Livre 2 : Index, jointures et Gestion des utilisateurs

4.0 01/09/2021

Bloc 2 - Conception, adaptation d'une base de données

Table des matières

L'interrogation de données - Les index et les vues

- Interroger une base de données : Les index et les vues
  - Les index
  - Les vues
- Exercice : Cas Infosynaps
- Exercice : Cas : Au père tranquille
- Exercice : Cas Tintinophile

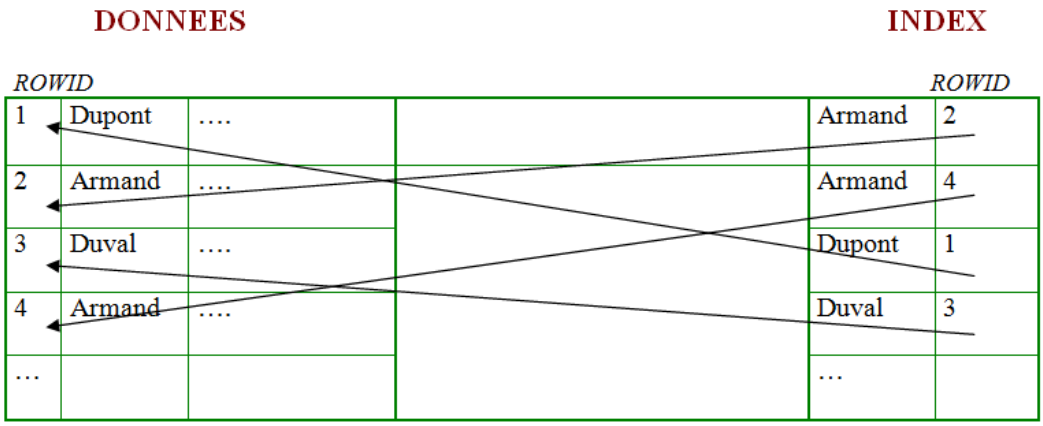
1. Interroger une base de données : Les index et les vues

La définition d'index et de vues relève du langage de définition de données (LDD) au même titre que la création de tables.

1.1. Les index

On crée des index sur les tables pour permettre un accès plus rapide à l'information. Un index concerne une ou plusieurs colonnes de la table et entraîne la création d'un espace comportant pour chaque valeur de la colonne ou du groupe de colonnes, le numéro du (ou des) tuple(s) correspondant(s).

L'illustration présente une table Client avec un index sur le nom.



Exemple d'index de la table CLIENT

Remarque **:**

- les données sont insérées de façon séquentielle dans la table Client au fur et à mesure des ajouts
- chaque ligne de la table est identifiée de façon unique par son identifiant appelé ROWID
- lorsqu'on crée un index sur le nom du client, les valeurs de la colonne indexée sont en permanence classées ; à chacune de ces valeurs, on fait correspondre le ROWID de la ligne de la table.

Un index va permettre d'éviter de balayer toute la table lors de la recherche d'une ligne répondant à un critère de sélection exprimé sur la colonne à indexer.

Exemple :

```sql
SELECT *
FROM Client
WHERE Nom_cli = 'Armand' ;
```

Exécution sans index sur le nom de client : Le SGBDR scrute toute la table Client (bloc par bloc et ligne par ligne) pour retrouver les clients nommés 'Armand'.

Exécution avec index sur le nom de client : Le SGBDR passe dans ce cas par l'index qui le renverra vers les lignes concernées dans la table. Cette seconde solution est nettement plus rapide car le nombre de blocs parcourus est beaucoup moins important.

¶ Syntaxe : Création d'un index

```sql
CREATE [UNIQUE] INDEX NomIndex
ON NomTable (NomColonne [ASC DESC] , NomColonne ...]);
```

- UNIQUE est à préciser si les doublons ne sont pas acceptés.
- Par défaut, le classement est croissant (ASC).
- Un index peut être composé de une à 16 colonnes.

Exemple :

CREATE INDEX idxcli ON Client (Nom_cli) ;

¶ Syntaxe : Suppression d'un index

```sql
DROP INDEX NomIndex ;
```

Conseil : Colonnes à indexer

- Les clés primaires : La création d'un index sur la clé primaire est automatique sous PostGreSQL.
- Les clés étrangères : Ces clés servent pour les jointures. C'est justement pour accélérer les opérations de jointure qu'un index est nécessaire sur les clés étrangères.
- Les colonnes servant souvent de critère de recherche : Ce sont les colonnes qui sont référencées dans une clause WHERE.
- Les colonnes servant souvent de critère de tri ou de groupement : Ce sont les colonnes qui sont référencées dans une clause ORDER BY ou GROUP BY.

Conseil : Colonnes à ne pas indexer

Il est recommandé de ne pas indexer :

- les colonnes présentant peu de valeurs distinctes,
- les colonnes souvent modifiées.

Remarque : Cas où l'index n'est pas utilisé

Bien qu'une colonne soit indexée, l'index ne sera pas utilisé si :

- la colonne indexée ne figure ni dans WHERE, ni dans ORDER BY et ni dans GROUP BY,
- on applique une fonction sur la colonne indexée :

```sql
SELECT *
FROM Client
WHERE UPPER(Nom_cli) LIKE 'P%';
```

- on teste l'inégalité entre la colonne indexée et une valeur :

```sql
SELECT *
FROM Client
WHERE Nom_cli != 'Dupont';
```

- on compare la colonne indexée avec un modèle de chaîne dont on ne connaît pas le début :

```
SELECT *
FROM client
WHERE Nom_cli LIKE '_A%';
```

Inconvénients des index

- L'insertion ou la suppression dans une table indexée de même que la modification de la valeur d'une colonne indexée va entraîner une mise à jour de l'espace d'index.

Les opérations DELETE et UPDATE sont néanmoins très souvent précédées par une recherche. Dans ce cas, l'index fait gagner du temps lors de la recherche et en fait perdre lors de l'opération proprement dite de modification ou suppression. Donc, globalement, l'index n'est pas vraiment pénalisant.

- L'opération INSERT ne fait pas de recherche préalable : d'éventuels index ne peuvent être que pénalisants.

Dans les cas de saisie massive initiale d'une table, on recommande de saisir la table d'abord et de créer les index après ; de cette façon, l'étape de saisie est plus rapide et l'espace disque occupé est moindre.

1.2. Les vues

Les vues sont des tables virtuelles créées à partir de requêtes SQL. Seule la définition de la vue (c'est à dire le texte de la requête) est stockée, l'ensemble de tuples correspondant n'est pas stocké contrairement aux tuples d'une table.

