

Interrogation de bases de données SQL

Table des matières

Objectifs	3
I - Cours	4
1. Questions simples avec le Langage de Manipulation de Données (SELECT)	4
1.1. Question (SELECT)	4
1.2. Opérateurs de comparaisons et opérateurs logiques.....	5
1.3. Renommage de colonnes et de tables avec les alias.....	6
1.4. Dédoublonnage (SELECT DISTINCT)	6
1.5. Tri (ORDER BY)	7
1.6. Projection de constante	7
1.7. Commentaires en SQL	8
1.8. Exercice : Représentation de représentants.....	8
2. Opérations d'algèbre relationnelle en SQL	10
2.1. Expression d'une restriction	10
2.2. Expression d'une projection	10
2.3. Expression du produit cartésien	11
2.4. Expression d'une jointure	12
2.5. Exercice : Tableau final	13
2.6. Expression d'une jointure externe	14
2.7. Exercice : Photos à gauche	16
2.8. Opérateurs ensemblistes	16
II - Exercices	18
1. Exercice : Location d'appartements.....	18
2. Exercice : Employés et salaires	18
Abréviations	20
Index	21

Objectifs



- Maîtriser les bases du SQL pour écrire des questions exigeant des jointures, projections, restriction
- Savoir préparer des résultats de requête. (renommage, tri, dédoublonnage...)



SQL* est un langage standardisé, implémenté par tous les SGBDR*, qui permet, indépendamment de la plate-forme technologique et de façon déclarative, de définir le modèle de données, de le contrôler et enfin de le manipuler.

1. Questions simples avec le Langage de Manipulation de Données (SELECT)

1.1. Question (SELECT)

Question



Fondamental

La requête de **sélection** ou **question** est la base de la recherche de données en SQL.

Sélection



Définition

La sélection est la composition d'un **produit cartésien**, d'une **restriction** et d'une **projection** (ou encore la composition d'une jointure et d'une projection).



Syntaxe

```
1 SELECT liste d'attributs projetés
2 FROM liste de relations du produit cartésien
3 WHERE condition de la restriction
```

- La partie SELECT indique le sous-ensemble des attributs qui doivent apparaître dans la réponse (c'est le schéma de la relation résultat).
- La partie FROM décrit les relations qui sont utilisables dans la requête (c'est à dire l'ensemble des attributs que l'on peut utiliser).
- La partie WHERE exprime les conditions que doivent respecter les attributs d'un tuple pour pouvoir être dans la réponse. Une condition est un prédicat et par conséquent renvoie un booléen. Cette partie est optionnelle.



Exemple

```
1 SELECT Nom, Prenom
2 FROM Personne
3 WHERE Age>18
```

Cette requête sélectionne les attributs Nom et Prénom des tuples de la relation Personne, ayant un attribut Age supérieur à 18.

Notation préfixée



Afin de décrire un attribut d'une relation en particulier (dans le cas d'une requête portant sur plusieurs relations notamment), on utilise la notation `relation.attribut`.



```
1 SELECT Personne.Nom, Personne.Prenom, Vol.Depart
2 FROM Personne, Vol
3 WHERE Personne.Vol=Vol.Numero
```

SELECT *



Pour projeter l'ensemble des attributs d'une relation, on peut utiliser le caractère `*` à la place de la liste des attributs à projeter.



```
1 SELECT *
2 FROM Avion
```

Cette requête sélectionne tous les attributs de la relation Avion.

Notons que dans cet exemple, la relation résultat est exactement la relation Avion

1.2. Opérateurs de comparaisons et opérateurs logiques

Introduction

La clause `WHERE` d'une instruction de sélection est définie par une condition. Une telle condition s'exprime à l'aide d'opérateurs de comparaison et d'opérateurs logiques. Le résultat d'une expression de condition est toujours un booléen.

Condition



```
1 Condition Élémentaire ::= Propriété <Opérateur de comparaison> Constante
2 Condition ::= Condition <Opérateur logique> Condition | Condition Élémentaire
```

Les opérateurs de comparaison sont :

- `P = C`
- `P <> C`
- `P < C`
- `P > C`
- `P <= C`
- `P >= C`
- `P BETWEEN C1 AND C2`
- `P IN (C1, C2, ...)`
- `P LIKE 'chaîne'`
- `P IS NULL`

Les opérateurs logiques sont :

- OR
- AND
- NOT

Opérateur LIKE



L'opérateur `LIKE` 'chaîne' permet d'insérer des jokers dans l'opération de comparaison (alors que l'opérateur `=` teste une égalité stricte) :

- Le joker `%` désigne 0 ou plusieurs caractères quelconques
- Le joker `_` désigne 1 et 1 seul caractère

On préférera l'opérateur `=` à l'opérateur `LIKE` lorsque la comparaison n'utilise pas de joker.

