SQL

Ch 3 – SELECT Jointures

I. Introduction

A. Exemple introductif

1. Approche de l'éclatement des tables

Un musée souhaite gérer les informations relatives aux œuvres des artistes qu'il expose. Une première table de données nous donne ceci :

id_oeuvre	titre	annee	nom	prenom	anNais	paysOrigine
1	David	1501	Buonarroti	Michelangelo	1475	Italie
2	Le penseur	1880	Rodin	Auguste	1840	France
3	Les bourgeois de Calais	1886	Rodin	Auguste	1840	France
4	Mademoiselle Pogany III	1933	Brancusi	Constantin	1874	Roumanie

On peut observer que les données relatives à un artiste sont répétées pour chacune de ses œuvres (*chacun des artistes a produit des centaines d'œuvres…)*
La **redondance** (*répétition, souvent inutile, de données*) doit être éliminée[1] dans une base de données : les méthodes de conception des bases de données et des règles de

[1] On doit absolument tendre à l'éliminer, mais il arrive que certaines situations obligent à dupliquer des informations

construction des relations offrent des outils pour les éviter (matrice des dépendances fonctionnelles pour la construction d'un MCD[2], formes normales pour la construction d'un MLD[3] relationnel).

2. Eclatement des tables

Les données relatives aux artistes sont extraites de la table **t_oeuvre** pour former une table indépendante **t_artiste** ; une colonne a été ajoutée à la table **t_artiste**, **id_artiste**, afin que chaque artiste ait un numéro permettant de l'identifier de manière sûre (si 2 artistes portaient le même nom – homonymes – ils auraient des numéros différents).

table « **t_oeuvre** »

id_oeuvre	titre	annee
1	David	1501
2	Le penseur	1880
3	Les bourgeois de Calais	1886
4	Mademoiselle Pogany III	1933

table « **t_artiste** »

Id_artiste	nom	prenom	anNais	paysOrigine
1	Buonarroti	Michelangelo	1475	Italie
2	Rodin	Auguste	1840	France
3	Brancusi	Constantin	1874	Roumanie

3. Du « bout de ficelle » à la clef étrangère

Il va nous falloir créer un lien entre la table **t_oeuvre** et la table **t_artiste** (*afin que chaque œuvre puisse faire référence à l'artiste qui l'a produite*).
Dans un bloc de fiches « papier », on aurait pu relier les fiches « œuvres » et « artistes » avec des bouts de ficelles !!! On prenant une fiche « œuvre », le fil nous relie directement à la fiche « artiste » correspondante.

La solution mise en œuvre dans les bases de données consiste à répéter simplement le numéro de l'artiste dans la table des œuvres : ainsi, à partir d'une œuvre on pourra directement accéder aux données concernant l'artiste.

table « **t_oeuvre** »

id_oeuvre	titre	annee	Id_artiste
1	David	1501	1
2	Le penseur	1880	2
3	Les bourgeois de Calais	1886	2
4	Mademoiselle Pogany III	1933	3

=

table « **t_artiste** »

Id_artiste	nom	prenom	anNais	paysOrigine
1	Buonarroti	Michelangelo	1475	Italie
2	Rodin	Auguste	1840	France
3	Brancusi	Constantin	1874	Roumanie

[2] Modèle Conceptuel de Données : premier niveau d'abstraction dans la représentation des données d'un SI (méthode MERISE)
[3] Modèle Logique de Données : représentation des données tenant compte d'une forme d'organisation future des données, ici le modèle relationnel (méthode MERISE)

Cette colonne se nomme clef étrangère : c'est une colonne d'une table (ici **id_artiste** dans la table **t_oeuvre**) qu'on va retrouver comme clef primaire (identifiant) dans une autre table (ici **id_artiste** dans **t_artiste**) et qui permettra, à partir d'un œuvre bien précise d'accéder à l'artiste qui l'a produite.

B. De la nécessité de re-joindre les tables

La nécessité de répartir les données d'un SI (éviter la redondance) nous oblige à définir des opérations qui permettront à nouveau de rassembler, de manière cohérente, les éléments séparés. C'est l'objet des opérations de jointures de l'algèbre relationnelle.

