

Introduction à SQL sous ORACLE

Serge Tahé, université d'Angers, 1991

AVERTISSEMENT

Ce polycopié a été initialement écrit en 1991 et pratiquement pas remanié depuis. Certaines des informations qu'il contient sont désormais obsolètes. On trouvera un cours plus récent à l'Url [<http://tahe.developpez.com/divers/sql-firebird/>]. Il présente le langage SQL avec le SGBD libre Firebird.

ST, septembre 2001.



L'essentiel de l'ouvrage est tiré de la documentation officielle d'ORACLE. Cependant certains points sont inspirés de l'excellent ouvrage de Christian MAREE et Guy LEDANT : SQL Initiation, Programmation et Maîtrise paru chez EYROLLES.

SQL (*Structured Query Language*) est un langage standard de création, de maintenance et d'interrogation de bases de données relationnelles. Il est indépendant du SGBD utilisé. Si les exemples de ce document ont été écrits à l'aide du SGBD Oracle, ils peuvent cependant, pour la plupart, être reproduits avec tout SGBD relationnel. Sous Windows, on trouvera dans ce domaine outre Oracle des produits moins lourds tels Access, MySQL, SQL Server. Ils acceptent tous le langage SQL mais parfois avec des limites vis à vis de ce qu'accepte Oracle. Si de document est utilisé avec un autre produit qu'Oracle, on pourra ignorer toutes les informations spécifiques à ce SGBD, essentiellement celles concernant SQLPLUS l'outil d'Oracle pour interroger des bases avec SQL. Dans l'annexe, ont été rassemblés divers documents :

- 1 comment faire du SQL avec Access. Ce SGBD est très répandu sur les machines windows personnelles et il se trouve qu'il respecte une grande partie de la norme SQL. C'est l'outil idéal pour appréhender SQL chez soi.
- 2 comment installer Oracle sous une machine Linux ou Windows. C'est une bonne méthode pour apprendre à administrer Oracle mais elle est coûteuse en espace disque, 1 Go environ et en performances. Une machine peu puissante est facilement écrasée par les ressources nécessaires à Oracle.
- 3 comment installer MySQL sous une machine Linux ou Windows. MySQL est une base de données moins complète mais beaucoup plus légère qu'Oracle. Contrairement à Access, ce SGBD peut être utilisé dans des applications réelles essentiellement sur des machines Linux.
- 4 comment faire du SQL avec l'outil Microsoft Query. Celui-ci permet de se connecter à quasiment toute base de données sous windows (Access, Oracle, MySQL,...).

1 L'environnement SQLPLUS d'Oracle

SQLPLUS est l'outil d'Oracle permettant l'utilisation du langage SQL. Cependant il offre en plus diverses commandes de manipulation de commandes SQL, de formatage des affichages écran etc ... formant ce que nous appellerons l'environnement SQLPLUS et qui est présenté partiellement dans ce chapitre.

SQLPLUS s'appelle à partir d'Unix par la commande :

sqlplus

Apparaît alors le message de connexion suivant :

```
SQL*Plus: Version 3.0.7.1.1 - Production on Thu Oct 10 13:24:03 1991
Copyright (c) Oracle Corporation 1979, 1989. All rights reserved.

Enter user-name: serge
Enter password:
Connected to: ORACLE RDBMS V6.0.30.2.1, transaction processing option - Production
PL/SQL V1.0.30.0.1 - Production
```

Il vous est demandé votre nom ainsi que votre mot de passe. Consultez votre enseignant pour connaître ces deux informations. Pour vous connecter, vous pouvez aussi utiliser la syntaxe

```
sqlplus nom_utilisateur/mot_de_passe
```

Par exemple

```
sqlplus serge/serge
```

Une fois la connexion avec Oracle établie, SQLPLUS affiche son message d'attente :

```
SQL>
```

indiquant qu'il attend une commande SQL ou SQLPLUS :

Les commandes SQL permettent de créer, mettre à jour et exploiter les tables de données.

```
Ex : select * from biblio;
```

Les commandes SQLPLUS permettent de manipuler l'environnement dans lequel vont s'exécuter les commandes SQL :

- a éditer, sauvegarder, récupérer des commandes SQL
- b préciser le formatage désiré pour le résultat des requêtes SQL
- c diverses commandes

```
Ex : describe biblio
```

1.1 Syntaxe des commandes SQL

Voici quelques règles d'écriture des commandes SQL :

- 1 Elles peuvent s'écrire indifféremment en majuscules ou minuscules. Par la suite, nous écrirons les noms des tables et colonnes en majuscules et le reste en minuscules.
- 2 Une commande SQL se termine par ; ou / ou une ligne blanche :

<code>;</code>	indique la fin de la commande et demande son exécution
<code>/</code>	idem à ; mais doit être seul sur sa ligne.
<code>ligne blanche</code>	termine la commande sans lancer son exécution

- 3 Une commande SQL peut s'étaler sur plusieurs lignes. Après chaque ligne, l'interpréteur génère une ligne supplémentaire numérotée et ce tant qu'il n'a pas rencontré la fin de la commande.

```
a select * from biblio;  
b select *  
  2 from biblio; <--- 2 est le N° de ligne
```

sont deux commandes identiques.

1.2 Syntaxe des commandes SQLPLUS

Voici quelques règles d'écriture des commandes SQLPLUS :

- a La commande peut être entrée indifféremment en majuscules ou minuscules.
- b La plupart des commandes SQLPLUS ont une abbréviation. Par exemple la commande **input** peut être abrégée par **i**.
- c Une commande SQLPLUS peut être tapée sur plusieurs lignes, chaque ligne intermédiaire étant terminée par **-**. SQLPLUS commence la ligne suivante par **>** :

```
SQL> column genre -  
> heading 'GENRE DU LIVRE'
```

- d Une commande SQLPLUS ne se termine par rien de particulier. Cependant le point-virgule est accepté.

1.3 Quelques commandes SQLPLUS

Nous nous proposons ici de voir quelques commandes SQLPLUS qui nous seront utiles dans notre étude du langage SQL.

1.3.1 Sortie de SQLPLUS

syntaxe	exit
action	ramène au système d'exploitation

1.4 Exécuter une commande système

syntaxe	host commande_système
---------	------------------------------

action	exécute la commande du système d'exploitation.
--------	--

syntaxe	host
---------	-------------

action	fait apparaître le "prompt" du système d'exploitation. On peut alors taper des commandes quelconques. On revient à SQLPLUS par la commande exit.
--------	--

Exemples :

```
SQL> host pwd <-- répertoire courant ?  
/users/serge/oracle/sqlplus <-- résultat
```

```
SQL> host ll<-- contenu du répertoire courant ?  
total 0<-- rien
```

```
SQL> host >fic<-- on crée un fichier vide
```

```
SQL> host ll<-- vérification  
total 0  
-rw-rw-r--  1 serge    enseign      0 Oct 11 15:14 fic
```

```
SQL> host mkdir rep<-- on crée un répertoire
```

```
SQL> host ll<-- vérification  
total 1  
-rw-rw-r--  1 serge    enseign      0 Oct 11 15:14 fic  
drwxrwxr-x  2 serge    enseign    512 Oct 11 15:15 rep
```

```
SQL> host cd rep <-- on change de répertoire courant
```

```
SQL> host pwd <-- vérification  
/users/serge/oracle/sqlplus <-- ça n'a pas marché
```

```
SQL> host ll<-- vérification : le répertoire courant n'a effectivement pas changé  
total 1  
-rw-rw-r--  1 serge    enseign      0 Oct 11 15:14 fic  
drwxrwxr-x  2 serge    enseign    512 Oct 11 15:15 rep
```

On remarque qu'on ne peut changer de répertoire courant par la commande **host**. Essayons l'autre méthode :

```
SQL> host <-- on appelle le système
```

```
$ pwd<-- on est sous le système. Répertoire courant ?  
/users/serge/oracle/sqlplus
```

```
$ mkdir rep <-- on crée un répertoire
```

```
$ ll <-- vérification  
total 1  
drwxrwxr-x  2 serge    enseign    512 Oct 11 15:25 rep
```

```
$ cd rep <-- changement de répertoire courant
```

```
$ pwd<-- vérification  
/users/serge/oracle/sqlplus/rep <-- ça a marché
```

```
$ exit <-- retour à SQLPLUS
```

```
SQL> host pwd <-- répertoire courant ?  
/users/serge/oracle/sqlplus <-- ce n'est plus rep mais SQLPLUS de nouveau  
SQLPlus
```

Les deux exemples précédents montrent que le répertoire courant pour SQLPLUS est celui à partir duquel il a été lancé. Il ne semble pas possible d'en changer. Cette notion de répertoire courant est importante car c'est là que SQLPLUS rangera certains des fichiers qu'il produira.

1.4.1 Gestion du buffer SQL

Sous SQLPLUS, on entre des commandes SQL ou SQLPLUS. La dernière commande SQL entrée au clavier est enregistrée dans une zone appelée buffer SQL. Tant qu'elle est présente dans ce buffer, la commande peut être modifiée, sauvegardée, relancée, etc... Les commandes de gestion du buffer SQL sont des commandes SQLPLUS et obéissent donc à la syntaxe déjà présentée. Noter que les commandes SQLPLUS émises ne sont pas mémorisées.

1.4.1.1 Edition du buffer

Les commandes d'édition du buffer s'appliquent à une seule des lignes constituant la commande SQL qui y est contenue. Celle-ci est signalée par une étoile et est appelée ligne courante.

Exemple :

```
SQL> select * <-- commande SQL sur 3 lignes
  2  from biblio <-- elle est automatiquement enregistrée dans le buffer
  3  where prix>100;
```

```
SQL> list <-- commande SQLPLUS visualisant le buffer SQL
  1  select *
  2  from biblio
  3* where prix>100 <-- la ligne 3 est ligne courante
```

```
SQL> list 2 <-- on demande à voir la ligne n°2 du buffer
  2* from biblio <-- elle est devenue ligne courante
```

Voici les commandes d'édition du buffer SQL :

Commande	Abbréviation	But
APPEND texte	A texte	ajoute texte à la fin de la ligne courante
CHANGE /ancien/nouveau/	C /ancien/nouveau/	change texte ancien en texte nouveau dans la ligne courante
CHANGE /texte	C /texte	supprime texte dans la ligne courante
DEL		supprime la ligne courante
INPUT	I	entre en saisie de lignes supplémentaires
INPUT texte	I texte	ajoute texte au buffer
LIST	L	visualise toutes les lignes
LIST n	L n	visualise la ligne n° n
LIST *	L *	visualise la ligne courante
LIST LAST	L LAST	visualise la dernière ligne
LIST m n	L m n	visualise les lignes m à n
CLEAR BUFFER	CL BUFF	vide le buffer

Exemples

```
SQL> list <-- contenu du buffer SQL
  1  select *
  2  from biblio
  3* where prix>100
```

```
SQL> clear buffer <-- vide le buffer
buffer cleared
```

```
SQL> list <-- vérification
No lines in SQL buffer.
```

```
SQL> input <-- ajoute des lignes au buffer
  1  select *
  2  from biblio
```

```

3  where prix>100
4 <-- on termine par une ligne blanche pour que la commande
<-- ne soit pas exécutée

SQL> 1 <-- vérification (l=list)
1  select *
2  from biblio
3* where prix>100

SQL> del <-- supprime la ligne courante (3 ici)

SQL> 1 <-- vérification
1  select *
2* from biblio

SQL> 1 1 <-- visualise ligne 1 qui devient ligne courante
1* select *

SQL> 1 2 <-- visualise ligne 2
2* from biblio

SQL> i <-- ajoute des lignes (i=input)
3  where prix>100
4

SQL> 1 <-- vérification
1  select *
2  from biblio
3* where prix>100

SQL> c/100/200/ <-- change 100 en 200 dans la ligne courante (ligne 3 ci-dessus)
3* where prix>200 <-- résultat

SQL> 1 2 <-- ligne 2 devient ligne courante
2* from biblio

SQL> a 2 <-- ajoute 2 en fin de ligne courante (a=append)
2* from biblio2 <-- résultat

SQL> 1
1  select *
2  from biblio2
3* where prix>200

```

Une autre manière d'éditer le buffer SQL est d'utiliser un éditeur de texte par la commande EDIT. Celle-ci appelle l'éditeur dont le nom est défini par la variable système _EDITOR. On peut obtenir la liste de ces variables par la commande DEFINE :

```

SQL> define <-- liste des variables définies

DEFINE _EDITOR= "vi" (CHAR) <-- l'éditeur est ici vi.
DEFINE _O_VERSION = "ORACLE RDBMS V6.0.30.2.1, transaction processing option - Production PL/SQL
V1.0.30.0.1 - Production" (CHAR)
DEFINE _O_RELEASE = "6003002" (CHAR)

```

Dans l'exemple précédent, la commande EDIT copie le buffer dans un fichier appelé afiedt.buf du répertoire courant puis appelle l'éditeur vi pour éditer ce fichier. On modifie et sauvegarde le fichier par les commandes habituelles de l'éditeur vi. Il sera recopié dans le buffer SQL.

```

SQL> 1 <-- liste le buffer
1  select *
2  from biblio2
3* where prix>200

SQL> edit <-- édition du buffer avec vi
// changer 200 en 100
Wrote file afiedt.buf<-- création du fichier afiedt.buf

SQL> host ll<-- vérification
total 1
-rw-rw-r-- 1 serge  enseign  38 Oct 11 15:35 afiedt.buf

SQL> host cat afiedt.buf <-- contenu de afiedt.buf
select *
from biblio
where prix>100
/

SQL> 1 <-- contenu du nouveau buffer
1  select *
2  from biblio
3* where prix>100

```

1.4.2 Sauvegarde et récupération du buffer

syntaxe	save fichier
action	sauvegarde le buffer SQL dans fichier.

syntaxe	get fichier
action	le buffer SQL est chargé avec le contenu de fichier

Exemples

```
SQL> 1 <-- contenu du buffer
      1 select *
      2 from biblio
      3* where prix>100

SQL> save cmd1<-- le buffer est sauvegardé dans cmd1
Created file cmd1

SQL> host ll<-- vérification
total 2
-rw-rw-r-- 1 serge      enseign      38 Oct 11 15:35 afiedt.buf
-rw-rw-r-- 1 serge      enseign      38 Oct 11 15:49 cmd1.sql
<-- le fichier a en fait le suffixe .sql

SQL> host cat cmd1.sql <-- contenu du fichier ?
select *
from biblio
where prix>100
/

SQL> clear buffer <-- on vide le buffer SQL
buffer cleared

SQL> 1 <-- contenu du buffer ?
No lines in SQL buffer.<-- rien

SQL> get cmd1 <-- on charge le buffer avec cmd1
      1 select * <-- résultat
      2 from biblio
      3* where prix>100

SQL> 1 <-- vérification
      1 select *
      2 from biblio
      3* where prix>100
```

1.4.3 Exécution du buffer SQL

syntaxe	run (abréviation r)
action	exécute la commande SQL du buffer

Exemple :

```
SQL> 1 <-- contenu du buffer SQL
      1 select *
      2 from biblio
      3* where prix>100

SQL> run <-- exécution du buffer
      1 select *
      2 from biblio
      3* where prix>100
```

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D
Vipere au poing	Bazin	Roman	01-JAN-91	130	O
L'adieu aux armes	Hemingway	Roman	01-FEB-91	150	O

1.4.4 Gestion des fichiers de commandes

Il est possible de rassembler des commandes SQL dans un fichier texte appelé **fichier de commandes**. Les commandes ont la même syntaxe qu'en mode interactif. Le fichier peut être construit à l'aide d'un éditeur :

```
syntaxe | edit fichier_de_commandes
```

Il sera exécuté par la commande

```
syntaxe | start fichier_de_commandes
```

Exemples

```
SQL> edit cmd2<-- création du fichier de commandes SQL
```

```
SQL> host ll
total 3<-- contenu du répertoire courant
-rw-rw-r-- 1 serge      enseign      38 Oct 11 15:35 afiedt.buf
-rw-rw-r-- 1 serge      enseign      38 Oct 11 15:49 cmd1.sql
-rw-rw-r-- 1 serge      enseign     107 Oct 11 16:07 cmd2.sql
  <-- en réalité, c'est le fichier cmd2.sql qui a été créé
```

```
SQL> host cat cmd2.sql <-- contenu du fichier de commandes
select titre,auteur,prix from biblio where prix>100;
select titre,auteur,prix from biblio where prix>140;
```

```
SQL> start cmd2 <-- exécution du fichier de commandes cmd2
```

```
TITRE      AUTEUR      PRIX    <-- résultat du premier select
-----
Vipere au poing      Bazin      130
L'adieu aux armes    Hemingway  150

TITRE      AUTEUR      PRIX    <-- résultat du second select
-----
L'adieu aux armes    Hemingway  150
```

```
SQL> l <-- qu'y a-t-il dans le buffer ?
1* select titre,auteur,prix from biblio where prix>140 <-- la dernière commande exécutée
```

Un fichier de commandes sera le plus souvent construit à l'aide d'un éditeur. On peut aussi le construire à l'aide des commandes d'édition du buffer SQL ainsi que des commandes **get** et **save**.

1.5 Conclusion

Nous avons vu quelques commandes SQLPLUS qui faciliteront notre étude du langage SQL. D'autres seront présentées dans les chapitres ultérieurs au gré des besoins. Notons qu'il existe des moyens beaucoup plus conviviaux que SQLPLUS pour travailler avec Oracle notamment avec des interfaces graphiques.

2 Introduction au langage SQL

2.1 Préliminaires

Dans ce chapitre nous présentons les commandes SQL nécessaires à la **création** et à la **maintenance** de tables. Nous en donnons une version courte permettant de travailler rapidement. Leur syntaxe complète est disponible dans les guides de référence d'Oracle.

2.2 Les types de données d'Oracle

Lors de la création d'une table, il nous faudra indiquer le type des données que peut contenir une colonne de table. Nous présentons ici, les types les plus courants :

2.3 Les chaînes de caractères

Les types sont les suivants :

CHAR	un caractère
CHAR (N)	chaîne d'au plus N caractères (N<=255)
LONG	chaîne d'au plus 65535 caractères

Les constantes chaînes sont entourées d'apostrophes. Une apostrophe dans le texte doit être doublée :

'Jean'
'aujourd'hui'

2.4 Les nombres

Les types numériques sont les suivants :

NUMBER	nombre
NUMBER (N)	nombre entier de N chiffres
NUMBER (N, M)	nombre sur N positions dont M après la virgule
NUMBER (N, -M)	nombre entier de N positions arrondi au M ième chiffre précédant la virgule. Exemple : 123 au format NUMBER(3,-1) est représenté par 120.

Ces types n'ont pas tous la même représentation interne :

NUMBER	est représenté par un format "flottant" (floating point format) c'est à dire sous la forme <i>mantisse * 2^{exposant}</i> L'exposant est dans l'intervalle [-129,124]. La précision de la mantisse est de 126 bits (environ 37 chiffres décimaux). Les nombres flottants sont représentés de façon approchée.
NUMBER (. .)	est représenté par un format "fixe" (fixed point format) c'est à dire sous une forme décimale avec la virgule à un endroit fixe du nombre : $d_n d_{n-1} \dots d_1 d_0, d_{-1} d_{-2} \dots$ Un nombre en format "fixe" est représenté de façon exacte.

Notation des constantes numériques

3.4, -10.3, 1.4e+45 (1.4*10⁴⁵)

2.4.1 Les dates

Le type à utiliser est ici le type **DATE**. Une date est stockée sous la forme d'un nombre représentant siècle, année, mois, jour, heure, minute, seconde. Ce type sert donc également à stocker des **heures**.

Le format par défaut des constantes de type **DATE** est le suivant :

'DD-MMM-YY' avec D : Day, M : Month, Y : Year

Exemple : '10-APR-85' pour le 10 avril 1985

On peut

- . ajouter un nombre (de jours) à une date, pour avoir une nouvelle date.
- . soustraire deux dates pour avoir le nombre de jours les séparant
- . comparer deux dates

2.4.2 Les données nulles

Dans une table, une ligne peut avoir des colonnes sans valeur. On dit que la valeur de la colonne est la constante NULL. On peut tester la présence de cette valeur à l'aide des opérateurs

IS NULL et **IS NOT NULL**

2.4.3 Les conversions de type

Diverses fonctions permettent des changements de type :

TO_NUMBER	convertit une chaîne de caractères en nombre
TO_CHAR	convertit un nombre ou une date en chaîne de caractères
TO_DATE	convertit une chaîne de caractères ou un nombre en date.

2.5 Création d'une table

syntaxe	CREATE TABLE <i>table</i> (<i>nom_colonne1</i> <i>type_colonne1</i> <i>contrainte_colonne1</i> , <i>nom_colonne2</i> <i>type_colonne2</i> <i>contrainte_colonne2</i> , ...)						
action	crée la table <i>table</i> avec les colonnes indiquées : <table><tr><td><i>nom_colonnei</i></td><td>nom de la colonne i à créer</td></tr><tr><td><i>type_colonnei</i></td><td>type des données de la colonne i : char(30) number(6,2)</td></tr><tr><td><i>contrainte_colonnei</i></td><td>contrainte que doivent respecter les données de la colonne i. En voici deux : NOT NULL aucune valeur nulle n'est permise dans la colonne. UNIQUE aucune valeur ne peut apparaître plusieurs fois dans la colonne.</td></tr></table>	<i>nom_colonnei</i>	nom de la colonne i à créer	<i>type_colonnei</i>	type des données de la colonne i : char(30) number(6,2)	<i>contrainte_colonnei</i>	contrainte que doivent respecter les données de la colonne i. En voici deux : NOT NULL aucune valeur nulle n'est permise dans la colonne. UNIQUE aucune valeur ne peut apparaître plusieurs fois dans la colonne.
<i>nom_colonnei</i>	nom de la colonne i à créer						
<i>type_colonnei</i>	type des données de la colonne i : char(30) number(6,2)						
<i>contrainte_colonnei</i>	contrainte que doivent respecter les données de la colonne i. En voici deux : NOT NULL aucune valeur nulle n'est permise dans la colonne. UNIQUE aucune valeur ne peut apparaître plusieurs fois dans la colonne.						

Par exemple, nous souhaitons créer une table enregistrant les livres vendus par une librairie. Nous lui donnerons la structure suivante :

Name	Type	Contrainte	Signification
TITRE	CHAR(20)	NOT NULL UNIQUE	Titre du livre
AUTEUR	CHAR(15)	NOT NULL	Son auteur
GENRE	CHAR(15)	NOT NULL	Son genre (Roman, Poésie, Policier, BD, ..)
ACHAT	DATE	NOT NULL	Date d'achat du livre

Name	Type	Contrainte	Signification
PRIX	NUMBER(6,2)	NOT NULL	Son prix
DISPONIBLE	CHAR(1)	NOT NULL	Est-il disponible ? O (oui), N (non)

La table s'appellera BIBLIO. La commande pour la créer est la suivante :

```
SQL> create table biblio
2  ( titre char(20) not null unique,
3  auteur char(15) not null,
4  genre char(15) not null,
5  achat date not null,
6  prix number(6,2) not null,
7  disponible char(1) not null);
```

Table created.