Une vue dans un SGBDR est assimilable à la notion de requête enregistrée sous Access mais offre plus de possibilités qu'une requête Access notamment au niveau de l'attribution des privilèges.

Remarque **:**

- La clause ORDER BY est interdite dans la requête de création de la vue (elle n'a absolument aucun intérêt dans ce contexte : une table est un « sac de billes » non ordonnées).
- La clause WITH CHECK OPTION permet de garantir que les tuples insérés ou modifiés au travers de la vue sont compatibles avec la définition de la vue. Bien sûr, cette clause n'a de sens que si l'insertion ou la modification est possible.

Avec PostGreSQL cette option n'est pas disponible et les **vues ne sont accessibles qu'en lecture seule**.

Syntaxe **: Création d'une vue**

```
CREATE VIEW NomVue
AS
Requête
[WITH CHECK OPTION] ;
```

Exemple :

TABLE CLIENT :

Num_cli	Nom_cli	Adr_cli	Cdp_cli	Vil_cli	Tel_cli
1	Dupont	69 rue de Turbigo	75003	PARIS	0143211234
2	Armand	05 rue Royale	75002	PARIS	0144332211
3	Duval	14 rue des Petits Champs	75002	PARIS	0142622222
4	Armand	06 rue de France	21000	DIJON	0380421312
5	Dupont	10 rue de Conte	75003	PARIS	0143211235
6	Petitjean	25 rue de Turin	80000	AMIENS	NULL
7	Durand	12 rue de la Paix	21000	DIJON	0349999555

VUE V_CLI_PAR :

Num_cli	Nom_cli	Tel_cli
1	Dupont	0143211234
2	Armand	0144332211
3	Duval	0142622222
5	Dupont	0143211235

Vue sur la table CLIENT

- on effectue ici une projection et une sélection.
- la manipulation de la vue V_CLI_PAR permet de masquer les lignes des clients 4, 6 et 7 et certaines colonnes pour les clients "visibles".

```
CREATE VIEW V_CLI_PAR
AS
SELECT Num_clin Nom_cli, Tel_cli
FROM Client
WHERE Vil_cli = 'PARIS' ;
```

Pour utiliser la vue afin de consulter les clients parisiens, il suffit d'écrire :

```
SELECT *
FROM V_CLI_PAR;
```

Exemple :

```
CREATE VIEW V_CLI_VILLE (V_ville, V_nb)
AS
SELECT Vil_cli, count(*)
FROM Client
GROUP BY Vil_cli;
```

Remarque :

On n'est pas obligé de spécifier les noms des colonnes de la vue si la liste de sélection de la requête ne comporte ni expression, ni fonction, ni colonnes en double.

Si on ne spécifie pas de nom, les colonnes de la vue héritent des noms des champs indiqués dans la liste de sélection (sans le nom de la table en préfixe). Il faut soit ne préciser aucun des noms de colonnes de la vue, soit les préciser tous.

Intérêt des vues

- Une vue permet d'assurer la confidentialité de certaines données : avec les vues, il est possible de mettre uniquement certaines colonnes et/ou certaines lignes à disposition de l'utilisateur (cf. : exemple ci-dessus).
- Une vue peut également être utilisée pour simplifier la formulation de requêtes lorsque celles-ci sont complexes.

Exemple :

La vue V_COMM permet de simplifier la formulation d'une requête complexe.

```
CREATE VIEW V_COMM
AS
SELECT C.num_com, C.dat_com, L.num_lig, L.num_art, A.des_art, L.qte_lig
FROM Commande C, Lg_comm L, Article A
WHERE C.num_com = L.num_com
AND L.num_art = A.num_art ;
```

- trois tables sont en jeu : LG_COMM, COMMANDE et ARTICLE,
- on effectue une projection et des jointures
- on écrit ensuite simplement la requête suivante pour avoir toutes les info sur une commande donnée (ici la 'C123'). Ce qui donne la requête suivante :

```
SELECT *
FROM V_COMM
WHERE num_com='C123';
```

Mise à jour (modification, insertion, suppression) à travers une vue (pas sous PostGreSQL)

Il n'est pas possible d'effectuer des mises à jour à partir de n'importe quelle vue.

La vue n'est pas modifiable dans les cas suivants :

- si elle ne contient pas tous les attributs définis comme NON NULL dans la table interrogée,
- ou si elle contient une jointure
- ou si elle contient une fonction agrégat.
- ou enfin, si elle ne contient un "SELECT DISTINCT".

Exemple :

Soit la vue V_CLI_DIJON :

```
CREATE VIEW V_CLI_DIJON
AS
SELECT *
FROM Client
WHERE Vil_cli = 'DIJON' ;
```

Modification à travers V_CLI_DIJON :

```
UPDATE V_CLI_DIJON
SET Tel_cli = '0342123456'
WHERE Num_cli = 4 ;
```

Insertion d'un nouveau client à partir de V_CLI_DIJON :

```
INSERT INTO V_CLI_DIJON VALUES
(8, ‘Portier', '03 rue de Lille', 75003, ‘PARIS', NULL) ;
1 ligne créée.
```

Attention :

Si la vue V_CLI_DIJON avait été définie avec la clause WITH CHECK OPTION, l'insertion n'aurait pas été acceptée :

```
INSERT INTO V_CLI_DIJON VALUES (8, ‘Portier', '03 rue de Lille', 75003, ‘PARIS', NULL) ;
=> message d'erreur :
ORA-01402: vue WITH CHECK OPTION - violation de clause WHERE
```

En effet, on essaye d'insérer un client habitant à Paris alors que la vue est créée exclusivement avec des clients habitant à Dijon !

Modification d'une vue

```
REPLACE VIEW nomVue AS
requête ...
```

On écrit le plus souvent, dans la création d'une vue :

```
CREATE OR REPLACE VIEW nomVue AS
requête...
```

... de manière à faciliter les corrections éventuelles.

Suppression d'une vue

```
DROP VIEW NomVue ;
```

2. Exercice : Cas Infosynaps

La société InfoSynaps met à disposition de ses collaborateurs un espace de communication et d'information sous la forme d'un site intranet.

Le site intranet est constitué de différentes rubriques dont l'accès est régi suivant le type d'utilisateur. Par exemple, M. Pierre DUBOIS, intervenant en clientèle, dispose de la rubrique « Formation technique », alors que M. Patrick VENTOUT, commercial, n'y a pas accès. En revanche, M. VENTOUT peut utiliser la rubrique « Gestion clients et prospects ». Par ailleurs, M. DUBOIS et M. VENTOUT ont tous les deux accès à la rubrique « Frais de déplacement ». Une base de données dont le schéma relationnel est fourni ci-après permet d'assurer la gestion des accès aux différentes rubriques du site intranet.