II. La jointure interne d'égalité, liens naturels entre tables

A. Introduction

La **JOINTURE INTERNE** est une opération permettant de **RASSEMBLER** (joindre), **EN UNE SEULE TABLE**, les données qui ont été éclatées en plusieurs tables.
La JOINTURE INTERNE D'EGALITE est une **MISE EN CORRESPONDANCE DE TOUTES LES LIGNES DE 2 TABLES POUR LESQUELLES LA VALEUR D'UNE COLONNE EST EGALE.**

B. Syntaxe SQL2 vs SQL89

Deux formes de syntaxe cohabitent pour exprimer la jointure en SQL :

- La forme « ancienne », SQL89, qui décrit la jointure comme « produit cartésien » (dans la clause FROM) puis « sélection » (dans la clause WHERE)
- La forme « nouvelle », SQL92/SQL2 qui décrit la jointure comme opération à part entière (dans la clause FROM seulement) : c'est cette forme qui sera privilégiée car définissant de manière claire et centralisée le « chemin d'accès aux données à travers les différentes tables ».

Syntaxe générale :

SQL92/99

```
SELECT . . .
  FROM nom_de_table1
    INNER JOIN nom_de_table2
       ON nom_de_table1.colonne1 = nom_de_table2.colonne2
  . . .
;
```

ou bien si le nom de la colonne est la même dans les tables jointes :

```
SELECT . . .
  FROM nom_de_table1
    INNER JOIN nom_de_table2
       USING (colonne, ...)
  . . .
;
```

SQL89

```
SELECT . . .
```

```
  FROM nom_de_table1, nom_de_table2
  WHERE nom_de_table1.colonne1 = nom_de_table2.colonne2
  . . .
;
```

La jointure permet donc de constituer une table temporaire qui va ensuite être utilisée par les clauses suivantes de l'ordre SELECT.

Remarque importante :
Les noms des colonnes doivent être préfixés par un nom de table :

- **soit pour la clarté de la requête SQL** : cela permet d'exprimer clairement l'origine des colonnes utilisées
- **soit pour lever une ambiguïté sur un nom de colonne** : en effet **quand 2 noms de colonnes sont identiques dans 2 tables jointes, il est indispensable de les préfixer par le nom des tables d'origine** (sinon le SGBD ne sait pas laquelle des 2 utiliser).

C. Exemples

table « **t_oeuvre** »

id_oeuvre	titre	annee	Id_artiste
1	David	1501	1
2	Le penseur	1880	2
3	Les bourgeois de Calais	1886	2
4	Mademoiselle Pogany III	1933	3

=

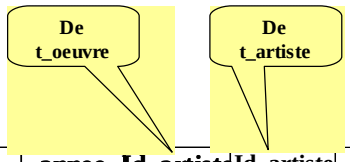
table « **t_artiste** »

Id_artiste	nom	prenom	anNais	paysOrigine
1	Buonarroti	Michelangelo	1475	Italie
2	Rodin	Auguste	1840	France
3	Brancusi	Constantin	1874	Roumanie

La colonne '**id_artiste**' de la table '**t_oeuvre**' est une clef étrangère qui fait référence à la colonne '**id_artiste**' de la table '**t_artiste**'.

Exemple 1 : Afficher les informations relatives aux œuvres et à l'artiste qui les a produites :

```
SELECT *
  FROM t_oeuvre
    INNER JOIN t_artiste
        ON t_oeuvre.id_artiste = t_artiste.id_artiste;
```



id_oeuvre	titre	annee	Id_artiste	Id_artiste	nom	prenom	anNais	paysOrigine
1	David	1501	1	1	Buonarroti	Michelangelo	1475	Italie
2	Le penseur	1880	2	2	Rodin	Auguste	1840	France
3	Les bourgeois de Calais	1886	2	2	Rodin	Auguste	1840	France
4	Mademoiselle Pogany III	1933	3	3	Brancusi	Constantin	1874	Roumanie

Exemple 2 : Afficher pour chaque œuvre, son titre, le nom et prénom de l'artiste qui l'a produite :

```
SELECT titre, nom, prenom
  FROM t_oeuvre
      INNER JOIN t_artiste
        ON t_oeuvre.id_artiste = t_artiste.id_artiste;
```

titre	nom	prenom
David	Buonarroti	Michelangelo
Le penseur	Rodin	Auguste
Les bourgeois de Calais	Rodin	Auguste
Mademoiselle Pogany III	Brancusi	Constantin

Exemple 3 : Afficher pour chaque œuvre, son titre, le numero, nom et prénom de l'artiste qui l'a produite :

```
SELECT titre, t_artiste.id_artiste, nom, prenom
  FROM t_oeuvre
    INNER JOIN t_artiste
        ON t_oeuvre.id_artiste = t_artiste.id_artiste;
```

titre	Id_artiste	nom	prenom
David	1	Buonarroti	Michelangelo
Le penseur	2	Rodin	Auguste
Les bourgeois de Calais	2	Rodin	Auguste
Mademoiselle Pogany III	3	Brancusi	Constantin

Remarque :
Dans la clause SELECT, la colonne '**id_artiste**' doit être préfixée afin de lever l'ambiguité (présence de cette colonne dans 2 tables)

D. « Surnoms » (ou alias) de tables

La norme SQL permet de simplifier l'écriture des jointures en associant à chaque table un autre nom (plus court), appelé **alias de table**.