2.6 Afficher la structure d'une table

syntaxe	describe table - commande SQLPLUS
action	affiche la structure de table

Exemple

```
SQL> describe biblio
Name                               Null?    Type
-----
TITRE                             NOT NULL CHAR(20)
AUTEUR                             NOT NULL CHAR(15)
GENRE                             NOT NULL CHAR(15)
ACHAT                             NOT NULL DATE
PRIX                              NOT NULL NUMBER(6,2)
DISPONIBLE                         NOT NULL CHAR(1)
```

2.7 Remplissage d'une table

syntaxe	insert into table [(colonne1, colonne2, ..)] values (valeur1, valeur2,)
action	ajoute une ligne (valeur1, valeur2, ..) à table. Ces valeurs sont affectées à colonne1, colonne2,... si elles sont présentes, sinon aux colonnes de la table dans l'ordre où elles ont été définies.

Exemple

```
SQL > edit insert          <-- construction du fichier de commandes insert
SQL> host cat insert.sql    <-- contenu du fichier de commandes insert.sql

insert into biblio values ('Candide','Voltaire','Essai','18-oct-85',140,'o');
insert into biblio values ('Les fleurs du mal','Baudelaire','Poème','01-jan-78',120,'n');
insert into biblio values ('Tintin au Tibet','Hergé','BD','10-nov-90',70,'o');
insert into biblio values ('Du côté de chez Swann','Proust','Roman','08-dec-78',200,'o');
insert into biblio values ('La terre','Zola','roman','12-jun-90',50,'n');
insert into biblio values ('Madame Bovary','Flaubert','Roman','12-mar-88',130,'o');
insert into biblio values ('Manhattan transfer','Dos Passos','Roman','30-aug-87',320,'o');
insert into biblio values ('Tintin en Amérique','Hergé','BD','15-may-91',70,'o');
```

```
SQL> start insert
```

```
1 row created.
1 row created.
1 row created.
```

```
insert into biblio values ('Du côté de chez Swann','Proust','Roman','08-dec-78',200,'o')
```

```
*
ERROR at line 1:  <--- 'Du côté de chez Swann' fait 21 caractères pour une colonne
                  <--- de 20 positions.
```

```
ORA-01401: inserted value too large for column
```

```
1 row created.
1 row created.
1 row created.
1 row created.
```

```
SQL> select * from biblio; <-- affichage du contenu de la table
```

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D
Candide	Voltaire	Essai	18-OCT-85	140	o
Les fleurs du mal	Baudelaire	Poème	01-JAN-78	120	n
Tintin au Tibet	Hergé	BD	10-NOV-90	70	o
La terre	Zola	roman	12-JUN-90	50	n
Madame Bovary	Flaubert	Roman	12-MAR-88	130	o
Manhattan transfer	Dos Passos	Roman	30-AUG-87	320	o
Tintin en Amérique	Hergé	BD	15-MAY-91	70	o

```
7 rows selected.
```

Remarques

- 1 Oracle ne tronque pas les chaînes trop grandes pour une colonne. Il les refuse en déclarant une erreur.
- 2 A l'affichage, chaque colonne est titrée avec son nom. Celui-ci est tronqué si la place manque. Exemple D pour DISPONIBLE.

Vérifions qu'une nouvelle insertion se fait en fin de table :

```
SQL> host cat insert2.sql <-- nouveau fichier de commandes
insert into biblio values ('Du côté de ch. Swann','Proust','Roman','08-dec-78',200,'o');
```

```
SQL> start insert2 <-- exécution de l'insertion
```

```
1 row created.
```

```
SQL> select * from biblio; <-- contenu de la table
```

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

```
8 rows selected.
```

2.8 Consultation de la table

2.8.1 Introduction

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 dans des chapitres ultérieurs.

syntaxe **SELECT** [**ALL** | **DISTINCT**] [***** | *expression1 alias1, expression2 alias2, ...*]
FROM *table*

action affiche les valeurs de *expressioni* pour toutes les lignes de table. *expressioni* 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
```

Exemples

```
SQL> select * from biblio;      <-- affichage de toutes les colonnes
```

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

8 rows selected.

```
SQL > select titre, auteur from biblio;
```

TITRE	AUTEUR
Candide	Voltaire
Les fleurs du mal	Baudelaire
Tintin au Tibet	Hergé
La terre	Zola
Madame Bovary	Flaubert
Manhattan transfer	Dos Passos
Tintin en Amérique	Hergé
Du côté de ch. Swann	Proust

8 rows selected.

```
SQL> select titre,prix from biblio;
```

TITRE	PRIX
Candide	140
Les fleurs du mal	120
Tintin au Tibet	70
La terre	50
Madame Bovary	130
Manhattan transfer	320
Tintin en Amérique	70
Du côté de ch. Swann	200

8 rows selected.

```
SQL> select titre TITRE_DU_LIVRE, prix PRIX_ACHAT from biblio;  
      <-- utilisation d'alias
```

TITRE_DU_LIVRE	PRIX_ACHAT
Candide	140
Les fleurs du mal	120
Tintin au Tibet	70
La terre	50
Madame Bovary	130
Manhattan transfer	320
Tintin en Amérique	70
Du côté de ch. Swann	200

8 rows selected.

Nous verrons ultérieurement des commandes SQLPLUS permettant un affichage plus sophistiqué.

2.8.2 Affichage des lignes vérifiant une condition

syntaxe		SELECT WHERE condition
---------	--	--------------------------------

action | seules les lignes vérifiant la *condition* sont affichées

Exemples

```
SQL> select titre,prix from biblio where prix>100;
```

TITRE	PRIX
Candide	140
Les fleurs du mal	120
Madame Bovary	130
Manhattan transfer	320
Du côté de ch. Swann	200

```
SQL> select titre,prix,genre
2   from biblio
3   where genre='Roman';
```

TITRE	PRIX	GENRE
Madame Bovary	130	Roman
Manhattan transfer	320	Roman
Du côté de ch. Swann	200	Roman

Un des livres a le genre 'roman' et non 'Roman'. Nous utilisons la fonction *upper* qui transforme une chaîne de caractères en majuscules pour avoir tous les romans.

```
SQL>
1  select titre,prix,genre
2  from biblio
3  where upper(genre)='ROMAN'
```

TITRE	PRIX	GENRE
La terre	50	roman
Madame Bovary	130	Roman
Manhattan transfer	320	Roman
Du côté de ch. Swann	200	Roman

Nous pouvons réunir des conditions par les opérateurs logiques

AND | ET logique
OR | OU logique
NOT | Négation logique

```
SQL> SQL>
1  select titre,prix,genre
2  from biblio
3  where upper(genre)='ROMAN'
4  and prix<100;
```

TITRE	PRIX	GENRE
La terre	50	roman

```
SQL> select titre,genre from biblio;
```

TITRE	GENRE
Candide	Essai
Les fleurs du mal	Poème
Tintin au Tibet	BD
La terre	roman
Madame Bovary	Roman
Manhattan transfer	Roman
Tintin en Amérique	BD
Du côté de ch. Swann	Roman

8 rows selected.

```
SQL>
1  select titre,genre from biblio
```

```

2 where upper(genre)='ROMAN' or upper(genre)='BD'

TITRE
-----
Tintin au Tibet      BD
La terre             roman
Madame Bovary        Roman
Manhattan transfer   Roman
Tintin en Amérique   BD
Du côté de ch. Swann Roman

6 rows selected.

SQL>
1 select titre,genre from biblio
2 where NOT( upper(genre)='ROMAN' or upper(genre)='BD')

```

```

TITRE
-----
Candide      Essai
Les fleurs du mal Poème

```

```
SQL> select titre,achat from biblio;
```

```

TITRE
-----
Candide      18-OCT-85
Les fleurs du mal 01-JAN-78
Tintin au Tibet 10-NOV-90
La terre      12-JUN-90
Madame Bovary 12-MAR-88
Manhattan transfer 30-AUG-87
Tintin en Amérique 15-MAY-91
Du côté de ch. Swann 08-DEC-78

```

```
8 rows selected.
```

```

SQL>
1 select titre,achat from biblio
2 where achat>'31-dec-87'

```

```

TITRE
-----
Tintin au Tibet 10-NOV-90
La terre        12-JUN-90
Madame Bovary   12-MAR-88
Tintin en Amérique 15-MAY-91

```

```

SQL> select titre,prix from biblio
2 where prix between 100 and 150; <-- prix dans l'intervalle [100..150]

```

```

TITRE
-----
Madame Bovary      136.5

```

2.8.3 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 <i>expression1</i> [asc desc], <i>expression2</i> [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 <i>expression1</i> 2 : en cas d'égalité de <i>expression1</i> , l'affichage se fait selon les valeurs de <i>expression2</i> etc ..

Exemples

```

SQL> select titre, genre,prix,achat from biblio
2 order by achat desc; <-- ordre décroissant des dates d'achat

```

```

TITRE
-----
Tintin en Amérique   BD

```

Tintin au Tibet	BD	70	10-NOV-90
La terre	roman	50	12-JUN-90
Madame Bovary	Roman	130	12-MAR-88
Manhattan transfer	Roman	320	30-AUG-87
Candide	Essai	140	18-OCT-85
Du côté de ch. Swann	Roman	200	08-DEC-78
Les fleurs du mal	Poème	120	01-JAN-78

8 rows selected.

```
SQL>
  1 select titre, genre,prix,achat from biblio
  2 order by prix;      <-- ordre croissant (par défaut) des prix
```

TITRE	GENRE	PRIX	ACHAT
La terre	roman	50	12-JUN-90
Tintin au Tibet	BD	70	10-NOV-90
Tintin en Amérique	BD	70	15-MAY-91
Les fleurs du mal	Poème	120	01-JAN-78
Madame Bovary	Roman	130	12-MAR-88
Candide	Essai	140	18-OCT-85
Du côté de ch. Swann	Roman	200	08-DEC-78
Manhattan transfer	Roman	320	30-AUG-87

8 rows selected.

```
SQL>
  1 select titre, genre,prix,achat from biblio
  2 order by genre desc;      <-- ordre décroissant du genre
```

TITRE	GENRE	PRIX	ACHAT
La terre	roman	50	12-JUN-90
Madame Bovary	Roman	130	12-MAR-88
Du côté de ch. Swann	Roman	200	08-DEC-78
Manhattan transfer	Roman	320	30-AUG-87
Les fleurs du mal	Poème	120	01-JAN-78
Candide	Essai	140	18-OCT-85
Tintin au Tibet	BD	70	10-NOV-90
Tintin en Amérique	BD	70	15-MAY-91

8 rows selected.

```
SQL >
  1 select titre, genre,prix,achat from biblio
  2 order by genre desc, prix;      <--- tri double critère
                                     <-- 1 : ordre décroissant du genre
                                     <-- 2 : ordre croissant du prix
```

TITRE	GENRE	PRIX	ACHAT
La terre	roman	50	12-JUN-90
Madame Bovary	Roman	130	12-MAR-88
Du côté de ch. Swann	Roman	200	08-DEC-78
Manhattan transfer	Roman	320	30-AUG-87
Les fleurs du mal	Poème	120	01-JAN-78
Candide	Essai	140	18-OCT-85
Tintin au Tibet	BD	70	10-NOV-90
Tintin en Amérique	BD	70	15-MAY-91

8 rows selected.

```
SQL>
  1 select titre, genre,prix,achat from biblio
  2 order by genre desc, prix desc;      <-- tri double critère
                                     <-- 1 : ordre décroissant du genre
                                     <-- 2 : ordre décroissant du prix
```

TITRE	GENRE	PRIX	ACHAT
La terre	roman	50	12-JUN-90
Manhattan transfer	Roman	320	30-AUG-87
Du côté de ch. Swann	Roman	200	08-DEC-78
Madame Bovary	Roman	130	12-MAR-88
Les fleurs du mal	Poème	120	01-JAN-78
Candide	Essai	140	18-OCT-85
Tintin au Tibet	BD	70	10-NOV-90
Tintin en Amérique	BD	70	15-MAY-91

8 rows selected.

2.9 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.

Exemples

```
SQL> select titre from biblio;    <-- liste des titres
```

```
TITRE
-----
Candide
Les fleurs du mal
Tintin au Tibet
La terre
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann
```

```
8 rows selected.
```

```
SQL> delete from biblio where titre='Candide';
```

```
1 row deleted.
```

```
SQL> select titre from biblio;    <-- vérification
```

```
TITRE
-----
Les fleurs du mal
Tintin au Tibet
La terre
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann
```

```
7 rows selected. <-- une ligne a été supprimée
```

2.10 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*.

Exemples

```
SQL> select genre from biblio;
```

```
GENRE
-----
Poème
BD
roman
Roman
Roman
BD
Roman
```

```
7 rows selected.
```

On met tous les genres en majuscules :

```
SQL> update biblio set genre=upper(genre);
```

```
7 rows updated.
```

```
SQL> select genre from biblio;    <--- vérification
```

```
GENRE
```

```

-----
POEME
BD
ROMAN
ROMAN
ROMAN
BD
ROMAN

7 rows selected.

```

```
SQL> select genre,prix from biblio;
```

GENRE	PRIX
POEME	120
BD	70
ROMAN	50
ROMAN	130
ROMAN	320
BD	70
ROMAN	200

```
7 rows selected.
```

Le prix des romans augmente de 5% :

```
SQL> update biblio set prix=prix*1.05 where genre='ROMAN';
```

```
4 rows updated.
```

```
SQL> select genre,prix from biblio; <-- vérification
```

GENRE	PRIX
POEME	120
BD	70
ROMAN	52.5
ROMAN	136.5
ROMAN	336
BD	70
ROMAN	210

```
7 rows selected.
```

2.11 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.

Exemples

```
SQL> select titre from biblio;<-- liste des titres
```

TITRE
Les fleurs du mal
Tintin au Tibet

```

La terre
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann

7 rows selected.

SQL> commit; <-- sauvegarde de l'état actuel de la base

Commit complete.

Suppression d'un titre

SQL> delete from biblio where titre='La terre';

1 row deleted.

SQL> select titre from biblio;    <-- vérification

TITRE
-----
Les fleurs du mal
Tintin au Tibet
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann

6 rows selected.    <--- le titre 'La terre' n'est plus présent

SQL> rollback;    <-- retour à l'état de la base lors du dernier commit.

Rollback complete.

SQL> select titre from biblio; <-- Vérification

TITRE
-----
Les fleurs du mal
Tintin au Tibet
La terre    <-- Le titre 'La terre' est récupéré
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann

7 rows selected.    <-- on est bien revenu à l'état antérieur.

SQL> select prix from biblio;

      PRIX
-----
      120
       70
      52.5
     136.5
      336
       70
      210

7 rows selected.

Tous les prix sont mis à zéro.

SQL> update biblio set prix=0;

7 rows updated.

SQL> select prix from biblio;    <-- vérification

PRIX
-----
0
0
0
0
0
0
0
0

7 rows selected.

```

```
SQL> rollback; <-- retour au dernier état validé de la base.
```

Rollback complete.

```
SQL> select prix from biblio; <-- vérification
```

```
      PRIX
-----
      120 <-- on est bien revenu à l'état antérieur
       70
      52.5
     136.5
      336
       70
      210
```

7 rows selected.

2.12 Mise en forme de l'affichage écran

Il existe diverses commandes SQLPLUS (et non SQL) permettant le contrôle de l'affichage écran.

2.12.1 Contrôle de l'affichage des colonnes

2.12.1.1 Titre de la colonne

syntaxe	COLUMN <i>nom_colonne</i> HEADING <i>titre_colonne</i>
action	donne le titre <i>titre_colonne</i> à la colonne indiquée.

Exemples

```
SQL> select titre,prix from biblio;
```

TITRE	PRIX
-----	-----
Les fleurs du mal	120
Tintin au Tibet	70
La terre	52.5
Madame Bovary	136.5
Manhattan transfer	336
Tintin en Amérique	70
Du côté de ch. Swann	210

7 rows selected.

```
SQL> column titre heading 'Titre du livre' <-- on donne un titre à la colonne titre
```

```
SQL> column prix heading 'Prix d'achat' <-- ainsi qu'à la colonne prix
```

```
SQL> select titre,prix from biblio; <-- vérification
```

Titre du livre	Prix d'achat
-----	-----
Les fleurs du mal	120
Tintin au Tibet	70
La terre	52.5
Madame Bovary	136.5
Manhattan transfer	336
Tintin en Amérique	70
Du côté de ch. Swann	210

7 rows selected.

2.12.2 Format d'une colonne numérique ou texte

syntaxe	COLUMN <i>nom_colonne</i> FORMAT <i>format</i>
action	La colonne indiquée est formatée au format indiqué. Différents formats existent pour les divers types de colonnes :

Colonne de texte : Le format **An** permet d'afficher les valeurs sur **n** positions.

Colonne de nombres

format	signification
9999	nombres cadrés sur 4 positions
0999	idem avec affichage des zéros de tête
9990	idem - affiche de plus 0 au lieu d'un blanc si la valeur est nulle
\$9999	fait précéder la valeur par \$
B9999	affiche une valeur nulle comme un blanc
9999MI	affiche - derrière une valeur négative
9999PR	affiche les valeurs négatives entre parenthèses
9,999	une virgule sera placée à la position indiquée
99.99	un point décimal sera placé à l'endroit indiqué
9.99EEEE	la valeur est affichée en notation scientifique

Exemples

```
SQL> column prix format 9999.99
SQL> select prix from biblio;
```

Prix d'achat

```
-----
      120.00
       70.00
       52.50
      136.50
      336.00
       70.00
      210.00
```

7 rows selected.

```
SQL> column prix format 0999.99
SQL> select prix from biblio
```

Prix d'achat

```
-----
      0120.00
      0070.00
      0052.50
      0136.50
      0336.00
      0070.00
      0210.00
```

7 rows selected.

```
SQL> column prix format 999.99EEEE
SQL> select prix from biblio;
```

Prix d'achat

```
-----
      1.20E+02
      7.00E+01
      5.25E+01
      1.37E+02
      3.36E+02
      7.00E+01
      2.10E+02
```

7 rows selected.

```
SQL> column titre format A20
SQL> select titre from biblio;
```

```

Titre du livre
-----
Les fleurs du mal
Tintin au Tibet
La terre
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann

7 rows selected.

SQL> column titre format A15
SQL> select titre from biblio;

Titre du livre
-----
Les fleurs du m    <-- les titres sont affichés à raison de 15 caractères par ligne.
al
Tintin au Tibet
La terre
Madame Bovary
Manhattan trans
fer
Tintin en Améri
que
Du côté de ch.
Swann
7 rows selected.

```

2.12.3 Affichage d'une colonne de dates

Le format implicite d'affichage d'une date est le format A9 : DD-MON-YY. Nous ne pouvons le changer. Ce qui est possible, c'est d'utiliser une fonction de conversion de la date en chaîne de caractères :

définition	TO_CHAR(date,format)
paramètres	date : la date à convertir format : le format de conversion
résultat	une date sous forme de chaîne de caractères.

Il existe de nombreux formats de conversion. Voici quelques-uns des éléments qui peuvent le composer :

Elément	Signification
YYYY, YYY, YY, Y	Les 4 chiffres de l'année, ou les 3 derniers chiffres, ou les 2 derniers, ou le dernier
YEAR	Année en toutes lettres
MM	N° du mois [1..12].
MONTH	Nom du mois sur 9 positions
MON	Nom du mois abrégé aux 3 premières lettres.
WW	N° de la semaine dans l'année [1..52].
DDD	Jour de l'année [1..366]
DD	Jour du mois [1..31]
D	Jour de la semaine [1..7]
DAY	Nom du jour
DY	Nom du jour abrégé aux 3 premières lettres.
"texte"	texte sera placé tel quel dans la date

Nous donnons dans la suite divers exemples de formatage de date. Nous utilisons pour cela :

la fonction **SYSDATE**

Elle donne la date du jour. Le résultat est du type date.

la table **DUAL**

C'est une table particulière définie sous Oracle et qui permet l'utilisation de la commande **SELECT** lorsqu'on n'a pas de colonne de table à afficher.

Exemples

```
SQL > select sysdate from dual;
```

```
SYSDATE
-----
16-OCT-91
```

```
SQL> select to_char(sysdate,'DD MONTH YYYY') from dual;
```

```
TO_CHAR(SYSDATE,'DDMONTHYYYY')
-----
16 OCTOBER 1991
```

```
SQL> select to_char(sysdate,'DD/MM/YY') from dual;
```

```
TO_CHAR(SYSDATE,'DD/MM/YY')
-----
16/10/91
```

```
SQL> select to_char(sysdate,'Day DD MONTH YYYY') from dual;
```

```
TO_CHAR(SYSDATE,'DAYDDMONTHYYYY')
-----
Wednesday 16 OCTOBER 1991
```

```
SQL> columnn achat format A8<-- colonne achat sur 8 positions
```

```
SQL> select titre, to_char(achat,'DD/MM/YY') ACHAT from biblio;
```

TITRE	ACHAT
Les fleurs du mal	01/01/78
Tintin au Tibet	10/11/90
La terre	12/06/90
Madame Bovary	12/03/88
Manhattan transfer	30/08/87
Tintin en Amérique	15/05/91
Du côté de ch. Swann	08/12/78

7 rows selected.

2.13 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*.

Exemples

```
SQL> create table romans as <-- on crée la table romans
2 select * from biblio <-- avec la structure de la table biblio
3 where genre='ROMAN'; <-- en ne conservant que les romans
```

Table created.

```
SQL> describe romans <-- vérification de la structure de la table romans
```

Name	Null?	Type
TITRE	NOT NULL	CHAR(20)
AUTEUR	NOT NULL	CHAR(15)
GENRE	NOT NULL	CHAR(15)
ACHAT	NOT NULL	DATE
PRIX	NOT NULL	NUMBER(6,2)

```
DISPONIBLE      NOT NULL CHAR(1)
```

```
SQL> select titre,genre from romans;  <-- Qu'y-a-t-il dans la table ?
```

TITRE	GENRE	
La terre	ROMAN	<-- Que des romans
Madame Bovary	ROMAN	
Manhattan transfer	ROMAN	
Du côté de ch. Swann	ROMAN	

```
SQL> create table cheap as          <-- création de la table cheap
  2  select titre,auteur,prix from biblio  <-- avec pour colonnes titre, auteur,prix
      <-- de biblio
  3  where prix<100;                  <-- n'enregistrant que les livres d'un prix
                                      <-- inférieur à 100 francs
```

Table created.

```
SQL> describe cheap <-- vérification de la structure de la table cheap
```

Name	Null?	Type
TITRE	NOT NULL	CHAR(20)
AUTEUR	NOT NULL	CHAR(15)
PRIX	NOT NULL	NUMBER(6,2)

```
SQL> select * from cheap; <-- contenu de la table cheap
```

TITRE	AUTEUR	PRIX	
Tintin au Tibet	Hergé	70	<-- que des prix inférieurs à 50 francs
La terre	Zola	52.5	
Tintin en Amérique	Hergé	70	

```
SQL> create table vide as <-- on crée une table appelée vide
  2  select * from biblio <-- qui a la structure de la table biblio
  3  where 1=2; <-- mais qui sera vide (condition 1=2 toujours fausse)
```

Table created.

