SQL

SQL pour communiquer avec un SGBDR

SQL : Structured Query Language

- Permet de communiquer avec un SGBDR

- Présente les avantages suivant :
  - langage efficace pour la communication avec une SGDBR
  - facile à apprendre et à utiliser
  - dispose de fonctionnalités complètes : définir, extraire et manipuler les données des tables

Objets manipulés par SQL

Les identificateurs

- SQL utilise des identificateurs pour désigner les objets qu'il manipule : utilisateurs, tables, colonnes, …
- Un identificateur est un mot formé d'au plus 30 caractères, commençant obligatoirement par une lettre de l'alphabet. Les caractères suivants peuvent être une lettre, un chiffre, ou l'un des symboles # $ et _

- SQL ne fait pas la différence entre les lettres minuscules et majuscules (sauf après la condition where attribut=valeur , voir TP)
- Les voyelles accentuées ne sont pas acceptées.

Objets manipulés par SQL

Les identificateurs

- Un identificateur ne doit pas figurer dans la liste des mot clés réservés

Quelques mot-clé réservés :
DATE, TABLE, INDEX
DECIMAL, DEFINITION, FILE, FORMAT, INDEX, LIST, MODE, OPTION, PARTITION, PRIVILEGES, PUBLIC, REF, REFERENCES, SELECT, SEQUENCE, SESSION, SET, TABLE, TYPE.

Objets manipulés par SQL

Les Tables

- Les relations (d’un schéma relationnel) sont stockées sous forme de tables composées de lignes et de colonnes.

Objets manipulés par SQL

Les Tables

Il est d'usage (mais non obligatoire) de mettre les noms de table au singulier : plutôt employe que employes pour une table d'employés.

Exemple : Table deptement des départements

deptement

numdept	nomd	ieu
10	Recherche	Rennes
20	vente	Metz
30	direction	Gif
40	fabrication	Reims

Objets manipulés par SQL

Les colonnes

- Les données contenues dans une colonne doivent être toutes d'un même type de données. Ce type est indiqué au moment de la création de la table qui contient la colonne

- Types de données :
  - caractères : char, varchar, …
  - valeurs numérique NUMBER : NUMBER(taille_maxi) ,

  - date/ heure : DATE

Instructions SQL

Interrogation de données :

SELECT	:	Extraction de données

Langage de manipulation de (LMD) :

INSERT : insertion
UPDATE : mise à jour
DELET : suppression de lignes

Langage de définition de données (LDD) :

CREATE : création d'un objet (table, …)
ALTER : modification d'un objet
DROP : supprimer un objet
RENAME : : renommer un objet

Interrogation de données

Interrogation de données

- Toute interrogation se fait par la commande « SELECT »
- Commande déclarative : décrit ce que l'on cherche sans décrire le moyen de le réaliser

Interrogation de données

Forme complète de la commande SELECT

- l'instruction SELECT doit inclure :
  - *) une clause SELECT qui détermine les colonnes à afficher
  - *) une clause FROM : détermine la table contenant les colonnes répertoriées dans la clause SELECT

```
SELECT [DISTINCT]  {*| column [alias] | expression1 [as alias] …}
FROM table1
[WHERE condition (s) ]
[GROUP BY expression]
[HAVING condition]
[ORDER BY {column, expr, alias} [ASC|DESC]]
[{UNION|INTERSECT|MINUS} (sous requête)] ;
```

Notes :

| : choix entre différentes options

{} : choix obligatoire

[] : facultatif

Exemple d'interrogation

Exemple d'interrogation simple : extraction de certains colonnes (projection)

>> Soit la relation : depatement(dept, nom, lieu)

numdept	nomd	ieu
10	Recherche	Rennes
20	vente	Metz
30	direction	Gif
40	fabrication	Reims

Nom de la colonnes de la projection

select nomd, lieu from departement

Nom de la table

Nomd	lieu
Direction	Gif
Recherche	Rennes
Fabrication	Reims
Vente	Metz

Interrogation de données

● Le nom complet d'une colonne d'une table est le nom de la table suivi d'un point et du nom de la colonne.
Par exemple : employe.nom

```
SELECT departement.nomd, departement.lieu FROM departement;
                    =
SELECT nomd, lieu FROM departement;
```

● Le nom de la table peut être omis quand il n'y a pas d'ambiguïté. Il doit être précisé s'il y a une ambiguïté, ce qui peut arriver quand on fait une sélection sur plusieurs tables à la fois et que celles-ci contiennent des colonnes qui ont le même nom

```
SELECT table1.nom,  table2.nom FROM table1,table2;
```

Extraction de toutes les colonnes

- Utilisation de * :
  - *) pas de projection
  - *) sélection de toute la table

SELECT * FROM Departement ;

Toutes les colonnes

Nom de la table

numdept	nomd	lieu
10	Recherche	Rennes
20	vente	Metz
30	direction	Gif
40	fabrication	Reims

Le contenu de toute la table

Alias

utilisation d'alias :

- permet de renommer des colonnes à l'affichage ou des tables dans la requête

== > alias colonne et alias table

Alias d'une colonne (1)

alias colonne :

- SELECT permet d'indiquer quelles colonnes, ou quelles expressions doivent être retournées par l'interrogation

SELECT [DISTINCT] *
ou
SELECT [DISTINCT] exp1 [[AS] nom1], exp2 [[AS] nom2], ...
- exp1, exp2, ... sont des expressions,

- nom1, nom2, ... sont des alias : des noms facultatifs de 30 caractères maximum, donnés aux expressions. Chacun de ces noms est inséré derrière l'expression, séparé de cette dernière par un blanc ou par le mot clé AS (optionnel) ; il constituera le titre de la colonne dans l'affichage du résultat de la sélection

Alias d’une colonne (2)

Exemple alias colonne :

Soit la relation : employe(nom,num,fonction,n_sup,embauche,salaire,comm,dept)

salaire mensuelle de chaque salarié :

```
SELECT nom  AS nomPersonne, sal*12 AS Salaire_Annuel FROM employe
```

