

Les objets de la base de données

Tables

Les tables sont les objets qui contiennent effectivement les données dans la base. Elles peuvent être de **deux types** :

Les tables systèmes

Contiennent les informations permettant le bon fonctionnement de la base de données

Les tables utilisateurs

Contiennent les données des utilisateurs

Vues

Une vue est « une table virtuelle » dont les données ne sont pas stockées dans une table de la base de données, et dans laquelle il est possible de rassembler des informations provenant de plusieurs tables.

Une vue est le résultat d'une requête.

Procédures stockées

Une procédure stockée est un ensemble d'instructions précompilées et exécutées sur demande ou bien encore de façons automatique (déclenché par un trigger).

Trigger

Lorsqu'une action a été exécutée sur une table ou sur le serveur.
Cela déclenche un événement.

Vous pouvez utiliser cet événement pour prendre certaines mesures.

Objets de la base de données

Index

L'index permettra de faire des recherches par dichotomie et d'accélérer les tris sur le champ concerné.

Un index est une table de correspondance où les données seront classées .

Deuxième partie

Le langage SQL

Le langage SQL

PROGRAMME

INTRODUCTION

- Qu'est-ce qu'un SGBDR ?
- **Le SQL**
- Le SQL adapté à Oracle

SQL PLUS : LES BASES

- Structure générale d'une requête
- Les différentes clauses
- Requête à plusieurs relations et sous-requêtes
- Opérations ensemblistes et multi-ensembles

- ❑ Structured Query Language (SQL)
- ❑ Le Langage de Définition de Données DDL
- ❑ Le Langage de Contrôle de Données (DCL)
- ❑ Le Langage de Manipulation de Données (DML)
- ❑ Le Langage de Contrôle de Transaction (TCL)

SYNTAXE SQL Plus

- LA SYNTAXE SQL est composée d'un ensemble de CLAUSES.
- Une instruction SQL commence par l'une des clauses : **SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW,**
- Et se termine par un **“;”**
- SQL n'est pas sensible à la casse.

Le langage SQL

Structured Query Language (SQL)

- Langage de requête structurée.

Comme tout langage qui sera traduit pour être exécuté par la machine, il a un **LEXIQUE**, une **SYNTAXE** et une **SÉMANTIQUE** (et possède un métalangage qu'on appelle aussi grammaire du langage).

- **Requête** : instruction demandant une action sur la base de données.
- SQL: Langage **standardisé** pour **interroger**, **manipuler** et **définir** des données, et pour fournir un contrôle des accès dans les BDDR
- Développé chez IBM fin des années 79 avec la forte participation d'Oracle standard SQL ANSI.

Les ensembles du langage SQL

Il y a 16 commandes SQL classées 4 sous ensembles d'utilisation :

- Le Langage de Définition de Données (**DDL**) pour créer et supprimer des objets dans la base (schéma de la BDD) :
■ CREATE ■ ALTER ■ DROP ■ RENAME ■ TRUNCATE ■ COMMENT
- Le Langage de Contrôle de Données (**DCL**) pour gérer les droits sur les objets de la base:
■ GRANT ■ REVOKE
- Le Langage de Manipulation de Données (**DML**) pour la recherche, l'insertion, la mise à jour et la suppression de données
■ SELECT ■ INSERT ■ UPDATE ■ DELETE ■ MERGE
- Le Langage de Contrôle de Transaction (**TCL**) pour la gestion des transactions de la base
■ COMMIT ■ ROLLBACK ■ SAVEPOINT

DDL : L'instruction CREATE

Elle concerne :

■ **Tables** ■ **Views** ■ **Synonyms** ■ **Indexes** ■ **Sequences**

LES CLAUSES SQL

Syntaxe

```
SELECT champs1, champs2, champs3 FROM table;
```

Exemple

```
SELECT  
    country_id, country_name, region_id  
FROM  
    countries;
```

Une projection est une sélection sur quelques attributs de table.

La projection

Une projection est une sélection sur quelques attributs de table.

Exemple :

Afficher l'id, le nom et le prénom de l'employé

```
SELECT  
    employee_id,  
    first_name,  
    last_name  
FROM  
    employees;
```

Exemple :

Pour chaque département, afficher l'id, le nom et l'id du manager

```
SELECT
    department_id,
    department_name,
    manager_id
FROM
    departments;
```


Exemple :

L'id des départements aux quels sont affectés des employés

```
select department_id from employees;
```

Le mot **DISTINCT** permet de supprimer les doublons

DEPARTMENT_ID
100
30
(null)
90
20
70
110
50
80
40
60
10

DEPARTMENT_ID
90
90
90
60
60
60
60
60
60
100
100

La clause **WHERE** permet filtrer les enregistrements

1- Syntaxe de la clause **WHERE** sous la forme

SELECT col1, col2....colX

FROM uneTable

WHERE CONDITION;

Condition : prédicats séparés par des
quantificateurs (et, ou, !)

Exemple :

Le nom des employés du Département 100

```
SELECT
    last_name
FROM
    employees
WHERE DEPARTMENT ID = 100;
```

Le nom des employés don le manager a pour ID 100

```
SELECT
    last_name
FROM
    employees
WHERE MANAGER_ID = 100;
```

1- Syntaxe de la clause **WHERE** avec les connecteurs logiques (et, ou, non)

SELECT col1, col2....colX

FROM uneTable

WHERE CONDITION [**AND** | **OR** | **NOT**] CONDITION;

Une **CONDITION** est constitué de termes et d'opérateur(s).

$(a = b)$: a et b sont les termes.

