Chapitre 1

Introduction

1.1 Présentation de SQL

SQL signifie "*Structured Query Language*" c'est-à-dire "Langage d'interrogation structuré".

En fait SQL est un langage complet de gestion de bases de données relationnelles. Il a été conçu par IBM dans les années 70. Il est devenu le langage standard des systèmes de gestion de bases de données (SGBD) relationnelles (SGBDR).

C'est à la fois :

- un langage de manipulation des données (LMD)
  - un langage d'interrogation (ordre SELECT)
  - un langage de manipulation des données (ordres UPDATE, INSERT, DELETE)
- un langage de définition des données (LDD ; ordres CREATE, ALTER, DROP),
- un langage de contrôle de l'accès aux données (LCD ; ordres GRANT, REVOKE).

Le langage SQL est utilisé par les principaux SGBDR : DB2, Oracle, Ingres, RDB,... Chacun de ces SGBDR a cependant sa propre variante du langage. Ce support de cours présente un noyau de commandes disponibles sur l'ensemble de ces SGBDR, et leur implantation dans Oracle Version 7.

1.2 Normes SQL

SQL a éte normalisé dès 1986 mais les premières normes, trop incomplètes, ont été ignorées par les éditeurs de SGBD.

La norme actuelle SQL-2 (appelée aussi SQL-92) date de 1992. Elle est acceptée par tous mais les SGBD relationnels qui dominent actuellement le marché (en particulier Oracle) ne sont toujours pas totalement adaptés à cette norme.

SQL-2 définit trois niveaux :

- Full SQL (ensemble de la norme)
- Intermediate SQL
- Entry Level (ensemble minimum à respecter pour se dire à la norme SQL-2)

1.3 Utilitaires associés

Comme tous les autres SGBD, Oracle comprend plusieurs utilitaires qui facilitent l'emploi du langage SQL et le développement d'applications de gestion s'appuyant sur une base de données relationnelle. En particulier SQLFORMS facilite grandement la réalisation des procédures transactionnelles (pour les traitements effectués pendant la saisie ou la modification des données en interactif par l'utilisateur). Il permet de dessiner les écrans de saisie et d'indiquer les traitements associés à cette saisie. D'autres utilitaires permettent de décrire les états de sorties imprimés, de sauvegarder les données, d'échanger des données avec d'autres logiciels, de travailler en réseau ou de constituer des bases de données réparties entre plusieurs sites.

Ce cours se limitant strictement à l'étude du langage SQL, nous n'étudierons pas tous ces utilitaires. Nous verrons les commandes essentielles d'un seul utilitaire, SQLPLUS, qui facilite l'utilisation interactive du langage SQL par un utilisateur. Ces commandes permettent de modifier les ordres SQL et de constituer des fichiers de commandes SQL que l'on peut réutiliser ensuite.

Les commandes du langage SQL peuvent être tapées directement au clavier par l'utilisateur ou elles peuvent être incluses dans un programme écrit dans un langage de troisième génération (Cobol, Langage C, Fortran, Ada,...) grâce à un précompilateur fourni par Oracle.

1.4 Connexion et déconnexion

On entre dans SQLPLUS par la commande :

`SQLPLUS` *`nom/mot-de-passe`*

Les deux paramètres, *nom* et *mot-de-passe*, sont facultatifs. Si on les omet sur la ligne de commande, SQLPLUS les demandera, ce qui est préférable pour *mot-de-passe* car une commande "*ps*" sous Unix permet de visualiser les paramètres d'une ligne de commande et donc de lire le mot de passe.

Pour se déconnecter, l'ordre à taper est **EXIT** (ou `exit` car on peut taper les commandes SQL en majuscules ou en minuscules).

1.5 Objets manipulées par SQL

1.5.1 Identificateurs

SQL utilise des identificateurs pour désigner les objets qu'il manipule : utilisateurs, tables, colonnes, index, fonctions, etc.

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. Les voyelles accentuées ne sont pas acceptées.

Un identificateur ne doit pas figurer dans la liste des mot clés réservés (voir manuel de référence Oracle). Voici quelques mots clés que l'on risque d'utiliser comme identificateurs : ASSERT, ASSIGN, AUDIT, COMMENT, DATE, DECIMAL, DEFINITION, FILE, FORMAT, INDEX, LIST, MODE, OPTION, PARTITION, PRIVILEGES, PUBLIC, SELECT, SESSION, SET, TABLE.

1.5.2 Tables

Les relations (d'un schéma relationnel ; voir polycopié du cours sur le modèle relationnel) sont stockées sous forme de tables composées de lignes et de colonnes.