1.3. Renommage de colonnes et de tables avec les alias

Alias de table



Il est possible de redéfinir le nom des relations au sein de la requête afin d'en simplifier la syntaxe.

```
1 SELECT t1.attribut1, t2.attribut2
2 FROM table1 t1, table2 t2
```



```
1 SELECT Parent.Prenom, Enfant.Prenom
2 FROM Parent, Enfant
3 WHERE Enfant.Nom=Parent.Nom
```

Cette requête sélectionne les prénoms des enfants et des parents ayant le même nom. On remarque la notation `Parent.Nom` et `Enfant.Nom` pour distinguer les attributs `Prenom` des relations `Parent` et `Enfant`.

On notera que cette sélection effectue une jointure sur les propriétés `Nom` des relations `Parent` et `Enfant`.

Alias d'attribut (AS)



Il est possible de redéfinir le nom des propriétés de la relation résultat.

```
1 SELECT attribut1 AS a1, attribut2 AS a2
2 FROM table
```

1.4. Dédoublonnage (SELECT DISTINCT)

SELECT DISTINCT



L'opérateur `SELECT` n'élimine pas les doublons (i.e. les tuples identiques dans la relation résultat) par défaut. Il faut pour cela utiliser l'opérateur `SELECT DISTINCT`.



```
1 SELECT DISTINCT Avion
2 FROM Vol
3 WHERE Date=31-12-2000
```

Cette requête sélectionne l'attribut Avion de la relation Vol, concernant uniquement les vols du 31 décembre 2000 et renvoie les tuples sans doublons.

1.5. Tri (ORDER BY)

Introduction

On veut souvent que le résultat d'une requête soit trié en fonction des valeurs des propriétés des tuples de ce résultat.

ORDER BY

§ Syntaxe

```
1 SELECT liste d'attributs projetés
2 FROM liste de relations
3 WHERE condition
4 ORDER BY liste ordonnée d'attributs
```

Les tuples sont triés d'abord par le premier attribut spécifié dans la clause ORDER BY, puis en cas de doublons par le second, etc.

Tri décroissant

Q Remarque

Pour effectuer un tri décroissant on fait suivre l'attribut du mot clé "DESC".

? Exemple

```
1 SELECT *
2 FROM Personne
3 ORDER BY Nom, Age DESC
```

1.6. Projection de constante

Projection de constante

§ Syntaxe

Il est possible de projeter directement des constantes (on utilisera généralement un alias d'attribut pour nommer la colonne).

```
1 SELECT constante AS nom
```

Cette requête renverra une table avec une seule ligne et une seule colonne à la valeur de *constante*.

? Exemple

```
1 SELECT '1' AS num
1 num
2 -----
3 1
```

? Exemple

```
1 SELECT 'Hello world' AS hw
1 hw
2 -----
3 Hello world
```

```
1 SELECT CURRENT_DATE AS now
```

```
1      now
2 -----
3 2016-10-21
```

1.7. Commentaires en SQL

L'introduction de commentaires au sein d'une requête SQL se fait :

- En précédant les commentaires de `--` pour les commentaires mono-ligne
- En encadrant les commentaires entre `/*` et `*/` pour les commentaires multi-lignes.

```
1 /*
2 * Creation of table Student
3 * Purpose : managing information about students in the system
4 */
5 CREATE TABLE person (
6 pknum VARCHAR(13) PRIMARY KEY, --Student national number
7 name VARCHAR(25)
8 );
9
10
```

1.8. Exercice : Représentation de représentants

[30 minutes]