TYPE_UTILISATEUR (code, libellé)
code : clé primaire
Exemple : Le type codé « IC » porte le libellé « Intervenant en clientèle ».

UTILISATEUR (login, nom, prénom, motDePasse, codeTypeUtil)
login : clé primaire
codeTypeUtil : clé étrangère en référence à code de TYPE_UTILISATEUR
Exemple : L'utilisateur de login « DUBOISP » correspond à « DUBOIS Pierre ».

RUBRIQUE (ref, intitulé)
ref : clé primaire
Exemple : La rubrique « FT » correspond à l'intitulé « Formation technique ».

ACCES (codeTypeUtil, refRubrique, nbVisites)
codeTypeUtil, refRubrique : clé primaire
codeTypeUtil : clé étrangère en référence à code de TYPE_UTILISATEUR
refRubrique : clé étrangère en référence à ref de RUBRIQUE
Exemples : La rubrique « FT » est accessible par le type utilisateur codé « IC » et enregistre actuellement 100 visites pour ce type d'utilisateur. La rubrique « GCP » est accessible par le type utilisateur codé « COM » (200 visites) et par le type d'utilisateur codé « IC » (92 visites).

Cas Infosynaps - Relations

Les contraintes d'intégrité de clé primaire et référentielle ont été omises lors de la création du schéma de la table ACCES qui contient maintenant des données.

Question 1

Écrire l'(les) ordre(s) SQL nécessaire(s) à l'ajout des contraintes d'intégrité absentes.

On considérera que la contrainte NOT NULL a été déclarée sur les colonnes refRubrique et codeTypeUtil lors de la création de la table ACCES.

Écrire les requêtes SQL permettant de réaliser les opérations suivantes :

Question 2

A. Obtenir la référence et l'intitulé des rubriques accessibles par l'utilisateur de login "DUBOISP".

Question 3

B. Mettre à jour la table ACCES suite à la visite de la rubrique de référence "FT" par un utilisateur de type codé "IC".

Question 4

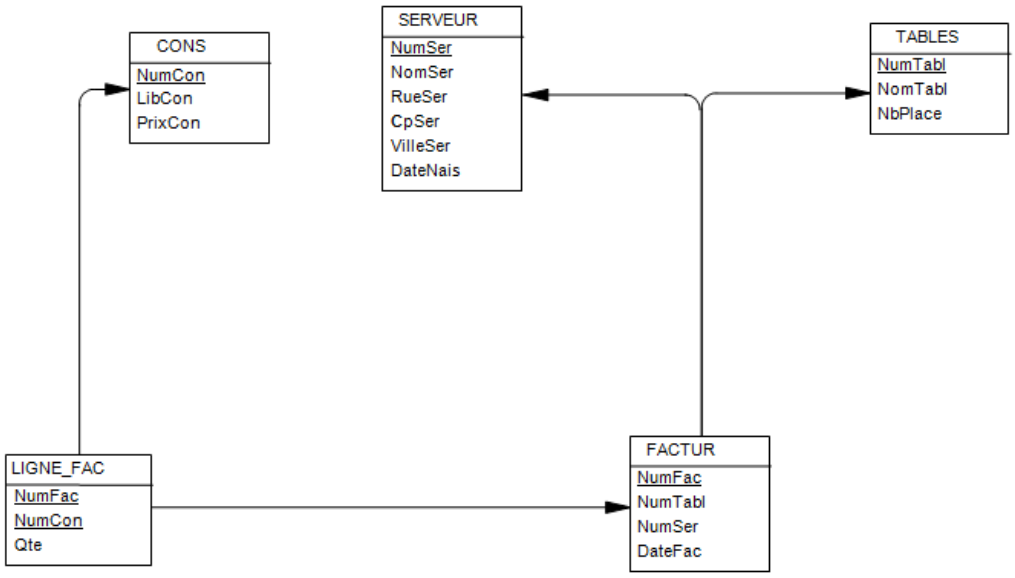
A. Créer la vue appelée "vFrequentationRubrique" qui donne par rubrique le nombre total de visites.

Les colonnes de la vue seront nommées ref, intitulé et nbTotalVisites.

Question 5

B. En utilisant la vue "vFrequentationRubrique", afficher l'(les) intitulé(s) de rubrique présentant le plus grand nombre total de visites.

3. Exercice : Cas : Au père tranquille



Au père tranquille - MLR

Question 1

On effectue souvent des recherches à partir du nom du serveur. On vous demande de trouver une solution pour accélérer ces requêtes et d'écrire l'instruction correspondante.

Question 2

On effectue souvent des jointures entre facture et serveur. On vous demande de trouver une solution pour accélérer ces requêtes et d'écrire l'instruction correspondante.

Question 3

• Créer une vue permettant d'afficher le chiffre d'affaires de la journée (vous utiliserez la fonction now() ou current_date() qui rend la date et l'heure système).

Vérifier que la vue est correcte. Pour cela, insérer les données nécessaires pour avoir au moins une facture (et les lignes de factures correspondantes) pour la date du jour (facultatif).

Question 4

Créer une vue permettant d'obtenir les factures établies par le serveur Pilou.

Vérifier que la vue est correcte (facultatif).

Question 5

Créer une vue permettant d'obtenir le chiffre d'affaire généré par les boissons de type "Café".

4. Exercice : Cas Tintinophile

Soit le modèle relationnel suivant :

ALBUM (**NOALB**, TITREALB, DATEALB)
PAYS (**NOPAYS**, NOMPAYS)
DEROULER (#**NOALB**, #**NOPAYS**)
PERSONNAGES (**NOPERS**, NOMPERS, PRENOMPERS, FONCTPERS, SEXEPERS, GENTILPERS)
PARTICIPER (#**NOALB**, #**NOPERS**)
JURONS (**NOJURON**, NOMJURON)
PRONONCER (#**NOPERS**, #**NOJURON**, #**NOALB**, **NOPAGE**)

Cas Tintinophile - MLR

Question 1

Retrouver le MCD correspondant :

Question 2

Créer la base de données sous PostGreSQL et intégrer le script suivant :

Cas Tintinophile - Script SQL [cf. script_tintin.zip]

Question 3

Ajouter la contrainte de clé primaire de "dérouler".

Question 4

Un album doit avoir obligatoirement un nom et celui ci doit être unique.

Question 5

Quand on supprime un personnage, on doit également supprimer ses participations aux différents albums.

Écrire et tester les requêtes qui permette de répondre aux questions suivantes :

Question 6

Combien de jurons la Castafiore prononce-t-elle ?

