# III- Opérateurs n-aires :

## 1 – Produit cartésien :

R1 de schéma S1 = ( $A_1$ , $A_2$ .... , $A_n$ )

R2 de schéma S2 = ( $B_1$ , $B_2$ .... , $B_p$ )

S1 et S2 sont disjoints (sinon renommer les attributs )

Le produit cartésien de R1 et R2 noté R1 × R2 est de schéma

S = ( $A_1$ , $A_2$ .... , $A_n$ , $B_1$ , $B_2$ .... , $B_p$)

( $a_1$ , $a_2$ .... , $a_n$ , $b_1$ , $b_2$ .... , $b_p$) appartient à R1 × R2 ssi ( $a_1$ , $a_2$ .... , $a_n$) appartient à R1 et ( $b_1$ , $b_2$ .... , $b_n$) appartient à R2

## Exemple :

**Livre**

| Titre | genre | Nom1 |
|---|---|---|
| Germinal | roman | Zola |
| Les fleurs de mal | poésie | Baudelaire |
| L'assommoir | roman | Zola |

**Auteur**

| Nom2 | Date-naissance |
|---|---|
| Zola | 1840 |
| Baudelaire | 1821 |

**Livre × Auteur**

| Titre | genre | Nom1 | Nom2 | Date_naissance |
|---|---|---|---|---|
| Germinal | roman | Zola | Zola | 1840 |
| Germinal | roman | Zola | Baudelaire | 1821 |
| Les fleurs de mal | Poésie | Baudelaire | Zola | 1840 |
| Les fleurs de mal | Poésie | Baudelaire | Baudelaire | 1821 |
| L'assommoir | roman | Zola | Zola | 1840 |
| L'assommoir | roman | Zola | Baudelaire | 1821 |

## Produit cartésien en langage SQL :

Le produit cartésien $R \times S$ s'exprime très simplement en incluant plusieurs relations dans la clause FROM de la commande SELECT:

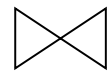
```
SELECT * FROM R , S ;
```

# 2- Jointure :

**La jointure consiste donc à rapprocher les lignes de deux relations pour lesquelles les valeurs d'un (ou plusieurs) attributs sont identiques.**

| Titre | genre | Nom-auteur |
|---|---|---|
| Germinal | roman | Zola |
| Les fleurs de mal | poésie | Baudelaire |
| L'assommoir | roman | Zola |

| Nom | Date-naissance |
|---|---|
| Zola | 1840 |
| Baudelaire | 1821 |

**Livre**  **Auteur**

**Nom-auteur=Nom**

| Titre | genre | Nom_auteur | Nom | Date_naissance |
|---|---|---|---|---|
| Germinal | roman | Zola | Zola | 1840 |
| Germinal | roman | Zola | Baudelaire | 1821 |
| Les fleurs de mal | Poésie | Baudelaire | Zola | 1840 |
| Les fleurs de mal | Poésie | Baudelaire | Baudelaire | 1821 |
| L'assommoir | roman | Zola | Zola | 1840 |
| L'assommoir | roman | Zola | Baudelaire | 1821 |

$\pi$ Titre, genre, Nom, Date_naissance (Livre $\bowtie$ Auteur)
Nom-auteur=Nom

| Titre | genre | Nom | Date_naissance |
|---|---|---|---|
| Germinal | roman | Zola | 1840 |
| Les fleurs de mal | Poésie | Baudelaire | 1821 |
| L'assommoir | roman | Zola | 1840 |

**En langage de l'algèbre relationnelle, la jointure des tables R1 et R2 selon une certaine contrainte sur les lignes a sa notation propre qui, selon les auteurs, peut s'écrire :**

**R1[contrainte]R2 ou R1 $\bowtie$ R2**
**[contrainte]**

**Peut être simplement définie comme équivalent à**

$$\sigma_{[Contrainte]}(R1 \times R2)$$

La jointure simple s'exprime en utilisant l'opérateur JOIN :

```
SELECT * FROM R1 JOIN R2 ON R1.A=R2.B
```

**Exemple :**

```
SELECT * FROM Livre JOIN Auteur ON Livre.Nom_Auteur= Auteur.Nom
```

**Remarque :** Jointure sans opérateur JOIN

```
SELECT * FROM Livre,Auteur where Livre.Nom_Auteur= Auteur.Nom
```

# III- Renommage :

**Le renommage permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 (ou plusieurs ) relations opérandes dans une opération.**

**On définit l'opérateur de renommage d'un attribut A en B (d'une Relation R) sous la forme:** $\rho_{A \to B}(R)$

**Exemple :** $\rho_{\text{nom- auteur} \to \text{nom1}}(\text{auteur})$

**SQL : SELECT** nom-auteur **AS** nom1 **FROM** auteur;

# Remarque :

**Si deux table possède le même nom d'attribut La première solution pour lever l'ambigüité est de préfixer un attribut par le nom de la table d'où il provient.**