Soit le schéma relationnel et le code SQL suivants :

```
1 REPRESENTANTS (#NR, NOMR, VILLE)
2 PRODUITS (#NP, NOMP, COUL, PDS)
3 CLIENTS (#NC, NOMC, VILLE)
4 VENTES (#NR=>REPRESENTANTS(NR), #NP=>PRODUITS(NP), #NC=>CLIENTS(NC), QT)

1 /* Les requêtes peuvent être testées dans un SGBDR, en créant une base de données
   avec le script SQL suivant */
2
3 /*
4 DROP TABLE VENTES ;
5 DROP TABLE CLIENTS ;
6 DROP TABLE PRODUITS ;
7 DROP TABLE REPRESENTANTS ;
8 */
9
10 CREATE TABLE REPRESENTANTS (
11   NR INTEGER PRIMARY KEY,
12   NOMR VARCHAR,
13   VILLE VARCHAR
14 );
15
16 CREATE TABLE PRODUITS (
17   NP INTEGER PRIMARY KEY,
18   NOMP VARCHAR,
19   COUL VARCHAR,
20   PDS INTEGER
21 );
22
23 CREATE TABLE CLIENTS (
```



```

24 NC INTEGER PRIMARY KEY,
25 NOMC VARCHAR,
26 VILLE VARCHAR
27 );
28
29 CREATE TABLE VENTES (
30 NR INTEGER REFERENCES REPRESENTANTS(NR),
31 NP INTEGER REFERENCES PRODUITS(NP),
32 NC INTEGER REFERENCES CLIENTS(NC),
33 QT INTEGER,
34 PRIMARY KEY (NR, NP, NC)
35 );
36
37 INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (1, 'Stephane', 'Lyon');
38 INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (2, 'Benjamin', 'Paris');
39 INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (3, 'Leonard', 'Lyon');
40 INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (4, 'Antoine', 'Brest');
41 INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (5, 'Bruno', 'Bayonne');
42
43 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (1, 'Aspirateur', 'Rouge', 3546);
44 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (2, 'Trottinette', 'Bleu', 1423);
45 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (3, 'Chaise', 'Blanc', 3827);
46 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (4, 'Tapis', 'Rouge', 1423);
47
48 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (1, 'Alice', 'Lyon');
49 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (2, 'Bruno', 'Lyon');
50 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (3, 'Charles', 'Compiègne');
51 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (4, 'Denis', 'Montpellier');
52 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (5, 'Emile', 'Strasbourg');
53
54 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 1, 1, 1);
55 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 1, 2, 1);
56 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (2, 2, 3, 1);
57 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (4, 3, 3, 200);
58 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 2, 190);
59 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 3, 2, 300);
60 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 2, 120);
61 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 4, 120);
62 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 4, 2);
63 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 1, 3);
64 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 1, 5);
65 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 3, 1);
66 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 5, 1);

```

Écrire en SQL les requêtes permettant d'obtenir les informations ci-après.

Question 1

Tous les détails de tous les clients.

Question 2

Les numéros et les noms des produits de couleur rouge et de poids supérieur à 2000.

Question 3

Les représentants ayant vendu au moins un produit.

Question 4

Les noms des clients de Lyon ayant acheté un produit pour une quantité supérieure à 180.

Question 5

Les noms des représentants et des clients à qui ces représentants ont vendu un produit de couleur rouge pour une quantité supérieure à 100.

2. Opérations d'algèbre relationnelle en SQL

2.1. Expression d'une restriction

§ Syntaxe

```
1 SELECT *
2 FROM R
3 WHERE <condition>
```

? Exemple

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R	A	B	C
	1	Alpha	10
	2	Bravo	10

Restriction(R1, C<20)

```
SELECT *
FROM R1
WHERE C<20
```

Exemple de restriction (SQL et Algèbre)

2.2. Expression d'une projection

§ Syntaxe

```
1 SELECT P1, P2, Pi
2 FROM R
```

? Exemple

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R	A	C
	1	10
	2	10
	3	20
	4	

Projection (R1,A,C)

```
SELECT A, C
FROM R1
```

Exemple de projection (SQL et Algèbre)

2.3. Expression du produit cartésien

§ Syntaxe

```
1 SELECT *
2 FROM R1, R2, Ri
```

? Exemple

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

A	B	C	X	Y
1	Alpha	10	10	Echo
1	Alpha	10	20	Fox
1	Alpha	10	30	Golf
2	Bravo	10	10	Echo
2	Bravo	10	20	Fox
2	Bravo	10	30	Golf
3	Charlie	20	10	Echo
3	Charlie	20	20	Fox
3	Charlie	20	30	Golf
4	Delta		10	Echo
4	Delta		20	Fox
4	Delta		30	Golf

Produit (R1,R2)

```
SELECT *
FROM R1,R2
```

Exemple de produit (SQL et Algèbre)

2.4. Expression d'une jointure

Jointure par la clause WHERE



En tant que composition d'un produit cartésien et d'une restriction la jointure s'écrit :

```
1 SELECT *
2 FROM R1, R2, Ri
3 WHERE <condition>
```