```
SQL> describe vide <--- structure de la table vide
```

Name	Null?	Type
TITRE	NOT NULL	CHAR(20)
AUTEUR	NOT NULL	CHAR(15)
GENRE	NOT NULL	CHAR(15)
ACHAT	NOT NULL	DATE
PRIX	NOT NULL	NUMBER(6,2)
DISPONIBLE	NOT NULL	CHAR(1)

```
SQL> select * from vide; <-- contenu de la table vide
```

```
no rows selected <-- elle est vide
```

2.14 Obtenir la liste des tables créées

Oracle enregistre des informations sur les tables créées par un utilisateur dans une table (une vue en fait) appelée **TABS** dont la structure est la suivante :

```
SQL> describe tabs
```

Name	Null?	Type
TABLE_NAME	NOT NULL	CHAR(30)
TABLESPACE_NAME	NOT NULL	CHAR(30)
CLUSTER_NAME	NOT NULL	CHAR(30)
PCT_FREE	NOT NULL	NUMBER
PCT_USED	NOT NULL	NUMBER
INI_TRANS	NOT NULL	NUMBER
MAX_TRANS	NOT NULL	NUMBER
INITIAL_EXTENT		NUMBER
NEXT_EXTENT		NUMBER
MIN_EXTENTS	NOT NULL	NUMBER


```

MAX_EXTENTS    NOT NULL    NUMBER
PCT_INCREASE   NOT NULL    NUMBER
BACKED_UP      CHAR(1)

```

Nous n'entrerons pas dans les détails de cette structure. Seulement nous remarquons que le nom des tables est enregistré dans la colonne **table_name**. Nous obtenons donc la liste des tables créées par la commande :

```
select table_name from tabs
```

Exemple

```
SQL> select table_name from tabs;
```

```

TABLE_NAME
-----
BIBLIO
CHEAP
ROMANS
VIDE

```

```
4 rows selected.
```

2.15 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 colonne1, colonne2, ... FROM table2 WHERE condition

```

action

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

Exemple

```
select * from biblio;
```

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D	
Les fleurs du mal	Baudelaire	POEME	01-JAN-78	60		n
Tintin au Tibet	Hergé	BD	10-NOV-90	70		o
La terre	Zola	ROMAN	12-JUN-90	52.5		n
Madame Bovary	Flaubert	ROMAN	12-MAR-88	136.5		o
Manhattan transfer	Dos Passos	ROMAN	30-AUG-87	336		o
Tintin en Amérique	Hergé	BD	15-MAY-91	70		o
Du côté de ch. Swann	Proust	ROMAN	08-DEC-78	210		o
Le père Goriot	Balzac	ROMAN	01-SEP-91	200		o

```
8 rows selected.
```

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

```
Table created.
```

```
SQL> select * from biblio2;
```

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D	
Les fleurs du mal	Baudelaire	POEME	01-JAN-78	60		n
La terre	Zola	ROMAN	12-JUN-90	52.5		n

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

```
6 rows created.
```

```
SQL> select * from biblio2;
```

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D	
Les fleurs du mal	Baudelaire	POEME	01-JAN-78	60		n
La terre	Zola	ROMAN	12-JUN-90	52.5		n
Tintin au Tibet	Hergé	BD	10-NOV-90	70		o
Madame Bovary	Flaubert	ROMAN	12-MAR-88	136.5		o
Manhattan transfer	Dos Passos	ROMAN	30-AUG-87	336		o

```
Tintin en Amérique    Hergé      BD      15-MAY-91    70    o
Du côté de ch. Swann  Proust    ROMAN   08-DEC-78    210   o
Le père Goriot        Balzac    ROMAN   01-SEP-91    200    o
```

8 rows selected.

```
SQL> drop table biblio2;
```

Table dropped.

2.16 Changer le nom d'une table

syntaxe	RENAME <i>table1</i> TO <i>table2</i>
action	change la nom <i>table1</i> en <i>table2</i>

Exemples

```
SQL> select table_name from tabs;    <-- liste des tables
```

```
TABLE_NAME
-----
ARTICLES          <-- tables existantes
BIBLIO
CLIENTS
CMD_ID
COMMANDES
SAUVEGARDE
STOCKS
```

7 rows selected.

```
SQL> rename biblio to livres;    <-- la table biblio devient la table livres
```

Table renamed.

```
SQL> select table_name from tabs;    <-- liste des tables
```

```
TABLE_NAME
-----
ARTICLES
CLIENTS
CMD_ID
COMMANDES
LIVRES            <-----
SAUVEGARDE
STOCKS
```

7 rows selected.

```
SQL> rename livres to biblio;    <-- la table livres redevient la table biblio
```

Table renamed.

```
SQL> select table_name from tabs;    <-- liste des tables
```

```
TABLE_NAME
-----
ARTICLES
BIBLIO            <----
CLIENTS
CMD_ID
COMMANDES
SAUVEGARDE
STOCKS
```

7 rows selected.

2.17 Suppression d'une table

syntaxe	DROP TABLE <i>table</i>
action	supprime <i>table</i>

Exemples

```
SQL> select table_name from tabs;

TABLE_NAME
-----
BIBLIO
CHEAP
ROMANS
VIDE

SQL> drop table romans;

Table dropped.

SQL> drop table cheap;

Table dropped.

SQL> drop table vide;

Table dropped.

SQL> select table_name from tabs;  <-- vérification

TABLE_NAME
-----
BIBLIO          <-- il ne reste que biblio

1 row selected.
```

2.18 Modification de la structure d'une table

syntaxe	ALTER TABLE <i>table</i> [ADD (<i>nom_colonne1 type_colonne1 contrainte_colonne1, nom_colonne2 type_colonne2 contrainte_colonne2, ...</i>)] [MODIFY (<i>nom_colonnea type_colonnea contrainte_colonnea, nom_colonneb type_colonneb contrainte_colonneb, ...</i>)]
action	permet d'ajouter (ADD) ou de modifier (MODIFY) des colonnes de table. La syntaxe <i>nom_colonnei type_colonnei contrainte_colonnei</i> est celle du CREATE TABLE.

Exemple

```
SQL> alter table biblio
  2  add (nbpages number(4))    <-- on ajoute une colonne
  3  modify (genre char(20));   <-- on en modifie une autre

Table altered.

SQL> describe biblio          <-- vérification

Name          Null?      Type
-----
TITRE         NOT NULL   CHAR(20)
AUTEUR        NOT NULL   CHAR(15)
GENRE         NOT NULL   CHAR(20)    <--- a changé
ACHAT        NOT NULL   DATE
PRIX          NOT NULL   NUMBER(6,2)
DISPONIBLE    NOT NULL   CHAR(1)
NBPAGES              NUMBER(4)    <--- a été créé

SQL> select titre,genre,nbpages from biblio;  <-- contenu de biblio

TITRE          GENRE      NBPAGES
-----
Les fleurs du mal    POEME
Tintin au Tibet      BD
La terre            ROMAN
Madame Bovary        ROMAN
Manhattan transfer   ROMAN
Tintin en Amérique   BD
Du côté de ch. Swann  ROMAN

7 rows selected.

SQL> alter table biblio
```

```

2 modify (titre char(15));      <-- on diminue la largeur de la colonne titre
modify (titre char(15))
*
ERROR at line 2:
ORA-01441: column to be modified must be empty to decrease column length

```

L'exemple précédent montre qu'on ne peut diminuer la largeur d'une colonne si celle-ci contient déjà des valeurs.

2.19 Les vues

Il est possible d'avoir une vue partielle d'une table ou de plusieurs tables. Une vue se comporte comme une table mais ne contient pas de données. Ses données sont extraites d'autres tables ou vues. Une vue comporte plusieurs avantages :

- 1 Un utilisateur peut n'être intéressé que par certaines colonnes et certaines lignes d'une table donnée. La vue lui permet de ne voir que ces lignes et ces colonnes.
- 2 Le propriétaire d'une table peut désirer n'en autoriser qu'un accès limité, à d'autres utilisateurs. La vue lui permet de le faire. Les autres utilisateurs qu'il aura nommés n'auront accès qu'à la vue qu'il aura définie.

2.19.1 Création d'une vue

syntaxe

```

CREATE VIEW nom_vue
AS SELECT colonne1, colonne2, ... FROM table WHERE condition
[ WITH CHECK OPTION ]

```

action

crée la vue *nom_vue*. Celle-ci est une table ayant pour structure *colonne1, colonne2, ...* de *table* et pour lignes, les lignes de *table* vérifiant *condition* (toutes les lignes s'il n'y a pas de condition)

WITH CHECK OPTION

Cette clause optionnelle indique que les insertions et les mises à jour sur la vue, ne doivent pas créer de lignes que la vue ne pourrait sélectionner.

Remarque La syntaxe de CREATE VIEW est en fait plus complexe que celle présentée et permet notamment de créer une vue à partir de plusieurs tables. Il suffit pour cela que la requête SELECT porte sur plusieurs tables (cf chapitre suivant).

Exemples

On crée à partir de la table **biblio**, une vue ne comportant que les romans (sélection de lignes) et que les colonnes **titre**, **auteur**, **prix** (sélection de colonnes) :

```

SQL> create view romans as select titre,auteur,prix
2 from biblio where genre='ROMAN';

```

View created.

```

SQL> select * from romans;      <-- contenu de la vue

```

TITRE	AUTEUR	PRIX
La terre	Zola	52.5
Madame Bovary	Flaubert	136.5
Manhattan transfer	Dos Passos	336
Du côté de ch. Swann	Proust	210

```

SQL> describe romans

```

Name	Null?	Type
TITRE	NOT NULL	CHAR(20)
AUTEUR	NOT NULL	CHAR(15)
PRIX	NOT NULL	NUMBER(6,2)

```

SQL> insert into biblio<-- on ajoute une ligne à la table biblio
2 values ('Le père Goriot','Balzac','Roman','01-sep-91',200,'o');

```

1 row created.

```

SQL> select * from romans;      <-- le nouveau titre est-il dans les romans ?

```

```

TITRE          AUTEUR      PRIX
-----
La terre       Zola         52.5
Madame Bovary  Flaubert    136.5
Manhattan transfer Dos Passos  336
Du côté de ch. Swann   Proust      210
  <-- il n'y est pas parce que le genre est Roman
  <-- et non ROMAN comme l'exige la vue créée

```

```

  <-- on met les genres en majuscules
SQL> update biblio set genre=upper(genre);

```

8 rows updated.

```

SQL> select * from romans;

```

```

TITRE          AUTEUR      PRIX
-----
La terre       Zola         52.5
Madame Bovary  Flaubert    136.5
Manhattan transfer Dos Passos  336
Du côté de ch. Swann   Proust      210
Le père Goriot  Balzac      200<-- le livre ajouté est là

```

Des exemples précédents, on voit un avantage de la vue par rapport à la table secondaire lorsqu'on veut filtrer une table principale. Rappelons les deux méthodes :

- 1 **create table** romans as select titre, auteur, prix from biblio where genre='ROMAN'
- 2 **create view** romans as select titre, auteur, prix from biblio where genre='ROMAN'

La méthode 1 crée une table qui devient obsolète dès qu'un nouveau roman est ajouté à la table *biblio*. La méthode 2 crée une vue qui elle, est toujours à jour.

2.19.2 Mise à jour d'une vue

Il est possible de mettre à jour une vue comme on le fait pour une table. Toutes les tables d'où sont extraites les données de la vue sont affectées par cette mise à jour. Voici quelques exemples :

```

SQL> delete from romans where prix=210;    <-- suppression de lignes dans la vue

```

1 row deleted.

```

SQL> select * from romans;                <-- vérification

```

```

TITRE          AUTEUR      PRIX
-----
La terre       Zola         52.5
Madame Bovary  Flaubert    136.5
Manhattan transfer Dos Passos  336
Le père Goriot  Balzac      200

```

```

SQL> select * from biblio;                <-- impact sur biblio

```

```

TITRE          AUTEUR      GENRE      ACHAT      PRIX      D
-----
Les fleurs du mal  Baudelaire  POEME      01-JAN-78    60  n
Tintin au Tibet   Hergé       BD         10-NOV-90    70  o
La terre          Zola        ROMAN      12-JUN-90    52.5 n
Madame Bovary     Flaubert    ROMAN      12-MAR-88    136.5 o
Manhattan transfer Dos Passos   ROMAN      30-AUG-87    336  o
Tintin en Amérique Hergé       BD         15-MAY-91    70  o
Le père Goriot    Balzac      ROMAN      01-SEP-91    200  o

```

7 rows selected.

```

SQL> update romans set prix=prix*1.05;    <-- modification de tous les prix de la vue

```

4 rows updated.

```

SQL> select * from romans;                <-- vérification

```

```

TITRE          AUTEUR      PRIX
-----
La terre       Zola         55.13
Madame Bovary  Flaubert    143.33
Manhattan transfer Dos Passos  352.8
Le père Goriot  Balzac      210

```

```
SQL> select * from biblio;      <-- impact sur biblio
```

TITRE	AUTEUR	GENRE	ACHAT	PRIX	D
Les fleurs du mal	Baudelaire	POEME	01-JAN-78	60	n
Tintin au Tibet	Hergé	BD	10-NOV-90	70	o
La terre	Zola	ROMAN	12-JUN-90	55.13	n
Madame Bovary	Flaubert	ROMAN	12-MAR-88	143.33	o
Manhattan transfer	Dos Passos	ROMAN	30-AUG-87	352.8	o
Tintin en Amérique	Hergé	BD	15-MAY-91	70	o
Le père Goriot	Balzac	ROMAN	01-SEP-91	210	o

7 rows selected.

2.19.3 Obtenir la liste des vues

La table système **user_views** contient des informations sur toutes les vues créées par l'utilisateur. Sa structure est la suivante :

```
SQL> describe user_views
```

Name	Null?	Type
VIEW_NAME	NOT NULL	CHAR(30)
TEXT_LENGTH		NUMBER
TEXT		LONG

<-- nom de la vue

On obtient donc la liste des vues par la commande :

```
select view_name from user_views
```

```
SQL> select view_name from user_views;
```

VIEW_NAME
ROMANS

On peut utiliser le concept de vue pour simplifier cette commande :

```
SQL> create view vues as select view_name from user_views;
<--- la vue nommée vues ne contient que la colonne view_name
<--- de la table user_views
View created.
```

```
SQL> select * from vues; <-- donne le nom des vues créées
```

VIEW_NAME
ROMANS
VUES

2.19.4 Supprimer une vue

syntaxe	DROP VIEW nom_vue
action	supprime la vue nommée

Exemple

```
SQL> drop view romans;
```

View dropped.

```
SQL> select * from vues; <-- affiche les vues créées
```

VIEW_NAME
VUES <-- la vue romans n'existe plus

2.20 Utilisation de fonctions de groupes

Il existe des fonctions qui au lieu de travailler sur chaque ligne d'une table, travaillent sur des groupes de lignes. Ce sont essentiellement des fonctions statistiques nous permettant d'avoir la moyenne, l'écart-type, etc ... des données d'une colonne.

syntaxe1 | **SELECT** *f1, f2, ..., fn* **FROM** *table*
[**WHERE** *condition*]

action | calcule les fonctions statistiques *fi* sur l'ensemble des lignes de table vérifiant l'éventuelle *condition*.

syntaxe2 | **SELECT** *f1, f2, ..., fn* **FROM** *table*
[**WHERE** *condition*]
[**GROUP BY** *expr1, expr2, ..*]

action | Le mot clé **GROUP BY** a pour effet de diviser les lignes de table par groupe. Chaque groupe contient les lignes pour lesquelles les expressions *expr1, expr2, ...* ont même valeur. Exemple : **GROUP BY genre** met dans un même groupe, les livres ayant le même genre. La clause **GROUP BY auteur, genre** mettrait dans le même groupe les livres ayant même auteur et même genre. Le **WHERE condition** élimine d'abord de la table les lignes ne vérifiant pas condition. Ensuite les groupes sont formés par la clause **GROUP BY**. Les fonctions *fi* sont ensuite calculées pour chaque groupe de lignes.

syntaxe3 | **SELECT** *f1, f2, ..., fn* **FROM** *table*
[**WHERE** *condition*]
[**GROUP BY** *expression*]
[**HAVING** *condition_de_groupe*]

action | La clause **HAVING** filtre les groupes formés par la clause **GROUP BY**. Elle est donc toujours liée à la présence de cette clause **GROUP BY**. Exemple : **GROUP BY genre HAVING genre != 'ROMAN'**

Les fonctions statistiques *fi* disponibles sont les suivantes :

AVG (expression)	moyenne de expression
COUNT (expression)	nombre de lignes pour lesquelles expression a une valeur
COUNT (*)	nombre total de lignes dans la table
MAX (expression)	max de expression
MIN (expression)	min de expression
STDDEV (expression)	écart-type de expression
SUM (expression)	somme de expression
VARIANCE (expression)	variance de expression

Exemples

SQL> select prix from biblio;

```

      PRIX
-----
      120
       70
      52.5
     136.5
      336
       70
      210
      200

```

8 rows selected.

```
SQL> select avg(prix), max(prix), min (prix) from biblio;
```

AVG(PRIX)	MAX(PRIX)	MIN(PRIX)
149.375	336	52.5

```
SQL> select titre, prix, genre from biblio;
```

TITRE	PRIX	GENRE
Les fleurs du mal	120	POEME
Tintin au Tibet	70	BD
La terre	52.5	ROMAN
Madame Bovary	136.5	ROMAN
Manhattan transfer	336	ROMAN
Tintin en Amérique	70	BD
Du côté de ch. Swann	210	ROMAN
Le père Goriot	200	ROMAN

8 rows selected.

```
SQL> select avg(prix) moyenne, stddev(prix) écart_type
  2  , max(prix) prix_maxi from biblio
  3  where genre='ROMAN';          <-- on ne s'intéresse qu'aux romans
```

MOYENNE	ECART_TYPE	PRIX_MAXI
187	104.330604	336

```
SQL> select count(*) from biblio
  2  where genre='BD';          <-- Combien y-a-t-il de BD ?
```

COUNT(*)
2

```
SQL> select count(*) from biblio where genre='ROMAN'
  2  and prix<100;          <-- Combien de romans à moins de 100 francs
```

COUNT(*)
1

```
SQL> select genre, prix from biblio;
```

GENRE	PRIX
POEME	60
BD	70
ROMAN	52.5
ROMAN	136.5
ROMAN	336
BD	70
ROMAN	210
ROMAN	200

8 rows selected.

```
SQL> select genre, avg(prix), count(*) from biblio
  2  group by genre;
```

GENRE	AVG(PRIX)	COUNT(*)
BD	70	2
POEME	60	1
ROMAN	187	5

```
SQL>
  1  select genre, avg(prix), count(*) from biblio
  2  group by genre
  3  having genre!='ROMAN';
```

GENRE	AVG(PRIX)	COUNT(*)
BD	70	2
POEME	60	1

```
SQL>
  1  select genre, avg(prix), count(*) from biblio
  2  where prix<65
```



```

3 group by genre
4 having genre!='ROMAN';

```

GENRE	AVG (PRIX)	COUNT (*)
POEME	60	1

```

SQL>
1 select genre,avg(prix) from biblio
2 group by genre
3 having avg(prix)>60

```

GENRE	AVG (PRIX)
BD	70
ROMAN	187

Remarque

De façon générale, la syntaxe HAVING ... peut s'employer lorsque les arguments du SELECT arg1, arg2, ..argn sont :

- a une constante ou une fonction sans paramètres (ex : SYSDATE)
- b une fonction de groupe telle que celles que nous venons de voir
- c une des expressions *expr1* de la clause *GROUP BY expr1, expr2,*

2.21 La pseudo-colonne ROWNUM

Une pseudo-colonne se comporte comme une colonne, si ce n'est qu'elle ne fait pas partie de la table. L'une d'entre-elles appelée ROWNUM numérote de 1 à n les lignes **résultat d'une requête**. Elle nous permet par exemple de limiter le nombre de lignes résultat de la requête : WHERE ROWNUM<5 par exemple, limite à 4 le nombre de lignes sélectionnées.

Exemple

```
SQL> select rownum,titre,prix from biblio;
```

ROWNUM	TITRE	PRIX
1	Les fleurs du mal	120
2	Tintin au Tibet	70
3	La terre	52.5
4	Madame Bovary	136.5
5	Manhattan transfer	336
6	Tintin en Amérique	70
7	Du côté de ch. Swann	210
8	Le père Goriot	200

8 rows selected.

```
SQL>select rownum,titre,prix from biblio
2 where prix<100;
```

ROWNUM	TITRE	PRIX
1	Tintin au Tibet	70
2	La terre	52.5
3	Tintin en Amérique	70

```
SQL> select titre,auteur from biblio where rownum <4;  <-- affichage des 3 premières lignes de biblio
```

TITRE	AUTEUR
Les fleurs du mal	Baudelaire
Tintin au Tibet	Hergé
La terre	Zola

2.22 Mémoriser et imprimer les commandes SQL et leurs résultats

Il est nécessaire d'avoir la possibilité de conserver et imprimer le résultat des requêtes notamment ceux du SELECT. Cela se fait avec la commande SQLPLUS (et non SQL)

syntaxe1	spool fichier
action	les affichages écran sont dupliqués dans fichier. Si celui-ci n'a pas de suffixe, il s'appellera en fait fichier.lst.

syntaxe2	spool off
action	arrête toute duplication de l'écran dans un fichier

syntaxe3	spool out
action	s'utilise à la place de <i>spool off</i> . Imprime alors le contenu du fichier dupliquant l'écran.

Exemples

```
SQL> spool copie          <-- duplication dans copie.lst

SQL> select titre,genre from biblio;

TITRE          GENRE    <-- affichage écran
-----
Les fleurs du mal    POEME
Tintin au Tibet      BD
La terre             ROMAN
Madame Bovary        ROMAN
Manhattan transfer   ROMAN
Tintin en Amérique   BD
Du côté de ch. Swann ROMAN
Le père Goriot       ROMAN

8 rows selected.

SQL> spool off            <-- fin de la duplication

SQL> host                 <-- on passe sous le système

$ ll copie.lst           <-- on vérifie que copie.lst existe bien
-rw-rw-r--  1 serge      enseignants  882 Oct 21 15:05 copie.lst

$ cat copie.lst          <-- son contenu

SQL> select titre,genre from biblio;  <-- tout ce qui est apparu à l'écran

TITRE          GENRE
-----
Les fleurs du mal    POEME
Tintin au Tibet      BD
La terre             ROMAN
Madame Bovary        ROMAN
Manhattan transfer   ROMAN
Tintin en Amérique   BD
Du côté de ch. Swann ROMAN
Le père Goriot       ROMAN

8 rows selected.

SQL> spool off            <-- dernière commande enregistrée

$ exit                  <-- retour à SQLPLUS
```

3 Les expressions du langage SQL

3.1 Introduction

Dans la plupart des commandes SQL, il est possible d'utiliser une expression. Prenons par exemple la commande SELECT.

syntaxe	SELECT <i>expr1, expr2, ...</i> from <i>table</i> WHERE <i>expression</i>
---------	---

SELECT sélectionne les lignes pour lesquelles *expression* est vraie et affiche pour chacune d'elles les valeurs de *expr1*.

