CHAPITRE 1

LES COMMANDES SQL

ALTER TABLE

Oracle offre la possibilité de modifier dynamiquement la définition d'une table:

- soit en y ajoutant la définition d'une colonne ou d'une contrainte d'intégrité,
- soit en y modifiant la définition d'une colonne,
- soit en y détruisant une contrainte d'intégrité. (Il n'est pas possible de détruire une colonne. Toute colonne qui n'est pas utilisée peut être mise à la valeur "null" auquel cas elle n'occupe plus d'espace disque).

```
ALTER TABLE nom de table
      ADD def_colonne                                          /* voir CREATE TABLE*/
        / [contrainte_sur_table]                               /* voir CREATE TABLE*/
        / (def_colonne/ contrainte_sur_table {, def_colonne/ contrainte_sur_table})
      MODIFY def_colonne
        / (def_colonne{, def_colonne})
      DROP   UNIQUE (nom_de_colonne {,nom_de_colonne})
        /      PRIMARY KEY
        /      CONSTRAINT nom_de_la_contrainte
```

Exemple:

```
ALTER TABLE FOURNISSEURS
      ADD (PAYS CHAR(20));
```

Il est possible de modifier la définition d'une colonne, à condition que la nouvelle définition soit compatible avec le contenu de la colonne.

Exemple:

```
ALTER TABLE FOURNISSEURS
      MODIFY (adr varchar(45));
```

BREAK

Cette commande crée des ruptures à l'édition.

BREAK {ON [<exp>/ROW/PAGE/REPORT]}
[SKIP<n>/SKIP PAGE]
[NODUPLICATES/DUPLICATES]

BREAK utilisé seul affiche les ruptures prises en cours.
CLEAR BREAKS annule toutes les ruptures.
DUPLICATES saute une ligne entre chaque enregistrement sélectionné.

Ex :

```
BREAK ON nfour SKIP 3;
select * from prix
order by nfou,prix;
```

COLUMN

Cette commande spécifie comment une colonne et un titre de colonne peuvent être présentés dans un état.

COLUMN <nom de colonne>
[HEADING] <titre de colonne>
[FORMAT] <format>
[LIKE] <nom de colonne>
[JUSTIFY] [LEFT/CENTER/RIGHT]
[ALIAS] <nouveau nom>

HEADING donne un titre à la colonne
FORMAT spécifie un format
LIKE spécifie un format à l'image d'une autre colonne déjà définie
JUSTIFY justifie le titre de la colonne
ALIAS définit un alias pour la colonne.

Ex :

```
COLUMN ADR HEADING ADRESSE;
COLUMN NOMFOU HEADING 'Nom des Fournisseurs';
/* Pour visualiser un titre sur 2 lignes, le couper par | */
COLUMN NOMFOU HEADING 'Nom des | Fournisseurs';
SELECT NFOU,NOMFOU,ADR FROM EMP;

COLUMN REFPIECE FORMAT A11;
COLUMN REFFOU LIKE REFPIECE;HEADING 'Référence des |
Fournisseurs';
SELECT REFPIECE,REFFOU FROM PRIX;

COLUMN PRIX * 1.186 ALIAS PTTC HEADING 'Prix TTC'
SELECT REFPIECE,PTTC FROM PRIX;
```

COMPUTE

Cette commande permet de faire différents calculs lorsque des ruptures ont été demandées, et doit être utilisée seulement si la commande BREAK a été exécutée. COMPUTE et BREAK doivent travailler sur les mêmes colonnes.

