# Oracle : SQL

Denis Roegel
roegel@loria.fr
IUT Nancy 2

1998/1999

## Table des matières

Ce document est une adaptation d'un chapitre de l'ouvrage *Oracle Unleashed* (SAMS Publishing, 1996).

# 1 Introduction

SQL (Structured Query Language) a été introduit par IBM comme le langage d'interface de son prototype de système de gestion de base de donnée relationnelle, System-R. Le premier système SQL disponible sur le marché a été introduit en 1979 par Oracle. Aujourd'hui, SQL est devenu un standard de l'industrie et Oracle est un leader dans la technologie des systèmes de gestion de bases de données relationnelles.

Comme SQL est un langage non procédural, des ensembles d'enregistrements peuvent être manipulés à la fois. La syntaxe est naturelle et souple, ce qui permet de se concentrer sur la présentation des données. Oracle a deux optimiseurs (basés sur le coût et des règles) qui vont analyser la syntaxe et la formater en une expression efficace avant que le moteur de la base de donnée ne le reçoive pour traitement. L'administrateur de la base de données détermine quel optimiseur est sélectionné pour chaque base de données.

### SQL–le standard

L'ANSI (American National Standards Institute) a déclaré SQL le langage standard pour les systèmes de gestion de bases de données relationnelles. La plupart des entreprises qui produisent des systèmes de gestion de bases de données relationnelles sont compatibles avec SQL et essaient de respecter le standard SQL89.

# 2 Types de données

Une règle générale pour écrire des expressions SQL valides est de ne pas mélanger des types de données. Des utilitaires de conversion sont disponibles pour passer d'un type à un autre. Ces fonctions de conversion sont décrites plus loin.

## 2.1 Numérique

Le type `NUMBER` est utilisé pour stocker zéro, les nombres négatifs, positifs, à virgule fixe et flottants jusqu'à 38 chiffres de précision. Les nombres peuvent s'échelonner entre $1.0 \times 10^{-130}$ et $1.0 \times 10^{126}$.

Les nombres peuvent être définis de l'une des trois manières suivantes :

```
NUMBER(p,s)
```

où `p` est la précision jusqu'à 38 chiffres et `s` est l'échelle (nombre de chiffres à la droite du point décimal). L'échelle peut s'étaler de $-84$ à 127.

```
NUMBER (p)
```

Ceci est un nombre à virgule fixe avec une échelle de zéro et une précision de `p`.

```
NUMBER
```

Ceci est un nombre à virgule flottante avec une précision de 38.

La table suivante montre comment Oracle stocke différentes échelles et précisions :

| Actual Data | 123456.789 | 123456.789 | 123456.789 | 123456.789 |
|---|---|---|---|---|
| Defined as | `NUMBER(6,2)` | `NUMBER(6)` | `NUMBER(6,-2)` | `NUMBER` |
| Stored as | 123456.79 | 123457 | 123400 | 123456.789 |

## 2.2 Date

Au lieu de stocker la date et l'heure dans une chaîne ou sous forme numérique, IBM a créé plusieurs types séparés. Pour chaque type `DATE`, les informations suivantes sont stockées :

```
Century Year Month Day Hour Minute Second
```

Il est facile de récupérer les date et heure courantes en appelant la fonction `SYSDATE`.

L'arithmétique sur les dates est possible en utilisant des constantes numériques ou d'autres dates. Seules l'addition et la soustraction sont admises. Par exemple, `SYSDATE + 7` va rendre la date dans une semaine.

Chaque base de donnée a un format de date par défaut qui est défini par le paramètre d'initialisation `NLS_DATE_FORMAT`. Ce paramètre est généralement mis à `DD-MON-YY`, où `DD` est le jour du mois (le premier jour du mois est 01), `MON` est l'abréviation du nom du mois et `YY` est une représentation à deux chiffres de l'année.

Si une heure n'est pas spécifiée, la valeur par défaut est minuit. Si seule l'heure est saisie, la date par défaut sera le premier jour du mois courant.