Question 7

Classer des personnages par nombre de jurons prononcés croissant ?

Question 8

Dans quel album le capitaine HADDOCK fait-il son apparition ?

Question 9

Quels sont les personnages qui ne jurent jamais ?

Question 10

Quels sont les albums (tous les renseignements) où il existe au moins 3 pays concernés par cet album (utiliser 2 formulations : sans puis avec un EXISTS).

Question 11

Quels sont les albums (tous les renseignements) où ne figurent que des personnages qui jurent ?

Question 12

Quels sont les personnages (numéro et nom) qui apparaissent dans tous les albums ?

II L'interrogation de données - Les jointures

- L'interrogation de données - Les jointures
    - Les requêtes sans jointure
    - Jointure naturelle et équijointure
    - Syntaxe normalisée des jointures
        - La jointure interne
        - La jointure externe
        - L'auto jointure
- Exercices d'application

1. L'interrogation de données - Les jointures

Les jointures permettent de mettre en relation plusieurs tables concourant à rechercher la ré-ponse à des interrogations et donc de combiner les colonnes de différentes tables.

Il existe en fait différentes natures de jointures que nous expliciterons plus en détail.

Retenez cependant que la plupart des jointures entre tables s'effectuent en imposant l'égalité des valeurs d'une colonne d'une table à une colonne d'une autre table. On parle alors de jointure na-turelle et équijointure. Mais on trouve aussi des jointures d'une table sur elle-même (on parle alors d'auto-jointure). Enfin il arrive que l'on doive procéder à des jointures externes, c'est à dire joindre une table à une autre, même si la valeur de liaison est absente dans une table ou l'autre.

Une jointure entre tables peut être mise en œuvre, soit à l'aide des éléments de syntaxe SQL que nous avons déjà vu (SELECT ... FROM T1, T2 WHERE T1.CHAMPS = T2.CHAMPS...) , soit à l'aide d'une clause spécifique du SQL, la clause JOIN.

1.1. Les requêtes sans jointure

¶ *Syntaxe* : Rappel de la syntaxe du SELECT

SELECT [DISTINCT ou ALL] * ou liste de colonnes

FROM nom des tables ou des vues

Exemple :

Récupération des n° des téléphones associés aux clients

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT, T_TELEPHONE ;
```

```
CLI_NOM  TEL_NUMERO
-------  --------------
DUPONT   01-45-42-56-63
DUPONT   01-44-28-52-52
DUPONT   01-44-28-52-50
DUPONT   06-11-86-78-89
DUPONT   02-41-58-89-52
DUPONT   01-51-58-52-50
DUPONT   01-54-11-43-21
DUPONT   06-55-41-42-95
DUPONT   01-48-98-92-21
...
```

Produit cartésien...

Dans ce cas, que calcule le compilateur SQL ? **Le produit cartésien des deux ensembles**.

Dans notre exemple la requête génère 17 400 lignes... (La table « T_CLIENT » possède 100 enregistrements et la table « T_TELEPHONE » en possède 174).

Il faut donc définir absolument un critère de jointure.

1.2. Jointure naturelle et équijointure

Nous allons commencer par voir comment à l'aide du SQL de base nous pouvons exprimer une jointure.

Dans le cas présent, ce critère est la correspondance entre les colonnes contenant la référence de l'identifiant du client (CLI_ID). Le résultat final est de 174 enregistrements.

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT, T_TELEPHONE
WHERE T_CLIENT.CLI_ID  = T_TELEPHONE.CLI_ID ;

Ou (avec des alias)

SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT C, T_TELEPHONE T
WHERE C.CLI_ID = T.CLI_ID ;
```

```
CLI_NOM   TEL_NUMERO
-------          --------------
DUPONT    01-45-42-56-63
DUPONT    01-44-28-52-52
DUPONT    01-44-28-52-50
BOUVIER   06-11-86-78-89
DUBOIS    02-41-58-89-52
DREYFUS   01-51-58-52-50
DUHAMEL   01-54-11-43-21
BOYER     06-55-41-42-95
MARTIN    01-48-98-92-21
MARTIN    01-44-22-56-21
...
```

Requête avec critère de jointure

Définition :

Une jointure naturelle est une jointure entre clefs primaires et clefs secondaires basée sur l'égalité des valeurs des colonnes.

Remarque : cas de non équijointure

Le fait de placer comme critère de jointure entre les tables, l'opérateur logique « égal » donne ce que l'on appelle une « équijointure ». Il est possible de faire des équijointures qui ne sont pas naturelles. Il s'agit là d'utiliser n'importe quelle condition de jointure entre deux tables, excepté la stricte égalité :

- > : supérieur
- >= : supérieur ou égal

- < : inférieur
- <= : inférieur ou égal
- <> ou != : différent de
- IN : dans un ensemble
- LIKE : correspondance partielle
- BETWEEN : entre deux valeurs
- EXISTS : dans une table

1.3. Syntaxe normalisée des jointures

Les jointures normalisées s'expriment à l'aide du mot clef JOIN dans la clause FROM.

¶ *Syntaxe* : Jointure interne

```
SELECT ...
FROM <table gauche> [INNER] JOIN <table droite>
ON <condition de jointure>...
```

¶ *Syntaxe* : Jointure externe

```
SELECT ...
FROM <table gauche> LEFT | RIGHT | FULL OUTER JOIN <table droite>
ON <condition de jointure>...
```

a) La jointure interne

Il s'agit de la plus commune des jointures.

🔎 *Exemple* :

Reprenons notre exemple de départ :

```
SELECT CLI_NOM, TEL_NUMERO
FROM   T_CLIENT
    ........ INNER JOIN T_TELEPHONE
        ..... ON T_CLIENT.CLI_ID = T_TELEPHONE.CLI_ID ;

Ou (avec des alias)


SELECT CLI_NOM, TEL_NUMERO
FROM   T_CLIENT C
        ........ INNER JOIN T_TELEPHONE T
            ........ ON C.CLI_ID = T.CLI_ID ;
```

CLI_NOM	TEL_NUMERO
DUPONT	01-45-42-56-63
DUPONT	01-44-28-52-52
DUPONT	01-44-28-52-50
BOUVIER	06-11-86-78-89
DUBOIS	02-41-58-89-52
DREYFUS	01-51-58-52-50
DUHAMEL	01-54-11-43-21
BOYER	06-55-41-42-95
MARTIN	01-48-98-92-21
MARTIN	01-44-22-56-21
...	