Exemples

```
select prix*1.186 from biblio
select to_char(achat,'dd/mm/yy') from biblio
select titre from biblio where prix between 100 and 150
```

Nous nous proposons dans ce chapitre d'expliciter la notion d'expression. Une expression élémentaire est du type : **opérateur1** **opérateur2** ou **fonction**(*paramètres*)

Exemple

Dans l'expression *GENRE = 'ROMAN'*

- GENRE est l'opérateur1
- 'ROMAN' est l'opérateur2
- = est l'opérateur

Dans l'expression

to_char(achat,'dd/mm/yy')

- TO_CHAR est une fonction
- achat et 'dd/mm/yy' sont les paramètres de cette fonction.

Nous traitons tout d'abord des expressions avec opérateurs, puis nous présenterons les fonctions disponibles sous ORACLE.

3.2 Expressions avec opérateur

Nous classifions les expressions avec opérateur suivant le type de leurs opérandes :

- . numérique
- . chaîne de caractères
- . date
- . booléen ou logique

3.2.1 Les expressions à opérandes de type numérique

3.2.1.1 Liste des opérateurs

Soient nombre1, nombre2, nombre3 des nombres. Les opérateurs utilisables sont les suivants :

Opérateurs relationnels

nombre1 > nombre2	: nombre1 plus grand que nombre2
nombre1 >= nombre2	: nombre1 plus grand ou égal à nombre2

<code>nombre1 < nombre2</code>	: nombre1 plus petit que nombre2
<code>nombre1 <= nombre2</code>	: nombre1 plus petit ou égal à nombre2
<code>nombre1 = nombre2</code>	: nombre1 égal à nombre2
<code>nombre1 != nombre2</code>	: nombre1 différent de nombre2
<code>nombre1 <> nombre2</code>	: idem
<code>nombre1 BETWEEN nombre2 AND nombre3</code>	: nombre1 dans l'intervalle [nombre2,nombre3]
<code>nombre1 IN (liste de nombres)</code>	: nombre1 appartient à liste de nombres
<code>nombre1 IS NULL</code>	: nombre1 n'a pas de valeur
<code>nombre1 IS NOT NULL</code>	: nombre1 a une valeur
<code>ANY, ALL, EXISTS</code>	

Opérateurs arithmétiques

<code>nombre1 + nombre2</code>	: addition
<code>nombre1 - nombre2</code>	: soustraction
<code>nombre1 * nombre2</code>	: multiplication
<code>nombre1 / nombre2</code>	: division

3.2.1.2 Opérateurs relationnels

Une expression relationnelle exprime une relation qui est vraie ou fausse. Le résultat d'une telle expression est donc un booléen ou valeur logique.

Exemples

```
SQL> select titre,prix from biblio
  2 where prix between 100 and 150;
```

TITRE	PRIX
Madame Bovary	136.5

```
SQL> select titre,prix from biblio
  2 where prix not between 100 and 150;
```

TITRE	PRIX
Les fleurs du mal	60
Tintin au Tibet	70
La terre	52.5
Manhattan transfer	336
Tintin en Amérique	70
Du côté de ch. Swann	210
Le père Goriot	200

7 rows selected.

```
SQL> select titre,prix from biblio where prix in (200,210);
```

TITRE	PRIX
Du côté de ch. Swann	210
Le père Goriot	200

3.2.1.3 Opérateurs arithmétiques

L'expression arithmétique nous est familière. Elle exprime un calcul à faire entre des données numériques. Nous avons déjà rencontré de telles expressions : on suppose que le prix mémorisé dans les fiches du fichier BIBLIO soit un prix hors taxes. On veut visualiser chaque titre avec son prix TTC pour un taux de TVA de 18.6% :

```
SELECT TTTRE, PRIX*1.186 FROM BIBLIO
```

Si les prix doivent augmenter de 3%, la commande sera

```
UPDATE BIBLIO SET PRIX = PRIX*1.03
```

On peut trouver plusieurs opérateurs arithmétiques dans une expression avec de plus des fonctions et des parenthèses. Ces éléments sont traités selon des priorités différentes :

1	<i>fonctions</i>	<---- plus prioritaire
2	<i>()</i>	
3	<i>* et /</i>	
4	<i>+ et -</i>	<---- moins prioritaire

Lorsque deux opérateurs de même priorité sont présents dans l'expression, c'est celui qui est le plus à gauche dans l'expression qui est évalué tout d'abord.

Exemples

$PRIX * TAUX + TAXES$

sera évaluée comme $(PRIX * TAUX) + TAXES$. C'est en effet l'opérateur de multiplication qui sera utilisé en premier.

$PRIX * TAUX / 100$

sera évaluée comme $(PRIX * TAUX) / 100$.

3.2.2 Les expressions à opérandes de type caractères

3.2.2.1 Liste des opérateurs

Les opérateurs utilisables sont les suivants :

Opérateurs relationnels

Soient *chaine1*, *chaine2*, *chaine3*, *modèle* des chaînes de caractères

<code>chaine1 > chaine2</code>	: chaine1 plus grande que chaine2
<code>chaine1 >= chaine2</code>	: chaine1 plus grande ou égale à chaine2
<code>chaine1 < chaine2</code>	: chaine1 plus petite que chaine2
<code>chaine1 <= chaine2</code>	: chaine1 plus petite ou égale à chaine2
<code>chaine1 = chaine2</code>	: chaine1 égale à chaine2
<code>chaine1 != chaine2</code>	: chaine1 différente de chaine2
<code>chaine1 <> chaine2</code>	: idem
<code>chaine1 BETWEEN chaine2 AND chaine3</code>	: chaine1 dans l'intervalle [chaine2,chaine3]
<code>chaine1 IN liste de chaines</code>	: chaine1 appartient à liste de chaines
<code>chaine1 IS NULL</code>	: chaine1 n'a pas de valeur
<code>chaine1 IS NOT NULL</code>	: chaine1 a une valeur
<code>chaine1 LIKE modèle</code>	: chaine1 correspond à modèle
<code>ANY, ALL, EXISTS</code>	

Opérateurs de concaténation

chaine1 || *chaine2* : *chaine2* concaténée à *chaine1*

3.2.2.2 Opérateurs relationnels

Que signifie comparer des chaînes avec des opérateurs tels que <, <=, etc ... ?

Tout caractère est codé sur un octet par un nombre compris entre 0 et 255. On appelle ce codage, le code ASCII des caractères. On trouvera ces codes ASCII en annexe. Ce codage permet la comparaison entre deux caractères. Ce sont en effet les codes ASCII (des nombres donc) des caractères qui sont comparés. Ainsi on dira que le caractère C1 est plus petit que le caractère C2 si le code ASCII de C1 est plus petit que le code ASCII de C2. A partir de la table des codes ASCII, on a les relations suivantes :

blanc < . < 0 < 1 < ... < 9 < ... < A < B < ... < Z < ... < a < b < ... < z

Dans l'ordre ASCII, les chiffres viennent avant les lettres, et les majuscules avant les minuscules. On remarquera que l'ordre ASCII respecte l'ordre des chiffres ainsi que l'ordre alphabétique auquel on est habitué.

3.2.2.3 Comparaison de deux chaînes

Soit la relation 'CHAT' < 'CHIEN'. Est-elle vraie ou fausse ? Pour effectuer cette comparaison, ORACLE compare les deux chaînes caractère par caractère sur la base de leurs codes ASCII. Dès que deux caractères sont trouvés différents, la chaîne à qui appartient le plus petit des deux est dite plus petite que l'autre chaîne. Dans notre exemple 'CHAT' est comparée à 'CHIEN'. On a les résultats successifs suivants :

```
'CHAT' 'CHIEN'
-----
'C' = 'C'
'H' = 'H'
'A' < 'I'
```

Après cette dernière comparaison, la chaîne 'CHAT' est déclarée plus petite que la chaîne 'CHIEN' dans l'ordre ASCII. La relation 'CHAT' < 'CHIEN' est donc vraie.

Soit à comparer maintenant 'CHAT' et 'chat'.

```
'CHAT' 'chat'
-----
'C' < 'c'
```

Après cette comparaison, la relation 'CHAT' < 'chat' est déclarée vraie.

Exemples

```
SQL> select titre from biblio;
```

```
TITRE
-----
Les fleurs du mal
Tintin au Tibet
La terre
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann
Le père Goriot
```

```
8 rows selected.
```

```
SQL> select titre from biblio where upper(titre) between 'L' and 'U';
```

```
TITRE
-----
Les fleurs du mal
Tintin au Tibet
La terre
Madame Bovary
Manhattan transfer
```

```
Tintin en Amérique
Le père Goriot

7 rows selected.
```

3.2.2.4 L'opérateur LIKE

L'opérateur LIKE s'utilise comme suit : *chaîne* **LIKE** *modèle*

La relation est vraie si *chaîne* correspond au *modèle*. Celui-ci est une chaîne de caractères pouvant comporter deux caractères génériques :

%	qui désigne toute suite de caractères
_	qui désigne 1 caractère quelconque

Exemples

```
SQL> select titre from biblio;

TITRE
-----
Les fleurs du mal
Tintin au Tibet
La terre
Madame Bovary
Manhattan transfer
Tintin en Amérique
Du côté de ch. Swann
Le père Goriot

8 rows selected.

SQL> select titre from biblio where titre like 'M%';

TITRE
-----
Madame Bovary
Manhattan transfer

SQL> select titre from biblio where titre like 'L_ %';

TITRE
-----
La terre
Le père Goriot
```

3.2.2.5 L'opérateur de concaténation

```
SQL> select 'chat' || 'eau' exemple from dual;  <-- concaténation de 'chat' avec 'eau'

EXEMPLE
-----
chateau  <-- résultat de la concaténation
```

3.2.3 Les expressions à opérandes de type date

Soient *date1*, *date2*, *date3* des dates. Les opérateurs utilisables sont les suivants :

Opérateurs relationnels

date1 < date2	est vraie si date1 est antérieure à date2
date1 <= date2	est vraie si date1 est antérieure ou égale à date2
date1 > date2	est vraie si date1 est postérieure à date2

<code>date1 >= date2</code>	est vraie si date1 est postérieure ou égale à date2
<code>date1 = date2</code>	est vraie si date1 et date2 sont identiques
<code>date1 <> date2</code>	est vraie si date1 et date2 sont différentes.
<code>date1 != date2</code>	idem
<code>date1 BETWEEN date2 AND date3</code>	est vraie si date1 est situé entre date2 et date3
<code>date1 IN (liste de dates)</code>	est vraie si date1 se trouve dans la liste de dates
<code>date1 IS NULL</code>	est vraie si date1 n'a pas de valeur
<code>date1 IS NOT NULL</code>	est vraie si date1 a une valeur
<code>date1 LIKE modèle</code>	est vraie si date1 correspond au modèle
<code>ALL, ANY, EXISTS</code>	

Opérateurs arithmétiques

<code>date1 - date2</code>	: nombre de jours séparant date1 de date2
<code>date1 - nombre</code>	: date2 telle que date1-date2=nombre
<code>date1 + nombre</code>	: date2 telle que date2-date1=nombre

Exemples

```
SQL> select achat from biblio;
```

```
ACHAT
-----
01-JAN-78
10-NOV-90
12-JUN-90
12-MAR-88
30-AUG-87
15-MAY-91
08-DEC-78
01-SEP-91
```

```
8 rows selected.
```

```
SQL> select achat from biblio
      where achat between '01-jan-88' and '31-dec-88';
```

```
ACHAT
-----
12-MAR-88
```

```
SQL> select achat from biblio
      where achat like '___-MA%';
```

```
ACHAT
-----
12-MAR-88
15-MAY-91
```

```
SQL> select sysdate from dual; <-- date & heure du moment
```

```
SYSDATE
-----
18-OCT-91
```

```
SQL> select sysdate-achat age from biblio; <-- âge des livres ?
```

```
      AGE
-----
5038.63328  <-- les chiffres derrière le point correspondent à la composante heure de la date
342.633275
493.633275
1315.63328
1510.63328
156.633275
4697.63328
47.6332755
```


8 rows selected.

```
SQL> select trunc(sysdate-achat) age from biblio; <-- trunc pour enlever la composante heure
```

```
-----
AGE
-----
5038
342
493
1315
1510
156
4697
47
```

8 rows selected.

3.2.4 Expressions à opérandes booléens

Rappelons qu'un booléen ou valeur logique a deux valeurs possibles : vrai ou faux. Sous Oracle, l'opérande logique est typiquement le résultat d'une expression relationnelle.

Soient *booléen1* et *booléen2* deux booléens. Il y a trois opérateurs possibles qui sont par ordre de priorité :

<i>booléen1</i> AND <i>booléen2</i>	est vraie si <i>booléen1</i> et <i>booléen2</i> sont vrais tous les deux.
<i>booléen1</i> OR <i>booléen2</i>	est vraie si <i>booléen1</i> ou <i>booléen2</i> est vrai.
NOT <i>booléen1</i>	a pour valeur l'inverse de la valeur de <i>booléen1</i> .

Exemples

```
SQL> select titre,genre,prix from biblio;
```

```
TITRE          GENRE      PRIX
-----
Les fleurs du mal    POEME        60
Tintin au Tibet      BD           70
La terre             ROMAN       52.5
Madame Bovary        ROMAN      136.5
Manhattan transfer   ROMAN      336
Tintin en Amérique   BD           70
Du côté de ch. Swann  ROMAN      210
Le père Goriot       ROMAN      200
```

8 rows selected.

```
SQL> select titre,genre,prix from biblio
      where prix>=50 and prix <=100;
```

```
TITRE          GENRE      PRIX
-----
Les fleurs du mal    POEME        60
Tintin au Tibet      BD           70
La terre             ROMAN       52.5
Tintin en Amérique   BD           70
```

```
SQL> select titre,genre,prix from biblio
      2  where prix <50 or prix >100;
```

```
TITRE          GENRE      PRIX
-----
Madame Bovary        ROMAN      136.5
Manhattan transfer   ROMAN      336
Du côté de ch. Swann  ROMAN      210
Le père Goriot       ROMAN      200
```

```
SQL> select titre,genre,prix from biblio
      2  where genre='ROMAN' and prix>200 or prix<100; <-- attention à la priorité des opérateurs
```

```
TITRE          GENRE      PRIX
-----
Les fleurs du mal    POEME        60
Tintin au Tibet      BD           70
La terre             ROMAN       52.5
Manhattan transfer   ROMAN      336
```

```
Tintin en Amérique    BD          70
Du côté de ch. Swann  ROMAN       210
```

6 rows selected.

```
SQL> select titre,genre,prix from biblio
      2* where genre='ROMAN' and (prix>200 or prix<100) <-- on met des parenthèses
```

TITRE	GENRE	PRIX
La terre	ROMAN	52.5
Manhattan transfer	ROMAN	336
Du côté de ch. Swann	ROMAN	210

3.3 Les fonctions prédéfinies d'ORACLE

Une expression peut comporter l'appel à des fonctions prédéfinies d'Oracle. Nous présentons ici les plus courantes. Nous les classifions selon le type prédominant de leurs paramètres ou selon leur rôle :

- . fonctions à paramètres de type numérique
- . fonctions à paramètres de type chaîne de caractères
- . fonctions à paramètres de type date
- . fonctions de conversion de type
- . fonctions diverses

Dans la syntaxe *SELECT fonction FROM table*

la fonction est évaluée le plus souvent pour chaque ligne de la table. On a donc affichage de n résultats pour n lignes. Certaines fonctions cependant, sont évaluées à partir des données de toutes les lignes de la table et ne donnent lieu qu'à un seul résultat. On les appellera des fonctions de groupe. On les connaît pour la plupart : ce sont les fonctions statistiques COUNT, SUM, AVG, STDDEV, VARIANCE, MAX, MIN.

Le lecteur pourra tester les fonctions présentées ci-après par la commande :

SELECT fonction FROM DUAL

La table DUAL comme il a été déjà expliqué est une table à une ligne et une colonne qui contient l'unique donnée 'X'.

```
SQL> describe dual;
```

Name	Null?	Type
DUMMY		CHAR(1)

<-- une seule colonne de type caractère

```
SQL> select * from dual; <-- contenu de la table DUAL
```

```
D
-
X <-- une seule ligne
```

La commande

SELECT fonction FROM DUAL

évalue *fonction* pour chaque ligne de la table DUAL, donc en fait pour une seule ligne. Il y a donc simplement affichage de la valeur de *fonction*.

3.3.1 Fonctions à paramètres de type numérique

abs (<i>nombre</i>)	valeur absolue de nombre <i>abs(-15)=15</i>
ceil (<i>nombre</i>)	plus petit entier plus grand ou égal à <i>nombre</i> <i>ceil(15.7)=16</i>

floor (<i>nombre</i>)	plus grand entier inférieur ou égal à <i>nombre</i> <i>floor</i> (14.3)=14
mod (<i>nombre1</i> , <i>nombre2</i>)	reste de la division entière (le quotient est entier) de <i>nombre1</i> par <i>nombre2</i> <i>mod</i> (7,3)=1 <i>mod</i> (7.5,3.2)=1.1
power (<i>nombre1</i> , <i>nombre2</i>)	<i>nombre1</i> élevé à la puissance <i>nombre2</i> qui doit être entier <i>power</i> (4,2)=16
round (<i>nombre1</i> , <i>nombre2</i>)	arrondit <i>nombre1</i> à <i>nombre2</i> chiffres après la virgule si <i>nombre2</i> >0, à - <i>nombre2</i> chiffres avant la virgule si <i>nombre2</i> <0. <i>round</i> (13.456,2)=13.46 <i>round</i> (13.456,-1)=10 <i>round</i> (16.456,-1)=20 <i>round</i> (16.456,0)=16
sign (<i>nombre</i>)	-1 si <i>nombre</i> <0 0 si <i>nombre</i> =0 +1 si <i>nombre</i> >0 <i>sign</i> (-6)=-1
sqrt (<i>nombre</i>)	racine carrée de <i>nombre</i> si <i>nombre</i> >=0 NULL si <i>nombre</i> <0 <i>sqrt</i> (16)=4
trunc (<i>nombre1</i> , <i>nombre2</i>)	<i>nombre1</i> est tronqué à <i>nombre2</i> chiffres après la virgule si <i>nombre2</i> >0 ou à - <i>nombre2</i> chiffres avant la virgule si <i>nombre2</i> <0. <i>trunc</i> (13.456,2)=13.45 <i>trunc</i> (13.456,-1)=10 <i>trunc</i> (16.456,-1)=10 <i>trunc</i> (16.456,0)=16

3.3.2 Fonctions à paramètres de type chaîne de caractères

chr (<i>nombre</i>)	caractère de code ASCII <i>nombre</i> <i>chr</i> (65)='A'
initcap (<i>chaîne</i>)	Met tous les mots de <i>chaîne</i> en minuscules sauf la première lettre mise en majuscule <i>initcap</i> ('gestion des stocks')='Gestion Des Stocks'
lower (<i>chaîne</i>)	met <i>chaîne</i> en minuscules <i>lower</i> ('INFO')='info'
lpad (<i>chaîne1</i> , <i>n</i> , <i>chaîne2</i>)	met <i>chaîne1</i> sur <i>n</i> positions, <i>chaîne1</i> étant cadrée à droite. Les caractères restant à gauche sont remplis par <i>chaîne2</i> . <i>lpad</i> ('chat',6,'*')='**chat' <i>lpad</i> ('chat',8,'xy')='xyxychat'
ltrim (<i>chaîne1</i> , <i>chaîne2</i>)	Les caractères de gauche de <i>chaîne1</i> sont supprimés jusqu'à rencontrer un caractère qui ne se trouve pas dans <i>chaîne2</i> . <i>ltrim</i> ('chaton','cb')='aton' <i>ltrim</i> ('chaton','bc')='aton' <i>ltrim</i> ('chaton','abc')='haton'
replace (<i>chaîne1</i> , <i>chaîne2</i> , <i>chaîne3</i>)	remplace <i>chaîne2</i> par <i>chaîne3</i> dans <i>chaîne1</i> . <i>replace</i> ('chat et chien','ch','**')='**at et **ien'
rpad (<i>chaîne1</i> , <i>n</i> , <i>chaîne2</i>)	idem lpad mais à droite <i>rpad</i> ('chat',6,'*')='chat**' <i>rpad</i> ('chat',8,'xy')='chatxyxy'
trim (<i>chaîne1</i> , <i>chaîne2</i>)	idem ltrim mais à droite <i>rtrim</i> ('chat','at')='ch' <i>rtrim</i> ('chat','ta')='ch'

substr (<i>chaîne,p,nombre</i>)	sous-chaîne de <i>chaîne</i> de <i>nombre</i> caractères commençant en position <i>p</i> . <i>substr('chaton',3,2)='at'</i>
translate (<i>chaîne,texte,traduction</i>)	remplace dans <i>chaîne</i> tout caractère se trouvant dans <i>texte</i> par le caractère correspondant se trouvant dans <i>traduction</i> <i>translate('abracadabra','ab','yz')='yzrydyzyry'</i>
ascii (<i>caractère</i>)	code ASCII de <i>caractère</i> <i>ascii('A')=65</i>
instr (<i>chaîne1,chaîne2,p,o</i>)	position de la <i>o</i> ième occurrence de <i>chaîne2</i> dans <i>chaîne1</i> , la recherche commençant à la position <i>p</i> de <i>chaîne1</i> . <i>instr('abracadabra','a',4,2)=6</i> <i>instr('abracadabra','a',5,3)=11</i>
length (<i>chaîne</i>)	nombre de caractères de <i>chaîne</i> <i>length('chaton')=6</i>

3.3.3 Fonctions de conversion

to_char (<i>nombre,format</i>)	transforme <i>nombre</i> en chaîne de caractères selon le <i>format</i> indiqué. Les formats utilisables sont ceux déjà présentés. <i>to_char(1000,'99999.99')=' 1000.00'</i> <i>to_char(1000,'99EEEE')=' 1E+03'</i>
to_char (<i>date,format</i>)	convertit <i>date</i> en chaîne de caractères selon le format indiqué. Cette fonction a été déjà présentée.
to_date (<i>chaîne,format</i>)	transforme <i>chaîne</i> en <i>date</i> . Le format décrit la chaîne. Les différents formats utilisables ont déjà été présentés. SQL> select to_date ('01/02/91' , 'dd/mm/yy') from dual; TO_DATE (' ----- 01-FEB-91 SQL> select to_date('01 january 91','dd month yy') from dual; TO_DATE (' ----- 01-JAN-91
to_number (<i>chaîne</i>)	transforme <i>chaîne</i> en nombre. Il faut pour cela que chaîne représente un nombre valide. <i>to_number('1987')=1987</i>

3.3.4 Fonctions de paramètres de type date

adddate (<i>date,n</i>)	<i>date</i> augmentée de <i>n</i> mois. Le résultat est une date. <i>date('01-jan-91',3)='01-apr-91'</i>
last_day (<i>date</i>)	date du dernier jour du mois contenu dans <i>date</i> <i>last_day('01-jan-91')='31-jan-91'</i>
months_between (<i>date1,date2</i>)	nombre de mois entre <i>date1</i> et <i>date2</i> . La partie décimale représente le pourcentage d'un mois de 31 jours. Si <i>date1</i> < <i>date2</i> le résultat est >0 sinon il est <0. <i>month_between('01-jan-91','14-feb-91')=-1.4193548</i>
next_day (<i>date,jour</i>)	donne la date du <i>jour</i> indiqué dans la semaine qui suit <i>date</i> . <i>next_day('01-jan-91','monday')='07-jan-91'</i>
sysdate	date du jour maintenue par le système

round (<i>date</i> , <i>format</i>)	<i>date</i> arrondie selon le <i>format</i> précisé Formats year : arrondit au 1er janvier le plus proche month : arrondit au 1er du mois le plus proche day : arrondit au dimanche le plus proche Exemples <pre>select sysdate from dual; SYSDATE ----- 21-OCT-91 SQL> select round(sysdate,'year') from dual; ROUND(SYS ----- 01-JAN-92 SQL> select round(sysdate,'month') from dual; ROUND(SYS ----- 01-NOV-91 SQL> select round(sysdate,'day') from dual; ROUND(SYS ----- 20-OCT-91</pre>
--	--

trunc (<i>date</i> , [<i>format</i>])	tronque date selon le format indiqué. Par défaut, c'est la composante heure qui est supprimée. Formats year : tronque au 1er janvier de l'année de date month : tronque au 1er du mois de date day : tronque au dimanche qui précède date. Exemples <pre>SQL> select sysdate from dual; SYSDATE ----- 21-OCT-91 SQL> select trunc(sysdate,'year') from dual; TRUNC(SYS ----- 01-JAN-91 SQL> select trunc(sysdate,'month') from dual; TRUNC(SYS ----- 01-OCT-91 SQL> select trunc(sysdate,'day') from dual; TRUNC(SYS ----- 20-OCT-91</pre>
---	---

3.3.5 Fonctions à paramètres de type variable

greatest (<i>expr1</i> , <i>expr2</i> , ..., <i>exprn</i>)	donne la plus "grande" des valeurs <i>expr1</i> , <i>expr2</i> , ..., <i>exprn</i> . Le type de <i>expr1</i> peut être numérique, caractères ou date.
least (<i>expr1</i> , <i>expr2</i> , ..., <i>exprn</i>)	donne la plus "petite" des valeurs <i>expr1</i> , <i>expr2</i> , ..., <i>exprn</i> . Le type de <i>expr1</i> peut être numérique, caractères ou date.