COMPUTE

[AVG]	effectue la moyenne
[COUNT]	compte le nombre de valeurs non nulles
[MAX]	retourne la plus grande des valeurs
[MIN]	retourne la plus petite des valeurs
[NUMBER]	compte le nombre de lignes
[STD]	calcule l'écart type (STandard Deviation)
[SUM]	effectue le somme
[VARIANCE]	calcule la variance

OF <nom de colonne>{,<nom de colonne>}
ON <nom de colonne ayant servi à créer la rupture>

CLEAR COMPUTES ; annule toutes les computes

Ex :

```
BREAK ON REFPIECE;
COMPUTE MIN OF PRIX ON REFPIECE;
SELECT REFPIECE,NFOU,REFFOU,PRIX
FROM PRIX
ORDER BY REFPIECE;
```

CREATE TABLE

1ère forme (Syntaxe simplifiée):

```
CREATE TABLE nom_de_table (
      nom_de_colonne   type_de_données   [DEFAULT expression]  [{contrainte_sur_colonne}]
   /  [contrainte_sur_table]
   /  {,nom_de_colonne type_de_données   [DEFAULT expression]  [{contrainte_sur_colonne}]
   /  [contrainte_sur_table]}
                                    );
```

avec :

- au maximum: 30 caractères pour un nom de table ou de colonne, 254 colonnes, 1 colonne de type long.
- contrainte_sur_colonne définie comme suit:

```
[CONSTRAINT nom_de_la_contrainte]
                    [NOT] NULL
                /   UNIQUE
                /   PRIMARY KEY
                /   REFERENCES nom_de_table [(nom_de_colonne)] [ON DELETE CASCADE]
                /   CHECK (condition)
```

- contrainte_sur_table définie comme suit:

```
[CONSTRAINT nom_de_la_contrainte]
                    UNIQUE / PRIMARY KEY  (nom_de_colonne {,nom_de_colonne})
                /   FOREIGN KEY (nom_de_colonne {,nom_de_colonne})
                        REFERENCES nom_de_table  [(nom_de_colonne {,nom_de_colonne})]
                            ON DELETE CASCADE
                /   CHECK (condition)
```

Exemple:

```
CREATE TABLE fournisseurs
(nfou       number(2)          primary key,
 nomfou     varchar(25)        not null check (nomfou=upper(nomfou)),
 adr        varchar(30),
 ville      varchar(20)        default 'ROUEN',
 region     varchar(3)         check (region in ('IDF','RHA',HTN','BSN')),
 c_post     varchar(5),
 constraint nom_adr unique (nomfou,adr)     );
```

2ème forme (Syntaxe simplifiée):

CREATE TABLE nom de table [(colonne [NOT NULL]
{,colonne [NOT NULL]})] as SELECT .....;

Exemple:

```
CREATE TABLE parisiens as
select nfou,nomfou,adr
 from fournisseurs
 where ville="PARIS";
```

CREATE INDEX

Cette commande crée un index sur une ou plusieurs colonnes pour une table. Plusieurs index peuvent être créés pour une même table. La tenue des index est faite automatiquement lors de la mise à jour des tables. Les requêtes SQL sont transparentes au fait qu'il existe ou non un index. C'est l'optimiseur d'Oracle qui, au moment de l'exécution de chaque requête, recherche s'il peut ou non s'aider d'un index. Les index sont stockés sous forme d'arbres équilibrés B_TREE.

CREATE [UNIQUE] INDEX <nom de l'index>
ON <nom de table> (<colonne1> ASC/DESC {,<colonne n> ASC/DESC})
[COMPRESS/NOCOMPRESS]
[PCTFREE <nombre>]

UNIQUE permet d'assurer l'unicité de la clef
NOCOMPRESS indique que l'on ne veut pas comprimer la clef. Par défaut, les index sont comprimés.
Dans certains cas, un index comprimé peut être plus efficace qu'un index non comprimé, si le seul accès au segment d'index est nécessaire, sans besoin d'accès au segment de données.
Ex : select nomfou from fournisseurs
where nomfou like 'F%';
PCTFREEE précise le pourcentage de place laissée libre dans les blocs d'index à la création de l'index. Cette place libre évitera une réorganisation de l'index lors des premières insertions de nouvelles clés. La valeur par défaut est 20%.