La jointure interne

✎ *Remarque* :

Le mot clef INNER est facultatif, il est implicite.

b) La jointure externe

Exemple : Client BOUVIER

Voici le résultat obtenu en interrogeant le client « BOUVIER ». Il possède un GSM, un TEL mais pas de FAX.

```
SELECT CLI_NOM, TEL_NUMERO,  TYP_CODE
FROM T_CLIENT C, T_TELEPHONE T  WHERE C.CLI_ID = T.CLI_ID
AND CLI_NOM = 'BOUVIER';
```

```
CLI_NOM      TEL_NUMERO      TYP_CODE
-------      --------------      --------
BOUVIER      06-11-86-78-89      GSM
BOUVIER      04-94-41-17-27      TEL
```

Client Bouvier - Résultat attendu

Cherchons à présent les clients dont les numéros de téléphone correspondent à un fax :

```
SELECT CLI_NOM, TEL_NUMERO
FROM T_CLIENT C, T_TELEPHONE T
WHERE C.CLI_ID = T.CLI_ID
 AND TYP_CODE = 'FAX' ;

⇔

SELECT CLI_NOM, TEL_NUMERO
FROM   T_CLIENT C
     INNER JOIN T_TELEPHONE  T
      ON C.CLI_ID = T.CLI_ID
WHERE  TYP_CODE = 'FAX' ;
```

```
CLI_NOM       TEL_NUMERO
-------       --------------
DUPONT        01-44-28-52-50
MARTIN        01-44-22-56-21
DUHAMEL       01-54-11-43-89
DUPONT        05-59-45-72-42
MARTIN        01-47-66-29-55
DUBOIS        04-66-62-95-64
DREYFUS       04-92-19-18-58
DUHAMEL       01-55-60-93-81
PHILIPPE      01-48-44-86-19
DAUMIER       01-48-28-17-95
...
```

Client BOUVIER - Résultat sans lui

Comme vous pouvez le constater, le nom du client BOUVIER n'apparaît pas. Il n'a pas été « oublié » par le traitement de la requête, mais le numéro de fax de ce client n'est pas présent dans la table T_TELEPHONE.

Or le moteur SQL recherche les valeurs de la jointure par égalité. Comme l'ID_CLI de BOUVIER n'est pas présent dans la table T_TELEPHONE, il ne peut effectuer la jointure et ignore donc la ligne concernant le client BOUVIER.

Méthode :

Que faut-il modifier dans la requête pour obtenir une ligne BOUVIER avec aucune référence de téléphone associée dans la réponse ?

Il suffit en fait d'opérer à l'aide d'une jointure externe :

```
SELECT CLI_NOM, TEL_NUMERO
FROM   T_CLIENT C
     LEFT OUTER JOIN T_TELEPHONE  T
        ON C.CLI_ID = T.CLI_ID
WHERE  TYP_CODE = 'FAX' ;
```

```
CLI_NOM    TEL_NUMERO
-------    --------------
DUPONT     01-44-28-52-50
DUPONT     05-59-45-72-42
MARTIN     01-47-66-29-55
BOUVIER    NULL
DUBOIS     04-66-62-95-64
DREYFUS    04-92-19-18-58
FAURE      NULL
LACOMBE    NULL
DUHAMEL    01-54-11-43-89
DUHAMEL    01-55-60-93-81
...
```

Résultat attendu correct

¶ *Syntaxe* :

```
SELECT ...
FROM <table gauche>
LEFT | RIGHT | FULL OUTER JOIN <table droite 1>
ON <condition de jointure>
[LEFT | RIGHT | FULL OUTER JOIN <table droite 2>
ON <condition de jointure 2>]
...

```

✎ *Remarque* :

Les mots clefs LEFT, RIGHT et FULL indiquent la manière dont le moteur de requête doit effectuer la jointure externe. Ils font référence à la table située à gauche (LEFT) du mot clef JOIN ou à la table située à droite (RIGHT) de ce même mot clef. Le mot FULL indique que la jointure externe est bilatérale.

¶ *Syntaxe* : LEFT OUTER JOIN

```
SELECT colonnes
FROM TGauche LEFT OUTER JOIN TDroite ON condition de jointure ...
```

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TGauche qui n'ont pas été prises en compte au titre de la satisfaction du critère.

¶ *Syntaxe* : RIGHT OUTER JOIN

```
SELECT colonnes
FROM TGauche RIGHT OUTER JOIN TDroite ON condition de jointure...
```

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TDroite qui n'ont pas été prises en compte au titre de la satisfaction du critère.

¶ *Syntaxe* : FULL OUTER JOIN

```
SELECT colonnes
FROM TGauche FULL OUTER JOIN TDroite ON condition de jointure
```

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table TGauche et TDroite qui n'ont pas été prises en compte au titre de la satisfaction du critère.

💬 *Conseil* :

Dans la mesure du possible, utiliser toujours un opérateur de jointure normalisé (mot clef JOIN).

En effet :

- Les jointures faites dans la clause WHERE (ancienne syntaxe de 1986 !) ne permettent pas de faire la distinction de prime abord entre ce qui relève du filtrage et ce qui relève de la jointure. La lisibilité des requêtes est donc plus grande en utilisant la syntaxe à base de JOIN.
- L'optimisation d'exécution de la requête est souvent plus pointue du fait de l'utilisation du JOIN.

c) L'auto jointure

🔑 *Définition* :

Une auto-jointure est une jointure qui permet de joindre la table sur elle-même.

🔎 *Exemple* : Chambres communicantes

Pour donner un exemple concret à nos propos nous allons modéliser le fait qu'une chambre puisse communiquer avec une autre (par une porte). Dès lors, le challenge est de trouver quelles sont les chambres qui communiquent entre elles.

Table T_Chambre

CHB_ID	CHB_COMMUNIQUE
9	7
7	9
12	14
14	12

Auto jointure - Table T_CHAMBRE

Pour formuler la recherche de chambres communicantes, il suffit de faire la requête suivante :

```
SELECT DISTINCT c1.CHB_ID, c1.CHB_COMMUNIQUE
FROM   T_CHAMBRE c1
       INNER JOIN T_CHAMBRE c2
             ON c1.CHB_ID = c2.CHB_COMMUNIQUE
```

```
CHB_ID  CHB_COMMUNIQUE
------  --------------
     7               9
     9               7
    12              14
    14              12
```

Auto-jointure - Résultat

...ou la table T_CHAMBRE figure deux fois par l'entremise de deux sur-nommages différents c1 et c2.

Notons que cette présentation n'est pas pratique car elle dédouble le nombre de couples de chambres communicantes, donnant l'impression qu'il y a 4 couples de chambres communicantes ce qui est faux...