## 2.3 Caractère

Il y a quatre types de données caractère disponibles :

1. Le type `CHAR` est utilisé quand des champs de taille fixe sont nécessaires. Toute longueur inférieure ou égale à 255 caractères peut être spécifiée. La longueur par défaut est 1. Quand des données sont entrées, tout espace résiduel est rempli de blancs. Tous les caractères alpha-numériques sont autorisés.
2. Le type `VARCHAR2` est utilisé pour des champs de longueur variable. Une longueur doit être fournie lorsque l'on utilise ce type de données. La longueur maximale est de 2000 caractères. Tous les caractères alpha-numériques sont autorisés.
3. Le type `LONG` est utilisé pour stocker de grandes quantités de texte de longueur variable. Toute longueur jusqu'à 2 gigaoctets peut être spécifiée. Ce type a des restrictions, telles que :
   - Une seule colonne d'une table peut être définie en `LONG`.
   - Une colonne de type `LONG` ne peut pas être indexée.
   - Une colonne de type `LONG` ne peut pas être passée en argument à une procédure.
   - Une fonction ne peut pas être utilisée pour rendre une colonne de type `LONG`.
   - Une colonne de type `LONG` ne peut pas être utilisée dans des clauses `where`, `order by`, `group by`, ou `connect by`.
4. Le type `VARCHAR` est synonyme de `VARCHAR2`. Oracle réserve ceci pour une utilisation future. Il ne faut pas utiliser ce type.

## 2.4 Binaire

Deux types de données, `RAW` et `LONGRAW`, sont disponibles pour stocker des données de type binaire telles que du son digitalisé et des images. Ces types de données ont des caractéristiques similaires aux types `VARCHAR2` et `LONG` déjà mentionnés.

Le type `RAW` est utilisé pour stocker des données binaires jusqu'à 2000 caractères et le type `LONGRAW` jusqu'à 2 gigaoctets.

Oracle ne stocke et n'extrait que des données binaires. Aucune manipulation de chaîne n'est autorisée. Les données sont extraites sous forme de valeurs de caractères hexadécimaux.

### 2.5 Autres

Chaque ligne de la base de donnée a une adresse. Celle-ci peut être obtenue en utilisant la fonction `ROWID`. Le format de `ROWID` est le suivant :

```
BLOCK.ROW.FILE
```

- `BLOCK` est le bloc de données des données `FILE` contenant la ligne `ROW`. Les données sont en format hexadécimal et de type `ROWID`.
- `MLSLABEL` est un type de donnée utilisé pour stocker le format binaire d'une étiquette utilisée sur un système d'exploitation sécurisé.

## 3 L'instruction `CREATE`

Cette instruction ouvre le monde à l'utilisateur. Seules quelques unes des instruction `CREATE` seront décrites ici.

### 3.1 Tables

Chaque concepteur de base de donnée doit créer une table un jour ou l'autre. Il est nécessaire d'avoir un privilège système pour exécuter la commande `CREATE TABLE`. L'administrateur de la base de données gère ces privilèges. La syntaxe pour créer une table est :

```
CREATE TABLE schema.TABLE (COLUMN DATATYPE
                           default expression
                           column constraint) table constraint

PCTFREE x1 PCTUSED x2 INITRANS x3 MAXTRANS x4
TABLESPACE name STORAGE clause CLUSTER cluster clause
ENABLE clause DISABLE clause AS subquery
```