La vue INDEXES permet à l'utilisateur de voir les index qu'il a créé.
La vue SYSINDEXES permet de voir les index sur toutes les tables auxquelles on a accès.

CREATE SEQUENCE (Syntaxe simplifiée)

Une séquence est un objet de la base de données à partir duquel les utilisateurs peuvent générer des entiers uniques. Ces séquences peuvent permettre de générer automatiquement les clés primaires.

CREATE SEQUENCE nom_de_séquence [INCREMENT BY entier] [START WITH entier]
[MAXVALUE entier / NOMAXVALUE] [MINVALUE entier / NOMINVALUE]

Deux variables, en rapport avec les séquences, sont utilisables sous SQL:

- CURVAL numéro de séquence en cours,
- NEXTVAL prochaine valeur à attribuer.

Exemples:

```
CREATE SEQUENCE num_com
   INCREMENT BY 1 START WITH 1000 MAXVALUE 99999;
.....
INSERT INTO entete
   VALUES (num_com.NEXTVAL, '06', SYSDATE, 'p');
```

Voir aussi:

```
ALTER SEQUENCE nom_de_séquence [INCREMENT BY entier] [START WITH entier]
   [MAXVALUE entier / NOMAXVALUE] [MINVALUE entier / NOMINVALUE]
DROP SEQUENCE nom_de_séquence
```

Exemples:

```
ALTER SEQUENCE num_com NOMAXVALUE;
.....
DROP SEQUENCE num_com ;
```

CREATE SYNONYM

Cette commande crée un synonyme pour une table, une vue, une séquence, une procédure, une fonction ou un autre synonyme.

CREATE [PUBLIC] SYNONYM nom_du_synonyme FOR nom_de_l'objet

Ex :

CREATE PUBLIC SYNONYM DEPARTEMENT FOR MARCEL.DEPT ;

CREATE VIEW

Cette commande crée une vue et provoque un commit.

CREATE VIEW <nom de la vue> [(alias{,alias})]
AS SELECT ..... ;

Si ALIAS est spécifié, la vue utilise comme nom de colonne le nom de l'alias.
Il est possible de mettre à jour ou d'effacer des lignes dans une vue, si la vue est basée sur une et une seule table, et si le SELECT ne contient pas de clauses GROUP BY..., DISTINCT, ou des fonctions d'ensemble. Même chose pour l'insertion de lignes si la vue ne contient pas de colonnes définies par des expressions.
Si une vue est construite à partir d'un SELECT * FROM X... et si on ajoute des colonnes à la table X, la définition de la vue n'est plus valide. Il faut la détruire et la reconstruire.

Ex : Moyenne des prix moyens par article ?

```
CREATE VIEW prixmoy
AS SELECT refpiece, AVG(prix) moy
FROM prix GROUP BY refpiece;

SELECT AVG(moy) FROM prixmoy;
```

DELETE

Cette commande permet la supression d'enregistrements.

DELETE FROM <table>
[WHERE <conditions>];

Si la condition n'est pas spécifiée, tous les enregistrements sont suprimés.

Ex :

```
DELETE FROM prix
WHERE reffou = '10120Q';
```

DROP

Cette commande permet la suppression de différents objets.

DROP TABLE nom de table;

DROP VIEW nom de vue;

DROP INDEX nom de l'index [ON table];

DROP CLUSTER nom du cluster;

DROP SEQUENCE nom de la séquence;

DROP SYNONYME nom du synonyme;

GRANT

Cette commande crée des utilisateurs, assigne des mots de passe, définit des privilèges aux utilisateurs sur des tables ou sur des vues et provoque également un commit.

