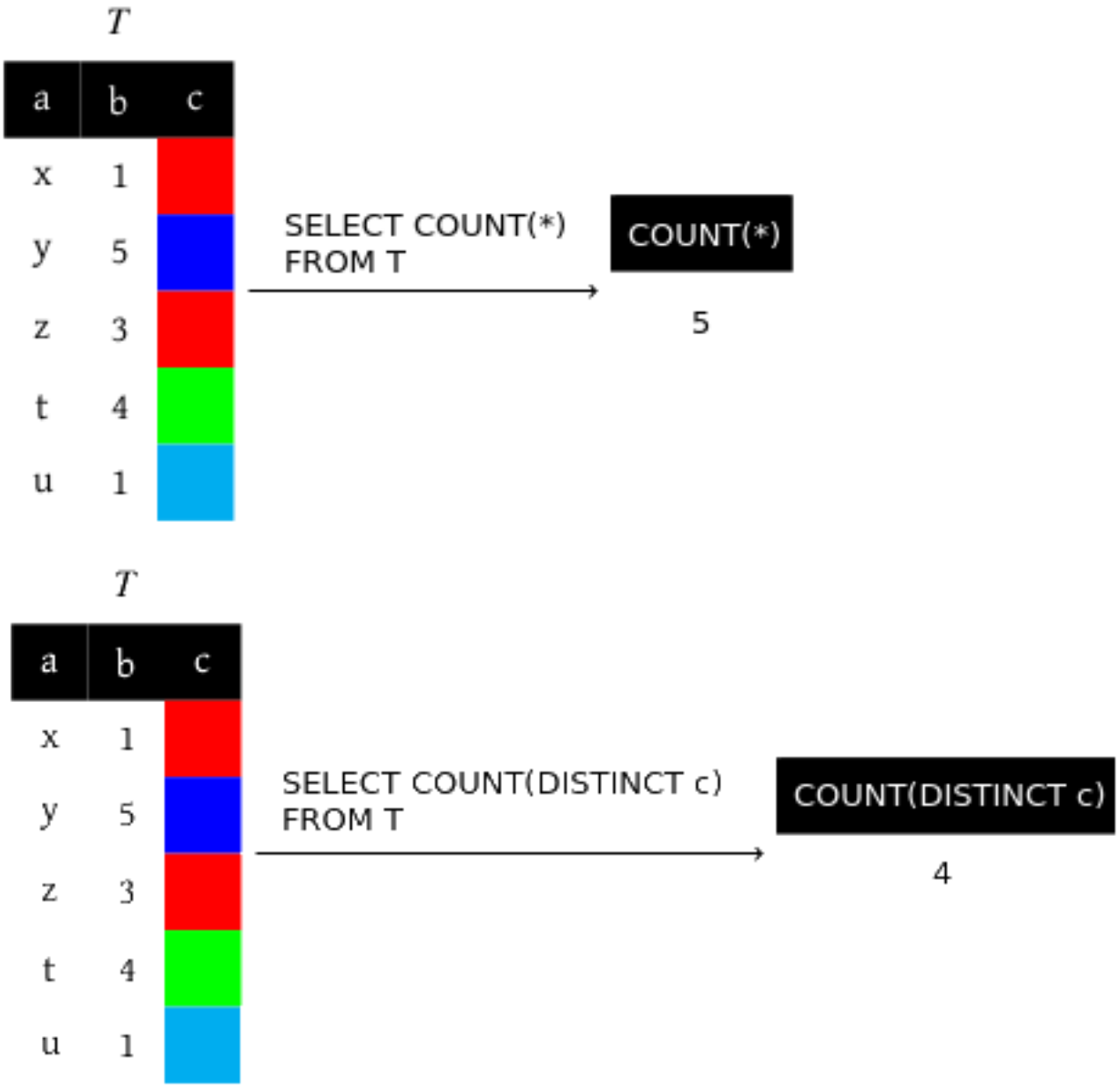
6. Fonctions d'agrégation

Mots-clefs : MIN, MAX, SUM, AVG, COUNT.

On obtient une ligne dans le résultat.