Dans cette syntaxe,

- `SCHEMA` est un paramètre optionnel pour identifier le schéma de la base de donnée dans laquelle cette table doit être placée. Par défaut, c'est celui de l'utilisateur.
- `TABLE` est obligatoire et est le nom de la table.
- `COLUMN DATATYPE` sont requis pour identifier chaque colonne dans la table. Les colonnes doivent être séparées par des virgules. Il y a au maximum 254 colonnes par table.
- L'expression `DEFAULT` est optionnelle et est utilisée pour donner une valeur par défaut à une colonne lorsque des insertions ultérieures ne réussissent pas à donner une valeur.
- `COLUMN CONSTRAINT` est optionnel. C'est utilisé pour définir une contrainte d'intégrité telle que `not null`.
- `TABLE CONSTRAINT` est optionnel et est utilisé pour définir une contrainte d'intégrité sur la table, comme par exemple la clé primaire.
- `PCTFREE` est optionnel mais a une valeur par défaut de 10. Ceci indique que 10 pour cents de chaque bloc sera réservé pour de futures mises à jour des lignes de la table. Les entiers de 1 à 99 sont autorisés.
- `PCTUSED` est optionnel mais a une valeur par défaut de 40. Ceci indique le pourcentage minimum d'espace utilisé qu'Oracle maintient avant qu'un bloc de données soit candidat pour une insertion de ligne. Les entiers de 1 à 99 sont autorisés. La somme de `PCTFREE` et `PCTUSED` doit être plus petite que 100.
- `INITRANS` est optionnel mais a une valeur par défaut de 1. Les entiers de 1 à 255 sont autorisés. Il est recommandé de ne pas modifier cette valeur. C'est une allocation du nombre d'entrées de transaction assignées au sein du bloc de données de la table.

- `MAXTRANS` est optionnel mais a une valeur par défaut qui est fonction de la taille des blocs de données. Ceci est utilisé pour identifier le nombre maximum de transactions parallèles pouvant faire des mises à jour dans un bloc de la table. Il est recommandé de ne pas modifier ce paramètre.
- `TABLESPACE` est optionnel mais a comme valeur par défaut le nom du « tablespace » du propriétaire du schéma. Un nom différent du nom par défaut peut être utilisé. Ces noms dépendent en général de l'application. L'administrateur de la base de données sera en mesure de donner des recommandations adéquates.
- `STORAGE` est optionnel et a des caractéristiques par défaut définies par l'administrateur de la base de données.
- `CLUSTER` est optionnel et spécifie qu'une table doit faire partie d'un groupe. Il faut identifier les colonnes de la table qui doivent en faire partie. Typiquement, les colonnes groupées sont des colonnes dans lesquelles se trouve la clé primaire.
- `ENABLE` est optionnel et active une contrainte d'intégrité.
- `DISABLE` est optionnel et désactive une contrainte d'intégrité.
- `AS SUBQUERY` est optionnel et insère les lignes rendues par la sous-requête dans la table à sa création.

Une fois que la table est créée, on peut utiliser la commande `ALTER TABLE` pour modifier la table. Pour modifier une contrainte d'intégrité, il faut d'abord utiliser `DROP` sur la contrainte, puis la recréer. Voyons deux exemples sur la création de tables.

```
CREATE TABLE ADDRESSES (ADRS_ID          NUMBER(6),
                        ACTIVE_DATE      DATE,
                        BOX_NUMBER       NUMBER(6),
                        ADDRS_1          VARCHAR2(40),
                        ADDRS_2          VARCHAR2(40),
                        CITY             VARCHAR2(40),
                        STATE            VARCHAR2(2),
                        ZIP              VARCHAR2(10));
```

Ceci est la forme la plus simple de création d'une table utilisant tous les paramètres par défaut. Le second exemple suit.

```
CREATE TABLE ADDRESSES  (ADRS_ID  NUMBER(6) CONSTRAINT PK_ADRS PRIMARY KEY,
                        ACTIVE_DATE   DATE DEFAULT SYSDATE,
                        BOX_NUMBER    NUMBER(6)      DEFAULT NULL,
                        ADDRS_1       VARCHAR2(40)   NOT NULL,
                        ADDRS_2       VARCHAR2(40)   DEFAULT NULL,
                        CITY          VARCHAR2(40)   DEFAULT NULL,
                        STATE         VARCHAR2(2)    DEFAULT 'NY',
                        ZIP           VARCHAR2(10))

PCTFREE 5
PCTUSED 65
TABLESPACE adrs_data
STORAGE (INITIAL 5140
         NEXT      5140
         MINEXTENTS  1
         MAXEXTENTS 10
         PCTINCREASE 10);
```

Dans cet exemple, des contraintes de données sont utilisées et certains paramètres de stockage seront actifs. L'utilisation de `PCTFREE` et `PCTUSED` est une bonne idée si les données sont relativement statiques.