Alias d’une colonne (3)

alias colonne :

Remarque : Si le nom contient des séparateurs (espace, caractère spécial), ou s'il est identique à un mot réservé SQL (exemple : DATE), il doit être mis entre guillemets

Exemple :
SELECT nom, sal*12 AS ”Salaire Annuel” FROM employe ;

Alias des tables

alias Table : si on définit l'alias d'une table , Il faut préfixer les colonnes par l'alias de la table

Exemple:

```
SELECT emp.nom FROM employe emp;
```

Duplicatas

Duplicatas :

la directive DESTINCT élimine les éventuelle duplicatas

Duplicatas

Table Pilote

brevet	nom	nbrHvol	prime	embauche	typeAvion	compa
PL-1	Gratien Viel	450	500	05/02/1965	A320	AF
PL-2	Didier Donsez	0		13/05/1965	A320	AF
PL-3	Richard Grin	1000		11/09/2001	A320	SING
PL-4	Placide Fresnais	2450	500	21/09/2001	1330	SING
PL-5	Daniel Vielle	400	600	16/01/1965	A340	AF
PL-6	Françoise Tort		0	24/12/2000	A340	CAST

SELECT compa FROM Pilote

Compa

AF
AF
SING
SING
AF
CAST

Avec duplicatas

SELECT DISTINCT compa FROM Pilote

Compa

AF
SING
AF
CAST

Sans duplicatas

Expressions arithmétiques

- Des expressions sur des données de type NUMBER et DATE

Opérateur	description	priorité
*	multiplication	1
/	division	1
+	addition	2
-	soustraction	2

- Les opérateurs présentant la même priorité sont évalués de gauche à droite
- Les parenthèses sont utilisées pour forcer des priorités et pour améliorer la clarté

Exercice : Simuler une augmentation de 10% des salaires des employés (utiliser la table emp)

== > SELECT sal * 1.10 augmentation FROM pilote;

Restriction : Instruction WHERE

➔WHERE prédicat :

SELECT FROM <nom de table>

WHERE <predicat> ;

Exemple : donnez la liste des employés dont le salaire est > 40000

Table emp

num	nom	fonction	num_sup	embauche	sal	comm	numdept
16712	MARTIN	directeur	25717	23-05-90	40000		30
17574	DUPONT	administratif	16712	03-05-95	9000		30
26691	DUPOND	commercial	27047	04-04-88	25000	2500	20
25012	LAMBERT	administratif	27047	14-04-91	12000		20
25717	JOUBERT	president		10-1à-82	50000		30
15155	GARDARIN	ingenieur	24533	22-03-85	24000		10
16034	LEBRETON	commercial	27047	01-01-91	15000	0	20
17147	MARTIN	commercial	27047	10-12-93	20000	500	20
27546	PAQUEL	commercial	27047	03-09-83	22000	2000	20
25935	LEFEBVRE	commercial	27047	11-01-84	23500	1500	20
15155	GARDARIN	ingenieur	24533	22-03-85	24000		10
26834	SIMON	ingenieur	24533	04-09-88	20000		10
16278	DELOBEL	ingenieur	24533	16-11-94	21000		10
25067	ADIBA	ingenieur	24533	05-11-87	30000		10
24533	CODD	directeur	25717	12-09-75	55000		10
27047	LAMERE	directeur	25717	07-09-89	45000		20
17232	BALIN	administratif	24533	03-10-87	13500		10
24831	BARA	administratif	16712	10-10-88	15000		30

== > SELECT nom FROM emp

WHERE sal >40000

Jaubert
Cdd
Lamer

Restriction : Instruction WHERE

- Instruction WHERE : prédicats

  - Un prédicat simple est la comparaison de deux expressions ou plus au moyen d'un opérateur :

```
WHERE exp1 = exp2
WHERE exp1 != exp2
WHERE exp1 < exp2
WHERE exp1 > exp2
WHERE exp1 <= exp2
WHERE exp1 >= exp2
WHERE exp1 BETWEEN exp2 AND exp3
WHERE exp1 LIKE exp2
WHERE exp1 NOT LIKE exp2
WHERE exp1 IN (exp2, exp3,...)
WHERE exp1 NOT IN (exp2, exp3,...)
WHERE exp IS NULL
WHERE exp IS NOT NULL
```

Restriction : Instruction WHERE

- Opérateur LIKE : exp1 LIKE exp2

  teste l'égalité de deux chaînes en tenant compte des caractères jokers dans la 2ème chaîne :

  _ remplace 1 caractère exactement

  % remplace une chaîne de caractères de longueur quelconque

Exemple :

- les noms des employés dont la première lettre est c :

  SELECT nom FROM employe where nom LIKE ’c%’;

- les noms des employés qui contiennent la chaîne de caractère at :

  SELECT nom FROM employe WHERE nom LIKE ‘%at%’;

Restriction : Instruction WHERE

Exemple :
soit la relation
Emp(empno,Ename job,embauche,sal,comm,#deptno)

- afficher le nom de tous les employés dont la deuxième lettre du nom est un ‘a’ et qui ont été embauchés en 1987
- afficher le nom et date d’embauche de tous les employés qui ont été embauché depuis 2000.

Restriction : Instruction WHERE

- IN : est l'opérateur qui permet de tester l'appartenance de la valeur d'une colonne à une liste

Exemples :

Liste des vols dont la ville d'arrivée est Nice ou Paris.

```
SELECT numvol
 FROM vol
WHERE va IN ('Nice ', 'Paris');
```

- BETWEEN : est l'opérateur qui permet de tester si une valeur appartient à n intervalle (les bornes sont incluses)

➔ SELECT … WHERE exp1 BETWEEN exp2 AND exp3

Exemple : Salaire et nom des employés gagnant entre 15000 et 18000

Restriction : Instruction WHERE

- **Opérateur SQL de négation**

  Dans certain cas il est plus simple de rechercher les lignes qui ne satisfont pas une condition ==> utilisation les opérateurs SQL avec une expression de négation

Exemple :

… WHERE dep !=75;

… WHERE job NOT LIKE ‘C%’;

… WHERE commission IS NOT NULL

… WHERE ville IS NOT IN ('Nice ', 'Paris');

- IS NULL et IS NOT NULL sont les opérateurs qui permettent de tester si une valeur a été définie ou pas pour une colonne.