T
a
b
c
x
1
y
5
z
3
t
4
u
1
SELECT MIN(b)
FROM T
MIN(b)
1
T
a
b
c
x
1
y
5
z
3
t
4
u
1
SELECT MAX(b)
FROM T
MAX(b)
5


T
a
b
c
x
1
y
5
z
3
t
4
u
1
SELECT SUM(b)
FROM T
SUM(b)
14
T
a
b
c
x
1
y
5
z
3
t
4
u
1
SELECT AVG(b)
FROM T
AVG(b)
2.8



COUNT(∗) : toutes les lignes.

COUNT(attribut) : toutes les lignes où attribut IS NOT NULL.

COUNT(DISTINCT attribut) : compte pour 1 si attribut a un doublon.

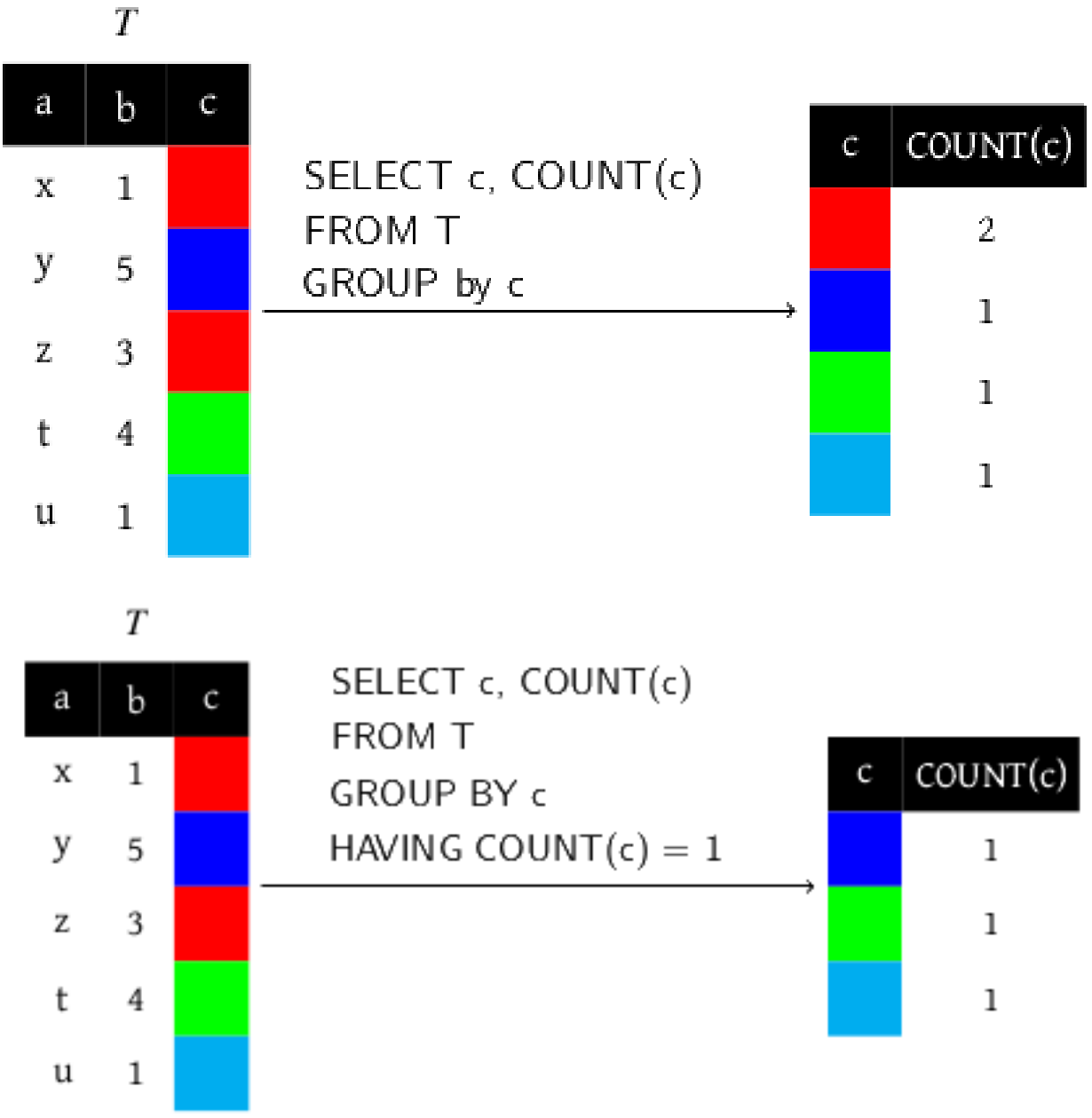
7. Regroupement et filtrage des groupes

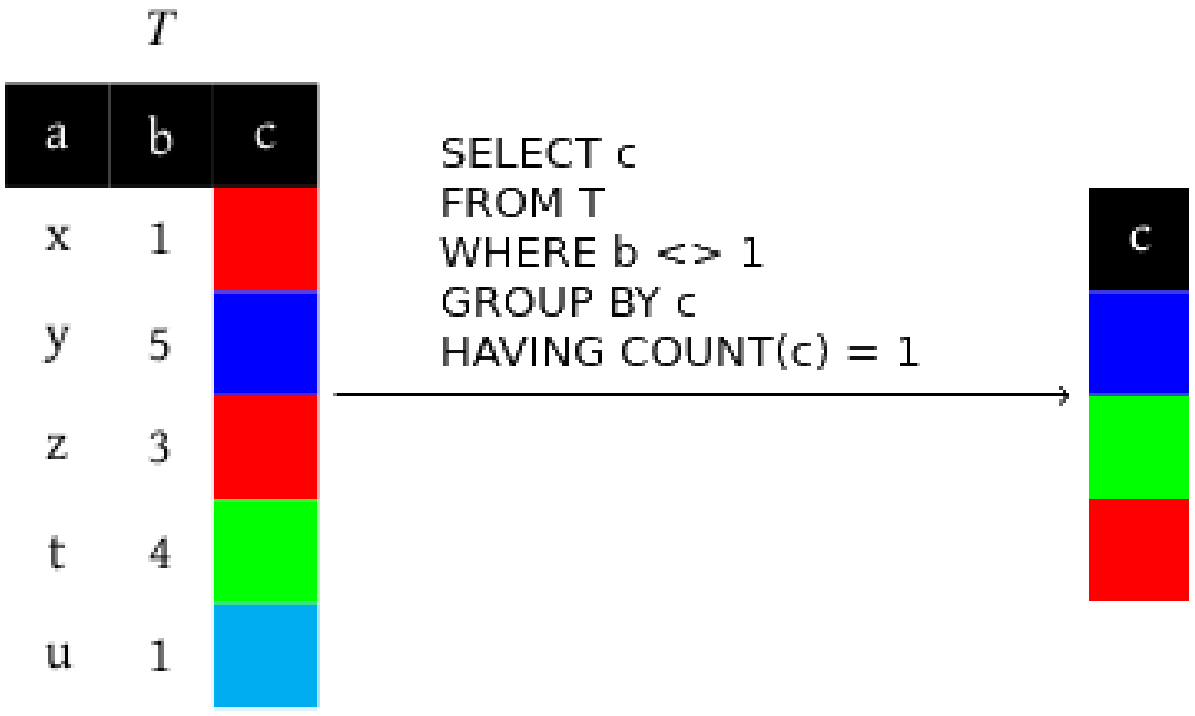
Mots-clefs : GROUP BY, HAVING.

Le WHERE filtre les enregistrements (avant que les groupes soient faits), le HAVING filtre les groupes.

Les fonctions d'agrégation s'appliquent sur chaque groupe.