Exemples

```
SQL> select greatest(100,200,300) grand, least(100,200,300) petit from dual
```

```
      GRAND PETIT
-----
      300    100
```

```
SQL> select greatest('chat','chien','veau') grand, least('chat','chien','veau')
petit from dual;
```

```
      GRAND PETIT
-----
      veau  chat
```

```
SQL> select greatest('01-jan-91','10-feb-91','15-sep-87') grand, least ('01-jan-91', '10-feb-91', '15-sep-
87') petit from dual;
```

```
      GRAND  PETIT
-----
15-sep-87 01-jan-91
```

```
SQL>      select      greatest(to_date('01-jan-91'),      to_date('10-feb-91'),to_date('15-sep-87'))      grand,
least(to_date('01-jan-91'), to_date('10-feb-91'),to_date('15-sep-87')) petit from dual
```

```
      GRAND  PETIT
-----
10-FEB-91 15-SEP-87
```

3.3.6 Fonctions diverses

<code>user</code>	nom de connexion Oracle de l'utilisateur
<code>uid</code>	n° identifiant chaque utilisateur Oracle
<code>userenv(option)</code>	options 'terminal' : identificateur du terminal de l'utilisateur 'language' : identificateur de la langue utilisée par Oracle

Exemples

```
SQL> select user,uid,userenv('terminal'),userenv('language') from dual;
```

```
USER      UID      USERENV( USERENV('LANGUAGE')
-----
SERGE      9      ttyc3d13 AMERICAN_AMERICA.US7ASCII
```

4 Approfondissement du langage SQL

4.1 Introduction

Dans ce chapitre nous voyons

- . d'autres syntaxes de la commande SELECT qui en font une commande de consultation très puissante notamment pour consulter plusieurs tables à la fois.
- . des syntaxes élargies de commandes déjà étudiées
- . les commandes de gestion de la sécurité des données.
- . Les commandes de gestion des performances des requêtes

Pour illustrer les diverses commandes étudiées, nous travaillerons avec les tables suivantes utilisées pour la gestion des commandes dans une PME de diffusion de livres :

la table CLIENTS

Elle mémorise des informations sur les clients de la PME. Sa structure est la suivante :

NOM - char(30)	: nom du client
STATUT - char(1)	: I=Individu, E=Entreprise, A=Administration
PRENOM - char(20)	: prénom dans le cas d'un individu
CONTACT - char(30)	: Nom de la personne à contacter chez le client (dans le cas d'une entreprise ou d'une administration)
RUE - char(25)	: Adresse du client - rue
VILLE - char(20)	: ville
CPOSTAL - char(5)	: code postal
TELEPH - char(20)	: Téléphone
DEPUIS - date	: Client depuis quelle date ?
IDCLI - char(6)	: n° identifiant le client de façon unique
DEBITEUR - char(1)	: O (Oui) si le client doit de l'argent à l'entreprise et N (Non) sinon.

exemple :

NOM	: LIBRAIRIE LA NACELLE
STATUT	: E
PRENOM	:
CONTACT	: Catherine Duchemin
RUE	: 115,Bd du Montparnasse
VILLE	: PARIS
CPOSTAL	: 75014
TELEPH	: 16-1-45-56-67-78
DEPUIS	: 06-APR-81
IDCLI	: 001006

DEBITEUR | : N

la table STOCKS

Elle mémorise des informations sur les produits vendus, ici des livres. Sa structure est la suivante :

ISBN - char(13)	: n° identifiant un livre de façon unique (ISBN= International Standard Book Number)
TITRE - char(30)	: Titre du livre
CODEDITEUR - char(7)	: Code identifiant un éditeur de façon unique
AUTEUR - char(30)	: nom de l'auteur
RESUME - char(400)	: résumé du livre
QTEANCOUR - number(4)	: Quantité vendue dans l'année
QTEANPREC - number(4)	: Quantité vendue l'année précédente
DERNVENTE - date	: date de la dernière vente
QTERECUE - number(3)	: Quantité de la dernière livraison
DERNLIV - date	: Date de la dernière livraison
PRIXVENTE - number(6,2)	: Prix de vente
COUT - number(6,2)	: Coût d'achat
MINCDE - number(3)	: Quantité minimale à commander
MINSTOCK - number(3)	: Seuil minimal du stock
QTESTOCK - number(3)	: Quantité en stock

exemple :

ISBN	: 0-913577-03-1
TITRE	: La gestion avec DBASE III Plus
CODEDITEUR	: 104
AUTEUR	: BYERS
RESUME	:
QTEANCOUR	: 32
QTEANPREC	: 187
DERNVENTE	: 08-AUG-89
QTERECUE	: 40
DERNLIV	: 07-JUL-89
PRIXVENTE	: 350
COUT	: 280
MINCDE	: 20
MINSTOCK	: 10
QTESTOCK	: 32

la table COMMANDES

Conclusion

Elle enregistre les informations sur les commandes faites par les clients. Sa structure est la suivante :

NOCMD - number(6)	: N° identifiant une commande de façon unique
IDCLI - char(6)	: N° du client faisant cette commande (cf fichier CLIENTS)
DATE_CMD - date	: Date de saisie de cette commande
ANNULE - char(1)	: O (Oui) si la commande a été annulée et N (Non) sinon.

exemple :

NOCMD	: 100204
IDCLI	: 001006
DATE	: 15-AUG-89
ANNULE	: N

la table ARTICLES

Elle contient les détails d'une commande, c'est à dire les références et quantités des livres commandés. Sa structure est la suivante :

NOCMD - number(6)	: N° de la commande (cf fichier COMMANDES)
ISBN - char(13)	: n° du livre commandé (cf fichier STOCKS)
QTE - number(3)	: Quantité commandée

Ainsi une commande demandant trois livres différents, donnera naissance à trois lignes dans la table ARTICLES, une pour chaque livre, avec cependant dans les trois cas, le même n° de commande.

exemple : pour la commande 100204 présentée ci-dessus, qui référencerait deux livres :

NOCMD	ISBN	QTE
100204	0-912677-45-7	100
100204	0-912677-16-3	50

Leur contenu est le suivant :

```
SQL> select nom,statut,ville,cpostal,depuis,idcli,debiteur from clients;
```

NOM	S	VILLE	CPOST	DEPUIS	IDCLI	D
LIBRAIRIE LA COMETE	E	ANGERS	49000	01-MAR-88	000001	N
LIBRAIRIE DU MARCHE	E	ANGERS	49100	01-APR-89	000002	O
TRESOR PUBLIC	A	ANGERS	49000	01-JAN-87	000003	N
MAIRIE D'ANGERS	A	SAUMUR	49700	01-FEB-78	000004	O
TELELOGOS	E	SEGRE	49500	01-OCT-77	000005	N
BARNARD	I	CHOLET	49800	01-SEP-77	000006	N
PREFECTURE DU M & L	A	ANGERS	49000	10-DEC-66	000007	O
ARTUS	E	AVRILLE	49350	14-JUN-88	000008	N
MECAFLUID	E	SAUMUR	49550	23-JUL-90	000009	N
PLUCHOT	I	SEGRE	49100	21-FEB-89	000010	O

10 rows selected.

```
SQL> select isbn,titre,qteancour,qteanprec,dernvente,qterecue
2 prixvente,cout,mincde,minstock,qtestock from stocks;
```

ISBN	TITRE	QTEAN COUR	QTEAN PREC	DERNV ENTE	QTERE CUE	PRIXV ENTE	CO UT	MIN CDE	MINST OCK	QTEST OCK
0-07-881551-7	DVORAK'S GUIDE TOO TELECOM	100	500	14-FEB-90	50	250	200	50	20	100
0-07-881309-3	USING 1-2-3 RELEASE 3	1000	5000	01-AUG-91	100	200	180	100	30	430
0-07-881524-X	USING SQL	800	1000	06-SEP-91	40	320	250	50	30	210
0-07-881497-9	DOS - THE COMPLETE REFERENCE	670	8000	14-SEP-91	500	340	270	100	100	780
0-07-881520-7	USING QUICK PASCAL	150	600	14-SEP-91	80	280	230	40	40	200
0-07-881537-1	USING WINDOWS 3	45	0	18-SEP-91	50	450	400	30	15	67

6 rows selected.

```
SQL> select * from commandes;
```

NOCMD	IDCLI	DATE_CMD	A
1	000003	21-SEP-91	N
2	000002	21-SEP-91	N
3	000005	23-SEP-91	N
4	000006	24-SEP-91	N
5	000002	25-SEP-91	N
7	000004	25-SEP-91	N
9	000005	26-SEP-91	N
10	000003	26-SEP-91	N

```
SQL> select * from articles;
```

NOCMD	ISBN	QTE
1	0-07-881551-7	2
1	0-07-881524-X	1
2	0-07-881537-1	4
3	0-07-881520-7	1
3	0-07-881551-7	1
3	0-07-881497-9	4
4	0-07-881551-7	2
4	0-07-881309-3	4
5	0-07-881309-3	1
7	0-07-881551-7	1
9	0-07-881524-X	3
10	0-07-881497-9	12

4.2 La commande SELECT

Nous nous proposons ici d'approfondir notre connaissance de la commande SELECT en présentant de nouvelles syntaxes.

4.2.1 Requête multi-tables

4.2.2 La jointure entre deux tables

Soient deux tables *table1* et *table2*. *table1* a les colonnes *col1* et *col2* et *table2* les colonnes *cola*, *colb*. Supposons que le contenu des tables soit le suivant :

table1	col1	col2	table2	cola	colb
	x	3		a	7
	y	4		b	4

Soit la commande :

```
SELECT col1, cola FROM table1, table2
WHERE table1.col2=table2.colb
```

Cette requête extrait des données de deux tables : *table1* et *table2* avec une condition entre deux colonnes de tables différentes. On appelle ce type de requête, une **jointure**. Comment fonctionne-t-elle ?

Une nouvelle table est construite avec pour colonnes, l'ensemble des colonnes des deux tables et pour lignes le produit cartésien des deux tables :

col1	col2	cola	colb
x	3	a	7
x	3	b	4
y	4	a	7
y	4	b	4

La condition *WHERE col2=colb* est appliquée à cette nouvelle table. On obtient donc la nouvelle table suivante :

col1	col2	cola	colb
y	4	b	4

Il y a ensuite affichage des colonnes demandées :

col1	cola
y	b

4.2.2.1 Syntaxe d'une requête multi-tables

```
syntaxe | SELECT colonne1, colonne2, ...
FROM table1, table2, ..., tablep
WHERE condition
ORDER BY ...
```

action | La nouveauté ici vient du fait que les colonnes *colonne1*, *colonne2*, ... proviennent de plusieurs tables *table1*, *table2*, ... Si deux tables ont des colonnes de même nom, on lève l'ambiguïté par la notation *tablei.colonnej*. La *condition* peut porter sur les colonnes des différentes tables.

Fonctionnement

- 1 La table produit cartésien de *table1*, *table2*, ..., *tablep* est réalisée. Si *ni* est le nombre de lignes de *tablei*, la table construite a donc $n1*n2*...*np$ lignes comportant l'ensemble des colonnes des différentes tables.
- 2 La *condition* du **WHERE** est appliquée à cette table. Une nouvelle table est ainsi produite
- 3 Celle-ci est ordonnée selon le mode indiqué dans **ORDER**.
- 4 Les colonnes demandées derrière **SELECT** sont affichées.

Exemples

On utilise les tables présentées précédemment. On veut connaître le détail des commandes passées après le 25 septembre :

```
SQL>
1 select articles.nocmd,isbn,qte from commandes,articles
2 where date_cmd>'25-sep-91'
3 and articles.nocmd=commandes.nocmd;
```

NOCMD	ISBN	QTE
9	0-07-881524-X	3
10	0-07-881497-9	12
10	0-07-881520-7	2

On remarquera que derrière FROM, on met le nom de toutes les tables dont on référence les colonnes. Dans l'exemple précédent, les colonnes sélectionnées appartiennent toutes à la table articles. Cependant la condition fait référence à la table commandes. D'où la nécessité de nommer cette dernière derrière le FROM. L'opération qui teste l'égalité de colonnes de deux tables différentes est souvent appelée une équi-jointure.

Continuons nos exemples. On désire le même résultat que précédemment mais avec le titre du livre commandé plutôt que son n° ISBN :

```
SQL>
1 select commandes.nocmd, stocks.titre, qte
2 from commandes,articles,stocks
3 where date_cmd>'25-sep-91'
4 and articles.nocmd=commandes.nocmd
5 and articles.isbn=stocks.isbn
```

NOCMD	TITRE	QTE
9	USING SQL	3
10	DOS - THE COMPLETE REFERENCE	12
10	USING QUICK PASCAL	2

On veut de plus le nom du client qui fait la commande :

```
SQL>
1 select commandes.nocmd, stocks.titre, qte ,clients.nom
2 from commandes,articles,stocks,clients
3 where date_cmd>'25-sep-91'
4 and articles.nocmd=commandes.nocmd
5 and articles.isbn=stocks.isbn
6 and commandes.idcli=clients.idcli;
```

NOCMD	TITRE	QTE	NOM
10	DOS - THE COMPLETE REFERENCE	12	TRESOR PUBLIC
10	USING QUICK PASCAL	2	TRESOR PUBLIC
9	USING SQL	3	TELELOGOS

On veut de plus les dates de commande et un affichage par ordre décroissant de ces dates :

```
SQL>
1 select commandes.nocmd, date_cmd, stocks.titre, qte ,clients.nom
2 from commandes,articles,stocks,clients
3 where date_cmd>'25-sep-91'
4 and articles.nocmd=commandes.nocmd
5 and articles.isbn=stocks.isbn
6 and commandes.idcli=clients.idcli
```

```

7 order by date_cmd desc;

NOCMD DATE_CMD  TITRE                                QTE  NOM
-----
10 26-SEP-91  DOS - THE COMPLETE REFERENCE  12  TRESOR PUBLIC
10 26-SEP-91  USING QUICK PASCAL           2  TRESOR PUBLIC
9  26-SEP-91  USING SQL                     3  TELELOGOS

```

Voici quelques règles à observer dans ces jointures :

- 1 Derrière SELECT, on met les colonnes que l'on désire obtenir à l'affichage. Si la colonne existe dans diverses tables, on la fait précéder du nom de la table.
- 2 Derrière FROM, on met toutes les tables qui seront explorées par le SELECT, c'est à dire les tables propriétaires des colonnes qui se trouvent derrière SELECT et WHERE.

4.2.2.2 L'auto-jointure

On veut connaître les livres qui ont un prix de vente supérieur à celui du livre 'USING SQL' :

```

SQL>
1 select a.titre from stocks a, stocks b
2   where b.titre='USING SQL'
3   and a.prixvente>b.prixvente

TITRE
-----
DOS - THE COMPLETE REFERENCE
USING WINDOWS 3

```

Les deux tables de la jointure sont ici identiques : la table *stocks*. Pour les différencier, on leur donne un alias : *from stocks a, stocks b*

L'alias de la première table s'appelle *a* et celui de la seconde, *b*. Cette syntaxe peut s'utiliser même si les tables sont différentes. Lors de l'utilisation d'un alias, celui-ci doit être utilisé partout dans la commande SELECT en lieu et place de la table qu'il désigne.

4.2.2.3 Jointure externe

Soit la jointure déjà étudiée :

```

SELECT col1, cola FROM table1, table2
WHERE table1.col2=table2.colb

```

Le produit cartésien des tables *table1* et *table2* est fait et le WHERE opère sur cette nouvelle table. Si pour une ligne de *table1*, on ne trouve aucune ligne de *table2* telle que *table2.colb=table1.col2*, la ligne correspondante de *table1* n'apparaîtra pas dans le résultat de la jointure. Parfois, on désire cependant que la ligne sorte avec l'indication qu'elle n'a pas de correspondant dans *table2*. On appelle cela, une **jointure externe** et sa syntaxe est la suivante :

```

SELECT col1, cola FROM table1, table2
WHERE table1.col2=table2.colb (+)

```

La syntaxe **WHERE** *table1.col2=table2.colb(+)* indique que les lignes de *table1* sans correspondant dans *table2*, seront associées à des lignes vides de *table2*. De même la syntaxe

```

SELECT col1, cola FROM table1, table2
WHERE table1.col2(+) = table2.colb

```

indique que les lignes de *table2* sans correspondant dans *table1*, seront associées à des lignes vides de *table1*.

Exemples

On veut connaître les clients qui ont acheté quelque chose en septembre avec la date de la commande. Les autres clients sont sortis sans cette date :

```
SQL>
1 select nom,date_cmd from clients,commandes
2   where date_cmd between '01-sep-91' and '30-sep-91'
3*  and clients.idcli=commandes.idcli (+)
```

NOM	DATE_CMD
LIBRAIRIE DU MARCHE	21-SEP-91
LIBRAIRIE DU MARCHE	25-SEP-91
TRESOR PUBLIC	21-SEP-91
TRESOR PUBLIC	26-SEP-91
MAIRIE D'ANGERS	25-SEP-91
TELELOGOS	23-SEP-91
TELELOGOS	26-SEP-91
BARNARD	24-SEP-91

On est étonné ici de ne pas avoir le bon résultat. Lorsqu'on réfléchit au fonctionnement de la jointure externe, on réalise que les clients n'ayant pas acheté ont été associées à une ligne vide de la table commandes et donc avec une date vide (valeur NULL dans la terminologie SQL). Cette date ne vérifie pas alors la condition fixée sur la date et le client correspondant n'est pas affiché. Essayons autre chose :

```
SQL>
1 select nom,date_cmd from clients,commandes
2   where ( date_cmd between '01-sep-91' and '30-sep-91'
3   or date_cmd is null)
4*  and clients.idcli=commandes.idcli (+)
```

NOM	DATE_CMD
LIBRAIRIE LA COMETE	
LIBRAIRIE DU MARCHE	21-SEP-91
LIBRAIRIE DU MARCHE	25-SEP-91
TRESOR PUBLIC	21-SEP-91
TRESOR PUBLIC	26-SEP-91
MAIRIE D'ANGERS	25-SEP-91
TELELOGOS	23-SEP-91
TELELOGOS	26-SEP-91
BARNARD	24-SEP-91
PREFECTURE DU M & L	
ARTUS	
MECAFLUID	
PLUCHOT	

On obtient cette fois-ci la réponse à notre question. Revenons sur la position du (+) dans la jointure. Que signifierait la commande :

```
SQL>
1 select nom,date_cmd from clients,commandes
2   where ( date_cmd between '01-sep-91' and '30-sep-91'
3   or date_cmd is null)
4   and clients.idcli (+)=commandes.idcli
```

Cette fois-ci ce sont les commandes qui n'auraient pas de correspondant (par *idcli*) dans la table *clients* qui seraient associées à des lignes vides de *clients*. Dans notre exemple, ce cas n'est pas possible : toute commande est faite par un client.

4.2.3 Requêtes imbriquées

syntaxe	SELECT <i>colonne[s]</i> FROM <i>table[s]</i> WHERE <i>expression opérateur requête</i> ORDER BY ...
---------	---

fonctionnement

requête est une commande SELECT qui délivre un groupe de 0, 1 ou plusieurs valeurs. On a alors une condition **WHERE** du type

expression opérateur (val1, val2, ..., vali)

expression et *vali* doivent être de même type. Si la requête délivre une seule valeur, on est ramené à une condition du type

expression opérateur valeur

que nous connaissons bien. Si la requête délivre une liste de valeurs, on pourra employer les opérateurs suivants :

IN	<i>expression IN (val1, val2, ..., vali)</i> vraie si <i>expression</i> a pour valeur l'un des éléments de la liste <i>vali</i> .
NOT IN	inverse de IN
ANY	doit être précédé de =, !=, >, >=, <, <= <i>expression</i> >= ANY (<i>val1, val2, ..., valn</i>) vraie si <i>expression</i> est >= à l'une des valeurs <i>vali</i> de la liste
ALL	doit être précédé de =, !=, >, >=, <, <= <i>expression</i> >= ALL (<i>val1, val2, ..., valn</i>) vraie si <i>expression</i> est >= à toutes les valeurs <i>vali</i> de la liste
EXISTS	<i>requête</i> vraie si la <i>requête</i> rend au moins une ligne.

Exemples

On reprend la question déjà résolue par une équi-jointure : Afficher les titres ayant un prix de vente supérieur à celui du livre 'USING SQL'.

```
SQL>
1 select titre from stocks
2 where prixvente > (select prixvente from stocks where titre='USING SQL')
```

```
TITRE
-----
DOS - THE COMPLETE REFERENCE
USING WINDOWS 3
```

Cette solution semble plus intuitive que celle de l'équi-jointure. On fait un premier filtrage avec un SELECT, puis un second sur le résultat obtenu. On peut opérer ainsi plusieurs filtrages en série.