Les exemples précédents peuvent ainsi être améliorés.

Ex. : Afficher les informations relatives aux œuvres et à l'artiste qui les a produites :

```
SELECT *
  FROM t_oeuvre O
    INNER JOIN t_artiste A
        ON O.id_artiste = A.id_artiste;
```

*Dans la clause FROM, on définit le surnom **O** pour la table **t_oeuvre** et le surnom **A** pour la table **t_artiste**. Ce surnom doit être ensuite utilisé partout où on aurait le nom des tables*.

Ex. : Afficher le titre de l'œuvre et les nom et prénom de l'artiste qui les a produites :

```
SELECT titre, nom, prenom
  FROM t_oeuvre O
    INNER JOIN t_artiste A
        ON O.id_artiste = A.id_artiste;
```

Ex. : Afficher le titre de l'œuvre et les numéro, nom et prénom de l'artiste qui les a produites :

```
SELECT titre, A.id_artiste, nom, prenom
  FROM t_oeuvre
    INNER JOIN t_artiste
        ON O.id_artiste = A.id_artiste;
```

III.Jointure croisée ou produit cartésien

LE PRODUIT CARTESIEN (ou jointure croisée) ASSOCIE CHACUNE DES LIGNES D'UNE TABLE AVEC TOUTES LES LIGNES D'UNE AUTRE TABLE
IL CONSERVE TOUTES LES LIGNES DE TOUTES LES TABLES JOINTES.

ATTENTION : IL RESULTE BIEN SOUVENT D'UN OUBLI DE SPECIFICATION DES CRITERES DE JOINTURE (particulièrement avec la notation SQL89).

Ex. : les nom et prénom des artistes associés à chaque membre :

SQL92/99

```
SELECT A.nom,A.prenom,B.nom, B.prenom
  FROM t_membre A
    CROSS JOIN t_artiste B;
```

SQL89

```
SELECT A.nom,A.prenom,B.nom, B.prenom
  FROM t_membre A , t_artiste B;
```

IV. Jointure interne

LA JOINTURE INTERNE (THETA-JOINTURE) RETOURNE LES LIGNES DES TABLES JOINTES QUI REPONDENT AUX CRITERES DE JOINTURE (COMPARAISON DE VALEURS DE COLONNES ISSUES DES 2 TABLES)

LES LIGNES QUI NE PEUVENT PAS ETRE JOINTES SONT ELIMINEES.

Les liens entre tables sont, le plus souvent réalisés entre la clef étrangère d'une table et la clef primaire d'un autre table. Cependant, les jointures peuvent être réalisées en utilisant d'autres colonnes[4]

1. JOINTURE NATURELLE

LA JOINTURE NATURELLE EXPRIME « NATURELLEMENT » LES LIENS ENTRE TABLES EN UTILISANT (sans qu'il y ait à la préciser) LES NOMS DE COLONNES IDENTIQUES DANS CHACUNE DES TABLES POUR CONSTRUIRE LA JOINTURE.

ATTENTION :

- ON OBTIENT UN PRODUIT CARTESIEN SI AUCUNE CORRESPONDANCE DE NOM N'EST TROUVEE
- ON OBTIENT UNE JOINTURE INCOHERENTE SI LA CORRESPONDANCE EST TROUVEE SUR DES COLONNES QUI N'AURAIENT PAS LE MÊME SENS

Ex. : Les membres et les activités auxquelles ils sont inscrits :

SQL92/99 (pour certains SGBD)

```
SELECT nom, prenom, intitule
FROM t_membre
    NATURAL JOIN t_inscrire
    NATURAL JOIN t_activite ;
```

SQL89

N'existe pas.

2. JOINTURES INTERNE, EQUI-JOINTURES

DANS L'EQUI-JOINTURE INTERNE, LE CRITERE DE JOINTURE EST EXPRIME SELON UNE EGALITE DE VALEURS DE COLONNES.

