Chapitre 3

Bases de données : Opérations entre tables

Dans ce chapitre, on reprend les tables présentées le chapitre 2, auxquelles on a rajouté une table nommé identité2 que voici :

rowid	Nom	Prenom
1	LE GUILLOUS	Cyrielle
2	MOLLE	Robin
3	BLANCHET	Lorry
4	LIN	Yiman

I Opérations ensemblistes entre tables

I.1 Union

L'union est une opération ensembliste que l'on peut réaliser sur des relations/tables à condition qu'elles aient la même structure, c'est-à-dire le même nombre d'attribut. Par exemple, si je veux trouver tous les noms des élèves apparaissant dans l'une des deux tables :

SQL

```
SELECT Nom, Prenom
FROM identité
UNION
Select Nom, Prenom
From identité2;
```

ce qui donne :

Nom	Prenom
ALBIN	Felix
BLANCHET	Lorry
CHERRAJ	Assia
DECOUPY	Victor
GOMES	Quentin
GOMIS	Jean-Marie
HAMELIN	Axel
HAMELIN	Dimitri
LE GUILLOUS	Cyrielle
LEGER	Maia
LIN	Yiman
MOLLE	Robin
POUBANNE	Lisa

I.2 Intersection

On peut également réaliser des intersections entre relations/tables ayant la même structure. Par exemple, si on cherche le (Nom,Prenom) des élèves apparaissant dans les deux tables :

SQL

```
SELECT Nom, Prenom
FROM identité
INTERSECT
Select Nom, Prenom
From identité2;
```

ce qui donne :

Nom	Prenom
LE GUILLOUS	Cyrielle

I.3 Produit cartésien

Il a le même principe qu'en algèbre, c'est-à-dire que si vous avez un ensemble A de tuples $(a_1, ..., a_n)$ et un ensemble B de tuples $(b_1, ..., b_m)$, le produit cartésien de ces deux ensemble donne l'ensemble des tuples $(a_1, ..., a_n, b_1, ..., b_m)$ avec $(a_1, ..., a_n)$ dans A et $(b_1, ..., b_m)$ dans B. Voici un exemple :

SQL

```
SELECT *, devoir_conf.rowid as numéro_devoir
FROM Devoir_conf
JOIN Devoir_type;
```

ce qui donne :

	type_num	coefficient	type	numéro_devoir
1	1	2	DS	1
2	1	2	DM	1
3	1	2	Interro	1
4	2	1	DS	2
5	2	1	DM	2
6	2	1	Interro	2
7	3	1	DS	3
8	3	1	DM	3
9	3	1	Interro	3
10	1	1	DS	4
11	1	1	DM	4
12	1	1	Interro	4
13	2	2	DS	5
14	2	2	DM	5
15	2	2	Interro	5
16	1	3	DS	6
17	1	3	DM	6
18	1	3	Interro	6
19	2	1	DS	7
20	2	1	DM	7
21	2	1	Interro	7
22	3	2	DS	8
23	3	2	DM	8

Remarque 3.1. *Le produit cartésien n'a pas vraiment d'intérêt ainsi. Dans le cas précédent, on remarque que cela aurait un intérêt de sélectionner que les triplets ou l'attribut type représente bien le type de devoir du devoir concerné. Autrement dit, il faut sélectionner les triplets où le numéro de la ligne de Devoir_type correspond au numéro indiqué dans l'attribut type_num. Le devoir numéro 1 a pour valeur dans type_num : 1, c'est donc un DS, il faut donc garder le triplet (1,2,DS,1) et supprimer tous les triplets de la forme (a,b,c,1), c'est le principe de la jointure que nous verrons un peu plus loin.*

I.4 Division cartésienne

La division cartésienne est l'opération inverse du produit cartésien : diviser une relation A par une relation B consiste à trouver une relation C contenant tous les n-uplets t tels que pour tout n-uplet t' de B, le n-uplet (t, t') soit dans A. Il n'y a pas d'opérateur permettant de faire cette opération directement. On la crée via les autres opérations selon les cas.