Il manque juste un petit quelque chose pour que cette requête soit parfaite. Il suffit en effet de ne retenir les solutions que pour des identifiants inférieurs ou supérieur d'une table par rapport à l'autre :

```
SELECT DISTINCT c1.CHB_ID, c1.CHB_COMMUNIQUE
FROM   T_CHAMBRE c1
       INNER JOIN T_CHAMBRE c2
             ON c1.CHB_ID = c2.CHB_COMMUNIQUE
                AND c1.CHB_ID <= c2.CHB_ID
```

```
CHB_ID  CHB_COMMUNIQUE
------  --------------
     7               9
    12              14
```

Auto-jointure - Résultat correct

¶ *Syntaxe* :

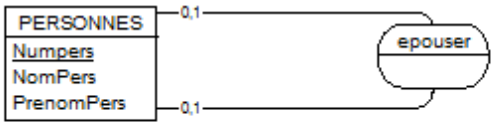
```
1 SELECT [DISTINCT ou ALL] * ou liste de colonnes
2 FROM laTable t1 INNER JOIN laTable t2
3 ON t1.laClef = t2.laPseudoClefEtrangère
4
```

✎ *Remarque* :

L'utilisation des alias est obligatoire dans ce type de jointure.

Il est assez fréquent que l'on ait besoin de telles auto-jointures. Voici quelques exemples de relation nécessitant une auto jointure de tables :

- dans une table des employées, connaître le supérieur hiérarchique de tout employé
- dans une table de composants pour savoir quels sont les composants nécessaires à la réalisation d'autres composants...
- dans une table de personnes, retrouver l'autre moitié d'un couple marié...



PERSONNES (Numpers, NomPers, prenomPers, Numconjoint#)

Mariage - MCD et MLR

Pour connaître le nom de la personne N° 1 et de son conjoint, on écrira :

```
SELECT p1.nom, p2.nom
FROM personnes p1, personnes p2
WHERE p2.numPers = p1.numconjoint
AND p1.numpers = 1 ;
```

ou :

```
SELECT P1.nom, P2.nom
FROM personnes P1, JOIN personnes P2
ON P2.numPers = P1.numconjoint
WHERE P1.numpers = 1 ;
```

2. Exercices d'application

Pas d'exercice d'application.

Remarque **:**

Il est possible (voire préférable) de reprendre tous les exercices précédents, en réécrivant les requêtes en SQL 2.

III Administrer une BDD - Les utilisateurs

- La gestion des utilisateurs
    - Les utilisateurs d'une base de Données
        - Création d'un utilisateur
        - Suppression d'un utilisateur
        - Liste des utilisateurs
    - Les privilèges sur les objets
        - Généralités
        - Attribution des privilèges objets
        - Révocation des privilèges objets
    - Les groupes d'utilisateurs
        - Création d'un groupe
        - Attribution de droits au groupe
        - Ajout d'un utilisateur dans un groupe
        - Suppression d'un utilisateur
        - Suppression d'un groupe
- Exercice : Cas Transair

1. La gestion des utilisateurs

La gestion des droits d'accès relève du langage de contrôle de données (LCD).

Il est indispensable d'introduire des éléments de sécurité dans une base de données. Ainsi, si par exemple un pirate exploite une faille de votre serveur web, il ne pourra ni détruire votre base de données ni consulter les données confidentielles qu'elle contient.

1.1. Les utilisateurs d'une base de Données

Les SGBDR utilisent souvent le concept de DBA. Le DBA est le « database administrator ». C'est-à-dire celui qui administre le serveur de SGBDR. C'est en quelque sorte le super-utilisateur. Avec PostGreSQL, il s'agit en général de l'utilisateur postgres.

Lorsque vous êtes connecté sur Linux sous le compte postgres, vous pouvez alors créer ou supprimer des bases de données ou des utilisateurs.

Sous PostGreSQL, un utilisateur est considéré comme un objet auquel on va attribuer des privilèges.

a) Création d'un utilisateur

```
1 CREATE USER NomUtilisateur WITH PASSWORD 'MotDePasse' ;
```

Remarque **:**

La syntaxe de création d'un utilisateur est très différente selon les SBGDR.

Un utilisateur créé de cette manière ne pourra pas créer de nouvelles bases de données ou de nouveaux utilisateurs. Il pourra accéder aux bases existantes dans la mesure où on le lui aura permis.

Pour l'autoriser à créer de nouvelles bases de données, ou de nouveaux utilisateurs il faudra lui attribuer ces droits à la création :

```
1 CREATE USER NomUtilisateur WITH PASSWORD 'MotDePasse' CREATEDB CREATEUSER ;
```

... ou les lui conférer après la création avec :

```
1 ALTER USER NomUtilisateur CREATEDB CREATEUSER ;
```

Ces droits peuvent lui être retirés par la commande :

```
1 ALTER USER NomUtilisateur NOCREATEDB NOCREATEUSER ;
```

Remarque :

Dans ces trois exemples les deux droits sont attribués ou retirés simultanément. On peut bien les dissocier l'un de l'autre (un utilisateur peut avoir le droit de créer des bases, mais pas d'utilisateurs ou vice-versa).

La commande ALTER USER peut également être utilisée pour modifier le mot de passe :

```
1 ALTER USER NomUtilisateur WITH PASSWORD 'MotDePasse' ;
```

b) Suppression d'un utilisateur

```
1 DROP USER NomUtilisateur ;
```

c) Liste des utilisateurs

```
1 SELECT * from pg_user ;
```

1.2. Les privilèges sur les objets

a) Généralités

Chaque objet d'une base de données, principalement les tables, possède des **privilèges** réglementant strictement les **accès**. Seul le **propriétaire** (créateur) d'un objet (et le DBA à une exception près) a tous les droits sur celui-ci. Le propriétaire (et lui seul) peut attribuer des droits sur ses objets de façon sélective à d'autres utilisateurs, voire à tous les utilisateurs (**PUBLIC**).

Les droits sur objet sont les suivants :

- SELECT : droit de lecture (sur table ou vue)
- INSERT : droit d'insertion de lignes (sur table ou vue)
- UPDATE : droit de modification de lignes (sur table ou vue)
- DELETE : droit de suppression de lignes (sur table ou vue)

Le mot clé **ALL** (PRIVILEGES) permet d'attribuer tous les droits.

b) Attribution des privilèges objets

```
1 GRANT PrivilègeObjet [,PrivilègeObjet ...]
2 ON NomObjet
3 TO NomUtilisateur1 [, NomUtilisateur2 ...]
4 [WITH GRANT OPTION];
```

Remarque :

NomObjet désigne la tale ou la vue.

