- dans une clause WHERE
- dans une clause ORDER BY (étudiée en 3.11)
- dans les ordres de manipulations de données (INSERT, UPDATE, DELETE étudiés au chapitre 2)

Le tableau qui suit donne le nom des principales fonctions (voir 3.10 pour plus de détails).

DE GROUPE	ARITHMETIQUES	DE CHAINES	DE DATES
SUM	NVL	NVL	NVL
COUNT	TO_CHAR	SUBSTR	TO_CHAR
VARIANCE	SQRT	LENGTH	ADD_MONTHS
MAX	ABS	INSTR	MONTHS_BETWEEN
MIN	POWER	TO_NUMBER	NEXT_DAY

Remarque 1.2
Toute expression dont au moins un des termes a la valeur NULL donne comme résultat la valeur NULL.

La fonction NVL (*Null VaLue*) permet de remplacer une valeur NULL par une valeur par défaut :

```
NVL (expr1, expr2)
```

prend la valeur *expr1*, sauf si *expr1* a la valeur NULL, auquel cas NVL prend la valeur *expr2*.

Exemple 1.6 `NVL(COMM, 0)`

Chapitre 2

Langage de manipulation des données

Le langage de manipulation de données est le langage permettant de modifier les informations contenues dans la base.

Il existe trois commandes SQL permettant d'effectuer les trois types de modification des données :

INSERT	ajout de lignes
UPDATE	mise à jour de lignes
DELETE	suppression de lignes

Ces trois commandes travaillent sur la base telle qu'elle était au début de l'exécution de la commande. Les modifications effectuées par les autres utilisateurs entre le début et la fin de l'exécution ne sont pas prises en compte (même pour les transactions validées).

2.1 Insertion

```
INSERT INTO table (col1,..., coln)
VALUES (val1,...,valn)
```

ou

```
INSERT INTO table (col1,..., coln)
SELECT ...
```

table est le nom de la table sur laquelle porte l'insertion. *col1,..., coln* est la liste des noms des colonnes pour lesquelles on donne une valeur. Cette liste est optionnelle. Si elle est omise, ORACLE prendra par défaut l'ensemble des colonnes de la table dans l'ordre où elles ont été données lors de la création de la table. Si une liste de colonnes est spécifiée, les colonnes ne figurant pas dans la liste auront la valeur NULL.

Exemples 2.1

(a)
```
INSERT INTO dept VALUES (10, 'FINANCES', 'PARIS')
```

(b)
```
INSERT INTO dept (lieu, nomd, dept)
    VALUES ('GRENOBLE', 'RECHERCHE', 20)
```

La deuxième forme avec la clause SELECT permet d'insérer dans une table des lignes provenant d'une table de la base. Le SELECT a la même syntaxe qu'un SELECT normal.

Exemple 2.2

Enregistrer la participation de MARTIN au groupe de projet numéro 10 :

```
INSERT INTO PARTICIPATION (MATR, CODEP)
SELECT MATR, 10
FROM EMP
WHERE NOME = 'MARTIN';
```

2.2 Modification

La commande UPDATE permet de modifier les valeurs d'un ou plusieurs champs, dans une ou plusieurs lignes existantes d'une table.

```
UPDATE table
SET col1 = exp1, col2 = exp2, ...
WHERE prédicat
```

ou

```
UPDATE table
SET (col1, col2,...) = (SELECT ...)
WHERE prédicat
```

table est le nom de la table mise à jour ; *col1*, *col2*, ... sont les noms des colonnes qui seront modifiées ; *exp1*, *exp2*,... sont des expressions. Elles peuvent aussi être un ordre SELECT renvoyant les valeurs attribuées aux colonnes (deuxième variante de la syntaxe).

Les valeurs de *col1*, *col2*... sont mises à jour dans toutes les lignes satisfaisant le prédicat. La clause WHERE est facultative. Si elle est absente, *toutes* les lignes sont mises à jour.