NULL = non défini

Restriction : Instruction WHERE

- **Opérateurs logiques : AND et OR**

  Peuvent être utilisés pour combiner plusieurs prédicats

  L'opérateur AND est prioritaire par rapport à l'opérateur OR

  L'opérateur NOT placé devant un prédicat en inverse le sens.

➔ SELECT … WHERE condition1 AND/OR condition2

Exemple :

- Sélectionner les employés du département 30 ayant un salaire supérieur à 1500 :

```
SELECT nom FROM emp WHERE dep=30 AND Sal >1500;
```

- Noms des directeur et des administratif du départementr 30

```
SELECT nom, num, fonction, n_dept FROM emp
WHERE (fonction = 'directeur' OR fonction ='administratif')
        AND n_dept = 30 ;
```

- Priorté : AND > OR ➔ utiliser les paranthéses pour forcer la priorité et pour plus de clareté

Restriction : Instruction WHERE

- Définir une valeur NULL

  - Une valeur NULL est une valeur non disponible, non attribuée
  - Une valeur NULL est différente d'un zéro ou d'un espace
  - la valeur d'une ligne est absente pour une colonne donnée, cette valeur est considéré comme NULL
  - Les colonnes, quel que soit leur type de données peuvent contenir des valeur NULL, excepté lorsque la colonne a été définie comme NOT NULL ou comme PRIAMARY KEY lors de sa création
  - **Attention :**

    NULL ≠ 0 ➔ « expr = NULL » : toujours FAUX !

    Tester avec : IS NULL et IS NOT NULL

Restriction : Instruction WHERE

- Définir une valeur NULL

- Exemple :

Soit la relation :

Emp(empno,Ename job,embauche,sal,comm,#deptno,tel)

Donnez la liste de employés dont le numéro de téléphone n'est pas renseignés :

```
== > SELECT nom
        FROM emp
          WHERE  tel IS NULL ;
```

Exercice : afficher la liste des employés qui n'ont pas de commission

Ordonnancement : *ORDER BY*

- *Clause ORDER BY : Classement des résultats d'un SELECT*

  Précise l'ordre dans lequel la liste des lignes sélectionnées sera donnée.

Expression générale :

```
SELECT col1, col2, fonction1Groupe, …
FROM table (s)
[WHERE condition (s) ]
[GROUP BY cole1[,col2, …]
[HAVING condition]
[ORDER BY {column | position| alias} [ASC|DESC]]
[{UNION|INTERSECT|MINUS} (sous requête)] ;
```

Ordonnancement : *ORDER BY*

Exemple : Donner pour chaque fonction la liste des noms des employés

```
SELECT ename, job
FROM emp
ORDER BY job
      == > trie selon la fonction
```

Ename	job
BARA	administratif
BALIN	administratif
LAMBERT	administratif
DUPONT	administratif
DUPOND	commercial
LEFEBVRE	commercial
PAQUEL	commercial
MARTIN	commercial
LEBRETON	commercial
…	

Ordonnancement : *ORDER BY*

- on peut lui associer :
  - ASC : ordre croissant (par défaut)
  - DESC : ordre décroissant

Ordonnancement : *ORDER BY*

Exemple : Donner pour chaque fonction la liste des noms des employés ainsi que leurs salaire trié dans l'ordre décroissant

```
SELECT ename, job, sal
FROM emp
ORDER BY job, sal  DESC
```

Ename	job	sal
BARA	administratif	15000
BALIN	administratif	13500
LAMBERT	administratif	12000
DUPONT	administratif	9000
DUPOND	commercial	25000
LEFEBVRE	commercial	23500
PAQUEL	commercial	22000
MARTIN	commercial	20000
LEBRETON	commercial	15000
…		

Ordonnancement : *ORDER BY*

```
SQL> select ename, job, sal from scott.emp order by job, sal desc;

ENAME      JOB              SAL
---------- --------- ----------
FORD       ANALYST         3000
SCOTT      ANALYST         3000
SMITH      CLERK           5000
MILLER     CLERK           1300
ADAMS      CLERK           1100
SMITH2     CLERK           1000
JAMES      CLERK            950
tito       hob             1500
JONES      MANAGER         2975
BLAKE      MANAGER         2850
CLARK      MANAGER         2450

ENAME      JOB              SAL
---------- --------- ----------
KING       PRESIDENT       5000
ALLEN      SALESMAN        1600
TURNER     SALESMAN        1500
MARTIN     SALESMAN        1250
WARD       SALESMAN        1250
```

Ordonnancement : *ORDER BY*

```
SQL> select ename, job, sal from scott.emp order by sal desc, job;

ENAME      JOB              SAL
---------- --------- ----------
SMITH      CLERK           5000
KING       PRESIDENT       5000
FORD       ANALYST         3000
SCOTT      ANALYST         3000
JONES      MANAGER         2975
BLAKE      MANAGER         2850
CLARK      MANAGER         2450
ALLEN      SALESMAN        1600
tito       hob             1500
TURNER     SALESMAN        1500
MILLER     CLERK           1300

ENAME      JOB              SAL
---------- --------- ----------
MARTIN     SALESMAN        1250
WARD       SALESMAN        1250
ADAMS      CLERK           1100
SMITH2     CLERK           1000
JAMES      CLERK            950
```

Ordonnancement : *ORDER BY*

Exercice :

Donnez pour chaque département (deptno) la liste des noms des employés ainsi que leurs salaire trié dans l'ordre décroissant

Rappel :

Emp(empno,Ename, job,embauche,sal,comm,#deptno)

Ordonnancement : *ORDER BY*

- Possibilité de faire le trie en fonction de la position : utile lors d'un tri en fonction d'une expression longue