Si on veut utiliser la table créée par un autre utilisateur, il faut spécifier le nom de cet utilisateur devant le nom de la table :

BERNARD.DEPT

Remarques 1.1

(a) Selon la norme SQL-2, le nom d'une table devrait être précédé d'un nom de schéma (voir 4.1).

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

Exemple 1.1

Table DEPT des départements :

DEPT	NOMD	LIEU
10	FINANCES	PARIS
20	RECHERCHES	GRENOBLE
30	VENTES	LYON
40	FABRICATION	ROUEN

1.5.3 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 (voir 1.7).

Chaque colonne est repérée par un identificateur unique à l'intérieur de chaque table. Deux colonnes de deux tables différentes peuvent porter le même nom. Il est ainsi fréquent de donner le même nom à deux colonnes de deux tables différentes lorsqu'elles correspondent à une clé étrangère à la clé primaire référencée. Par exemple, la colonne "Dept" des tables DEPT et EMP.

Une colonne peut porter le même nom que sa table.

Le nom complet d'une colonne est en fait celui de sa table, suivi d'un point et du nom de la colonne. Par exemple, la colonne DEPT.LIEU

Le nom de la table peut être omis quand il n'y a pas d'ambiguïté sur la table à laquelle elle appartient, ce qui est généralement le cas.

1.6 Types de données

Les types de données d'Oracle ne suivent pas la norme SQL-2.

1.6.1 Type numérique

Types numériques SQL-2

Les types numériques de la norme SQL-2 sont :

- Types exacts : INTEGER, SMALLINT, NUMERIC, DECIMAL
- Types non exacts : REAL, DOUBLE PRECISION, FLOAT

La définition de ces types dépend du SGBD. Reportez-vous au manuel du SGBD que vous utilisez pour plus de précisions.

Type numérique d'Oracle

Oracle n'a qu'un seul type numérique NUMBER. Par soucis de compatibilité, Oracle permet d'utiliser les types SQL-2 mais ils sont ramenés au type NUMBER.

Lors de la définition d'une colonne de type numérique, on peut préciser le nombre maximum de chiffres et de décimales qu'une valeur de cette colonne pourra contenir :

```
NUMBER
NUMBER(taille_maxi)
NUMBER(taille_maxi, décimales)
```

Si le paramètre *décimales* n'est pas spécifié, 0 est pris par défaut. Si le paramètre *taille_maxi* n'est pas spécifié il n'y aura aucun contrôle effectué par ORACLE sur le nombre de chiffres des valeurs contenues dans la colonne (maximum 38 chiffres). La valeur absolue du nombre doit être inférieure à 10^{128}.

L'insertion d'un nombre avec plus de chiffres que *taille_maxi* sera refusée (*taille_maxi* ne prend en compte ni le signe ni le point décimal). Les décimales seront éventuellement arrondies en fonction des valeurs données pour *taille_maxi* et *décimales*.

Exemple 1.2

SALAIRE NUMBER(8,2)
définit une colonne numérique SALAIRE. Les valeurs auront au maximum 2 décimales et 8 chiffres au plus au total (donc 6 chiffres avant le point décimal).

Les constantes numériques ont le format habituel : -10, 2.5, 1.2E-8 (ce dernier représentant 1.2 x 10^{-8}).

1.6.2 Type chaîne de caractères

Les constantes chaînes de caractères sont entourées par des apostrophes ('). Si la chaîne contient une apostrophe, celle-ci doit être doublée. Exemple : `'aujourd''hui'`.

Il existe deux types pour les colonnes qui contiennent des chaînes de caractères :

- le type `CHAR` pour les colonnes qui contiennent des chaînes de longueur constantes ne contenant pas plus de 255 caractères.

  La déclaration de type chaîne de caractères de longueur constante a le format suivant :

  `CHAR (longueur)`

  où *longueur* est la longueur maximale (en nombre de caractères) qu'il sera possible de stocker dans le champ. *longueur* doit être obligatoirement spécifiée. L'insertion d'une chaîne dont la longueur est supérieure à *longueur* sera refusée. Une chaîne plus courte que *longueur* sera complétée par des espaces (important pour les comparaisons de chaînes).

- le type `VARCHAR2` pour les colonnes qui contiennent des chaînes de longueurs variables ne contenant pas plus de 2000 caractères. On déclare ces colonnes par :

  `VARCHAR2 (longueur)`

  longueur indique la longueur maximale des chaînes contenues dans la colonne (VARCHAR dans la norme SQL-2).

1.6.3 Type date

La déclaration de type date a le format suivant :

`DATE`

Une constante de type "date" est une chaîne de caractères entre apostrophes. Le format dépend des options que l'administrateur a choisies au moment de la création de la base. S'il a choisi de "franciser" la base, le format d'une date est "jour/mois/année", par exemple, '25/11/92' (le format "américain" par défaut donnerait '25-NOV-92'). L'utilisateur peut saisir des dates telles que '3/8/93' mais les dates enregistrées dans la base ont toujours deux chiffres pour chacun des nombres, par exemple, '03/08/93'.