Avec `Condition` permettant de joindre des attributs des `Ri`

Jointure par la clause ON



On peut également utiliser la syntaxe dédiée suivante :

```
1 SELECT *
2 FROM R1 INNER JOIN R2
3 ON <condition>
```

Et pour plusieurs relations :

```
1 SELECT *
2 FROM (R1 INNER JOIN R2 ON <condition>) INNER JOIN Ri ON <condition>
3
```



R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox

Jointure (R1 ,R2 ,R1 .C=R2 .X)

```
SELECT *
FROM R1 INNER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure (SQL et Algèbre)

Une jointure naturelle



```
1 SELECT *
2 FROM R1, R2
3 WHERE R2.NUM = R1.NUM
```

Auto-jointure



Pour réaliser une auto-jointure, c'est à dire la jointure d'une relation avec elle-même, on doit utiliser le renommage des relations. Pour renommer une relation, on note dans la clause FROM le nom de renommage après le nom de la relation : "FROM NOM_ORIGINAL NOUVEAU_NOM".

Auto-jointure



```

1 SELECT E1.Nom
2 FROM Employe E1, Employe E2
3 WHERE E1.Nom= E2.Nom

```

2.5. Exercice : Tableau final

Que renvoie la dernière instruction SQL de la séquence ci-dessous ?

```

1 CREATE TABLE R (a INTEGER, b INTEGER);
2 INSERT INTO R VALUES (1,1);
3 INSERT INTO R VALUES (1,2);
4 INSERT INTO R VALUES (3,3);
5 SELECT * FROM R R1, R R2 WHERE R1.a=R2.a;

```

☐

1	1
1	2
3	3

☐

1	1
1	1
1	2
1	2
3	3
3	3

☐

1	1	1	1
1	1	1	2
1	1	3	3
1	2	1	1
1	2	1	2
1	2	3	3
3	3	1	1
3	3	1	2
3	3	3	3

☐

1	1	1	1
1	1	1	2
1	2	1	1
1	2	1	2
3	3	3	3



1	1	1	2
1	1	3	3
1	2	1	1
1	2	3	3
3	3	1	1
3	3	1	2

2.6. Expression d'une jointure externe

Jointure externe, gauche ou droite



Pour exprimer une jointure externe on se base sur la syntaxe INNER JOIN en utilisant à la place OUTER JOIN, LEFT OUTER JOIN ou RIGHT OUTER JOIN.

Jointure externe gauche



```
1 SELECT Num
2 FROM Avion LEFT OUTER JOIN Vol
3 ON Avion.Num=Vol.Num
```

Cette requête permet de sélectionner tous les avions, y compris ceux non affectés à un vol.



Remarquons que "Avion LEFT OUTER JOIN Vol" est équivalent à "Vol RIGHT OUTER JOIN Avion" en terme de résultat.

Intuitivement, on préfère utiliser la jointure gauche pour sélectionner tous les tuple du côté N d'une relation 1:N, même si il ne sont pas référencés ; et la jointure droite pour sélectionner tous les tuples d'une relation 0:N, y compris ceux qui ne font pas de référence. Cette approche revient à toujours garder à gauche de l'expression "JOIN" la relation "principale", i.e. celle dont on veut tous les tuples, même s'ils ne référencent pas (ou ne sont pas référencés par) la relation "secondaire".



R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox

Jointure (R1, R2, R1.C=R2.X)

```
SELECT *
FROM R1 INNER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure (SQL et Algèbre)

? Exemple

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox
	4	Delta			

JointureExterneGauche (R1, R2, R1.C=R2.X)

```
SELECT *
FROM R1 LEFT OUTER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure externe gauche (SQL et Algèbre)

? Exemple

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox
				30	Golf

JointureExterneDroite (R1, R2, R1.C=R2.X)

```
SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure externe droite (SQL et Algèbre)

Trouver les enregistrements non joints



Méthode

La jointure externe sert en particulier pour trouver les enregistrements d'une table qui ne sont pas référencés par une clé étrangère. Il suffit de sélectionner, après la jointure externe, tous les enregistrements pour lesquels la clé de la relation référençante est nulle, on obtient alors ceux de la relation référencée qui ne sont pas référencés.