$=$: est l'opérateur

$=$ égalité

\neq non égalité

\neq différent

$>$ Supérieur

$<$ inférieur

\geq

\leq

\nless

\ngtr

Exemple : La liste des programmeurs dont le manager
a pour ID = 103;

```
SELECT  
    MANAGER_ID, LAST_NAME,  
    EMAIL, FIRST_NAME, JOB_ID, MANAGER_ID  
FROM  
    employees  
WHERE JOB_ID='IT PROG' AND MANAGER_ID = '103';
```

Exemple : La liste des employés dont le **JOB_ID** est **'IT_PROG'** ou qui travaille au département 10;

```
select * from employees
where
JOB_ID='IT_PROG' or DEPARTMENT_ID = '10';
```


1- la clause **IN**

SELECT col1, col2....colN

FROM laTable

WHERE col_name **IN** (val1, val2,...valN);

1- la clause **IN**.

Exemple :

les employés des départements 10 – 20 – 30 – 100 – 80

```
SELECT
  *
FROM
  employees
WHERE DEPARTMENT_ID IN ('10', '20', '30', '100', '80');
```

1- la clause **BETWEEN**

```
SELECT col1, col2....colN  
    FROM uneTable  
WHERE col BETWEEN val1 AND val2;
```

1- Liste des **employees** dont l'**ID** est entre **109** et **114**

```
SELECT  
    *  
FROM  
    employees  
WHERE EMPLOYEE_ID BETWEEN  
    '109' AND '114';
```

Exécutons la requête :

```
select hire_date from employees;
```

Liste des employés embauchés du 30/10 au 31/12 2005

```
select * from employees  
where hire_date BETWEEN '30/10/05' and '31/12/05';
```

1- la clause **LIKE**

```
SELECT col1, col2....colN  
FROM une_table  
WHERE col LIKE expression;
```

Dans un expression les jokers sont :

% : toute suite de caractères

_ (dash) : remplace un caractère;

1- la clause **LIKE**

Liste des employés dont le nom commence par Al

```
select * from employees  
where FIRST_NAME like 'Al%';
```

1- la clause **LIKE**

Liste des employés dont le nom commence par Al

```
select * from employees  
where FIRST_NAME like 'Al%';
```

```
select * from employees  
where FIRST_NAME like 'A_b%';
```

1- la clause **LIKE**

Liste des employés dont le nom commence par **A**
Et le 3ème caractère est : **b**

```
select * from employees  
where FIRST_NAME like 'A_b%';
```


1- la clause **ORDER BY**

```
SELECT col1, col2....colN  
  
    FROM uneTable  
    { WHERE CONDITION }  
  
ORDER BY column_name {ASC|DESC};
```

Exemple : ORDER BY

```
select * from employees  
ORDER BY FIRST_NAME;
```

Function : SUBSTR (Col, Départ, nbChar)

Permet d'extraire une sous chaine d'une chaine..

A l'aide de cette fonction donnez
le nombre de recrutement par mois et par an

```
SELECT substr(hire_date,7,2) ,  
       substr(hire_date,4,2) ,  
       count(*)  
       from employees  
group by substr(hire_date,7,2),  
       substr(hire_date,4,2);
```

La clause **COUNT**

Permet de compter le nombre

```
SELECT COUNT(*) FROM employees;
```

```
select count(DEPARTMENT_ID) from employees;
```

```
select count (distinct DEPARTMENT_ID) from employees;
```

```
select distinct substr(hire_date,7,2)  
from employees;
```

```
select count (distinct substr(hire_date,7,2))  
from employees;
```

Exemple : GROUP BY

La fonction de groupage consiste à faire des calculs sur un critère.

Exemple somme des salaires par département.

```
SELECT DEPARTMENT_ID, SUM(salary)
FROM employees
GROUP BY DEPARTMENT_ID
ORDER BY DEPARTMENT_ID;
```

Les ALIAS

On peut utiliser des alias pour renommer les colonnes

```
SELECT substr(hire_date,7,2) an,  
       substr(hire_date,4,2) mois,  
       count(*) total  
from employees  
group by substr(hire_date,7,2),  
         substr(hire_date,4,2)  
order by an,mois;
```

Exemple : HAVING

La having est l'équivalent du WHERE pour les champs calculés

Exemple somme des salaires par département.

```
select department_id, sum(salary) total
from employees
group by department_id
having sum(salary) > 50000;
```

Les opérateurs arithmétiques

SQL permet d'utiliser les opérateurs arithmétique classiques

Somme des salaires de tous les employés

```
select sum(SALARY) from employees;
```

Somme de 1% du salaire de chaque employé

```
select sum(SALARY)*1/100 from employees;
```

On souhaite simuler l'augmentation de 4% du salaire de chaque employé

	Nom	Salaire	Majoration	Nouveau Salaire
1	TJ	2100	105	2205
2	Steven	2200	110	2310

Les opérateurs arithmétiques

SQL permet d'utiliser les opérateurs arithmétique classiques

Somme des salaires de tous les employés

```
select sum(SALARY) from employees;
```

Somme de 1% du salaire de chaque employé

```
select sum(SALARY)*1/100 from employees;
```

On souhaite simuler l'augmentation de 4% du salaire de chaque employé

```
select FIRST_NAME "Nom", salary "Salaire", SALARY*5/100  
salary+SALARY*5/100 "Nouveau Salaire" from employees  
order by SALARY asc;
```

Les opérateurs arithmétiques

Sous ORACLE, la clause FROM est obligatoire

```
SELECT 30+20 ;
```

```
SELECT 30+20 FROM DUAL;
```