TP 2: Définition et manipulation des données

1. Introduction au langage SQL

1.1 Création d'une table

Question1 : Créer une table qui s'appelle BIBLIO enregistrant les livres vendus par une librairie. Nous lui donnerons la structure suivante :

Nom	type	contrainte	signification
Titre	Char(20)	NOT NULL UNIQUE	Titre du livre
Auteur	Char(12)	NOT NULL	Son auteur
Genre	Char(7)	NOT NULL	Son genre (policier, roman, BD,…)
Achat	Date	NOT NULL	Date d'achat du livre
Prix	Number(6,2)	NOT NULL	Son prix
Disponible	Char(1)	NOT NULL	Est-il disponible ? o (oui), n (non).

1.2 Remplissage d'une table

Question2 : Remplir, comme suit, la table Biblio déjà créée

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D
Candide	Voltaire	Essai	18/10/85	140	o
Les fleurs du mal	Baudelaire	Poème	01/01/78	120	n
Tintin au Tibet	Hergé	BD	10/11/90	70	o
La terre	Zola	roman	12/06/90	50	n
Madame Bovary	Flaubert	Roman	12/05/88	130	o
Manhattan transfer	Dos Passos	Roman	30/08/87	320	o
Tintin en Amérique	Hergé	BD	15/05/91	70	o
Du côté de ch Swann	Proust	Roman	08/12/78	200	o

1.3 Consultation de la table

La commande **SELECT** de consultation d'une table a une syntaxe très riche. Nous abordons maintenant celle-ci mais nous aurons l'occasion d'y revenir ultérieurement.

Syntaxe :

SELECT [ALL|DISTINCT] *[*|expression_1 alias_1, expression_2 alias_2, ...]*
FROM nom_table

Action :

Affiche les valeurs de *expression_i* pour toutes les lignes de table. *Expression_i* peut être une colonne ou une expression plus complexe. Le symbole * désigne l'ensemble des colonnes. Par défaut, toutes les lignes de table (**ALL**) sont affichées. Si **DISTINCT** est présent, les lignes identiques sélectionnées ne sont

affichées qu'une fois. Les valeurs de *expressioni* sont affichées dans une colonne ayant pour titre *expressioni* ou *aliasi* si celui-ci a été utilisé.

Remarque :

La commande SELECT affiche ses résultats en série. S'il y en a beaucoup, on ne voit que les derniers ... Pour avoir un affichage page par page, il faut utiliser la commande SQLPLUS : **set pause on**
Il faut valider avec la touche *Entrée* pour démarrer l'affichage. Après chaque page, on peut avoir la suivante par la touche *Return* ou arrêter par la touche *Suppr*. L'affichage page par page restera en vigueur tant qu'on n'aura pas émis la commande inverse set pause off.

Question3 : Afficher toutes les colonnes et lignes de la table biblio.
Question4 : Afficher les titres des livres, leurs auteurs et leurs prix.

1.4 Affichage des lignes vérifiant une condition

Syntaxe :

SELECT ....
WHERE *condition*

Action : seules les lignes vérifiant la *condition* sont affichées.

Question5 : Quels sont les livres (titre et prix) qui coûtent plus de 100 ?
Question6 : Quels sont les prix des livres de genre 'roman' ? (titre, prix et genre).

Nous pouvons réunir des conditions par les opérateurs logiques
AND ET logique
OR OU logique
NOT Négation logique

Question7 : Quels sont les livres de genre roman et qui coutent moins de 100 ?
Question8 : Quels sont les livres de genre roman ou BD ?
Question9 : Quels sont les livres qui ne sont ni roman ni bande dessinée ?

1.5 Affichage des lignes selon un ordre déterminé

Aux syntaxes précédentes, il est possible d'ajouter une clause indiquant l'ordre d'affichage désiré :

Syntaxe:

SELECT ....
ORDER BY *expression1* [**asc**|**desc**], *expression2* [**asc**|**dec**], ...

Action :

Les lignes résultat de la sélection sont affichées dans l'ordre de
1 : ordre croissant (**asc** qui est la valeur par défaut) ou décroissant (**desc**) de *expression1*
2 : en cas d'égalité de *expression1*, l'affichage se fait selon les valeurs de *expression2*
etc ..

Question10 : Ordonner la table biblio suivant des achats décroissants.
Question11 : Afficher la table biblio suivant l'ordre croissant des prix.
Question12 : Afficher la table biblio suivant l'ordre décroissant des prix et du genre.

1.6 Suppression de lignes dans une table

Syntaxe:

DELETE FROM *table* [**WHERE** *condition*]

Action :

Supprime les lignes de *table* vérifiant *condition*. Si cette dernière est absente, toutes les lignes sont détruites.

Question13 : Supprimer de la table biblio les livres de titre 'candide'.

1.7 Modification du contenu d'une table

Syntaxe :

update table set *colonne1 = expression1, colonne2 = expression2, ...*
[**where** *condition*]

Action :

Pour les lignes de *table* vérifiant *condition* (toutes les lignes s'il n'y a pas de condition), *colonnei* reçoit la valeur *expressioni*.

Question14 : Mettre tous les genres en majuscule.
Question15 : Augmenter les prix des romans de 5%.

1.8 Mise à jour définitive d'une table

Lorsqu'on apporte des modifications à une table, Oracle les génère en fait sur une copie de la table. Elles peuvent être alors rendues définitives ou bien être annulées par les commandes **COMMIT** et **ROLLBACK**.

Syntaxe : **COMMIT**
Action : rend définitives les mises à jour faites sur les tables depuis le dernier COMMIT.
Syntaxe : **ROLLBACK**
Action : Annule toutes modifications faites sur les tables depuis le dernier COMMIT.
Remarque : Un COMMIT est fait implicitement aux moments suivants :
a) A la déconnexion d'Oracle
b) Après chaque commande affectant la structure des tables : CREATE, ALTER, DROP.

1.9 Création d'une table à partir d'une autre table

On utilise ici une variante de CREATE :

Syntaxe:

CREATE TABLE *table2*
AS SELECT *colonne1, colonne2, ...*
FROM *table1* [**WHERE** *condition*]

Action :

Crée *table2* avec pour structure *colonne1, colonne2, ...* de *table1*.
Par ailleurs *table2* reçoit les lignes de *table1* vérifiant *condition*.

```
SQL>  create table romans as <-- on crée la table romans
        select * from biblio <-- avec la structure de la table biblio
        where upper(genre)='ROMAN'; <-- en ne conservant que les romans
Table created.
```

Question16 : Créer une table cheaps, avec pour colonnes : titre, auteur, prix de la table biblio n'enregistrant que les livres d'un prix inférieur à 100 francs.

Question17 : Crée une table appelée vide qui a la structure de la table biblio mais qui sera vide.

1.10 Ajout de lignes dans une table en provenance d'une autre table

Il est possible d'ajouter des lignes d'une table à une autre table lorsque leurs structures sont compatibles :

Syntaxe :

INSERT INTO *table1* [(*colonne1, colonne2, ...*)]
SELECT *colonnea, colonneb, ...* **FROM** *table2* **WHERE** *condition*

Action :

Les lignes de *table2* vérifiant *condition* sont ajoutées à *table1*. Les colonnes *colonnea, colonneb, ...* de *table2* sont affectées dans l'ordre à *colonne1, colonne2, ...* de *table1* et doivent donc être de type compatible.

Exemple

```
SQL> select * from biblio;
SQL> create table biblio2 as select * from biblio where disponible='n';
SQL> select * from biblio2;
SQL> insert into biblio2 select * from biblio where disponible='o';
SQL> select * from biblio2;
```