On veut connaître les titres ayant un prix de vente supérieur au prix moyen de vente :

```
SQL>
1 select titre from stocks
2 where prixvente > (select avg(prixvente) from stocks)
```

```
TITRE
-----
USING SQL
DOS - THE COMPLETE REFERENCE
USING WINDOWS 3
```

Quels sont les clients ayant commandé les titres résultat de la requête précédente ?

```
SQL>
1 select distinct idcli from commandes ,articles
2 where articles.isbn in (select isbn from stocks where prixvente
3 > (select avg(prixvente) from stocks))
4 and commandes.nocmd=articles.nocmd
```

```
IDCLI
-----
000002
000003
000005
```

Explications

a on sélectionne dans la table articles, les codes ISBN se trouvant parmi les livres de prix supérieur au prix moyen

- b dans les lignes sélectionnées de la table articles, il n'y a pas le code client IDCLI. Il se trouve dans la table commandes. Le lien entre les deux tables se fait par le n° de commande nocmd, d'où l'équi-jointure commandes.nocmd = articles.nocmd.
- c Un même client peut avoir acheté plusieurs fois l'un des livres concernés, auquel cas on aura son code IDCLI plusieurs fois. Pour éviter cela, on met le mot clé DISTINCT derrière SELECT.
- d pour avoir le nom du client, il nous faudrait faire une équi-jointure supplémentaire entre les tables commandes et clients.

Trouver les clients qui n'ont pas fait de commande depuis le 24 septembre :

```
SQL>
1  select nom from clients
2  where clients.idcli not in (select distinct commandes.idcli
3  from commandes where date_cmd>='24-sep-91')
```

```
NOM
-----
LIBRAIRIE LA COMETE
PREFECTURE DU M & L
ARTUS
MECAFLUID
PLUCHOT
```

Nous avons vu qu'on pouvait filtrer des lignes autrement par la clause WHERE : en utilisant la clause HAVING en conjonction avec la clause GROUP BY. La clause HAVING filtre des groupes de lignes.

De la même façon que pour la clause WHERE, la syntaxe

HAVING *expression opérateur requête*

est possible, avec la contrainte déjà présentée que *expression* doit être l'une des expressions *expr1* de la clause

GROUP BY *expr1, expr2, ...*

Exemples

Quelles sont les quantités vendues pour les livres de plus de 100F ?

Sortons d'abord les quantités vendues par titre :

```
SQL>
1  select titre,sum(qte) from stocks, articles
2  where articles.isbn=stocks.isbn
3  group by titre
```

```
TITRE                SUM(QTE)
-----
DOS - THE COMPLETE REFERENCE  16
DVORAK'S GUIDE TOO TELECOM    6
USING 1-2-3 RELEASE 3         5
USING QUICK PASCAL            3
USING SQL                     4
USING WINDOWS 3              4
```

Maintenant, filtrons les titres :

```
SQL>
1  select titre,sum(qte) from stocks,commandes,articles
2  where articles.isbn=stocks.isbn
3  group by titre
4  having titre in (select titre from stocks where prixvente>200)
```

```
TITRE                SUM(QTE)
-----
DOS - THE COMPLETE REFERENCE  16
DVORAK'S GUIDE TOO TELECOM    6
USING QUICK PASCAL            3
USING SQL                     4
USING WINDOWS 3              4
```

De façon peut-être plus évidente on aurait pu écrire :

```
SQL>
1  select titre,sum(qte) from stocks,commandes,articles
2  where articles.nocmd=commandes.nocmd
```



```

3 and articles.isbn=stocks.isbn
4 and prixvente>200
5* group by titre

```

TITRE	SUM(QTE)
DOS - THE COMPLETE REFERENCE	16
DVORAK'S GUIDE TOO TELECOM	6
USING QUICK PASCAL	3
USING SQL	4
USING WINDOWS 3	4

4.2.4 Requêtes corrélées

Dans le cas des requêtes imbriquées, on a une requête parent (la requête la plus externe) et une requête fille (la requête la plus interne). La requête mère n'est évaluée que lorsque la requête fille l'a été complètement.

Les requêtes corrélées ont la même syntaxe au détail près suivant : la requête fille fait une jointure sur la table de la requête mère. Dans ce cas, l'ensemble requête mère-requête fille est évaluée de façon répétée pour chaque ligne de la table mère.

Exemple

Nous reprenons l'exemple où nous désirons les noms des clients n'ayant pas fait de commande depuis le 24 septembre :

```

SQL>
1 select nom from clients
2 where not exists
3 (select idcli from commandes
4   where date_cmd>='24-sep-91'
5   and commandes.idcli=clients.idcli)

```

```

NOM
-----
LIBRAIRIE LA COMETE
PREFECTURE DU M & L
ARTUS
MECAFLUID
PLUCHOT

```

La requête mère s'exerce sur la table *clients*. La requête fille fait une jointure entre les tables *clients* et *commandes*. On a donc une requête corrélée. Pour chaque ligne de la table *clients*, la requête fille s'exécute : elle cherche le code *idcli* du client dans les commandes faites après le 24 septembre. Si elle n'en trouve pas (*not exists*), le nom du client est affiché. Ensuite, on passe à la ligne suivante de la table *clients*.

4.2.5 Critères de choix pour l'écriture du SELECT

Nous avons vu, à plusieurs reprises, qu'il était possible d'obtenir un même résultat par différentes écriture du SELECT. Prenons un exemple : Afficher les clients ayant commandé quelque chose :

Jointure

```

SQL>
1 select distinct nom from clients,commandes
2 where clients.idcli=commandes.idcli

```

```

NOM
-----
BARNARD
LIBRAIRIE DU MARCHE
MAIRIE D'ANGERS
TELELOGOS
TRESOR PUBLIC

```

REQUETES IMBRIQUEES

```

SQL>
1 select nom from clients
2 where idcli in (select idcli from commandes)

```

```
NOM
-----
LIBRAIRIE DU MARCHE
TRESOR PUBLIC
MAIRIE D'ANGERS
TELELOGOS
BARNARD
```

REQUETES CORRELEES

```
SQL>
1  select nom from clients
2  where exists (select * from commandes where commandes.idcli=clients.idcli)
```

```
NOM
-----
LIBRAIRIE DU MARCHE
TRESOR PUBLIC
MAIRIE D'ANGERS
TELELOGOS
BARNARD
```

Les auteurs Christian MAREE et Guy LEDANT, dans leur livre 'SQL, Initiation, Programmation et Maîtrise' proposent quelques critères de choix :

Performances

L'utilisateur ne sait pas comment ORACLE "se débrouille" pour trouver les résultats qu'il demande. Ce n'est donc que par expérience, qu'il découvrira que telle écriture est plus performante qu'une autre. MAREE et LEDANT affirment par expérience que les requêtes corrélées semblent généralement plus lentes que les requêtes imbriquées ou les jointures.

FORMULATION

La formulation par requêtes imbriquées est toujours plus lisible et plus intuitive que la jointure. Elle n'est cependant pas toujours utilisable. La règle est simple :

Les tables propriétaires des colonnes arguments du SELECT (SELECT col1, col2, ...) doivent être nommées derrière FROM. Le produit cartésien de ces tables est alors effectué, ce qu'on appelle une jointure.

Lorsque la requête affiche des résultats provenant d'une seule table, et que le filtrage des lignes de cette dernière impose la consultation d'une autre table, les requêtes imbriquées s'imposent.

4.3 Extensions de syntaxe

Pour des questions de commodité, nous avons le plus souvent présenté des syntaxes réduites des différentes commandes. Dans cette section, nous en présentons des syntaxes élargies. Elles se comprennent d'elles-mêmes car elles sont analogues à celles de la commande SELECT largement étudiée.

CREATE

syntaxe1	CREATE TABLE (<i>colonne1 type1 contrainte1, colonne2 type2 contrainte2 ...</i>)
explication	Cette syntaxe a déjà été étudiée. Nous avons déjà vu deux contraintes : UNIQUE et NOT NULL. Il en existe d'autres, mais qui dans la version actuelle d'Oracle ne sont pas testées, quoique la syntaxe les accepte ! Donc nous ne les présentons pas.

syntaxe2	CREATE TABLE (<i>colonne1 type1 contrainte1, colonne2 type2 contrainte2 ...</i>)
	AS requête
explication	Cette syntaxe a déjà été étudiée. Rappelons qu'elle permet de créer une table à partir d'une autre table.

INSERT

Conclusion

syntaxe1 | **INSERT INTO** *table* (*col1*, *col2*, ..) **VALUES** (*val1*, *val2*, ...)

syntaxe2 | **INSERT INTO** *table* (*col1*, *col2*, ..) (*requête*)

explication | Ces deux syntaxes ont été présentées

DELETE

syntaxe1 | **DELETE FROM** *table* **WHERE** *condition*

explication | Cette syntaxe est connue. Ajoutons que la condition peut contenir une requête avec la syntaxe *WHERE expression opérateur (requête)*

UPDATE

syntaxe1 | **UPDATE** *table*
SET *col1=expr1*, *col2=expr2*, ...
WHERE *condition*

explication | Cette syntaxe a déjà été présentée. Ajoutons que la condition peut contenir une requête avec la syntaxe **WHERE** *expression opérateur (requête)*

syntaxe2 | **UPDATE** *table*
SET (*col1*, *col2*, ..)= *requête1*, (*cola*, *colb*, ..)= *requête2*, ...
WHERE *condition*

explication | Les valeurs affectées aux différentes colonnes peuvent provenir d'une requête.

4.4 Gestion de l'accès concurrent aux données

Nous avons jusqu'à maintenant utilisé des tables dont nous étions les seuls utilisateurs. Dans la pratique, sur une machine multi-utilisateurs, les données sont le plus souvent partagées entre différents utilisateurs. Se pose alors la question : qui peut utiliser telle ou telle table et sous quelle forme (consultation, insertion, suppression, ajout, ...) ?

Appelons environnement de l'utilisateur, les tables et vues auxquelles il peut accéder. Cet environnement est décrit dans la table système **ACCESSIBLE_TABLES** :

```
SQL> describe accessible_tables
```

Name	Null?	Type
OWNER	NOT NULL	CHAR(30)
TABLE_NAME	NOT NULL	CHAR(30)
TABLE_TYPE		CHAR(11)

```
SQL> select * from accessible_tables where rownum<=10;
```

OWNER	TABLE_NAME	TABLE_TYPE
SYS	AUDIT_ACTIONS	TABLE
SYS	USER_AUDIT_TRAIL	VIEW
SYS	USER_AUDIT_CONNECT	VIEW
SYS	USER_AUDIT_RESOURCE	VIEW
SYS	DUAL	TABLE
SYS	USER_CATALOG	VIEW
SYS	ALL_CATALOG	VIEW
SYS	ACCESSIBLE_TABLES	VIEW
SYS	USER_CLUSTERS	VIEW
SYS	USER_CLU_COLUMNS	VIEW

```
10 rows selected.
```

On constate que l'utilisateur a accès à un grand nombre de tables. Ce sont pour la plupart des tables du système Oracle formant ce qu'on appelle le "Dictionnaire des données". Nous y reviendrons ultérieurement.

4.4.1 Les privilèges d'accès aux tables et vues

Lorsqu'un utilisateur crée une table ou une vue, il en est le propriétaire exclusif. Les autres utilisateurs n'y ont pas accès. Il peut cependant en donner un droit d'utilisation à d'autres par la commande **GRANT**.

syntaxe	GRANT <i>privilège1, privilège2, ...</i> ALL PRIVILEGES ON <i>table/vue</i> TO <i>utilisateur1, utilisateur2, ...</i> PUBLIC [WITH GRANT OPTION]
action	accorde des privilèges d'accès <i>privilègei</i> ou tous les privilèges (ALL PRIVILEGES) sur la <i>table</i> ou <i>vue</i> aux utilisateurs <i>utilisateuri</i> ou à tous les utilisateurs (PUBLIC). La clause WITH GRANT OPTION permet aux utilisateurs ayant reçu les privilèges de les transmettre à leur tour à d'autres utilisateurs.

Les privilèges *privilègei* qui peuvent être accordés sont les suivants :

ALTER	droit d'utiliser la commande ALTER TABLE sur la table.
DELETE	droit d'utiliser la commande DELETE sur la table ou vue.
INSERT	droit d'utiliser la commande INSERT sur la table ou vue
SELECT	droit d'utiliser la commande SELECT sur la table ou vue
UPDATE	droit d'utiliser la commande UPDATE sur la table ou vue. Ce droit peut être restreint à certaines colonnes par la syntaxe : GRANT update (col1, col2, ...) ON table/vue TO utilisateur1, utilisateur2, ... PUBLIC [WITH GRANT OPTION]
INDEX	droit d'utiliser la commande CREATE INDEX sur la table.

Trois tables du système donnent des indications sur les privilèges accordés ou reçus :

USER_TAB_GRANTS	tables pour lesquelles on a accordé ou reçu des privilèges
USER_TAB_GRANTS_MADE	tables pour lesquelles on a accordé des privilèges
USER_TAB_GRANTS_RECV	tables pour lesquelles on a reçu des privilèges

Leur structure est la suivante :

```
SQL> describe user_tab_grants
```

Name	Null?	Type	
GRANTEE	NOT NULL	CHAR(30)	<-- celui qui reçoit le privilège
OWNER	NOT NULL	CHAR(30)	<-- le propriétaire de la table ou vue
TABLE_NAME	NOT NULL	CHAR(30)	<-- la table ou vue
GRANTOR	NOT NULL	CHAR(30)	<-- celui qui accorde les privilèges
SELECT_PRIV		CHAR(1)	<-- privilège SELECT (Y : yes ou N : no)
INSERT_PRIV		CHAR(1)	<-- privilège INSERT
DELETE_PRIV		CHAR(1)	<-- privilège DELETE
UPDATE_PRIV		CHAR(1)	<-- privilège UPDATE
REFERENCES_PRIV		CHAR(1)	
ALTER_PRIV		CHAR(1)	<-- privilège ALTER
INDEX_PRIV		CHAR(1)	<-- privilège INDEX
CREATED	NOT NULL	DATE	<-- date d'octroi des privilèges

```
SQL> describe user_tab_grants_made
```

Name	Null?	Type
GRANTEE	NOT NULL	CHAR(30)
TABLE_NAME	NOT NULL	CHAR(30)
GRANTOR	NOT NULL	CHAR(30)
SELECT_PRIV		CHAR(1)
INSERT_PRIV		CHAR(1)
DELETE_PRIV		CHAR(1)
UPDATE_PRIV		CHAR(1)
REFERENCES_PRIV		CHAR(1)
ALTER_PRIV		CHAR(1)
INDEX_PRIV		CHAR(1)
CREATED	NOT NULL	DATE

```
SQL> describe user_tab_grants_recd
```

Name	Null?	Type
OWNER	NOT NULL	CHAR(30)
TABLE_NAME	NOT NULL	CHAR(30)
GRANTOR	NOT NULL	CHAR(30)
SELECT_PRIV		CHAR(1)
INSERT_PRIV		CHAR(1)
DELETE_PRIV		CHAR(1)
UPDATE_PRIV		CHAR(1)
REFERENCES_PRIV		CHAR(1)
ALTER_PRIV		CHAR(1)
INDEX_PRIV		CHAR(1)
CREATED	NOT NULL	DATE

```
SQL> select table_name from tabs;
```

TABLE_NAME
ARTICLES
BIBLIO
BIDON <----
CLIENTS
CMD_ID
COMMANDES
SAUVEGARDE
STOCKS

8 rows selected.

```
SQL> grant select,update on bidon to allo;
```

Grant succeeded.

```
SQL> select grantee,table_name,select_priv,update_priv from user_tab_grants_made;
```

GRANTEE	TABLE_NAME	S	U
ALLO	BIDON	Y A	<-- A : ALL (update autorisé sur toutes les colonnes)

Remarque

Un utilisateur ayant reçu des droits d'un utilisateur U sur une table T ne peut référencer celle-ci que par la syntaxe U.T sauf si la table T a un synonyme public lequel peut être créé par l'administrateur avec la commande

create public synonym nom1 for table1

Par exemple si on veut accorder à tous un droit de lecture sur une table T d'un utilisateur U :

- le propriétaire U accordera les droits nécessaires
grant select to public on T
- l'administrateur donnera un synonyme public à la table T de U
create public synonym S for U.T
- tout utilisateur a maintenant accès à la table S
*select * from S*

4.4.2 Suppression des privilèges accordés

Le propriétaire d'un objet peut revenir sur sa décision d'accorder des privilèges d'accès à d'autres utilisateurs par la commande REVOKE :

syntaxe	REVOKE <i>privilege1, privilege2, ...</i> ALL PRIVILEGES ON <i>table/vue</i> FROM <i>utilisateur1, utilisateur2, ...</i> PUBLIC
action	supprime des privilèges d'accès <i>privilegei</i> ou tous les privilèges (ALL PRIVILEGES) sur la <i>table</i> ou <i>vue</i> aux utilisateurs <i>utilisateuri</i> ou à tous les utilisateurs (PUBLIC).

Exemple

```
SQL> revoke all privileges on bidon from allo;
```

Revoke succeeded.

```
SQL> select grantee,table_name,select_priv,update_priv from user_tab_grants_made
```

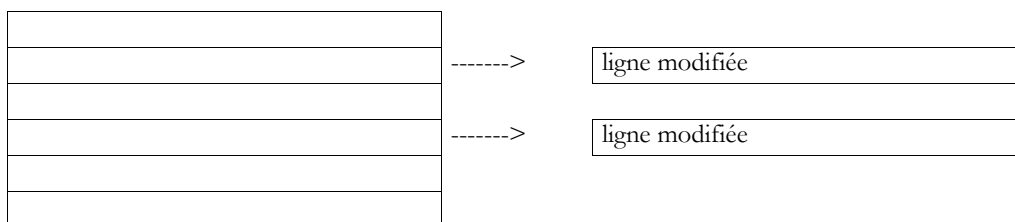
```
no rows selected      <-- il n'y a plus de privilèges sur la table bidon
```

4.4.3 Les transactions

Une transaction est l'ensemble des commandes SQL émises entre un COMMIT/ROLLBACK et le COMMIT/ROLLBACK suivant. Rappelons quelques points sur ces commandes :

- 1 En **consultation**, un utilisateur travaille sur la table originale.
- 2 En **modification** (*update, insert, delete*), les lignes modifiées sont créées en-dehors de la table originale. Celle-ci n'est donc pas modifiée

Original

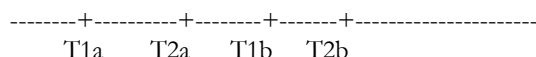


- 3 les lignes de la table originale ne sont accessibles qu'en lecture aux autres utilisateur (ceux qui ont des droits dessus). Toute modification de leur part est bloquée tant que l'utilisateur ayant le premier modifié la table originale n'a pas validé (COMMIT) ou invalidé (ROLLBACK) ses modifications, c.a.d. tant qu'il n'a pas terminé sa transaction.

Quelques commandes génèrent un COMMIT implicite :

- a Les commandes qui modifient la structure des tables :
CREATE TABLE, ALTER TABLE, DROP TABLE, GRANT
- b La déconnexion d'Oracle qu'elle soit normale ou anormale (panne système).

Dans un contexte multi-utilisateurs, la transaction revêt d'autres aspects. Soient deux utilisateurs *U1* et *U2* travaillant sur la même table *TAB* :



La transaction de l'utilisateur *U1* commence au temps *T1a* et finit au temps *T1b*.

La transaction de l'utilisateur *U2* commence au temps *T2a* et finit au temps *T2b*.

U1 travaille sur une photo de *TAB* prise au temps *T1a*. Entre *T1a* et *T1b*, il modifie *TAB*. Les autres utilisateurs n'auront accès à ces modifications qu'au temps *T1b*, lorsque *U1* fera un COMMIT.

U2 travaille sur une photo de *TAB* prise au temps *T2a*, donc la même photo qu'utilisée par *U1* (si d'autres utilisateurs n'ont pas modifié l'original entre-temps). Il ne "voit" pas les modifications qu'a pu apporter l'utilisateur *U1* sur *TAB*. Il ne pourra les voir qu'au temps *T1b*.

Cet exemple donne une nouvelle dimension à la transaction : une transaction est une suite cohérente de commandes SQL qui doit former un tout indissociable. Prenons un exemple en comptabilité : *U1* et *U2* travaillent sur des comptes. *U1* crédite *comptex* d'une somme et débite *comptey* de la même somme pendant que *U2* consulte les comptes.

Supposons que la transaction de *U1* consiste en la mise à jour de *comptex* suivie de la mise à jour de *comptey* et que la transaction de *U2* consiste en la visualisation de l'état des comptes. Nous considérons les deux cas possibles :

- 1 *U1* termine sa transaction après que *U2* ne commence la sienne.
- 2 *U1* termine sa transaction avant que *U2* ne commence la sienne.

Cas 1

Le schéma temporel est le suivant :

```

-----+-----+-----+-----+-----
      T1a   T2a   T1b   T2b

```

Lorsque *U2* consulte, il voit les anciennes valeurs des comptes *comptex* et *comptey*. Cependant la table est cohérente : le total des CREDITS est égal au total des DEBITS.

Cas 2

Le schéma temporel est le suivant :

```

-----+-----+-----+-----+-----
      T1a   T1b   T2a   T2b

```

Lorsque *U2* consulte, il voit les nouvelles valeurs des comptes *comptex* et *comptey*. La table est cohérente : le total des CREDITS est égal au total des DEBITS.

Supposons maintenant que *U1* crédite *comptex* puis valide pour ensuite débiter *comptey* et valider. Il y alors deux transactions de la part de *U1*. L'une entre *T1a* et *T1b*, l'autre entre *T1b* et *T1c*. Supposons que *U2* commence sa transaction au temps *T2* suivant :

```

-----+-----+-----+-----+-----
      T1a   T1b   T2   T1c

```

Au temps *T2*, une photo de la table des comptes est prise pour *U2*. Sur cette photo, il apparaîtra que les comptes ne sont pas équilibrés : *comptex* apparaît crédité alors que *comptey* n'apparaît pas encore débité. C'est une situation anormale.

Généralement, ces problèmes de transaction seront réglés par programme : c'est le concepteur du programme qui choisira les groupes de commandes qui doivent être indissociables et donc faire l'objet d'une transaction. Les transactions resteront le plus souvent inconnues des utilisateurs du programme.

4.4.4 Lecture cohérente

Le mécanisme de la photo est appelé sous Oracle "Read Consistency" que nous traduirons par "Lecture cohérente". Explicitons cette notion sur un exemple. Soit un utilisateur *U1* faisant une requête de consultation sur la table *TAB*, alors qu'un utilisateur *U2* est en train de la modifier selon le schéma temporel suivant :

```

-----+-----+-----+-----+-----
      T1a   T2a   T2b   T1b

```

L'utilisateur *U1* fait sa requête en *T1a*. Celle-ci est terminée en *T1b*.

L'utilisateur *U2* fait sa modification en *T2a*. Celle-ci est terminée en *T2b*.

Entre le temps T1a et le temps T1b, l'état de la table TAB peut changer : notamment au temps T2b lorsque l'utilisateur U2 valide son travail. Cependant, la requête de l'utilisateur U1 ne travaille que sur la photo de la table prise au temps T1a. C'est la notion de lecture cohérente. Une requête de consultation travaille sur une photo prise à un instant T où toutes les données sont dans un état cohérent, stable. Elle ne tient pas compte des changements d'état de la table qui peuvent survenir lors de son exécution.