1ère forme :

GRANT [CONNECT][RESSOURCE][,DBA]
TO <utilisateur>{,<utilisateur>}
IDENTIFIED BY <mot de passe>{,<mot de passe>};

Cette première forme définit des utilisateurs, assigne des mots de passe et définit des privilèges aux utilisateurs. Tout utilisateur est autorisé ainsi à changer son mot de passe. Mais les autres droits ne peuvent être utilisés que par le DBA.
CONNECT permet à un utilisateur de se connecter.
RESSOURCE permet de créer des tables ou divers objets dans la base.
DBA permet d'utiliser toutes les commandes propres au DBA.
Il est possible de combiner ces différents droits.

Ex :

GRANT CONNECT,RESSOURCE TO ora1 IDENTIFIED BY ora;

2ème forme :

GRANT <privilège>{,<privilège>}/ALL
ON <nom de table> TO <utilisateur>/PUBLIC
[WITH GRANT OPTION]

Cette seconde forme distribue des privilèges à des utilisateurs sur certaines tables ou vues. Un privilège spécifie les commandes qu'un utilisateur est autorisé à utiliser sur une table ou une vue. Les privilèges sont : ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE.
ALL accorde tous les privilèges.
UPDATE peut être suivi d'une liste de colonnes.
GRANT UPDATE (<colonne>{,<colonne>}) TO ...;

Les mises à jour ne se font que sur les colonnes spécifiées.
<nom de table> peut être un nom de table ou de vue.

Si c'est une vue, les seuls privilèges autorisés sont : DELETE, INSERT, SELECT, UPDATE.
PUBLIC spécifie tous les utilisateurs.
WITH GRANT OPTION permet à celui qui reçoit un ou plusieurs privilèges de transmettre tout ou partie de ces privilèges.

Ex :

```
GRANT ALL ON PRIX TO ALAIN;

GRANT SELECT, UPDATE(statut) ON entete TO PUBLIC;
```

INSERT

Cette commande permet l'ajout de nouveaux enregistrements dans la table ou vue spécifiée.

INSER INTO <nom de table ou vue> [(<colonne{,<colonne>}]
VALUES (<valeur){,<valeur>}] / SELECT.....

Ex :

```
INSERT INTO prix (refpiece,nfou,reffou,prix)
VALUES ('51180','6','10120Q','0.60);
```

Une table *fou2* ayant la même structure que *fournisseurs* peut voir son contenu transféré de la façon suivante:

```
INSERT INTO fournisseurs
SELECT nfou,nomfou,adr,ville,region,c_post
FROM fou2;
```

RENAME

Cette commande permet de renommer une table ou une vue.

Exemple :

RENAME fournisseurs TO fourn;

REVOKE

Cette commande ôte des privilèges à des utilisateurs sur des tables ou sur des vues et provoque un commit.

REVOKE [CONNECT][,RESSOURCE][,DBA]
FROM <utilisateur>{,<utilisateur>}

Cette commande qui ne peut être utilisée que par le DBA supprime des privilèges à des utilisateurs.

Ex :

REVOKE RESSOURCE FROM ORA1,ORA2;
REVOKE DBA FROM ALAIN;

REVOKE <privilège>{,<privilège>} /ALL
ON <nom de table> FROM <utilisateur>{,utilisateur}/PUBLIC;

Cette forme supprime les privilèges sur une table ou une vue.
Mais si un utilisateur est autorisé à accéder à une table par plusieurs autres autorisations, cet utilisateur pourra continuer ses accès jusqu'au retrait de toutes les autorisations sur cet objet.
Supprimer un privilège à un utilisateur revient à supprimer tous les privilèges accordés par cet utilisateur à d'autres.

Ex :

REVOKE DELETE,INSERT ON prix FROM alain;

REVOKE ALL ON entete FROM PUBLIC;

SELECT

SELECT [ALL/DISTINCT] * / <exp>[(<alias>]{,<exp>[<alias>]}
FROM <table> [<alias>] {,<table> [<alias>]}
[WHERE <condition>]
[CONNECT BY <condition> [START WITH <condition>]]
[GROUP BY <exp> {,<exp>}
[HAVING <condition>]]
[{UNION/INTERSECT/MINUS SELECT ....}]
[ORDER BY <exp> [ASC/DESC]{,<exp> [ASC/DESC]}]
[FOR UPDATE OF <colonne>{,<colonne>} [NOWAIT]];

Retourne une relation dont les données proviennent d'une ou plusieurs autres relations.