Dans Oracle une donnée de type DATE inclut un temps en heures, minutes et secondes. En SQL-2 ce n'est pas le cas et il existe les types TIME et TIMESTAMP pour indiquer un temps en heures minutes, secondes et fractions de secondes.

1.6.4 Valeur NULL

Une colonne qui n'est pas renseignée, et donc vide, est dite contenir la valeur "**NULL**". Cette valeur n'est pas zéro, c'est une absence de valeur.

1.7 Création d'une table

L'ordre CREATE TABLE permet de créer une table en définissant le nom et le type (voir 1.6) de chacune des colonnes de la table. Nous ne verrons ici que trois des types de données utilisés dans SQL : numérique, chaîne de caractères et date. Nous nous limiterons dans cette section à une syntaxe simplifiée de cet ordre (voir 4.2.1 et 1.8 pour des compléments) :

```
CREATE TABLE table
(colonne1 type1,
colonne2 type2,
.........
.........)
```

table est le nom que l'on donne à la table ; *colonne1, colonne2,..* sont les noms des colonnes ; *type1, type2,..* sont les types des données qui seront contenues dans les colonnes.

On peut ajouter après la description d'une colonne l'option NOT NULL qui interdira que cette colonne contienne la valeur NULL. On peut aussi ajouter des contraintes d'intégrité portant sur une ou plusieurs colonnes de la table (voir 1.8).

Exemple 1.3

```
CREATE TABLE article
( ref     CHAR (10) NOT NULL,
  prix    NUMBER (9,2),
  datemaj DATE)
```

1.8 Contrainte d'intégrité

Dans la définition d'une table, on peut indiquer des contraintes d'intégrité portant sur une ou plusieurs colonnes. Les contraintes possibles sont :

UNIQUE, PRIMARY KEY, FOREIGN KEY...REFERENCES, CHECK

Chaque contrainte doit être nommée (ce qui permettra de la désigner par un ordre ALTER, et ce qui est requis par les nouvelles normes SQL) en la faisant précéder de :

```
CONSTRAINT nom-contrainte contrainte
```


Il existe des contraintes :

sur une colonne : la contrainte porte sur une seule colonne. Elle suit la définition de la colonne dans un ordre CREATE TABLE (pas possible dans un ordre ALTER TABLE).

sur une table : la contrainte porte sur une ou plusieurs colonne. Elles se place au même niveau que les définition des colonnes dans un ordre CREATE TABLE ou ALTER TABLE.

1.8.1 Types de contraintes

```
(pour une contrainte sur une table :)
PRIMARY KEY (colonne1, colonne2,...)
(pour une contrainte sur une colonne :)
PRIMARY KEY
```

indique la clé primaire de la table (contrainte d'intégrité d'entité). Aucune des colonnes qui composent cette clé ne doit avoir une valeur NULL.

```
(pour une contrainte sur une table :)
UNIQUE (colonne1, colonne2,...)
(pour une contrainte sur une colonne :)
UNIQUE
```

interdit qu'une colonne (ou la concaténation de plusieurs colonnes) contienne deux valeurs identiques.

```
(pour une contrainte sur une table :)
FOREIGN KEY (colonne1, colonne2,...)
REFERENCES tableref[(col1, col2,...)]
[ON DELETE CASCADE]
(pour une contrainte sur une colonne :)
REFERENCES tableref[(col1)]
[ON DELETE CASCADE]
```

indique que la concaténation de *colonne1, colonne2,...* (ou la colonne que l'on définit pour une contrainte sur une colonne) est une clé étrangère qui fait référence à la concaténation des colonnes *col1, col2,...* de la table *tableref* (contrainte d'intégrité référentielle). Si aucune colonne de *tableref* n'est indiquée, c'est la clé primaire de *tableref* qui est prise par défaut.

Cette contrainte ne permettra pas d'insérer une ligne de la table si la table *tableref* ne contient aucune ligne dont la concaténation des valeurs de *col1, col2,...* est égale à la concaténation des valeurs de *colonne1, colonne2,...*

col1, col2,... doivent avoir la contrainte PRIMARY KEY ou UNIQUE. Ceci implique qu'une valeur de *colonne1, colonne2,...* va référencer une et une seule ligne de *tableref.*

L'option "**ON DELETE CASCADE**" indique que la suppression d'une ligne de *tableref* va entrainer automatiquement la suppression des lignes qui la référencent dans la

table. Si cette option n'est pas indiquée, il est impossible de supprimer des lignes de *tableref* qui seraient référencées par des lignes de la table.

