Support de Cours : Algèbre Relationnelle & SQL

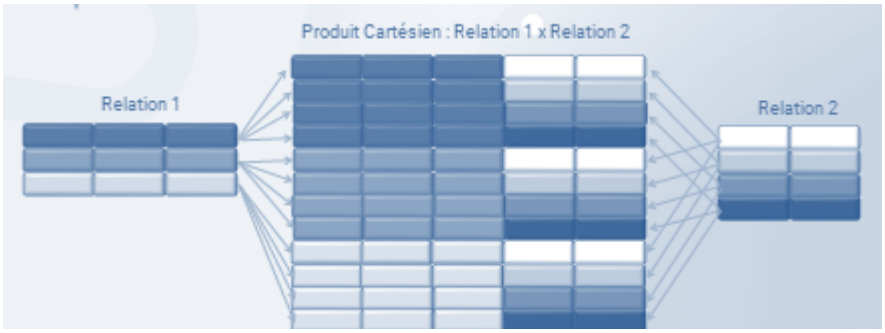
Séquence 2 : Grain 3.1

I- INTRODUCTION

Après avoir étudié la sélection (filtrer des lignes) et la projection (choisir des colonnes), nous abordons un opérateur clé pour interroger des données liées : **la jointure**. L'o**bjectif du cours** est de Comprendre les concepts de **produit cartésien** et de jointure (appelée aussi **θ-jointure)** en algèbre relationnelle et leur traduction en SQL pour construire des requêtes élaborées.
Souvent, on imagine une jointure comme un simple moyen de «coller » deux tables ensemble. Cependant, cette représentation mentale est bien trop limitée pour écrire des requêtes complexes.
Alors, comment ça marche *vraiment* ?
La clé, c'est de comprendre que la jointure est un **filtrage actif**… à partir d'un *produit cartésien.*
Mais, c'est quoi d'abord un produit cartésien ?

II- PRODUIT CARTESIEN

Définition : Le produit cartésien est probablement l'opérateur dont la compréhension et la plus fondamentale pour penser et écrire des requêtes non triviales. Le produit cartésien combine deux tables (R1 et R2) pour générer une nouvelle table contenant toutes les combinaisons possibles entre chaque ligne de R1 et chaque ligne de R2. Plus formellement, c'est une combinaison de toutes les lignes de deux tables R1 et R2, générant une table avec nb_lignes(R1)×nb_lignes(R2) entrées.



- **Notation** : R1×R2
- **En SQL** : SELECT * FROM R1 CROSS JOIN R2;
- **EXEMPLE** : **MATIERE (ID_M, LIBELLE, PROF) X NOTES (ETUD, COURS, NOTE)**

Matière

ID_M	LIBELLÉ	PROF
1	Python	7
2	BD	6
3	Java	4
4	IA	1

MATIERE(ID_M, LIBELLE, PROF)

Matière X NOTES

ID_M	LIBELLÉ	PROF	ETUD	COURS	NOTE
1	Python	7	3	2	14
1	Python	7	3	3	12
1	Python	7	5	2	15
2	BD	6	3	2	14
2	BD	6	3	3	12
2	BD	6	5	2	15
3	Java	4	3	2	14
3	Java	4	3	3	12
3	Java	4	5	2	15
4	IA	1	3	2	14
4	IA	1	3	3	12
4	IA	1	5	2	15

Notes

ETUD	COURS	NOTE
3	2	14
3	3	12
5	2	15

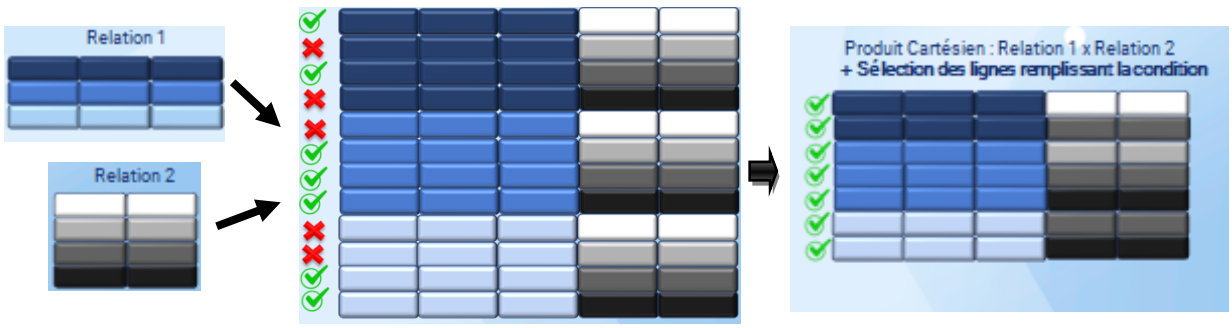
NOTES (ETUD, COURS, NOTE)