Exemples 2.3

(a) Faire passer MARTIN dans le département 10 :

```
UPDATE EMP SET DEPT = 10
WHERE NOME = 'MARTIN';
```

(b) Augmenter de 10 % les commerciaux :

```
UPDATE EMP
SET SAL = SAL * 1.1
WHERE POSTE = 'COMMERCIAL';
```

(c) Donner à CLEMENT un salaire 10 % au dessus de la moyenne des salaires des secrétaires :

```
UPDATE EMP
SET SAL = (SELECT AVG(SAL) * 1.10
           FROM EMP
           WHERE POSTE = 'SECRETAIRE')
WHERE NOME = 'CLEMENT'
```

On remarquera que la moyenne des salaires sera calculée pour les valeurs qu'avaient les salaires au début de l'exécution de la commande UPDATE et que les modifications effectuées sur la base pendant l'exécution de cette commande ne seront pas prises en compte.

(d) Enlever (plus exactement, mettre à la valeur "NULL") la commission de MARTIN :

```
UPDATE EMP
SET COMM = NULL
WHERE NOME = 'MARTIN'
```


2.3 Suppression

L'ordre DELETE permet de supprimer des lignes d'une table.

```
DELETE FROM table
WHERE prédicat
```

La clause WHERE indique quelles lignes doivent être supprimées. ATTENTION : cette clause est facultative ; si elle n'est pas précisée, TOUTES LES LIGNES DE LA TABLE SONT SUPPRIMEES (heureusement qu'il existe ROLLBACK !).

Exemple 2.4 `DELETE FROM dept WHERE dept = 10`

2.4 Transactions : Commit/Rollback

Une transaction est un ensemble de modifications de la base qui forment un tout indivisible : il faut effectuer ces modifications entièrement ou pas du tout, sous peine de laisser la base dans un état incohérent.

Une transaction commence au début d'une session de travail ou juste après la fin de la transaction précédente. L'utilisateur peut à tout moment valider (et terminer) la transaction en cours par la commande **COMMIT**. Les modifications deviennent alors définitives et visibles à toutes les autres transactions.

IMPORTANT : tant que la transaction n'est pas validée, les insertions, modifications et suppressions qu'elle a exécutées n'apparaissent pas aux autres transactions.

L'utilisateur peut annuler (et terminer) la transaction en cours par la commande **ROLLBACK**. Toutes les modifications depuis le début de la transaction sont annulées.

Ce mécanisme est utilisé par Oracle pour assurer l'intégrité de la base en cas de fin anormale d'une tâche utilisateur : Oracle lance automatiquement un ROLLBACK des transactions non terminées.

Si une erreur survient lors d'une transaction, un ROLLBACK est automatiquement effectué. Cela signifie par exemple que, si une erreur survient au cours d'une instruction UPDATE qui modifie plusieurs lignes, aucune ligne ne sera modifiée.

Remarque 2.1

Certains ordres SQL, notamment ceux de définitions de données, provoquent une validation automatique de la transaction en cours. Le fait de quitter SQLPLUS par EXIT provoque également un COMMIT.

On peut indiquer si la transaction écrira ou non dans la base :

```
SET TRANSACTION {READ ONLY | READ WRITE}
```

On peut aussi indiquer le degré de concurrence que l'on accepte pour cette transaction par rapport aux autres transactions (voir chapitre 5).

Chapitre 3

Interrogations

3.1 Syntaxe générale

L'ordre SELECT possède six clauses différentes, dont seules les deux premières sont obligatoires. Elles sont données ci-dessous, dans l'ordre dans lequel elles doivent apparaître, quand elles sont utilisées :

```
SELECT ...
FROM ...
WHERE ...
GROUP BY ...
HAVING ...
ORDER BY ...
```

3.2 Clause SELECT

Cette clause 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 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 ; il constituera le titre de la colonne dans l'affichage du résultat de la sélection. Le mot clé AS est optionnel.

"*" signifie que toutes les colonnes de la table sont sélectionnées.

Le mot clé facultatif DISTINCT ajouté derrière l'ordre SELECT permet d'éliminer les duplications : si, dans le résultat, plusieurs lignes sont identiques, une seule sera conservée.

Exemples 3.1

(a) `SELECT * FROM DEPT`