Si le privilège objet est UPDATE et qu'on souhaite autoriser la modification uniquement sur certaines colonnes, il faut préciser la liste des colonnes pour lesquelles la modification est autorisée.

L'option WITH GRANT OPTION permet d'autoriser la liste des utilisateurs cités dans la commande GRANT à donner à leur tour le droit qu'ils viennent de recevoir.

Attention :

Cette clause n'est pas disponible sous PostGreSQL.

Simulation :

L'utilisateur "essai" peut sélectionner des clients, en ajouter, les modifier ou encore les supprimer :

```
1 GRANT SELECT, INSERT, UPDATE, DELETE ON client TO essai ;
```

... Ou

```
1 GRANT ALL ON client TO essai ;
```

Si le privilège objet est UPDATE et qu'on souhaite autoriser la modification uniquement sur certaines colonnes, il est possible de préciser la liste des colonnes pour lesquelles la modification est autorisée...

¶ Syntaxe :

```
1 UPDATE (NomColonne1 [,Nomcolonne2 ...] ) ;
```

Attention :

Non disponible sous PostGreSQL avant la version 8.

c) Révocation des privilèges objets

```
1 REVOKE PrivilègeObjet [, PrivilègeObjet...] ON NomObjet
2 FROM NomUtilisateur1 [, NomUtilisateur2 ...] ;
```

Simulation :

L'utilisateur ‘essai' a été autorisé à sélectionner, ajouter, modifier ou encore supprimer des clients ; il ne doit plus pouvoir en supprimer :

```
1 REVOKE DELETE ON CLIENT FROM essai ;
```

1.3. Les groupes d'utilisateurs

Gérer les droits au niveau de l'utilisateur n'est en général pas une bonne idée. En effet, les utilisateurs, on en ajoute et on en enlève au fil de la maintenance de l'application. Il est donc préférable de créer des groupes d'utilisateurs.

On gérera les droits au niveau du groupe, et on ajoutera tout simplement des utilisateurs dans ce groupe lorsqu'on en aura besoin.

a) Création d'un groupe

```
1 CREATE GROUP NomGroupe [WITH USER util1, util2, ...] ;
```

Simulation : Exemple :

```
1 CREATE GROUP visiteurs ;
```

b) Attribution de droits au groupe

C'est le même principe que pour attribuer des privilèges objet à un utilisateur excepté qu'il faut indiquer un nom de groupe dans la clause **TO**.

Simulation **:**

Autoriser le groupe ‘'visiteurs' à sélectionner, supprimer, insérer et modifier la table des clients :

```
GRANT SELECT, INSERT, UPDATE, DELETE ON CLIENT TO GROUP visiteurs ;
```

De la même façon, on utilisera **REVOKE** pour les retirer.

c) Ajout d'un utilisateur dans un groupe

```
ALTER GROUP NomGroupe
ADD USER NomUtilisateur1 [, NomUtilisateur2 ...] ;
```

Simulation **:**

Ajouter l'utilisateur essai dans le groupe "visiteurs" :

```
ALTER GROUP visiteurs ADD USER essai ;
```

d) Suppression d'un utilisateur

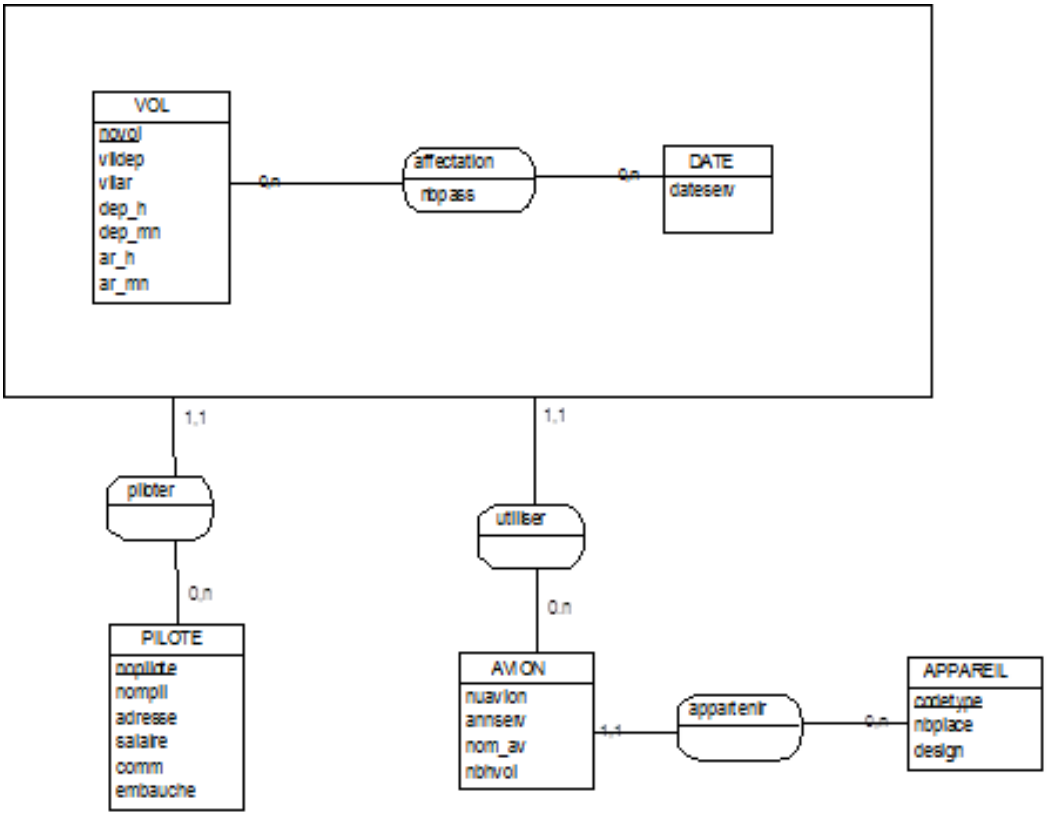
```
ALTER NomGroupe
DROP NomUtilisateur1 [, NomUtilisateur2 ...] ;
```

e) Suppression d'un groupe

```
DROP GROUP NomGroupe ;
```

2. Exercice : Cas Transair

On vous donne le schéma relationnel suivant :



Cas TRANSAIR - MLR

Cela donne le schéma suivant :

```
VOL(novol, vildep,vilar,dep_h, dep_mn, ar_h, ar_mn)
PILOTE(nopilote, nompil, adresse, salaire, comm, embauche)
AVION(nuavion, annserv, nom_av, nbhvol, #type)
APPAREIL(codetype, nbplace, design)
AFFECTATION(novol, dateserv, nbpass, #pilote, #avion)
```

Cas Transair - MLR

Vous disposez du script SQL permettant la création du jeu d'essai.