SELECT nom, date_emb, 12*sal FROM emp ORDER BY 3;

== > trie les résultats en fonction du troisième élément de l'instruction SELECT

Fonctions de groupes

Les fonctions de groupes peuvent apparaître dans le Select ou le Having. Ci-dessous quelques fonctions usuelles:

AVG moyenne
SUM somme
MIN plus petite des valeurs
MAX plus grande des valeurs
VARIANCE variance
STDDEV écart type (déviation standard)
COUNT(*) nombre de lignes
COUNT(col) nombre de valeurs non nulles de la colonne
COUNT(DISTINCT col) nombre de valeurs non nulles différentes

Fonctions de groupes

-**Fonctions de groupe :** Les fonctions de groupe permettent de faire des calculs sur l'ensemble des valeurs d'une colonne

Exemple :

```
SELECT SUM(sal)
FROM employe
WHERE dep = 10
```

→ Renvoie le total des salaires du département 10

```
SELECT nom, fonction, salaire
FROM employe
WHERE salaire = (SELECT
MAX(salaire) FROM emp) ;
```

→ Renvoie le nom, la fonction et le salaire de l'employé (ou des employés) ayant le salaire le plus élevé

- **Cas des valeurs « NULL » :** non prises en compte, exceptées par COUNT(*)

Fonctions de groupes

Exercice : soit la table Pilote ci-dessous

Table Pilote

brevet	nom	nbrHvol	prime	embauche	typeAvion	compa
PL-1	Gratien Viel	450	500	05/02/1965	A320	AF
PL-2	Didier Donsez	0		13/05/1965	A320	AF
PL-3	Richard Grin	1000		11/09/2001	A320	SING
PL-4	Placide Fresnais	2450	500	21/09/2001	1330	SING
PL-5	Daniel Vielle	400	600	16/01/1965	A340	AF
PL-6	Françoise Tort		0	24/12/2000	A340	CAST

Ecrire les requêtes qui permettent de donner :

1- Moyenne des heures de vol et des primes des pilotes de la compagnie ‘AF’

2-Nombre de pilotes, nombre total d’heures effectuées par tous les pilotes, nombre de primes (toutes et distinctes)

3- le nombre d’heures de vol le plus élevé, le nombre d’heure de vol le moins élevé

4- date d’embauche la plus récente, date d’embauche la plus ancienne, la prime la plus faible, la prime la plus élevée, la moyenne des primes,

5 - le salaire le plus bas, le salaire le plus haut, le salaire moyenne

Fonctions de groupes

Solution :

1- Moyenne des heures de vol et des primes des pilote de la compagnie ‘AF’ :

SELECT AVG(nbHVol),AVG(prime) FROM pilote WHERE compa=‘AF’;

Regroupement : GROUP BY … HAVING …

Exemple :

Table Pilote

brevet	nom	nbrHvol	prime	embauche	typeAvion	compa
PL-1	Gratien Viel	450	500	05/02/1965	A320	AF
PL-2	Didier Donsez	0		13/05/1965	A320	AF
PL-3	Richard Grin	1000		11/09/2001	A320	SING
PL-4	Placide Fresnais	2450	500	21/09/2001	1330	SING
PL-5	Daniel Vielle	400	600	16/01/1965	A340	AF
PL-6	Françoise Tort		0	24/12/2000	A340	CAST

En regroupant sur la colonne compa, trois ensemble de lignes (groupement) sont composés ➔ possibilité d'appliquer des fonctions de groupe à chacun de ces ensembles

brevet	nom	nbrHvol	prime	embauche	typeAvion	compa
PL-1	Gratien Viel	450	500	05/02/1965	A320	AF
PL-2	Didier Donsez	0		13/05/1965	A320	AF
PL-5	Daniel Vielle	400	600	16/01/1965	A340	AF
PL-6	Françoise Tort		0	24/12/2000	A340	CAST
PL-3	Richard Grin	1000		11/09/2001	A320	SING
PL-4	Placide Fresnais	2450	500	21/09/2001	1330	SING

Possibilité de grouper sur plusieurs colonnes : exemple compa et typeAvion pour classifier les pilotes selon ces deux critères

Regroupement : GROUP BY … HAVING …

- *Clause GROUP BY :*
  - constituer des groupements de lignes
  - liste les colonne du groupement
    == > Un SELECT de groupe (avec GROUP BY) ne donnera qu'une ligne par groupe

liste les colonnes du select doivent apparaitre dans el groupe by

Expression générale :

```
SELECT col1 [,col2, …], fonction1Groupe, …
FROM table (s)
[WHERE condition ]
[GROUP BY cole1[,col2, …]
[HAVING condition]
;
```

Permet d'exclure des lignes pour chaque groupement ou de rejeter des groupements entiers. Elle s'aplpique à la totalité de la table

liste les colonnes du groupement.
!! Pas de fct de groupe

Permet de poser des conditions sur chaque groupement

Regroupement : GROUP BY … HAVING …

S'appliquent au groupe par groupe

```
SELECT  SUM(sal) , numdept
FROM employe
GROUP BY numdept ;
```

SUM(sal)	numdept
163500	10
162500	20
114000	30

Un SELECT de groupe (avec GROUP BY) ne donnera qu'une ligne par groupe

Liste les colonnes du groupement

Que des fonctions de groupe ou des expressions du GROUP BY

!! Les colonnes présentant dans le SELECT doivent apparaitre dans Group By . Seules des fonctions ou expressions peuvent exister en plus dans le SELECT

Regroupement : GROUP BY … HAVING …

- Utilisation de la clause GROUP BY expr1, expr2, … :
  Parmi les lignes ayant même valeur pour expr1, on regroupe celle ayant même valeur pour expr2, …

Exemple :
soit la relation Pilot(brevet,nom,nbrHvol,prime,embauche,typeAvion,compa)

Donner le nombre de pilote par compagnie et par type d'appareil

== > *SELECT compa, typeAvion,COUNT(brevet) FROM Pilote GROUP BY compa, typeAvion*;

compa	typeAvion	COUNT(brevet)
AF	A320	2
AF	A340	1
CAST	A340	1
SING	A320	1
SING	A330	1

GROUP BY : Exemple

Exemple : le nombre d'employés par département et par fonction (job).