I.5 Jointure

L'opération de jointure est une opération permettant de joindre deux tables en une seule en suivant un critère donné. Formellement, c'est la composition d'un produit cartésien des deux tables qu'on veut joindre et de la sélection des n-uplets de la table issue du produit qui vérifient un critère.

Par exemple, si on reprend le produit cartésien précédent, en conservant les triplets qui on un sens :

SQL

```sql
SELECT *, devoir_conf.rowid as numéro_devoir
FROM Devoir_conf
JOIN Devoir_type ON devoir_type.rowid=devoir_conf.type_num;
```

	type_num	coefficient	type	numéro_devoir
1	1	2	DS	1
2	2	1	DM	2
3	3	1	Interro	3
4	1	1	DS	4
5	2	2	DM	5
6	1	3	DS	6
7	2	1	DM	7
8	3	2	Interro	8
9	1	1	DS	9
10	2	2	DM	10
11	1	3	DS	11
12	2	1	DM	12
13	1	1	DS	13
14	2	1	DM	14
15	1	2	DS	15
16	1	3	DS	16

Ce qui permet d'indiquer directement le type de chaque devoir (l'attribut type_num correspond bien à l'attribut type).

Deuxième exemple, si on veut retrouver le nom et le prénom des élèves avec les notes :

SQL

```
SELECT Nom, Prenom, devoir, part_1, part_2
FROM notes
JOIN identité ON identité.rowid=notes.élèves;
```

ce qui donne :

Nom	Prenom	Devoir	part_1	part_2
ALBIN	Felix	1	1	4
ALBIN	Felix	2	7	3
ALBIN	Felix	3	5	4
ALBIN	Felix	4	4	9
ALBIN	Felix	5	9	3
ALBIN	Felix	6	3	8
ALBIN	Felix	7	6	6
ALBIN	Felix	8	8	5
ALBIN	Felix	9	9	4
ALBIN	Felix	10	9	7
ALBIN	Felix	11	3	3
ALBIN	Felix	12	4	9
ALBIN	Felix	13	7	9
ALBIN	Felix	14	6	8
CHERRAJ	Assia	1	10	9
CHERRAJ	Assia	2	9	3
CHERRAJ	Assia	3	9	6
CHERRAJ	Assia	4	3	9
CHERRAJ	Assia	5	8	1
CHERRAJ	Assia	6	6	1
CHERRAJ	Assia	7	7	5
CHERRAJ	Assia	8	6	10
CHERRAJ	Assia	9	2	7
CHERRAJ	Assia	10	1	8

II Exercices

Exercice 1 Joindre les attributs Nom et Prenom à la table : notes. Puis, à l'aide de l'opérateur + permettant de sommer deux colonnes, calculer les notes de chaque élève à chaque devoir.

Exercice 2 Trouver, pour chaque élève, la plus mauvaise note qu'ils ont obtenue à un devoir. La table doit renvoyer le nom, le prénom, le numéro du devoir et la note obtenue.

Exercice 3 Trouver, pour chaque élève, la plus mauvaise note qu'ils ont obtenue pour chaque type de devoir. La table doit renvoyer le nom, le prénom, le numéro du devoir, le type de devoir et la note obtenue.

Exercice 4 Calculer la moyenne obtenue par chaque étudiant selon le type de devoir. La table doit renvoyer le nom, le prénom, le type du devoir, la moyenne obtenue sur ce type.

Exercice 4 Calculer la moyenne obtenue par chaque étudiant. La table doit renvoyer le nom, le prénom et la moyenne obtenue.

Exercice 5 Le professeur, légèrement compatissant, décide de ne pas compter dans la moyenne de l'élève sa plus mauvaise note dans chaque type de devoir. Recommencer l'exercice précédent en tenant compte de cette information. Pour cela, on pourra utiliser l'opérateur IN qui renvoie TRUE si un n-uplet est dans une table, FALSE sinon.