En tant que nouveau « responsable informatique », vous êtes chargé de gérer les utilisateurs de la base de données décrite précédemment :

A télécharger et dé-zipper
Cas Transair - Script SQL [cf. Transair_SQL.zip]

On identifie plusieurs utilisateurs dont :

- L'administrateur de la base de données (vous-même)
- La directrice des ressources humaines (DRH) qui peut ajouter ou supprimer des pilotes.
- Les contrôleurs aériens qui consultent les vols et les informations sur les avions.
- Les comptables qui gèrent la paie des pilotes et peuvent modifier leur salaire.
- Les responsables de la programmation des vols (Michel ou Joël) qui sont chargés d'enregistrer les nouveaux vols, de les modifier ou de les supprimer et de gérer les affectations qui en dépendant.

Partie 1 - Interrogation de la base de données (30 questions)

Question 1

Établir la liste des pilotes dont le salaire est compris entre 19000 et 23000. Afficher le nom et le salaire des pilotes.

Question 2

Établir la liste des vols qui arrivent à Londres avant 12 heures. Afficher le numéro de vol, la ville de départ, l'heure de départ et l'heure d'arrivée. la liste sera présentée par ordre alphabétique des villes de départ et heures de départ croissantes.

Question 3

Établir la liste des avions qui appartiennent à un type d'appareil dont le premier caractère du code est "7". Afficher les numéros d'avion et les types.

Question 4

Afficher les numéros de vols qui ont eu lieu le 2 mars 2004.

Question 5

Donner la liste alphabétique des pilotes qui habitent PARIS et qui ont été embauchés avant le 1er janvier 2001 ou après le 1er janvier 2002. Afficher les noms des pilotes, leur date d'embauche et leur adresse.

Question 6

Afficher par ordre alphabétique des villes de départ et par heures de départ croissantes, les vols enregistrés dans la base de données.

Question 7

Donner la liste des pilotes qui ne perçoivent pas de commission et qui ont un salaire supérieur à 20000. Afficher le nom du pilote et son salaire.

Question 8

Donner la liste alphabétique des pilotes qui ont effectués un vol le 2 mars 2004. Afficher le nom du pilote, le numéro de vol, la ville de départ et la ville d'arrivée.

Question 9

Donner pour chaque pilote qui habite à Lyon et qui a utilisé un avion totalisant moins de 12000 heures de vol, le type d'appareil correspondant. Afficher le nom du pilote et le libellé du type d'appareil.

Question 10

Donner pour chaque pilote de la base de données , la liste des villes à partir desquelles il a effectué un vol. Afficher le numéro du pilote, son nom et la ville de départ.

Question 11

Donner pour chaque pilote, la liste des villes desquelles il n'a jamais décollé. Afficher le numéro de pilote, le nom du pilote et la ville.

Question 12

Donner la liste des vols qui correspondent à des allers-retours entre deux villes. Afficher le numéro de vol, la ville de départ et la ville d'arrivée.

Question 13

Donner pour chaque pilote qui est passé par PARIS, le numéro de vol et la date correspondante. Afficher le numéro du pilote, son nom, le numéro de vol et la date du vol.

Question 14

Donner la liste des appareils qui sont utilisés pour la liaison LYON-LONDRES. Afficher le code type et le libellé de l'appareil.

Question 15

Établir la liste des avions qui sont partis au moins une fois de Paris ou de Lyon. Afficher uniquement les numéros d'avions.

Question 16

Donner la liste des pilotes qui ont le même nom, mais des adresses différentes. Afficher leur nom et leur adresse.

Question 17

Donner la liste des vols pour lesquels au moins un des pilotes ayant effectué la liaison habite la ville d'arrivée. Pour chaque vol, afficher le numéro de vol, la ville de départ et la ville d'arrivée. (2 lignes)

Question 18

Établir la liste des pilotes qui ont volé sur tous les types d'avion excepté le type SSC. Afficher le numéro et le nom de chaque pilote.

Question 19

Donner la liste des pilotes embauchés le même mois et la même année que le pilote N° 3421. Afficher le numéro, le nom du pilote, le mois et l'année d'embauche.

Question 20

20. Donner le nombre de vols <> auxquels le pilote N° 6723 a été affecté (résultat : 4)

Question 21

Établir la liste donnant, pour chaque mois de l'année 2004 le nombre de vols réalisés et le nombre de passagers transportés.

Question 22

Quel est le salaire moyen des pilotes ?

Question 23

Établir la liste des pilotes dont le salaire est supérieur au salaire moyen. Afficher le nom du pilote et son salaire.

Question 24

Indiquer le nombre d'avions par type d'appareil. Afficher le type d'appareil et le nombre d'avions.

Question 25

Indiquer les types d'appareils dont la compagnie possède au moins deux exemplaires. Afficher le type d'appareil et le nombre d'avions.

Question 26

Pour chaque avion en service, indiquer le nombre de pilotes distincts qui le conduisent. Afficher le numéro d'avion et le nombre de pilotes.

Question 27

Pour chaque pilote en service, indiquer le nombre de vols assurés. Afficher le numéro du pilote, son nom et le nombre de vols.

Question 28

Établir la liste des pilotes qui assurent plus d'un vol au départ de LYON. Afficher uniquement le numéro de pilote.

Question 29

Établir la capacité moyenne des avions par ville de destination. Afficher la ville de destination et la capacité moyenne. Cette dernière colonne sera renommée ‘capacité_moyenne'.

Question 30

Pour chaque pilote en service, donner la liste des vols qu'ils assurent. Si un pilote n'est pas en service, indiquer ‘****' à la place du numéro de vol. Afficher le numéro du pilote, son nom et le numéro de vol.

Partie 2 – Gestion des comptes utilisateurs (6 questions)

Question 31

Créer les utilisateurs mentionnés ci-dessus en prenant soin de faire précéder leur nom de votre nom de login et en leur affectant un mot de passe.

Question 32

Donner aux différents utilisateurs les droits nécessaires en fonction de leurs besoins.

Question 33

Se connecter à la base sous leur identité et vérifier s'ils peuvent effectuer les manipulations requises.

Question 34

4. Modifier le mot de passe de Michel.

Question 35

5. Retirer les droits de Michel et Joel.

Créer un groupe pour les responsables de la programmation des vols : RESP_VOL et y affecter Michel et Joel.

Attribuer les droits requis au groupe.

Question 36

Se reconnecter en tant qu'un des deux responsables des vols et vérifier.