Ex. : Les membres et les activités auxquelles ils sont inscrits :

SQL92

```
SELECT nom, prenom, intitule
  FROM t_membre A
    INNER JOIN t_inscrire B
        ON A.nummembre = B.nummembre
    INNER JOIN t_activite C
```

[4] les performances sont dégradées si les colonnes de jointures ne sont pas associées à des index (clefs primaires, clefs étrangères, index supplémentaires)

```
        ON B.numactivite = C.numactivite;
```

Ou bien si les colonnes portent le même nom :

```
SELECT nom, prenom, intitule
  FROM t_membre A
    INNER JOIN t_inscrire B
        USING (nummembre)
    INNER JOIN t_activite C
        USING (numactivite);
```

SQL89

```
SELECT nom, prenom, intitule
  FROM t_membre A, t_inscrire B, t_activite C
  WHERE A.nummembre = B.nummembre
    AND B.numactivite = C.numactivite;
```

3. AUTO JOINTURE

L'AUTO JOINTURE EST UN CAS PARTICULIER DE LA JOINTURE OU LES TABLES JOINTES SONT LES MEMES (ON EFFECTUE UNE JOINTURE D'UNE TABLE AVEC UNE COPIE « ALIASEE » DE CETTE MEME TABLE)

Ex. : Les membres et leur parrain :

SQL92

```
SELECT A.nom, A.prenom, B.nom AS NomParrain, B.prenom AS
PrenomParrain
  FROM t_membre A
    INNER JOIN t_membre B
        ON A.mem_nummembre = B. nummembre;
```

SQL89

```
SELECT A.nom, A.prenom, B.nom AS NomParrain, B.prenom AS
PrenomParrain
  FROM t_membre A, t_membre B
  WHERE A.mem_nummembre = B.nummembre;
```

4. SEMI JOINTURE

LA SEMI-JOINTURE EST UN CAS PARTICULIER DE LA JOINTURE INTERNE OU LES COLONNES RECUPEREES SONT CELLES D'UNE SEULE DES TABLES JOINTES

Ex. : Les membres qui ont un parrain :

SQL92

```
SELECT A.*
  FROM t_membre A
    INNER JOIN t_membre B
        ON A.mem_nummembre = B.nummembre;
```

SQL89

```
SELECT A.*
  FROM t_membre A, t_membre B
  WHERE A.mem_nummembre = B.nummembre;
```

5. JOINTURES INTERNE, NON EQUI-JOINTURES

DANS LA NON EQUI-JOINTURE INTERNE, LE CRITERE DE JOINTURE N'EST PAS EXPRIME SELON UN CRITERE D'EGALITE, MAIS GRACE A UN AUTRE OPERATEUR DE COMPARAISON : <, <=, >, >=, <>.

Ex. : Les membres avec les membres qui sont plus jeunes :

SQL92

```
SELECT A.nom, A.prenom, A.datenaissance, B.nom, B.prenom,
B.datenaissance
  FROM t_membre A
    INNER JOIN t_membre B
        ON A.datenaissance < B.datenaissance;
```

SQL89

```
SELECT A.nom, A.prenom, A.datenaissance, B.nom, B.prenom,
B.datenaissance
  FROM t_membre A, t_membre B
  WHERE A.datenaissance < B.datenaissance;
```

V. Jointure externe

LA **JOINTURE EXTERNE** RETOURNE LES LIGNES DES TABLES JOINTES QUI REPONDENT AUX CRITERES DE JOINTURES ET EGALEMENT, DANS CERTAINS CAS, CELLES QUI NE REPONDENT PAS AUX CRITERES.

Les valeurs des colonnes provenant des lignes manquantes sont positionnées à NULL.

LES LIGNES QUI NE PEUVENT PAS ETRE JOINTES SONT CONSERVEES, DANS CERTAINS CAS.

table gauche OUTER JOIN table droite

ON critère de jointure

1. JOINTURES EXTERNE GAUCHE

LES LIGNES DE LA TABLE DE GAUCHE QUI NE PEUVENT PAS ETRE JOINTES SONT CONSERVEES.

Ex. : Les membres avec, éventuellement, les intitulés des activités auxquels ils se sont inscrits :

SQL92

```
SELECT A.nom, A.prenom, C.intitule
FROM t_membre A
    LEFT OUTER JOIN t_inscrire B
        ON A.nummembre = B.nummembre
    LEFT OUTER JOIN t_activite C
        ON B.numactivite = C.numactivite;
```

SQL89

```
.opérateur *= sous SQL Server ( * du côté où on conserve)
.opérateur (+)= Sous Oracle ( (+) du côté où on conserve)
```

2. JOINTURES EXTERNE DROITE

LES LIGNES DE LA TABLE DE DROITE QUI NE PEUVENT PAS ETRE JOINTES SONT CONSERVEES.