**Application :** Soit une base de données d'un hôtel contenant les tables suivantes :

*batiment*

| nom | etoiles |
|---|---|
| Rose | 3 |
| Jasmin | 2 |
| Lys | 3 |

*Chambre*

| numero | batiment | fenetres |
|---|---|---|
| 1 | Rose | 2 |
| 2 | Rose | 1 |
| 3 | Rose | 1 |
| 1 | Jasmin | 1 |
| 2 | Jasmin | 0 |
| 3 | Jasmin | 1 |
| 4 | Jasmin | 1 |
| 1 | Lys | 3 |
| 2 | Lys | 2 |
| 3 | Lys | 2 |

*lit*

| idlit | chambre | batlit |
|---|---|---|
| 1 | 1 | Rose |
| 2 | 1 | Rose |
| 3 | 2 | Rose |
| 4 | 3 | Rose |
| 5 | 1 | Jasmin |
| 6 | 2 | Jasmin |
| 7 | 2 | Jasmin |
| 8 | 3 | Jasmin |
| 9 | 3 | Jasmin |
| 10 | 4 | Jasmin |
| 11 | 1 | Lys |
| 12 | 2 | Lys |
| 13 | 3 | Lys |

*nuits*

| Client | Lit | date |
|---|---|---|
| Lennon | 1 | 15-08-2014 |
| McCartney | 8 | 18-08-2014 |
| Starr | 3 | 03-07-2014 |
| Harrison | 2 | 01-08-2014 |
| Page | 10 | 05-08-2014 |
| Plant | 1 | 13-08-2014 |
| Jones | 11 | 05-08-2014 |
| Bonham | 7 | 02-08-2014 |
| Townshend | 1 | 08-08-2014 |

1- Identifier dans chaque table les attributs qui définissent sa clé primaire

*batiment*

| nom | etoiles |
|---|---|
| Rose | 3 |
| Jasmin | 2 |
| Lys | 3 |

*Chambre*

| numero | batiment | fenetres |
|---|---|---|
| 1 | Rose | 2 |
| 2 | Rose | 1 |
| 3 | Rose | 1 |
| 1 | Jasmin | 1 |
| 2 | Jasmin | 0 |
| 3 | Jasmin | 1 |
| 4 | Jasmin | 1 |
| 1 | Lys | 3 |
| 2 | Lys | 2 |
| 3 | Lys | 2 |

*lit*

| idlit | chambre | batlit |
|---|---|---|
| 1 | 1 | Rose |
| 2 | 1 | Rose |
| 3 | 2 | Rose |
| 4 | 3 | Rose |
| 5 | 1 | Jasmin |
| 6 | 2 | Jasmin |
| 7 | 2 | Jasmin |
| 8 | 3 | Jasmin |
| 9 | 3 | Jasmin |
| 10 | 4 | Jasmin |
| 11 | 1 | Lys |
| 12 | 2 | Lys |
| 13 | 3 | Lys |

*nuits*

| Client | Lit | date |
|---|---|---|
| Lennon | 1 | 15-08-2014 |
| McCartney | 8 | 18-08-2014 |
| Starr | 3 | 03-07-2014 |
| Harrison | 2 | 01-08-2014 |
| Page | 10 | 05-08-2014 |
| Plant | 1 | 13-08-2014 |
| Jones | 11 | 05-08-2014 |
| Bonham | 7 | 02-08-2014 |
| Townshend | 1 | 08-08-2014 |

**1- Obtenir le nom des clients ayant séjourné dans le bâtiment Jasmin.**
**2 Obtenir le nom des clients ayant séjourné dans un bâtiment 3 étoiles.**
**3 Obtenir le nom des clients ayant séjourné dans une chambre ayant au moins 2 fenêtres.**

# IV- Sous requête :

**Dans le langage SQL une sous-requête (aussi appelé “requête imbriquée” ou “requête en cascade”) consiste à exécuter une requête à l’intérieur d’une autre requête.**

***Quel sont les noms des clients qui ont passé une nuit dans le batiment contenant le lit numéro 1 ?***

```
SELECT Client FROM nuits,Lit WHERE lit=idlit AND batlit = (SELECT batlit FROM lit WHERE idlit = 1)
```

# V- Division cartésienne :

La division cartésienne est utilisée pour répondre à des requêtes du type: "Quels sont les références des produits achetés par tous les clients ?"

R1

| A | B |
|---|---|
| a1 | x1 |
| a2 | x2 |
| a3 | x2 |
| a1 | x3 |
| a2 | x1 |

R2

| A |
|---|
| a1 |
| a2 |

| X |
|---|
| x1 |

**Division: R1÷R2**

| ENSEIGNEMENT | |
|---|---|
| **enseignant** | **nom** |
| Germain | Dubois |
| Fidus | Pascal |
| Robert | Dubois |
| Germain | Pascal |
| Fidus | Dubois |
| Germain | Durand |
| Robert | Durand |

| ETUDIANT |
|---|
| **nom** |
| Dubois |
| Pascal |

| ENSEIGNEMENT ÷ ETUDIANT |
|---|
| **enseignant** |
| Germain |
| Fidus |