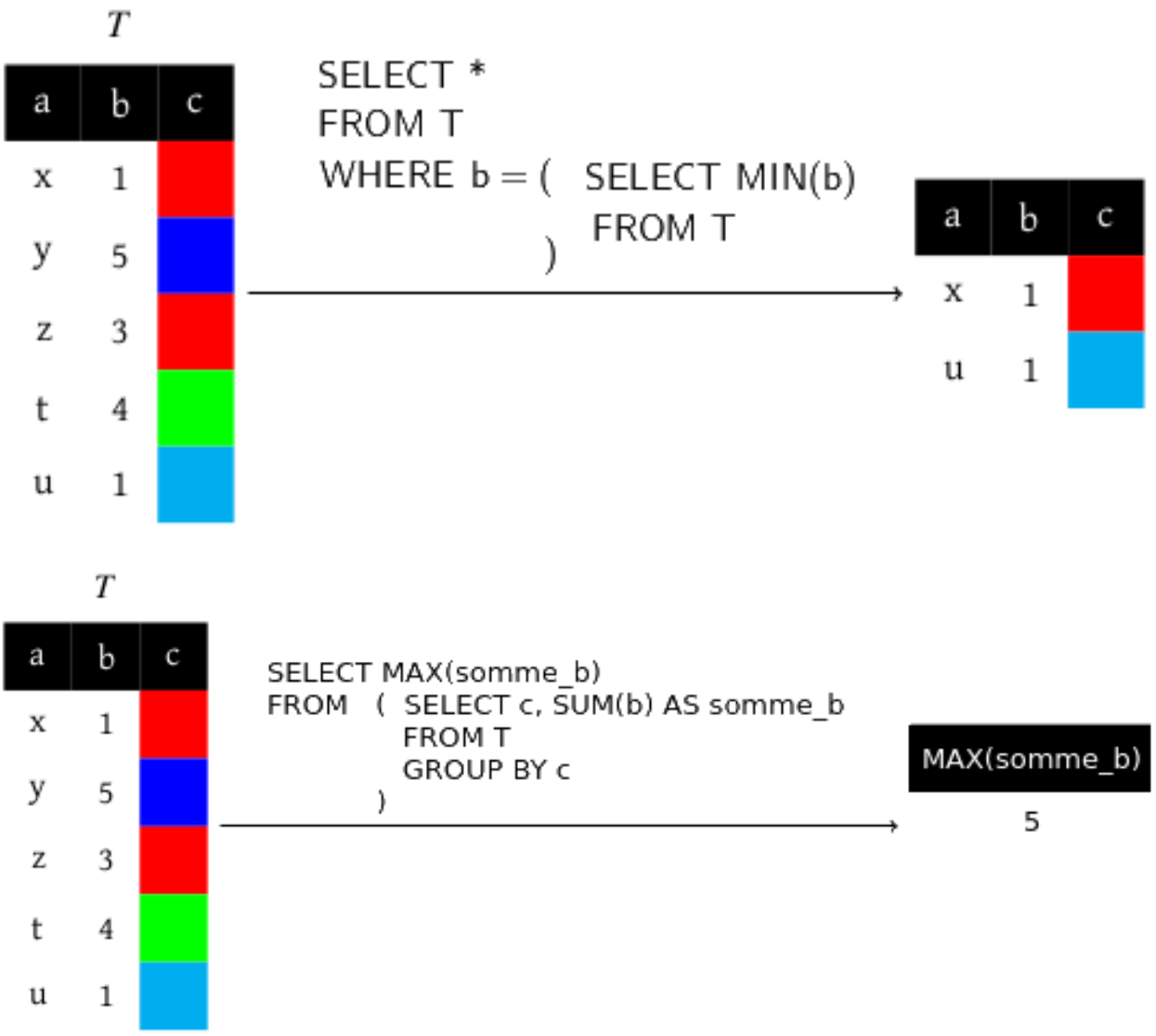
Avec having count(∗) <> 2. La première ligne n'apparaît plus.





8. Requêtes imbriquées

Pour imbriquer une requête, on la met entre parenthèses.



On ne peut pas imbriquer les fonctions d'agrégation, on utilise donc une requête imbriquée.

```
SELECT UNION | INTERSECT | EXCEPT DISTINCT
FROM
JOIN ... ON
.
.
.
WHERE
GROUP BY
HAVING
ORDER BY ASC|DESC
LIMIT
OFFSET
```

Souvent aux concours :