Ex. : Les activités avec, éventuellement, les nom et prénom des membres qui s'y sont inscrits :

SQL92

```
SELECT A.nom, A.prenom, C.intitule
FROM t_membre A
    RIGHT OUTER JOIN t_inscrire B
        ON A.nummembre = B.nummembre
    RIGHT OUTER JOIN t_activite C
        ON B.numactivite = C.numactivite;
```

SQL89

```
.opérateur =* sous SQL Server ( * du côté où on conserve)
.opérateur =(+) Sous Oracle ( (+) du côté où on conserve)
```

3. JOINTURES EXTERNE TOTALE / ENTIERE

LES LIGNES DES TABLES QUI NE PEUVENT PAS ETRE JOINTES SONT CONSERVEES, LES VALEURS DES COLONNES DES LIGNES NON JOINTES SONT POSITIONNEES A LA VALEUR « NULL ».

Ex. : Les membres avec 'éventuellement, les activités auxquelles ils se sont inscrits et les activités même si aucun membre ne s'y sont inscrits :

SQL92

```
SELECT A.nom, A.prenom, C.intitule
FROM t_membre A
    FULL OUTER JOIN t_inscrire B
        ON A.nummembre = B.nummembre
```

```
    FULL OUTER JOIN t_activite C
        ON B.numactivite = C.numactivite;
```

SQL89

```
...
```

VI. Organiser les niveaux de jointures

Dans le cas où plus de 2 tables doivent être jointes, il est parfois nécessaire de préciser l'ordre dans lequel ces jointures doivent être réalisées,

- D'abord **par obligation** : **quand plusieurs formes de jointures** sont réalisées
- puis par clarté, pour la lisibilité de l'expression globale de la jointure.

VII. Exercices

Soient les définitions des tables t_personne et t_chien:

- t_personne (nomPersonne, agePersonne)
- t_chien (nomChien, #nomPersonne, poids, ageChien)

Les colonnes soulignées représentent les identifiants (clefs primaires), les colonnes préfixées par #, les clefs étrangères.

Soient les contenus des 2 tables :

table « **t_personne** »

nomPersonne	**agePersonne**
Tim	12
John	23
Bill	30
Paul	50

table « **t_chien** »

nomChien	**#nomPersonne**	**poids**	**ageChien**
Pollux	Paul	12	5
Nestor	Tim	20	10
Rex	Bill	18	15
Boby	Tim	10	8

Pour chacune des questions, proposez une requête SQL et donnez le nombre de lignes que comportera le résultat final.

Question 1 : obtenir la liste des chiens avec leur maître (nom du chien et nom de la personne)

```
SELECT nomChien, nomPersonne
  FROM t_chien ;
```

(La jointure n'est pas nécessaire ici, toutes les colonnes peuvent être trouvées dans la table t_chien)

Question 2 : obtenir la liste des personnes avec leur(s) chien(s) (nom de la personne et nom du chien)

```
SELECT t_personne.nomPersonne, nomChien
  FROM t_personne INNER JOIN t_chien
       ON t_personne.numPersonne = t_chien.nomPersonne ;
```

```
SELECT A.nomPersonne, nomChien
  FROM t_personne A INNER JOIN t_chien B
       ON A.nomPersonne = B.nomPersonne ;
```

On a du préfixer le nom de la personne dans la clause SELECT : on le trouve en effet dans les 2 tables, il est indispensable de lever l'ambiguïté.

Question 3 : obtenir la liste des personnes avec leur(s) chien(s) si le chien a plus de 10 ans (nom et age de la personne et nom et age du chien)

```
SELECT A.nomPersonne,agePersonne, nomChien, ageChien
  FROM t_personne A INNER JOIN t_chien B
       ON A.nomPersonne = B.nomPersonne
```

```
  WHERE ageChien > 10;
```

Question 4 : obtenir la liste des personnes avec leur(s) chien(s) si la personne a plus de 25 ans (nom et age de la personne et nom et age du chien)

```
SELECT A.nomPersonne,agePersonne, nomChien, ageChien
  FROM t_personne A INNER JOIN t_chien B
       ON A.nomPersonne = B.nomPersonne
  WHERE agePersonne > 25;
```

Question 5 : obtenir la liste des personnes avec le poids total de leur(s) chien(s) (par personne) (nom de la personne et total des poids)

```
SELECT A.nomPersonne,SUM(poids) AS poidsTotal
  FROM t_personne A INNER JOIN t_chien B
       ON A.nomPersonne = B.nomPersonne
  GROUP BY A.nomPersonne ;
```