SELECT deptno, job, COUNT(*) FROM emp
GROUP BY deptno, job

```
SQL> SELECT deptno, job, COUNT(*) FROM emp
  2  GROUP BY deptno,job;

    DEPTNO JOB                              COUNT(*)
---------- ------------------------------ ----------
        20 MANAGER                                 1
        10 COMMERCIAL                              1
        10 President                               1
        30 MANAGER                                 1
        30 COMMERCIAL                              1
        20 COMMERCIAL                              2
        10 MANAGER                                 1
        30 Ingenieur                               1
        20 ANALYST                                 2
        30 INGENIEUR                               3
```

Trie aléatoire

GROUP BY : Exemple

Exemple : le nombre d'employés par département et par fonction (job).

SELECT deptno, job, COUNT(*) FROM emp
GROUP BY deptno, job ORDER BY deptno;

Trie du résultat :

```
SQL> SELECT deptno, job, COUNT(*) FROM emp
  2  GROUP BY deptno,job
  3  order by deptno;

    DEPTNO JOB                              COUNT(*)
---------- ------------------------------ ----------
        10 COMMERCIAL                              1
        10 MANAGER                                 1
        10 President                               1
        20 ANALYST                                 2
        20 COMMERCIAL                              2
        20 MANAGER                                 1
        30 COMMERCIAL                              1
        30 INGENIEUR                               3
        30 Ingenieur                               1
        30 MANAGER                                 1
```

Regroupement : GROUP BY … HAVING …

- Utilisationn de HAVING:

- HAVING joue le rôle de filtre pour la clause GROUP BY, mais ne porte que sur des caractéristiques de groupe :
  - *) fonction de groupe
  - *) expressions présentes dans le GROUP BY

Exemple : Liste des salaires moyens par fonction (job) ayant plus de deux employés.

```
SELECT fonction,COUNT(*), AVG(sal)
FROM employe
GROUP BY fonction
HAVING COUNT(*) >= 2;
```

Regroupement : GROUP BY … HAVING …

- soit la relation :

Pilote(brevet,nom,nbHvol,typeavion,compa)

Ecrire les requêtes répondant aux questions suivantes :

1- Moyenne des heures de vol et des primes par compagnie

2- Nombre de pilote par compagnie

3- Nombre d'heures de vol le plus élevé, date d'embauche la plus récente par compagnie

4- somme des heures de vol des pilotes volant sur A320 par compagnie

5- Nombre de pilotes qualifié (nombre de brevet) par compagnie et par type d'appareil

6- le nombre de pilotes par compagnie ayant plus d'un pilote

Regroupement : GROUP BY … HAVING …

-Solution :

1- Moyenne des heures de vol et des primes pour chaque compagnie :

```
SELECT compa, AVG(nbHvol), AVG(prime) FROM Pilote
GROUP By compa;
```

Compa	AVG(nbHvol)	AVG(prime)
AF	283,33	550
CAST		0
SING	1725	500

2- Nombre de pilote par compagnie :

```
SELECT compa, count(*) From Pilote
GROUP By compa;
```

Compa	COUNT(*)
AF	3
CAST	1
SING	2

Regroupement : GROUP BY … HAVING …

3- Nombre d'heures de vol le plus élevé, date d'embauche la plus récente par compagnie:

```
 SELECT compa, MAX(nbHvol), MAX(embauche) as 'embauche recent'
 FROM Pilote
GROUP By compa;
```

Compa	MAX(nbHvol)	date embauche recente
AF	450	13/O5/95
CAST		24/12/00
SING	2450	21/09/01

4- somme des heures de vol des pilotes volant sur A320 par compagnie:

```
 SELECT compa,  SUM(nbHvol) Frome Pilote
 WHERE typAvion='A320'
GROUP By compa;
```

Compa	SUM(NBHVOL)
AF	450
SING	1000

Regroupement : GROUP BY … HAVING …

5- Nombre de pilotes qualifié (nombre de brevet) par compagnie et par type d'appareil :

```
 SELECT compa, typAvion, count(brevet) FROM Pilote
GROUP By compa, typeAvion;
```

Compa	typeAvion	count(brevet)
AF	A320	2
AF	A340	1
CAST	A340	1
SING	A320	1
SING	A330	1

6- le nombre de pilotes par compagnie ayant plus d'un pilote:

```
 SELECT compa, count(brevet) Frome Pilote
GROUP By compa
HAVING count(brevet)>=2;
```

Compa	COUNT(brevet)
AF	3
SING	2

Jointure

Les jointures permettent d'extraire les données issues de plusieurs tables
== > met en relation deux tables sur la base d'une condition de jointure (comparaison de colonne)

➔ Tout simplement en spécifiant plusieurs tables après le FROM

- Une condition de jointure exprime une relation existant entre les données d'une colonne d'une table et de celle d'une autre colonne d'une autre table
== > généralement fait intervenir une clé étrangère d'une table avec une clé primaire d'une autre table

Jointure

Exemple de jointure :

- La clé secondaire de la table emp correspond à la clé primaire de dep

Table emp

num	nom	fonction	num_sup	embauche	sal	comm	numdept
16712	MARTIN	directeur	25717	23-MAY-90	40000		30
17574	DUPONT	administratif	16712	03-MAY-95	9000		30
26691	DUPOND	commercial	27047	04-APR-88	25000	2500	20
25012	LAMBERT	administratif	27047	14-APR-91	12000		20
…							

Table dept

numdept	Nomd	lieu
10	Recherche	Rennes
20	vente	Metz
30	direction	Gif
40	fabrication	Reims

Exemple : liste des employés et de leur lieux de travail

```
SELECT emp.nom, lieu
FROM emp, dept
WHERE emp.numdept = dept.numdept ;
```