Ex :

Select * from fournisseurs;
Renvoie toutes les lignes et colonnes de fournisseurs.

Select nfou from fournisseurs
where nomfou='Fnac Rennes';

Select refpiece,nfou from prix
where prix>100;

Select nomfou from fournisseurs
where nomfou like 'F%';

Select nfou,refpiece reference,prix*1.186 PTTC
from prix;
where nfou='27' or nfou='6';

Select distinct ville from fournisseurs;

Select ville,nomfou,from fournisseurs
order by ville,nomfou desc;
(<=> order by 1,2 desc)

```
Select refpiece,prix from prix
where prix between 1200 and 1400;

Select ncom,date,nfou from entete
where date<'1-Jun-84';
```

Exemples avec jointures.

```
Select fournisseurs.nomfou
from fournisseurs,entete
where statut="p"
and entete.nfou=fournisseurs.nfou;

Select f.nomfou
from fournisseurs f,entete e
where statut="p"
and e.nfou=f.nfou;
```

Rappel : La jointure est l'opération qui permet d'obtenir une relation constituée d'informations provenant de différentes relations (Voir Chapitre 1, § B, p.19)
Jointure externe : L'opérateur + permet de faire figurer dans le résultat les lignes satisfaisant la condition de jointure, plus celles n'ayant pas de correspondant.

```
Select nomfou,statut
from fournisseurs f,entete e
where ville="Paris"
and e.nfou=f.nfou(+);
```

Tous les fournisseurs habitant à Paris apparaîtront , même s'ils ne sont pas présents dans entete

Exercices résolus :

Donner le nom du fournisseur et les dates de commandes pour toutes les commandes en attente (statut P) ainsi que les fournisseurs sans commandes en attente. Le résultat se présentant sous la forme suivante :

Fnac Rennes	30/06/98
Fournisseur Express	10/06/98
Gibert Jeunes	Pas de commande en cours
Maison de Robert	Pas de commande en cours

```
Select f.nomfou,e.date
from fournisseurs f,entete e
where f.nfou=e.nfou
and e.statut='P'
UNION
select nomfou,'Pas de commande en cours'
from fournisseurs
where nfou NOT IN (select nfou from entete where statut='p');
```

Explications :

Le premier select fournit un ensemble de données, la condition portant sur le statut de la table entete.
Le deuxième select donne un deuxième ensemble de données égal au complément du premier dans fournisseurs.

TTITLE BTITLE

TTITLE : Affiche un titre sur chaque page lors de l'édition
BTITLE : Affiche un titre après chaque page lors de l'édition.

TTITLE [<spécification>] <texte> /OFF/ON
BTITLE .....Idem.....

Spécification pouvant prendre une des valeurs :

- LEFT
- RIGHT
- CENTER
- SKIP<n> (passage à la ligne)
- COL<n> (positionnement à la colonne n).

OFF supprime l'affichage du titre sans en supprimer la définition.

Ex :

```
TTITLE LEFT 'Vendredi 10 mars 1989'
SKIP COL10 'Liste des fournisseurs';
SELECT nomfou, adr,ville from fournisseurs;

BTITLE RIGHT 'Tournez la page SVP'
SELECT refpiece,prix from prix;
```

UPDATE

Modifie la valeur des champs pour les enregistrements répondant à la condition.

UPDATE <table>
SET <colonne>=<exp> {,<colonne>=<exp>}
WHERE <condition>

UPDATE <table>
SET (<colonne>{,<colonne>})=(SELECT.....)
WHERE <condition>;

Ex :

```
UPDATE prix
SET PRIX=1.1
WHERE nfou='27';
```