❖ **Equivalent SQL** : **SELECT * FROM MATIÈRE CROSS JOIN NOTES;**

Le produit cartésien permet donc de produire toutes les possibilités de combinaison entre les lignes de deux tables sans qu'il y ait de lien entre elles. Le lien sera amené par le prochain opérateur : LA JOINTURE

III- JOINTURE

Définition : La **Jointure** se construit sur le **produit cartésien** mais amène du sens on ne conservant que les lignes pertinentes remplissant une condition donnée.
La **Jointure** se note avec deux petits triangles qui se font face (pour donner la forme d'une **cravate**). Elle prend deux relations et une **condition** (expression logique) pour produire une nouvelle relation



La Jointure est appelée aussi : Thêta-jointure ou θ-Jointure. En anglais JOIN. Le nom Thêta-jointure est utilisé pour des raisons historiques. Dans les premiers articles sur l'algèbre relationnelle le symbole θ désignait une condition. On parle de produit cartésien suivi d'une restriction (filtre) basée sur une condition logique θ

- **Notation :** $R1 \bowtie_{\theta} R2$
- **En SQL :** SELECT * FROM R1 JOIN R2 ON condition_theta;
- **Types de conditions θ:** Comparaisons (=,>,<=,>,<), opérations logiques (ET, OU, NON).
- **EXEMPLE : MATIERE (ID_M, LIBELLE, PROF)** $\bowtie_{(NOTE>12)}$ **NOTES (ETUD, COURS, NOTE)**
  - ➔ **Condition :** Conserver uniquement les combinaisons où Note > 12

Matière

ID_M	LIBELLÉ	PROF
1	Python	7
2	BD	6
3	Java	4
4	IA	1

Notes

ETUD	COURS	NOTE
3	2	14
3	3	12
5	2	15

ID_M	LIBELLÉ	PROF	ETUD	COURS	NOTE
1	Python	7	3	2	14
~~1~~	~~Python~~	~~7~~	~~3~~	~~3~~	~~12~~
1	Python	7	5	2	15
2	BD	6	3	2	14
~~2~~	~~BD~~	~~6~~	~~3~~	~~3~~	~~12~~
2	BD	6	5	2	15
3	Java	4	3	2	14
~~3~~	~~Java~~	~~4~~	~~3~~	~~3~~	~~12~~
3	Java	4	5	2	15
4	IA	1	3	2	14
~~4~~	~~IA~~	~~1~~	~~3~~	~~3~~	~~12~~
4	IA	1	5	2	15

Matière ⋈ NOTES
Note > 12

ID_M	LIBELLÉ	PROF	ETUD	COURS	NOTE
1	Python	7	3	2	14
1	Python	7	5	2	15
2	BD	6	3	2	14
2	BD	6	5	2	15
3	Java	4	3	2	14
3	Java	4	5	2	15
4	IA	1	3	2	14
4	IA	1	5	2	15

❖ **Equivalent SQL :** **SELECT * FROM MATIÈRE JOIN NOTES ON NOTES.NOTE >12;**

Dans cet exemple, la jointure commence par générer un produit cartésien entre les tables Matière et Notes. Par exemple, ici Matière contient 4 entrées (Python, BD, JAVA et IA) et Notes 3 entrées (14, 12 et 15), cette étape produit temporairement 12 lignes. Ensuite, la jointure applique un filtre conditionnel pour ne conserver que les associations répondant à un critère précis. Dans ce cas, seules les combinaisons où la Note est strictement supérieure à 12 sont conservées. Ainsi, les lignes avec les notes 14 et 15 restent, tandis que celles avec la note inférieure ou égale à 12 seront éliminées. Ici, la condition θ (NOTE >12) filtre le produit cartésien initial.

IV- CONCLUSION

Trois points clés à retenir :

- Une jointure (ou **θ** –jointure) est un **produit cartésien** suivi d'un **filtre intelligent (une restriction)**.
- Le symbole **θ** représente la condition qui donne son sens à la jointure.
- Maîtriser cette logique permet de concevoir des requêtes **précises et efficaces**.

Pour conclure, la θ-Jointure transforme une combinaison brute de données en un résultat cohérent et exploitable, en éliminant le « bruit » généré par le produit cartésien. Nous approfondirons ces notions en classe à travers des exercices pratiques et des études de cas.