Jointure entre tables « emp » et « dept »

Jointure

- Equi-Jointure entre deux tables : jointure avec condition d'égalité

```
SELECT nom, lieu
FROM emp, dept
WHERE emp.numdept = dept.numdept ;
```

« num_dept » est un nom de colonne commun aux deux tables : il faut préfixer par le nom de la table

Nom	lieu
MARTIN	Gif
DUPONT	Gif
DUPOND	Metz
LAMBERT	Metz

…

Pas d'affichage de Reims

Equi-Jointure utilise l'opérateur d'égalité dans la clause de jointure et compare généralement des clés primaire avec des clés étrangères

Jointure

- Auto-jointure : cas particulier de l'équijointure
  Relie une table à elle-même
  - Pourquoi ? : pour rassembler des informations venant de 2 lignes différentes d'une même table
  - Exemple : noms des employés et noms de leurs managers

```
SELECT emp.nom, mgr.nom

FROM emp,  emp mgr

WHERE emp.num_sup = mgr.num
```

Besoin de renommer la table pour indiquer de quelle ligne vient la colonne citée

Jointure

- Jointure externe :

On rappel les deux tables

Table dept

numdept	nomd	ieu
10	Recherche	Rennes
20	vente	Metz
30	direction	Gif
40	fabrication	Reims

Table emp

num	nom	fonction	num_sup	embauche	sal	comm	numdept
16712	MARTIN	directeur	25717	23-05-90	40000		30
17574	DUPONT	administratif	16712	03-05-95	9000		30
26691	DUPOND	commercial	27047	04-04-88	25000	2500	20
25012	LAMBERT	administratif	27047	14-04-91	12000		20
25717	JOUBERT	president		10-1à-82	50000		30
15155	GARDARIN	ingenieur	24533	22-03-85	24000		10
16034	LEBRETON	commercial	27047	01-01-91	15000	0	20
17147	MARTIN	commercial	27047	10-12-93	20000	500	20
27546	PAQUEL	commercial	27047	03-09-83	22000	2000	20
25935	LEFEBVRE	commercial	27047	11-01-84	23500	1500	20
15155	GARDARIN	ingenieur	24533	22-03-85	24000		10
26834	SIMON	ingenieur	24533	04-09-88	20000		10
16278	DELOBEL	ingenieur	24533	16-11-94	21000		10
25067	ADIBA	ingenieur	24533	05-11-87	30000		10
24533	CODD	directeur	25717	12-09-75	55000		10
27047	LAMERE	directeur	25717	07-09-89	45000		20
17232	BALIN	administratif	24533	03-10-87	13500		10
24831	BARA	administratif	16712	10-10-88	15000		30

Jointure

- Jointure externe :

permettent d'extraire des enregistrements qui ne répondent pas aux critères de jointure

le lieu de travail des employés :

```
SELECT emp.nom, lieu
FROM emp, dept
WHERE emp.numdept =
dept.numdept ;
```

Sans jointure externe le site de Reims ne figure pas dans la réponse (car aucun employé n'y travaille!)

BARA	Gif
MARTIN	Gif
DUPONT	Gif
DUPOND	Metz
….	
BARA	Gif

Avec la jointure externe il sera ajouté :

BARA	Gif
MARTIN	Gif
DUPONT	Gif
DUPOND	Metz
….	
BARA	Gif
	Reims

Jointure

Jointure externe :

- Il s'agit de prendre en compte les lignes d'une table n'ayant pas eu de correspondance dans les autres tables.
- Par défaut ces lignes sont éliminés de la jointure.
- Avec une jointure « externe » elles seront ajoutées au résultat :

```
SELECT emp.nom, lieu
FROM emp,dept
WHERE emp.numdept (+) = dept.numdept ;
```

On ajoute « (+) » au nom de la colonne dont les valeurs ne couvrent pas assez de champ (là où il « manque » des valeurs).

BARA	Gif
MARTIN	Gif
DUPONT	Gif
DUPOND	Metz
….	
BARA	Gif
	Reims

Jointure

Exercice :

Soient les deux relation :

Pilote(brevet,nom,nbHvol,typeavion,#compa)

Compagnie(comp,adresse, nomComp)

Qualifs(brevet,typeAvion,validité)

1- Liste des pilotes et les noms de leurs compagnies même les compagnies n'ayant pas de pilote

2-Liste des pilotes et le numéro de leurs brevet même les pilotes n'ayant pas encore leurs qualification

Jointure

- inéqui-jointure :

Les requêtes d'inéqui-jointure font intervenir tout type d'opérateur (<>,>,<,>=,<=,BETWEEN,LIKE,IN). A l'inverse de l'équi-jointure, la clause d'une équijointure n'est pas basée sur l'égalité des clés primaires et les clés étrangères

- Exemple 1 :

Soit : Pilote(brevet,nom,nbHvol,typeavion,#compa)

- les noms et numéros des brevets des pilotes ayant plus d'expérience (celui qui a comptabilisé plus d'heur de vol) que le pilote de numéro de brevet 'PL-2' :

```
SELECT p1.nom,p1.brevet FROM Pilote p1, Pilote p2
WHERE p1.nbHvol>p2.nbHvol AND p2.brevet='PL-2';
```

- Exemple 2 : les noms, salaires et fonctions des employés gagnant plus que l'employé SIMON

```
SELECT emp.nom, emp.salaire, emp.fonction
FROM emp, emp j
WHERE emp.salaire > j.salaire   AND      j.nom ='SIMON' ;
```

Opérateurs ensemblistes

- But : combiner dans un résultat unique des lignes venant d'interrogations différentes
- Opérateurs ensemblistes possibles :

  Union (UNION et UNION ALL), Intersection (INTERSECT) et différence relationnelle (MINUS)

Syntaxe :

SELECT … FROM nomTable [WHERE …]
{UNION | INTERSECT | MINUS}
SELECT … FROM nomTable [WHERE …]

- ! Après projection et avant combinaison, les lignes doivent avoir rigoureusement le même nombre de colonnes avec des types identiques