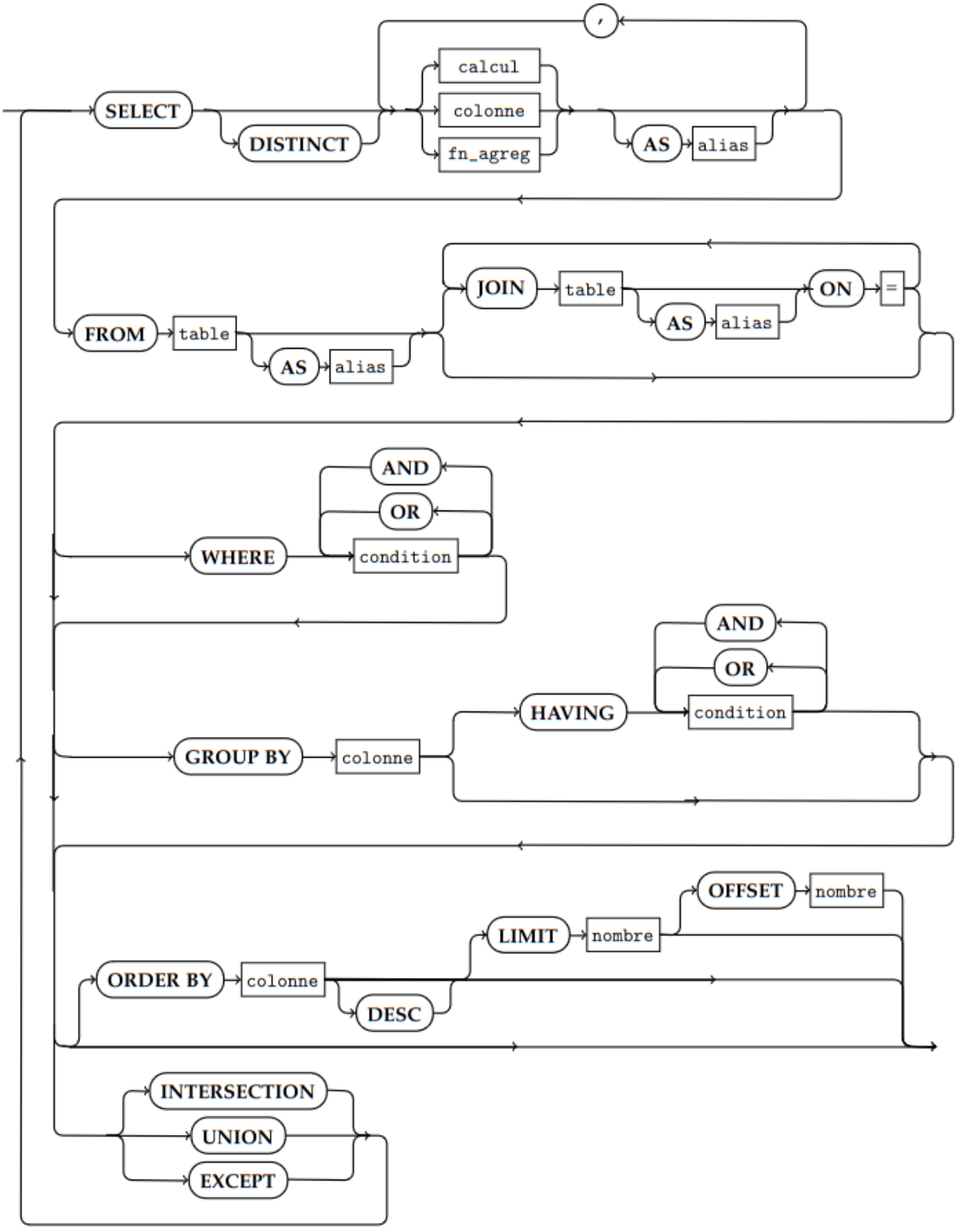
- Jointures
- Group by/Having
- Requêtes imbriquées

Remarque :

On peut écrire $R_1.c$ ou c au lieu de FROM R_1
GROUP BY c

On peut se passer du nom de la relation que si il n'y a aucun doublon.

Conclusion


,
SELECT
DISTINCT
calcul
colonne
fn_agreg
AS
alias
FROM
table
AS
alias
JOIN
table
AS
alias
ON
=
AND
OR
WHERE
condition
AND
OR
HAVING
condition
GROUP BY
colonne
OFFSET
nombre
LIMIT
nombre
ORDER BY
colonne
DESC
INTERSECTION
UNION
EXCEPT