La lecture cohérente s'applique normalement à une commande SQL isolée. On peut l'étendre à une transaction. Prenons un exemple : l'utilisateur U1 fait des statistiques sur la table TAB :

```
1 select avg(col) from tab; au temps T1
2 select stddev(col) from tab; au temps T2
```

La commande 1 travaille sur une photo prise au temps T1, et la commande 2 sur une photo prise au temps T2. Entre-temps, la table TAB a pu changer. Ce serait gênant dans notre exemple, puisqu'on désire obtenir la moyenne et l'écart-type sur les mêmes données. Ce n'est pas garanti ici.

On peut, pour résoudre le problème, utiliser une transaction spéciale appelée une transaction à lecture seulement (**Read Only Transaction**). Cette transaction prend, lorsqu'elle démarre, une photo de la base et travaille ensuite uniquement sur cette photo, jusqu'à la fin de la transaction. Une telle transaction ne doit comporter que des lectures de la base. Sa syntaxe est la suivante :

```
SET TRANSACTION READ ONLY;  <-- début de la transaction
Commande1;
....
Commanden;
COMMIT;                      <-- fin de la transaction
```

4.4.5 Contrôle par défaut des accès concurrents

Le mécanisme de la lecture cohérente, nous permet de travailler sur un état stable d'une table et non sur un état en cours d'évolution et inachevé. Il ne résout pas tous les problèmes d'accès concurrents aux tables. Considérons de nouveau l'exemple de nos deux utilisateurs U1 et U2 travaillant sur la même table TAB avec le schéma temporel suivant :

```
-----+-----+-----+-----+-----
      T1a   T2a   T1b   T2b
```

La transaction de l'utilisateur U1 commence au temps T1a et finit au temps T1b.
La transaction de l'utilisateur U2 commence au temps T2a et finit au temps T2b.

Supposons que U1 et U2 travaillent avec un système de réservation de places de train. Au temps T1a, une photo de la table des réservations est prise pour U1 : le siège 12 du wagon 20 du TGV 2040 est libre. L'application le réserve pour U1. Au temps T2a, une photo des réservations est prise pour U2. C'est la même que pour U1 qui n'a pas encore terminé sa transaction. L'application risque donc de réserver pour U2 le même siège que pour U1.

Oracle évite ces situations de la façon suivante :

a La commande **SELECT**

Cette commande ne fait que lire des données. C'est le mécanisme de la photo qui assure que les données récupérées sont des données stables et non en cours d'évolution.

b Commandes de **modification du contenu** des tables : **UPDATE, INSERT et DELETE**.

- 1 Une photo cohérente de la table est prise
- 2 La commande de mise à jour obtient un usage exclusif des lignes à mettre à jour. Elle est la seule à pouvoir les modifier jusqu'à ce que la transaction se termine.
- 3 Les lectures par d'autres transactions des lignes bloquées sont autorisées. Une photo cohérente leur sera fournie.
- 4 Les autres transactions peuvent faire ce qu'elles veulent sur les lignes non bloquées.
- 5 Les lignes sont libérées dès que la transaction qui les a bloquées est terminée.

c Commandes de **modification de la structure** d'une table : **CREATE, ALTER, DROP** :

- 1 Un COMMIT implicite est généré.
- 2 La commande obtient un usage exclusif de la table modifiée.
- 3 Les autres transactions ne peuvent que consulter la table (avec une photo de l'ancienne structure) , elles ne peuvent la modifier.
- 4 Après exécution de la commande, un COMMIT implicite est généré libérant la table pour les autres transactions.

De nouveau, l'importance de la transaction apparaît ici : un utilisateur faisant un UPDATE sur une table bloque les lignes mises à jour jusqu'à la fin de la transaction les rendant indisponibles aux autres utilisateurs qui voudraient les modifier. Il est donc souhaitable d'émettre un COMMIT après modification des données. Le plus souvent, c'est un programme qui le fera.

Remarque

Lorsqu'un utilisateur obtient l'usage exclusif d'une table ou de lignes de la table, on dit que la table ou les lignes sont verrouillées.

4.4.6 Contrôle explicite des accès concurrents

Le comportement par défaut d'Oracle dans la gestion des accès concurrents peut être ignoré par un verrouillage explicitement demandé par l'utilisateur. Il y a peu de cas où cela s'avère nécessaire. Citons-en quelques-uns :

- 1 La base que vous utilisez est très demandée. Les temps d'attente deviennent longs. Vous êtes prioritaire : vous demandez alors l'usage exclusif des tables que vous utilisez. Les autres n'y auront plus accès.
- 2 Vous voulez travailler avec un état stable de la base pendant toute la durée d'une transaction. Vous voulez modifier la base. La transaction à lecture seulement ne suffit donc pas : vous demandez un usage exclusif de la table.

Le verrouillage explicite est demandé par :

syntaxe `LOCK TABLE table1, table2,... IN mode MODE [NOWAIT]`

action verrouille *table1*, *table2*, ... dans le mode indiqué par *mode* :

EXCLUSIVE

seules les requêtes sont autorisées sur la table. Celui qui verrouille a seul la possibilité de modifier son contenu.

ROW SHARE

autorise les accès concurrents à la table comme dans la méthode par défaut. Empêche de plus un autre utilisateur de poser un verrou exclusif sur la table.

La clause **NOWAIT** indique qu'il ne faut pas attendre si le verrouillage demandé ne peut être obtenu. En son absence, l'utilisateur est mis en attente de la libération de l'objet demandé.

La fin de la transaction (COMMIT/ROLLBACK) enlève tout verrouillage.

4.5 Gestion des performances

Nous l'avons déjà dit : l'utilisateur ne sachant pas comment Oracle procède pour répondre à ses demandes, il lui est difficile de les formuler de façon à optimiser les temps de réponse. Il peut parfois augmenter les performances à l'aide de deux outils : les *index* et les *clusters*. Nous n'évoquons ici que les index.

4.5.1 Les index

Considérons le fichier BIBLIO. Vous voulez des informations sur le livre 'Manhattan Transfer' :

```
SELECT * FROM biblio WHERE TITRE='Manhattan Transfer'
```

En l'absence d'index, Oracle ira consulter toute la table BIBLIO afin de trouver le livre en question. Le temps de consultation dépendra de la taille de la table.

En faisant de la colonne `TITRE`, un index, un fichier d'index est créé. Oracle ira alors consulter ce fichier pour rechercher le titre 'Manhattan Transfer'. Ce fichier est organisé de manière à favoriser les recherches. Elle est très rapide et peu dépendante de la taille de la table indexée.

L'indexation ralentit la mise à jour des tables, puisque le fichier index doit suivre ces mises à jour.

4.5.2 Création d'un index

syntaxe	CREATE [UNIQUE] INDEX <i>nom_index</i> ON <i>table</i> (<i>expr1</i> , <i>expr2</i> , ..)
action	crée l'index nommé <i>nom_index</i> à partir des expressions <i>expr1</i> , <i>expr2</i> , ... de <i>table</i> . Améliore les temps de recherche sur les expressions participant à l'index. La clause UNIQUE demande à ce que deux lignes différentes de <i>table</i> aient deux index différents.

Remarque

On indexera en général, sur les colonnes intervenant souvent dans la sélection des lignes, c'est à dire se trouvant dans la clause **WHERE** du **SELECT**.

Exemples

```
SQL> select titre, auteur from biblio;

TITRE          AUTEUR
-----
Les fleurs du mal  Baudelaire
Tintin au Tibet   Hergé
La terre          Zola
Madame Bovary     Flaubert
Manhattan transfer Dos Passos
Tintin en Amérique Hergé
Le père Goriot    Balzac

7 rows selected.

SQL> create index titre on biblio(titre);      <-- indexation sur les titres des livres

Index created.

SQL> create index auteur on biblio(auteur);    <-- indexation sur la colonne auteur

Index created.

SQL> create index a date_achat on biblio(achat); <-- indexation sur la date d'achat du livre

Index created.

SQL> create index genre_prix on biblio (genre,prix) <-- index sur deux colonnes

Index created.
```

4.5.3 Obtenir la liste des index

La vue **IND** rassemble les index de l'utilisateur :

```
SQL> describe IND

Name          Null?     Type
-----
INDEX_NAME    NOT NULL CHAR(30)
TABLE_OWNER   NOT NULL CHAR(30)
TABLE_NAME    NOT NULL CHAR(30)
TABLE_TYPE    CHAR(11)
UNIQUENESS    CHAR(9)
TABLESPACE_NAME NOT NULL CHAR(30)
INI_TRANS     NOT NULL NUMBER
MAX_TRANS     NOT NULL NUMBER
INITIAL_EXTENT NUMBER
```

```

NEXT_EXTENT      NUMBER
MIN_EXTENTS      NOT NULL NUMBER
MAX_EXTENTS      NOT NULL NUMBER
PCT_INCREASE     NOT NULL NUMBER

```

```
SQL> select index_name,table_name,uniqueness from ind;
```

INDEX_NAME	TABLE_NAME	UNIQUENES
AUTEUR	BIBLIO	NONUNIQUE
DATE_ACHAT	BIBLIO	NONUNIQUE
GENRE_PRIX	BIBLIO	NONUNIQUE
TITRE	BIBLIO	NONUNIQUE

4.5.4 Abandon d'un index

syntaxe	DROP INDEX <i>nom_index</i>
action	abandonne l'index nommé

Exemples

```

SQL> drop index genre_prix;      <-- abandon d'un index

Index dropped.

SQL> select index_name from ind;  <-- vérification

INDEX_NAME
-----
AUTEUR
DATE_ACHAT
TITRE

```

4.5.5 Conseils

Nous regroupons ici des conseils glanés dans les ouvrages déjà cités :

- 1 Créer un index sur les colonnes intervenant souvent dans la sélection des lignes, c'est à dire dans les clauses WHERE, GROUP BY, ORDER BY
- 2 Créer un index sur les colonnes qui servent à faire des jointures entre tables.
- 3 Créer un index pour les grande tables (plusieurs centaines de lignes)
- 4 Ne pas créer d'index sur des colonnes ayant peu de valeurs différentes (la colonne genre de la table BIBLIO par exemple).
- 5 Ne pas créer d'index pour une requête qui retournera plus du quart des lignes d'une table.

Un index n'est pas toujours utilisé. C'est "l'optimiseur" d'Oracle qui décide de l'utiliser ou pas selon des règles qui lui sont propres.

4.6 Le dictionnaire des données

Oracle gère un certain nombre de tables contenant diverses informations sur tous les objets (tables, vues, index, privilèges d'accès, ..) de tous les utilisateurs. Elles donnent des renseignements souvent intéressants pour l'utilisateur. Les informations concernant les objets d'un utilisateur donné sont contenues dans les vues ou les tables suivantes :

TABS	vue - liste des tables créées par l'utilisateur
USER_VIEWS	table - liste des vues créées par l'utilisateur
OBJ	vue - liste des objets appartenant à l'utilisateur
MYPRIVS	vue - privilèges de l'utilisateur
IND	vue - index créés par l'utilisateur

USER_TAB_GRANTS_MADE	table - privilèges accordés par l'utilisateur sur des objets lui appartenant
USER_TAB_GRANTS_REC'D	table - privilèges reçus par l'utilisateur sur des objets ne lui appartenant pas.
COLS	vue - liste des colonnes appartenant à l'utilisateur

De très nombreuses autres tables et vues sont accessibles à l'utilisateur et lui donnent des renseignements souvent intéressants. Leur liste se trouve dans la table DICT :

```
SQL> describe dict
```

Name	Null?	Type
TABLE_NAME		CHAR(30)
COMMENTS		CHAR(255)

```
SQL> select * from dict;
```

TABLE_NAME	COMMENTS
ACCESSIBLE_COLUMNS	Columns of all tables, views and clusters
ACCESSIBLE_TABLES	Tables and Views accessible to the user
ALL_CATALOG	All tables, views, synonyms, sequences accessible to the user
ALL_COL_COMMENTS	Comments on columns of accessible tables and views
ALL_COL_GRANTS	Synonym for COLUMN_PRIVILEGES
ALL_COL_GRANTS_MADE	Grants on columns for which the user is owner or grantor
ALL_COL_GRANTS_REC'D	Grants on columns for which the user or PUBLIC is the grantee
ALL_CONSTRAINTS	Constraint definitions on accessible tables
ALL_CONS_COLUMNS	Information about accessible columns in constraint definitions
ALL_DB_LINKS	Database links accessible to the user
ALL_DEF_AUDIT_OPTS	Auditing options for newly created objects
ALL_INDEXES	Descriptions of indexes on tables accessible to the user
ALL_IND_COLUMNS	COLUMNS comprising INDEXES on accessible TABLES
ALL_OBJECTS	Objects accessible to the user
ALL_SEQUENCES	Description of SEQUENCES accessible to the user
ALL_SYNONYMS	All synonyms accessible to the user
ALL_TABLES	Description of tables accessible to the user
ALL_TAB_COLUMNS	Synonym for ACCESSIBLE_COLUMNS
ALL_TAB_COMMENTS	Comments on tables and views accessible to the user
ALL_TAB_GRANTS	Synonym for TABLE_PRIVILEGES
ALL_TAB_GRANTS_MADE	User's grants and grants on user's objects
ALL_TAB_GRANTS_REC'D	Grants on objects for which the user or PUBLIC is the grantee
ALL_USERS	Information about all users of the database
ALL_VIEWS	Text of views accessible to the user
AUDIT_ACTIONS	Description table for audit trail action type codes. Maps action type numbers to action type names
CAT	Synonym for USER_CATALOG
CLU	Synonym for USER_CLUSTERS
COLS	Synonym for USER_TAB_COLUMNS
COLUMN_PRIVILEGES	Grants on columns for which the user is the grantor, grantee, or owner, or PUBLIC is the grantee
CONSTRAINT_COLUMNS	Information about accessible columns in constraint definitions
CONSTRAINT_DEFS	Constraint Definitions on accessible tables
DBA_AUDIT_CONNECT	Synonym for USER_AUDIT_CONNECT
DBA_AUDIT_RESOURCE	Synonym for USER_AUDIT_RESOURCE
DBA_AUDIT_TRAIL	Synonym for USER_AUDIT_TRAIL
DICT	Synonym for DICTIONARY
DICTIONARY	Description of data dictionary tables and views
DICT_COLUMNS	Description of columns in data dictionary tables and views
DUAL	
IND	Synonym for USER_INDEXES
MYPRIVS	Synonym for USER_USERS
OBJ	Synonym for USER_OBJECTS
SEQ	Synonym for USER_SEQUENCES
SYN	Synonym for USER_SYNONYMS
TABLE_PRIVILEGES	Grants on objects for which the user is the grantor, grantee, or owner, or PUBLIC is the grantee
TABS	Synonym for USER_TABLES
USER_AUDIT_CONNECT	Audit trail entries for user logons/logoffs
USER_AUDIT_TRAIL	Audit trail entries relevant to the user
USER_CATALOG	Tables, Views, Synonyms, Sequences accessible to the user
USER_CLUSTERS	Descriptions of user's own clusters
USER_CLU_COLUMNS	Mapping of table columns to cluster columns
USER_COL_COMMENTS	Comments on columns of user's tables and views
USER_COL_GRANTS	Grants on columns for which the user is the owner, grantor or grantee
USER_COL_GRANTS_MADE	All grants on columns of objects owned by the user

USER_COL_GRANTS_REC'D	Grants on columns for which the user is the grantee
USER_CONSTRAINTS	Constraint definitions on accessible tables
USER_CONS_COLUMNS	Information about accessible columns in constraint definitions
USER_CROSS_REFS	Cross references for user's views, synonyms, and constraints
USER_DB_LINKS	Database links owned by the user
USER_EXTENTS	Extents comprising segments owned by the user
USER_FREE_SPACE	Free extents in tablespaces accessible to the user
USER_INDEXES	Description of the user's own indexes
USER_IND_COLUMNS	COLUMNS comprising user's INDEXes or on user's TABLES
USER_OBJECTS	Objects owned by the user
USER_SEGMENTS	Storage allocated for all database segments
USER_SEQUENCES	Description of the user's own SEQUENCES
USER_SYNONYMS	The user's private synonyms
USER_TABLES	Description of the user's own tables
USER_TABLESPACES	Description of accessible tablespaces
USER_TAB_AUDIT_OPTS	Auditing options for user's own tables and views
USER_TAB_COLUMNS	Columns of user's tables, views and clusters
USER_TAB_COMMENTS	Comments on the tables and views owned by the user
USER_TAB_GRANTS	Grants on objects for which the user is the owner, grantor or grantee
USER_TAB_GRANTS_MADE	All grants on objects owned by the user
USER_TAB_GRANTS_REC'D	Grants on objects for which the user is the grantee
USER_TS_QUOTAS	Tablespace quotas for the user
USER_USERS	Information about the current user
USER_VIEWS	Text of views owned by the user
V\$ACCESS	Synonym for V_\$ACCESS
V\$BGPROCESS	Synonym for V_\$BGPROCESS
V\$DBFILE	Synonym for V_\$DBFILE
V\$FILESTAT	Synonym for V_\$FILESTAT
V\$LATCH	Synonym for V_\$LATCH
V\$LATCHHOLDER	Synonym for V_\$LATCHHOLDER
V\$LATCHNAME	Synonym for V_\$LATCHNAME
V\$LOCK	Synonym for V_\$LOCK
V\$LOGFILE	Synonym for V_\$LOGFILE
V\$PARAMETER	Synonym for V_\$PARAMETER
V\$PROCESS	Synonym for V_\$PROCESS
V\$RESOURCE	Synonym for V_\$RESOURCE
V\$ROLLNAME	Synonym for V_\$ROLLNAME
V\$ROLLSTAT	Synonym for V_\$ROLLSTAT
V\$ROWCACHE	Synonym for V_\$ROWCACHE
V\$SESSION	Synonym for V_\$SESSION
V\$SESSTAT	Synonym for V_\$SESSTAT
V\$SGA	Synonym for V_\$SGA
V\$STATNAME	Synonym for V_\$STATNAME
V\$SYSSTAT	Synonym for V_\$SYSSTAT
V\$TRANSACTION	Synonym for V_\$TRANSACTION
V\$_LOCK	Synonym for V_\$_LOCK

Pour utiliser l'une de ces tables, on peut vérifier sa structure par *DESCRIBE table* puis consulter le contenu par *SELECT col1, col2, ... FROM table*.

5 Conclusion

Nous avons vu l'essentiel d'Oracle pour l'utilisateur moyen. Nous n'avons en revanche que peu abordé le fonctionnement interne de ce SGBDR. Celui-ci est décrit dans la documentation Oracle "Guide pour l'Administrateur de la Base". On y trouve des renseignements précieux permettant une compréhension fine du fonctionnement d'Oracle et donnant la vue d'ensemble dont a besoin l'Administrateur de la base. Sa lecture est très vivement conseillée pour une utilisation avancée d'Oracle. Enfin, nous n'avons pas abordé la programmation, que ce soit le langage PL/SQL propre à Oracle ou le langage Pro*C permettant d'intégrer des ordres SQL dans un programme C.

ANNEXES

On trouvera dans ces annexes comment :

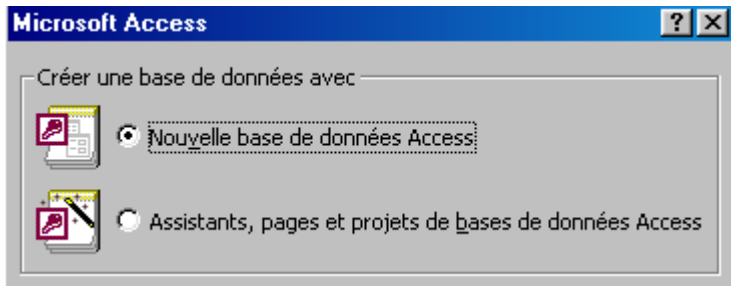
- utiliser Access et Microsoft Query pour faire du SQL
- définir et utiliser des sources de données ODBC
- installer Oracle sous Windows
- installer Oracle sous Linux
- installer MySQL sous Windows
- installer MySQL sous Linux
- faire des échanges de données entre Access, Oracle, MySQL
- utiliser Microsoft Query pour interroger des bases Oracle ou mySQL

6 SQL avec ACCESS

Access est un SGBD qu'on trouve couramment sur les machines windows personnelles. Si cet outil est trop limité pour un faire un SGBD en production supportant des dizaines d'utilisateurs simultanés, c'est un bon outil pour apprendre SQL car il supporte une grande partie du standard.

6.1 Créer une base

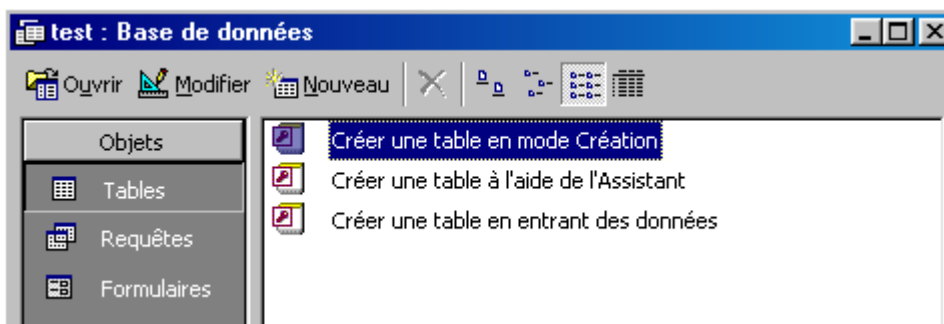
Lorsqu'on lance ACCESS, on a une boîte de dialogue demandant si on veut créer une nouvelle base. Faites [OK] et indiquez ensuite le nom du fichier .mdb qui contiendra les différents objets de la base, par exemple *test.mdb*.