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
				30	Golf

```

Restriction(
    JointureExterneDroite(R1,R2,R1.C=R2.X),
    R1.A IS NULL)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1.C=R2.X
WHERE R1.A IS NULL

```

Exemple de sélection d'enregistrements non référencés (SQL et Algèbre)

2.7. Exercice : Photos à gauche

Soit les trois relations instanciées suivantes :

Numero	Nom	Prenom
1	Jacques	Brel
2	Georges	Brassens
3	Léo	Ferré

Relation Personne

Personne	Photo
1	1
2	1
3	1

Relation PersonnesPhotos

Numero	Date	Lieu
1	10/12/1965	Paris
2	18/03/2002	Lille

Relation Photo

Combien de tuples renvoie l'instruction SQL suivante :

```

1 SELECT *
2 FROM Photo ph
3 LEFT JOIN PersonnesPhotos pp ON ph.Numero=pp.Photo
4 LEFT JOIN Personne pe ON pp.Personne=pe.Numero;

```

2.8. Opérateurs ensemblistes

Introduction

Les opérateurs ensemblistes ne peuvent être exprimés à l'aide de l'instruction de sélection seule.

Union**§ Syntaxe**

```
1 SELECT * FROM R1
2 UNION
3 SELECT * FROM R2
```

Intersection**§ Syntaxe**

```
1 SELECT * FROM R1
2 INTERSECT
3 SELECT * FROM R2
```

Différence**§ Syntaxe**

```
1 SELECT * FROM R1
2 EXCEPT
3 SELECT * FROM R2
```

🔍 Remarque

Les opérations INTERSECT et EXCEPT n'existent que dans la norme SQL2, et non dans la norme SQL1. Certains SGBD sont susceptibles de ne pas les implémenter.

Exercices



1. Exercice : Location d'appartements

[30 min]

Soit le schéma relationnel suivant gérant le fonctionnement d'une agence de location d'appartements.

```
1 APPARTEMENT(#code_appt:String, adresse:String, type:{studio,F1,F2,F3,F4,F5+},  
  prix_loyer:Real)  
2 LOCATAIRE(#code_loc:String, nom:String, prenom:String)  
3 LOCATION(#code_loc=>Locataire, #code_appt=>Appartement)  
4 PAIEMENT_LOYER(#code_loc=>Locataire, #code_appt=>Appartement, #date_payement:Date,  
  prix_paye:Real)
```

Question 1

En algèbre relationnelle et en SQL afficher tous les paiements effectués par un locataire avec le code X.

Question 2

En algèbre relationnelle et en SQL afficher l'adresse de l'appartement loué par un locataire avec le code X.

Question 3

En algèbre relationnelle et en SQL proposer une requête qui affiche tous les appartements libres.

2. Exercice : Employés et salaires

[20 minutes]

Soit le schéma relationnel :

```
1 Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)  
2 Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)  
3 Societe (#Nom, Pays, Activite)
```

Question 1

Écrivez une requête SQL permettant de sélectionner les noms de tous les directeurs de France.

Question 2

Écrivez une requête SQL permettant d'afficher le salaire de tous les employés en francs (sachant que le salaire dans la table est en euros et que un euro vaut 6.55957 franc).

Question 3

Écrivez une requête SQL permettant de vérifier que les salaires de chaque employé correspondent bien à ce qui est autorisé par leur fonction.

Question 4

Écrivez une requête SQL permettant le passage aux 35 heures :

- en modifiant le nombre d'heures travaillées pour chaque fonction (ramené à 35 s'il est supérieur à 35),
- et en réajustant les échelles de salaire au pro-rata du nombre d'heures travaillées avant le passage aux 35 heures.

Abréviations



SGBDR : Système de Gestion de Bases de Données Relationnelles

SQL : Structured Query Language

Index



Algèbre.....	10	UNION.....	16
Alias.....	6	WHERE.....	4, 10, 12
AND	5		
AS	6		
BETWEEN.....	5		
Comparaison	5		
Condition.....	5		
DATE.....	7		
DISTINCT.....	6		
EXCEPT.....	16		
FROM.....	4, 6, 7		
IN.....	5		
INNER.....	12		
INTERSECT.....	16		
IS NULL	5		
JOIN	12, 14		
Langage	4		
LEFT	14		
LIKE	5		
LMD	4, 10		
Logique.....	5		
NOT	5		
Opérateur	5		
OR	5		
ORDER BY.....	7		
OUTER.....	14		
Question	4		
Requête.....	4		
RIGHT	14		
SELECT.....	4, 6, 6, 7, 10, 11		
SQL.....	4, 10		
Tri.....	7		