## 3.2 Index

Les index sont utilisés pour améliorer la performance de la base de données. Un index est créé sur une ou plusieurs colonnes d'une table ou d'un groupe. On peut avoir plusieurs index par table. Le privilège système de `CREATE INDEX` est nécessaire pour exécuter cette commande. L'administrateur de la base de données est responsable de ces privilèges. La syntaxe de création d'un index est :

```
CREATE INDEX schema.index ON schema.table (COLUMN  ASC/DESC)

CLUSTER schema.cluster  INITRANS x MAXTRANS x TABLESPACE
          name STORAGE  clause PCTFREE x NOSORT
```

Dans cette syntaxe,

- `SCHEMA` est un paramètre optionnel identifiant le schéma de la base dans lequel doit se trouver l'index. Par défaut, c'est le schéma de l'utilisateur.
- `INDEX` est obligatoire et est le nom de l'index.
- `ON` est un mot réservé obligatoire.
- `TABLE` est un nom de table sur lequel est construit l'index.
- `COLUMN` est le nom de colonne à indexer. S'il y a plus d'une colonne, il faut s'assurer qu'elles sont dans l'ordre de priorité.
- `ASC/DESC` sont des paramètres optionnels. Les index sont construits en ordre croissant par défaut. `DESC` permet d'avoir l'ordre décroissant.
- `CLUSTER` est nécessaire seulement si cet index est destiné à un groupe.
- `INITRANS` est optionnel mais a la valeur par défaut de 1. Les entiers de 1 à 255 sont permis. Il est recommandé de ne pas changer ce paramètre. Il s'agit d'une allocation du nombre d'entrées de transactions assignées dans le bloc de données pour l'index.
- `MAXTRANS` est optionnel mais a une valeur par défaut qui est fonction de la taille du bloc de données. Il est utilisé pour identifier le nombre maximal de transactions qui peuvent mettre à jour en parallèle un bloc de données pour l'index. Il est recommandé de ne pas changer ce paramètre.
- `TABLESPACE` est optionnel mais a comme valeur par défaut le nom du « tablespace » du propriétaire du schéma. Un nom différent du nom par défaut peut être utilisé. L'administrateur de la base de données sera en mesure de donner des recommandations adéquates.
- `STORAGE` est optionnel et a des caractéristiques par défaut définies par l'administrateur de la base de données.
- `PCTFREE` est optionnel mais a une valeur par défaut de 10. Ceci indique que 10 pour cents de chaque bloc seront réservés pour de futures mises à jour de l'index. Les entiers de 1 à 99 sont autorisés.
- `NOSORT` est un paramètre optionnel qui permet de gagner du temps lors de la création de l'index si les données de la table sont déjà stockées dans l'ordre croissant. Ceci ne peut pas être utilisé si un index de groupe est utilisé.

En utilisant la table `ADRESSES` définie dans l'exemple du `CREATE TABLE`, deux index vont être créés dans le prochain exemple :

```
CREATE INDEX x_adrs_id ON ADDRESSES (ADRS_ID);
```

Ceci créera un index sur la colonne `adrs_id` seulement.

```
CREATE INDEX x_city_state ON ADDRESSES (CITY,STATE)
TABLESPACE application_indexes;
```

Cet index a deux colonnes ; `CITY` est la colonne primaire. Pour que les requêtes puissent utiliser un index, les noms des colonnes doivent faire partie de l'instruction `SELECT`. Si une instruction `SELECT` inclut `STATE` mais non `CITY`, l'index ne sera pas utilisé. Toutefois, si l'instruction `SELECT` contient une référence à `CITY` mais pas à `STATE`, une partie de l'index sera utilisée car `CITY` est la première colonne de l'index.

**Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.**