Opérateurs ensemblistes

Exemple :

Soit la relation :
Avion(immatruculation,typeAvion,compa,PrixAchat)

- **Les types d'avions communs que les deux compagnies AF et SING utilisent**

SELECT typeAvion From avion where compa ='AF'
INTERSECT
SELECT typeAvion From avion where compa ='SING'

== > typeAvion

A320
A340

Table Avion

immatriculation	typeAvion	compa	PrixAchat
F-WTS	A320	AF	104500
F-GTMP	A320	AF	104500
F-GTMP	A320	SING	198000
A-TNP	1330	SING	204500
F-WXF	A340	AF	120000
F-ITR	A340	CAST	104500
F-LAV	Concord	AF	15600
F-BIO	A340	SING	198000
F-INR	A330	SING	204500

Opérateurs ensemblistes

- **Tous les types d'avions que les deux compagnies AF et SING exploietent**

```
SELECT typeAvion From avion where compa ='AF'
UNION
SELECT typeAvion From avion where compa ='SING'
```

== > typeAvion

```
--------------
A320
A340
A330
concorde
1330
```

PS : on peut mettre UNION ALL = > on aura des doublons

- **Les avions exploités par la compagnies AF mais pas par SING**

```
SELECT typeAvion From avion where compa ='AF'
MINUS
SELECT typeAvion From avion where compa ='SING'
```

== > typeAvion

```
--------------
Concorde
```

sous-interrogations

- Un critère de recherche employé dans une clause WHERE peut être lui-même le résultat d'un SELECT

Exemple 1: sous-interrogation simple ramenant une seule valeur

```
SELECT nom
FROM emp
WHERE fonction=(SELECT job
FROM emp
WHERE nom ='toto');
```

Renvoi le nom de salariés ayant même job que toto

Exemple 2:

```
SELECT nom
FROM emp
WHERE (fonction, numdep)=(SELECT fonction, numdep FROM emp WHERE
nom='toto');
```

Renvoi le nom des salariés ayant même fonction que toto et appartenant au même département que toto

L’opérateur EXISTS

- Utilisé dans une requête principale
- Il est suivi d’une sous requête entre parenthèse :

  EXISTS (sous requête);

- Permet de tester si la sous requête renvoi au moins un enregistrement ou non :

  renvoi un vrai si oui un faux sinon

L'opérateur EXISTS

Soit la relation :

Emp(empno,Ename job,embauche,sal,comm,#deptno,tel)

Dept(deptno,nomD)

Les noms des départements qui ont au moins un employé ayant comme salaire plus de 1000

```
SELECT nomd FROM Dept
WHERE EXISTS ( SELECT * FROM Emp
               WHERE  sal>1000 and
               Emp.numpdet=Dept.numdept
             );
```

Sous-requête corrélée == > sous-requête est exécuté pour chaque tuple de la table Dept

L’opérateur NOT EXISTS

- Utilisé dans une requête principale
- Il est suivi d’une sous requête entre parenthèse :

  NOT EXISTS (sous requête);

- Permet de tester si la sous requête renvoi au moins unenregistrement ou non :

  renvoi un vrai si non faux si oui

L'opérateur NOT EXISTS

Soit la relation :

Emp(empno,Ename job,embauche,sal,comm,#deptno,tel)

Dept(deptno,nomD)

Les noms des départements qui n'ont aucun employé ayant comme salaire plus de 1000

```
SELECT nomD FROM Dept
WHERE NOT EXISTS ( SELECT * FROM Emp
                   WHERE  sal>1000 and
                   Emp.numpdet=Dept.numdept
                 );
```

Sous-requête corrélée == > sous-requête exécutée pour chaque tuple de la table Dept

À éviter si possible

L'opératuer NOT EXISTS

- Utilisation parfois incontournable
- Utiliser pour exprimer la division : permet de traduire le terme « tous les »

Exemple :

Etudiant(numE,nom prénom)
UV(numUV,intitulé)
Inscrit(numE,numE)

Donner les noms étudiants inscrits dans tous les UV :
== > équivalent à : Les noms des étudiants pour lesquels il n'existe pas d'UV dans lequel ils ne soient pas inscrits :

```
SELECT nom FROM Etudiant E
WHERE NOT EXISTS(  SLECT * FROM UV
                   WHERE NOT EXISTS(  SELECT * FROM Inscrit I
                                      WHERE  I.numE=E.numE
                                      and I.numV=UV.numUV
                                   )
                 );
```

définition des données : le Langage de Définition de Données (*LDD*)

Création d'une table

Syntaxe :

```
CREATE TABLE nomTable(
    colonne1 type1  [DEFAULT val1]  [NOT NULL],
    colonne2 type2 [DEFAULT val2]  [NOT NULL],
    …
   CONSTRAINT nomContrainte TypeContrainte
  );
```

nomTable : le nom de la table

Colonne<i> et type<i> : nom de la colonne avec son type (CHAR, VARCHAR, NUMBER,DATE, …)

nomContrainte : nom de la contrainte

typeContrainte : le type de la contrainte (clé primaire, clé étrangère,etc)

DEFAULT : Possibilité de donner une valeur par défaut pour une colonne si la colonne n'est pas renseignée :

Exemple : datemaj DATE DEFAULT CURRENT_DATE,
ville Varchar(30) DEFAULT 'Paris',

Création d'une table

Contraintes :

Objectif : programmer des règles de gestion au niveau des colonnes des tables

➔ alléger les programme client

Syntaxe :

CONSTRAINT nom_de_la_contrainte

- PRIMARY KEY(colonne)
- FOREIGN KEY (colonne) REFERENCES TABLE(colonne)
- CHECK (condition)
- UNIQUE (colonne) => pas de valeurs distinct

Ajout de contrainte :

ALTER TABLE <nomTable> ADD CONSTRAINT nom_contrainte typeContrainte;

Suppression de contrainte (Oracle) :

ALTER TABLE <nomTAble> DROP CONSTRAINT nomContrainte [CASCADE];