`CHECK(condition)`

donne une condition que la ou les colonnes devront vérifier (exemples dans la section suivante). On peut ainsi indiquer des contraintes d'intégrité de domaines dont on a des exemples dans la section 1.8.2.

Des contraintes d'intégrité peuvent être ajoutées (uniquement des contraintes liées à la table et pas à une colonne) ou supprimées par la commande ALTER (exemple à la fin de la section 1.8.2). Mais pour modifier une contrainte, il faut la supprimer et ajouter ensuite la contrainte modifiée.

Il est parfois intéressant d'enlever temporairement des contraintes. Oracle le permet par la commande ALTER TABLE ... DISABLE/ENABLE mais n'utilise pas la commande de la norme SQL-2 (SET CONSTRAINTS ... DEFERRED/IMMEDIATE).

1.8.2 Exemples de contraintes

Quelques contraintes sur des colonnes :

```
CREATE TABLE EMP (
  MATR NUMBER(5) CONSTRAINT KEMP PRIMARY KEY,
  NOME CHAR(9) CONSTRAINT NOM_UNIQUE UNIQUE
    CONSTRAINT MAJ CHECK (NOME = UPPER(NOME)),
  .........
  DEPT NUMBER(2) CONSTRAINT R_DEPT REFERENCES DEPT(DEPT)
    CONSTRAINT NDEPT CHECK (DEPT IN (10, 20, 30, 35, 40)))
```

Des contraintes de colonne peuvent être mises au niveau de la table :

```
CREATE TABLE EMP (
  MATR NUMBER(5),
  NOME CHAR(9) CONSTRAINT NOM_UNIQUE UNIQUE
     CONSTRAINT MAJ CHECK (NOME = UPPER(NOME)),
  .........
  CONSTRAINT NDEPT DEPT NUMBER(2) CHECK (DEPT IN (10, 20, 30, 35, 40)),
  CONSTRAINT KEMP PRIMARY KEY (MATR),
  CONSTRAINT R_DEPT FOREIGN KEY (DEPT) REFERENCES DEPT(DEPT))
```

Certaines contraintes portent sur plusieurs colonnes et ne peuvent être indiquées que comme contraintes de table :

```
CREATE TABLE PARTICIPATION (
MATR NUMBER(5) CONSTRAINT R_EMP REFERENCES EMP,
CODEP VARCHAR2(10) CONSTRAINT R_PROJET REFERENCES PROJET,
....,
CONSTRAINT PKPART PRIMARY KEY(MATR, CODEP))
```

Exemple avec ALTER TABLE :

```
ALTER TABLE EMP
  DROP CONSTRAINT NOM_UNIQUE
```



```
  ADD (CONSTRAINT SAL_MIN CHECK(SAL + NVL(COMM,0) > 5000))
```

1.9 Sélection simple

L'ordre pour retrouver des informations stockées dans la base est "SELECT".

Nous étudions dans ce chapitre une partie simplifiée de la syntaxe, suffisant néanmoins à la plupart des interrogations courantes. Une étude plus complète sera vue au chapitre 3.

```
SELECT exp1, exp2,...
FROM table
WHERE prédicat
```

table est le nom de la table sur laquelle porte la sélection. *exp1, exp2,...* est la liste des expressions (colonnes, constantes,... ; voir 1.10) que l'on veut obtenir. Cette liste peut être "*", auquel cas toutes les colonnes de la table sont sélectionnées.

Exemples 1.4

(a) `SELECT * FROM DEPT`

(b) `SELECT NOME, POSTE FROM EMP`

(c) `SELECT NOME , SAL + NVL(COMM,0) FROM EMP`

La clause WHERE permet de spécifier quelles sont les lignes à sélectionner.

Le prédicat peut prendre des formes assez complexes. La forme la plus simple est "exp1 *op* exp2", où *exp1* et *exp2* sont des expressions (voir 1.10) et *op* est un des opérateurs =, != (différent), >, >=, <, <=.

Exemple 1.5

```
SELECT MATR, NOME, SAL * 1.15
FROM EMP
WHERE SAL + NVL(COMM,0) >= 12500
```

1.10 Expressions

Les expressions acceptées par SQL portent sur des colonnes, des constantes, des fonctions.

Ces trois types d'éléments peuvent être reliés par des opérateurs arithmétiques (+ - * /), maniant des chaînes de caractères (|| pour concaténer des chaînes), des dates (- donne le nombre de jours entre deux dates).

Les priorités des opérateurs arithmétiques sont celles de l'arithmétique classique (* et /, puis + et -). Il est possible d'ajouter des parenthèses dans les expressions pour obtenir l'ordre de calcul que l'on désire.

Les expressions peuvent figurer :

– en tant que colonne résultat d'un SELECT