Création d'une table

Pilote

brevet	nom	nbrHvol	prime	embauche	typeAvion	compa

compagnie

compa	nrue	rue	ville	nomcompa

```
CREATE TABLE compagnie (
    compa VARCHAR(10),
   nomComp VARCHAR(30) NOT NULL,
   nrue NUMBER(3),
   rue VARCHAR(20),
    ville VARCHAR(15) DEFAULT 'Paris',
    CONSTRAINT pk_Comppagnie PRIMARY KEY(comp),
);
```

```
CREATE TABLE pilote (
    brevet  varchar(6), nom varchar(15) IS NOT NULL, nbHvol NUMBER(7),
compa varchar(4),
    CONSTRAINT pk_pilote PRIMARY KEY(brevet),
   CONSTRAINT  un_nom UNIQUE(nom),
   CONSTRAINT fk_pilote_compa FOREIGN KEY (compa)  REFERENCES
compagnie(compa)
);
```

Supprimer une table

Syntaxe :

DROP TABLE <nomTable>;

impossible de supprimer une table référencée par une contrainte d'intégrité référentielle

Possiblité d'utiliser l'option CASCADE CONSTRAINTS par Oracle :

DROP TABLE <nomTable>

[CASCADE CONSTRAINTS];

L'option `CASCADE CONSTRAINTS` est requise s'il s'agit de la table parent d'une relation de clé étrangère.

Renommer une table

Syntaxe :

ALTER TABLE <nomTable> RENAME To <newname>

Modification structurelles d'une table

● Ajout de colonne :

ALTER TABLE <nomTable> ADD (col1 type1, col2 type2 …);

Exemple :

ALTER TABLE Pilote ADD(tel VARCHAR(10));

● Suppression de colonne :

Syntaxe :

ALTER TABLE <nomTable> DROP COLUMN <nomColonne>;

Exemple :

ALTER TABLE Pilote DROP COLUMN adresse;

PS : La suppression n'est possible que si l colonne n'est pas l'objet d'une contrainte d'intégrité

● Renommer une colonne :

ALTER TABLE <nomTable> RENAME COLUMN <nomColonne> To <newColumnName>;

Exemple :

ALTER TABLE Pilote RENAME COLUMN ville To adresse;

Modification structurelles d'une table

- Modifier le type d'une colonne

Exemple :
- changement de la taille de la colonne compa et change de la contrainte par défaut.

ALTER TABLE Pilote MODIFY compa VARCHAR (6) DEFAULT 'SING' ;
- Rendre possible l'insertion de valeur nulle dans la colonne compa :

ALTER TABLE Pilote MODIFY compa ;

Modification des données

- Modification des données :
  langage de manipulation de données
- En SQL :
  - INSERT, UPDATE, DELETE
  - COMMIT, ROLLBACK : pour valider ou annuler les transactions

Modification des données

- Ajout de lignes : INSERT
- INSERT permet d'insérer une ligne en spécifiant les valeurs :

  Syntaxe :

  INSERT INTO nom_table(nom_col1,nom_col2, ...)
  VALUES (val1,val2...)

- les colonnes ne sont pas précisés , elles sont considérées dans l'ordre de la déclaration de la table

Exemple :

INSERT INTO compa VALUES ('AC',124, 'Champs Elysées', 'Aire France') ;

INSERT INTO emp VALUES (7369, 'DUPON, 'commercial', 27047,17-12-80, 26000, 2500, 20) ;

Modification des données

UPDATE :

- UPDATE permet de modifier les valeurs d'une ou plusieurs colonnes dans une ou plusieurs lignes déjà existantes

- Les nouvelles valeurs peuvent être énoncées par l'utilisateur ou provenir d'un ou plusieurs SELECT

- Syntaxe :

```
UPDATE nom_table
SET nom_col1 = {expression1 | ( SELECT ...) },
 nom_col2 = {expression2 | ( SELECT ...) }
 WHERE predicat;
```

Exemple:

```
UPDATE Employee SET salary = salary * 1.25 WHERE name =
'Bob'
```

DELETE: suppression de ligne

- DELETE permet de supprimer une ou plusieurs lignes d'une table

```
DELETE  FROM nom_table
WHERE predicat ;
```

  - Sans clause WHERE : toutes les lignes sont supprimées

Exemple : supprimer tous les employés travaillant à Rennes

```
DELETE  FROM emp
WHERE emp .numdept = (SELECT emp.numdept FROM dept
    WHERE lieu='Rennes')
```

Doit retourner une valeur et une seule

TRUNCATE:

- TRUNCATE : supprime tous les enregistrements d'une table et libère éventuellement de l'espace de stockage utilisé par la table

Syntaxe :

TRUNCATE TABLE <nomTable];

PS : Il n'est pas possible de tronquer une table qui est référenciée par des clés étrangères actives.

Soit le schéma suivant :

Hopital(codeHopital,nom,adresse,ville)

Laboratoire(codeLabo,nomLabo,#codeHopital)

Service(codeService,nomService,#codeHopital,nombreLits,#Matriculechef)

Medecin(Matricule,nom,adresse,specialite,fonction,#codeLabo,#codeService, #codeHopital,mail)

Patient(numSecu, nom,prenom,adresse,datenaissance,mail)

Consultation(#Matriule, #numSecu,montant, TypePaiment)

- Ecrire, en SQL, les requêtes suivantes :

a- Donner les noms des médecins pour chaque hôpital et pour chaque service.

b- Les noms des médecins de la même spécialité que le Dr. 'Toto'

c- Les noms des services, ainsi que l'hôpital de leurs appartenances, qui ont un nombre de lits supérieur au service de génécologie de l'hôpital de 'Ibn Tofail' de la ville de Marrakech.

d- Les noms des patients qui ont le même nom qu'un médecin qu'ils ont consultés

e- Le nombre de médecins par hôpital et par service

f- Le nombre de médecins par hôpital et par service ayant plus de vingt médecins, trié dans l'ordre alphabétique du nom d'hôpital.