Nom

test.mdb

Ceci fait, Access propose de créer une table :



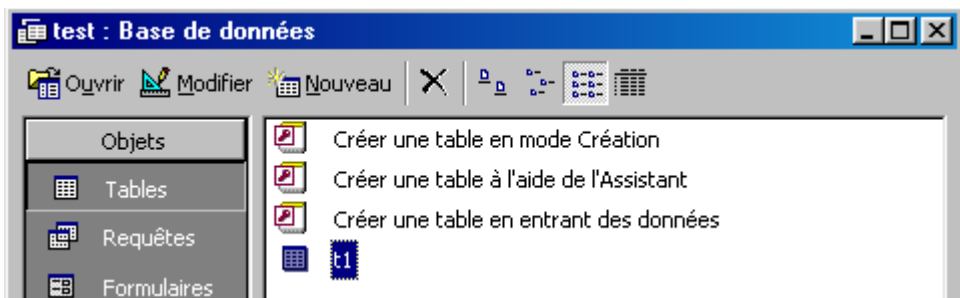
Choisissons "Créer une table en mode Création" et définissons une première table de la façon qui suit :

	Nom du champ	Type de données
	id	Texte
	cola	Texte
	colb	Numérique

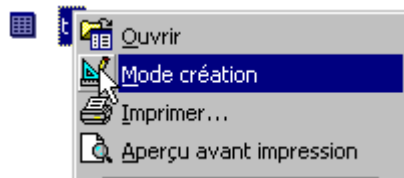
Fermer la fenêtre de définition de la table et donnez à celle-ci le nom T1 :



La table créée apparaît dans la structure de la base *test* :



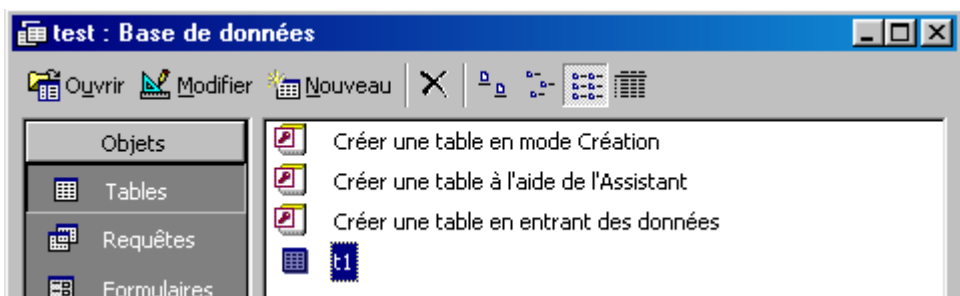
La structure de la table t1 peut être modifiée. Cliquez droit sur *t1* et choisissez *Mode création* :



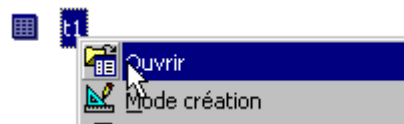
Faisons du champ *id* de *t1* une clé primaire. Pour cela, sélectionnez le champ *id* et l'option de menu *Edition/Clé primaire*. Une clé apparaît alors à gauche du champ *id*.

t1 : Table		
	Nom du champ	Type de données
	id	Texte
	cola	Texte
	colb	Numérique

Fermez la fenêtre et sauvegardez. On revient à la structure de la base :



Cliquez droit sur *t1* et choisissez l'option *Ouvrir* :



Créez maintenant quelques lignes dans la table *t1* :

t1 : Table			
	id	cola	colb
	id1	a1	1
	id2	a2	2
	id3	a3	3
	id4	a4	4
	id5	a5	5

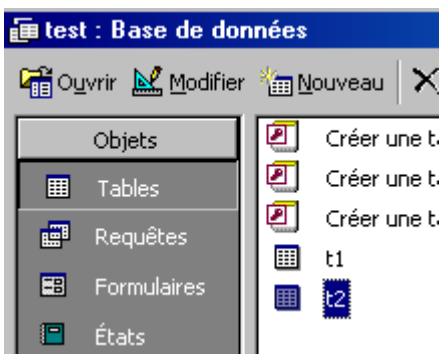
Fermez la fenêtre et sauvegardez. De la même façon que précédemment, créez une autre table *t2* qui aurait la structure suivante :

t2 : Table	
Nom du champ	Type de données
idT1	Texte
idT2	Texte
colC	Texte
colD	Texte

Le champ *idT2* est la clé primaire de la table. Le champ *idT1* est une clé étrangère référençant le champ *id* de la table *t1*. Le contenu de *t2* pourrait être le suivant :

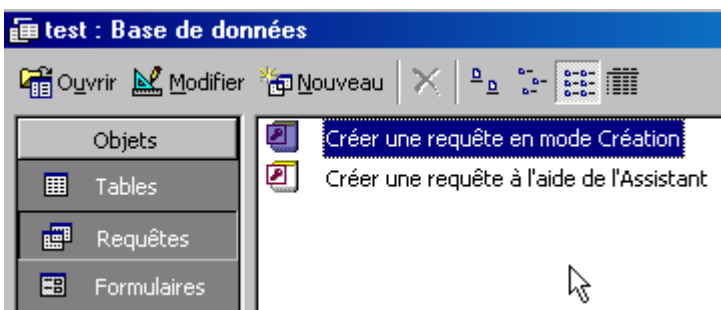
idT1	idT2	colC	colD
id3	t2-1	c1	d1
id1	t2-2	c2	d2
id3	t2-3	c3	d3
id2	t2-4	c4	d4
id1	t2-5	c5	d5

Nous avons maintenant deux tables dans notre base test.

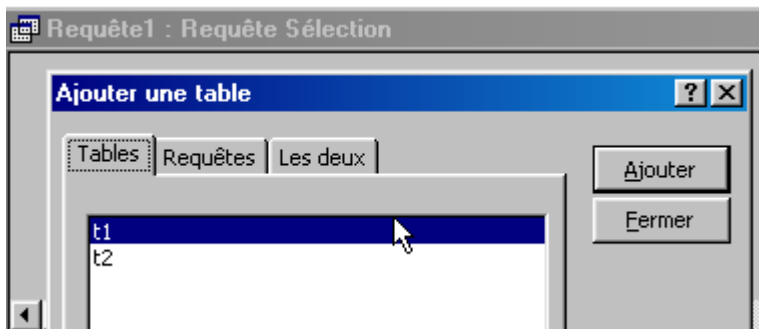


6.2 Générer des requêtes SQL

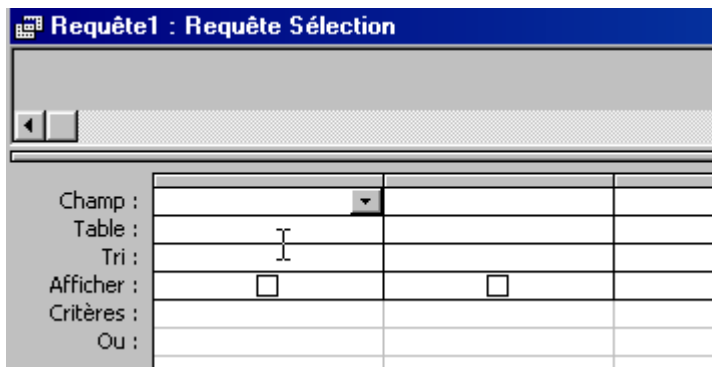
Dans la fenêtre ci-dessus de structure de la base, on peut choisir dans la fenêtre de gauche l'option Requetes :



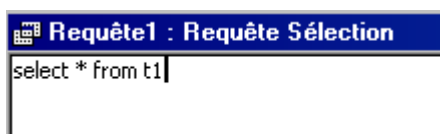
Choisissons "Créer une requête en mode création". On obtient alors un ensemble de fenêtres pas très lisible.



Fermez la fenêtre ci-dessus avec le bouton [Fermer]. Au passage, on note que nos deux tables t1 et t2 sont bien présentes. Nous sommes maintenant face à la fenêtre suivante permettant la construction graphique de requêtes SQL. Nous l'utiliserons très peu préférant taper directement le texte de la commande SQL.



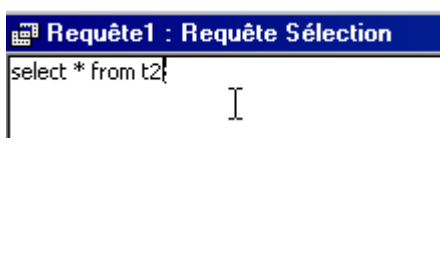
Prenez maintenant l'option *Affichage/Mode SQL* pour avoir la fenêtre suivante :



C'est une fenêtre d'édition dans laquelle nous pouvons taper une commande SQL. Nous exécutons celle ci-dessus par l'option *Requête/Exécuter*. Nous obtenons alors le résultat suivant :

	id	cola	colb
▶	d1	a1	1
	id2	a2	2
	id3	a3	3
	id4	a4	4
	id5	a5	5

Nous pouvons faire de même pour la table t2 Refaire *Affichage/Mode SQL* pour obtenir de nouveau la fenêtre d'édition.



	idT1	idT2	colC	colD
▶	id3	t2-1	c1	d1
	id1	t2-2	c2	d2
	id3	t2-3	c3	d3
	id2	t2-4	c4	d4
	id1	t2-5	c5	d5

Essayons une requête sur les deux tables :

Requête1 : Requête Sélection	
<pre>select t1.id,t1.colA,t1.colB,t2.colC,t2.colD from t1,t2 where t1.id=t2.idT1</pre>	

Requête1 : Requête Sélection		colA	colB	colC	colD
	id3	a3	3	c1	d1
	id1	a1	1	c2	d2
	id3	a3	3	c3	d3
	id2	a2	2	c4	d4
	id1	a1	1	c5	d5

On peut aussi utiliser des requêtes SQL de définition de données :

Requête1 : Requête Définition des données	
<pre>create table t3 (idT3 char(4), colE integer, colF varchar(20))</pre>	

On peut alors vérifier la présence de la table t1 dans la structure de la base :

test : Base de données	
Ouvrir	Modifier
Objets	
Tables	Créer u
Requêtes	Créer u
Formulaires	Créer u
États	t1
	t2
	t3

et vérifier sa structure :

t3 : Table	
Nom du champ	Type de données
idT3	Texte
colE	Numérique
colF	Texte

On a donc là un outil convivial pour apprendre SQL. On peut également utiliser Microsoft Query, un outil livré avec MS Office.

7 Sources de données ODBC

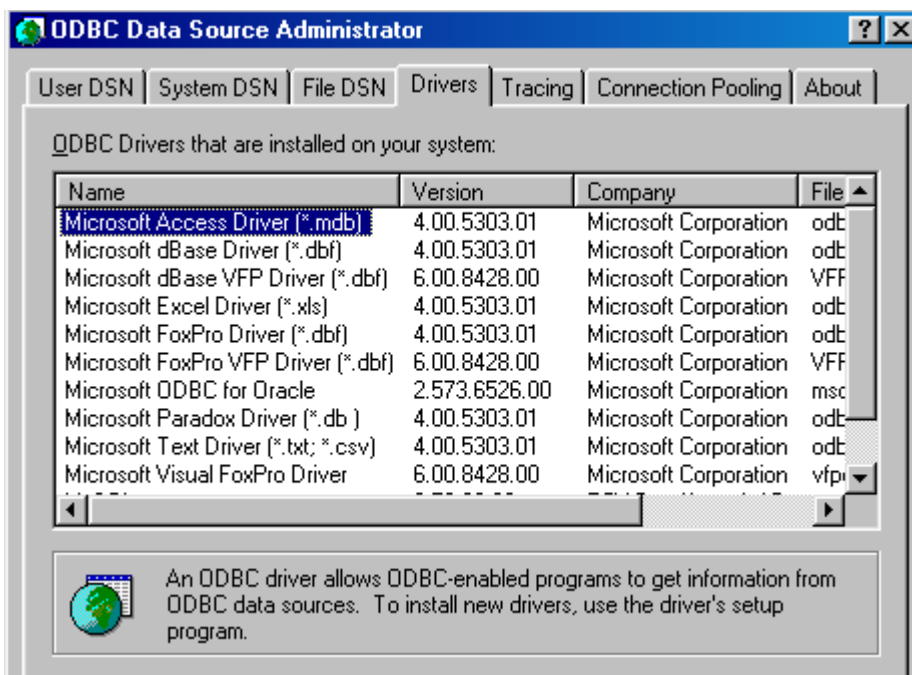
7.1 Pilotes ODBC

Sous Windows, un pilote ODBC (Open DataBase Connectivity) sert à masquer aux applications les particularités des bases de données. Ainsi une application Windows pourra utiliser l'interface standard des pilotes ODBC sans se préoccuper de la base de données qui est derrière. Celle-ci peut changer, l'application elle n'aura pas à être réécrite. Cette souplesse se paie par une moindre performance vis à vis des pilotes écrits spécialement pour la base de données.

Vous pouvez voir la liste des pilotes ODBC déjà installés sur votre machine par *Démarrer/Paramètres/Panneau de configuration*.



L'une des icônes du Panneau de configuration est *ODBC Data Sources*. C'est l'outil de configuration des bases de données ODBC c'est à dire des bases ayant un pilote ODBC. Lorsqu'on ouvre cette application, on obtient un classeur à plusieurs pages dont celui des pilotes ODBC :

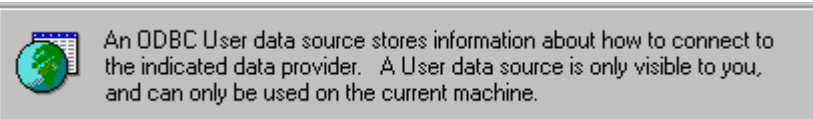


Ci-dessus, vous avez la liste des pilotes ODBC installés sur une machine Windows 9x.

7.2 Sources de données ODBC

Une source de données ODBC est une source de données accessible via un pilote ODBC. Dans la terminologie Windows les noms de ces sources ODBC portent un nom appelé DSN (Data Source Name). Il existe trois sortes de sources ODBC comme le montre l'écran ci-dessus :

User DSN



Une source de données ODBC "utilisateur" est donc une source réservée à un utilisateur particulier. Cette notion n'a de sens que sur une machine ayant

plusieurs utilisateurs en général NT.

System DSN



An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

Une source de données ODBC "utilisateur" est donc une source connue de tous les utilisateurs d'une machine. Là encore, cette notion a surtout un sens pour NT.

File DSN



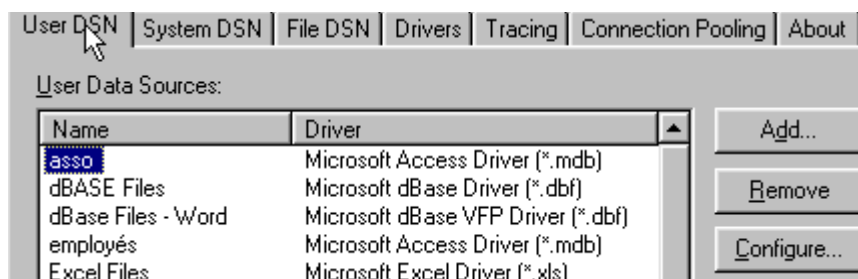
An ODBC File data source allows you to connect to a data provider. File DSNs can be shared by users who have the same drivers installed.

?

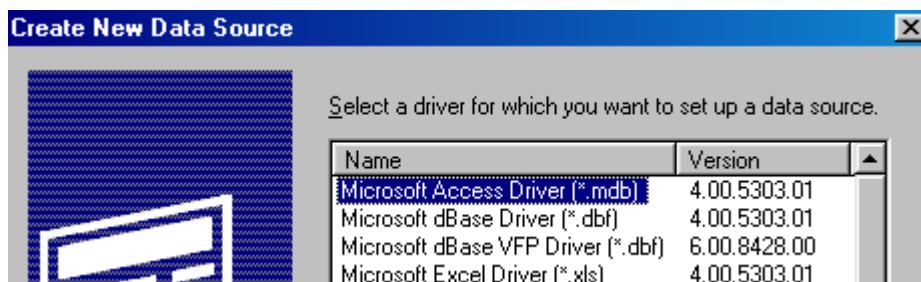
Sur une machine Windows 9x, on choisira User DSN ou System DSN. Il n'y aura une différence que s'il y a plusieurs utilisateurs déclarés sur la machine. L'intérêt d'une source de données ODBC est que les applications windows y ont accès via le nom qu'on lui donne, indépendamment de son emplacement physique. Cela permet de déplacer physiquement les sources de données sans avoir à réécrire les applications.

7.3 Créer une sources de données ODBC

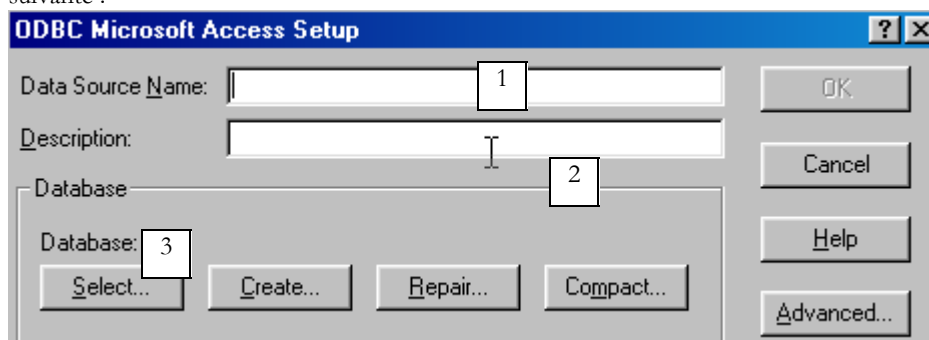
Montrons comment nous pouvons faire d'une base Access une source de données ODBC. Choisir la page User DSN :



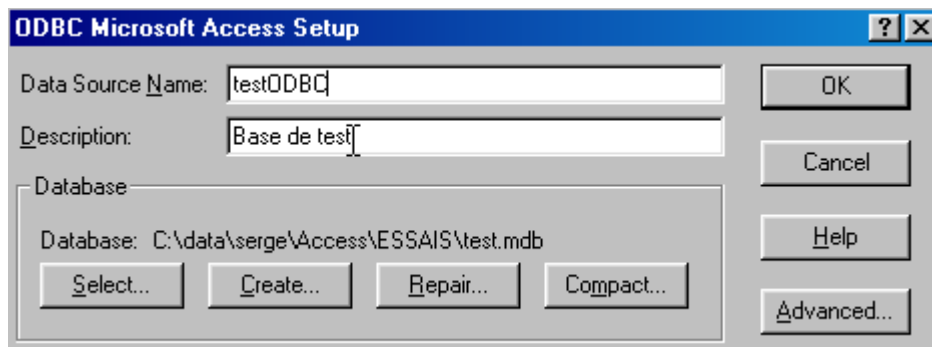
Faire [Add] pour créer une nouvelle source ODBC.



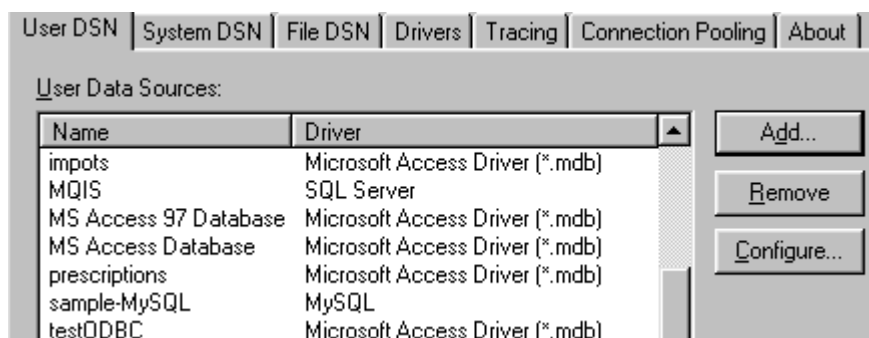
Il vous faut choisir le pilote ODBC de la source que vous allez construire. Choisissez *Microsoft Access driver* puis cliquez sur [Terminer]. On obtient ensuite une page d'informations à remplir, page spécifique au pilote choisi. Celle d'Access est la suivante :



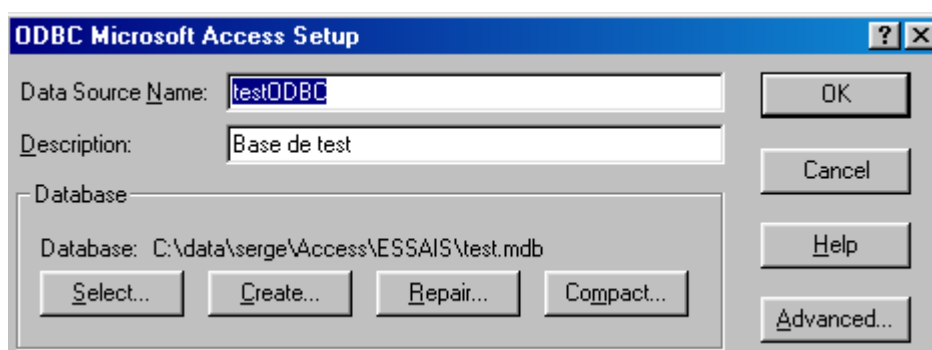
- 1 nom de la source - n'est pas forcément le nom de la base ACCESS
- 2 texte libre de description de la source
- 3 bouton permettant d'aller désigner la base Access qui



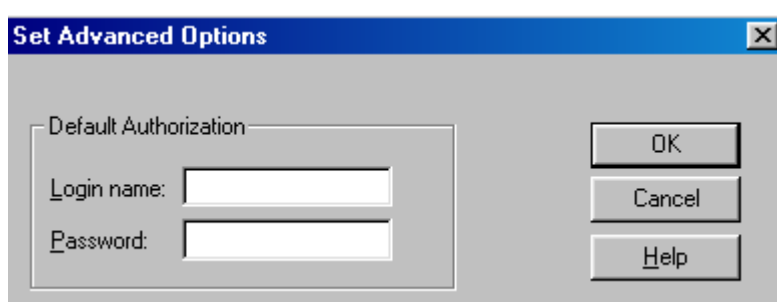
On fait [OK] pour valider la source qui doit maintenant apparaître avec les autres dans les sources ODBC "utilisateur".



Ces sources peuvent être supprimées avec le bouton [Remove] et reconfigurées avec le bouton [Configure]. Reconfigurons la source *testODBC* :



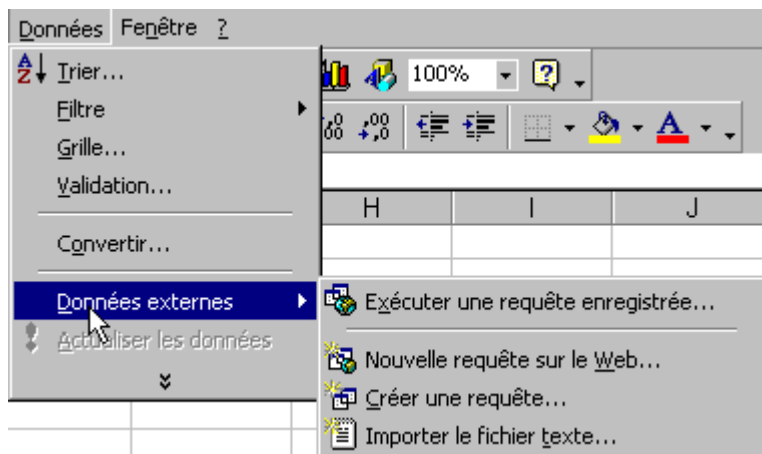
et utilisons le bouton [Advanced].



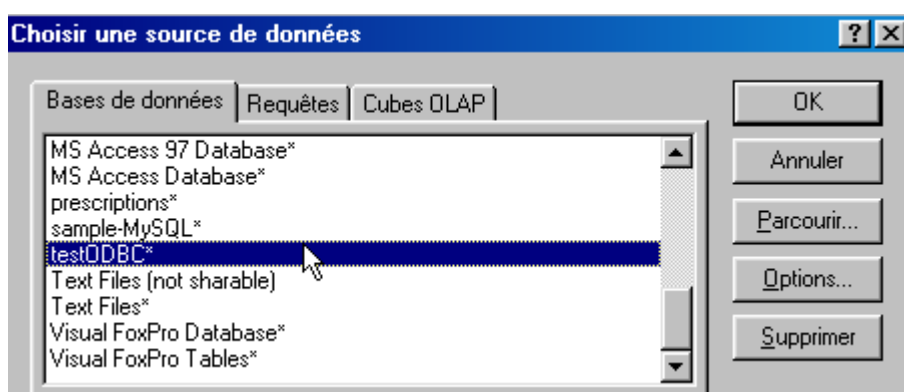
On voit ici qu'on pourrait définir un login/mot de passe pour notre source ODBC.

7.4 Utiliser une source de données ODBC

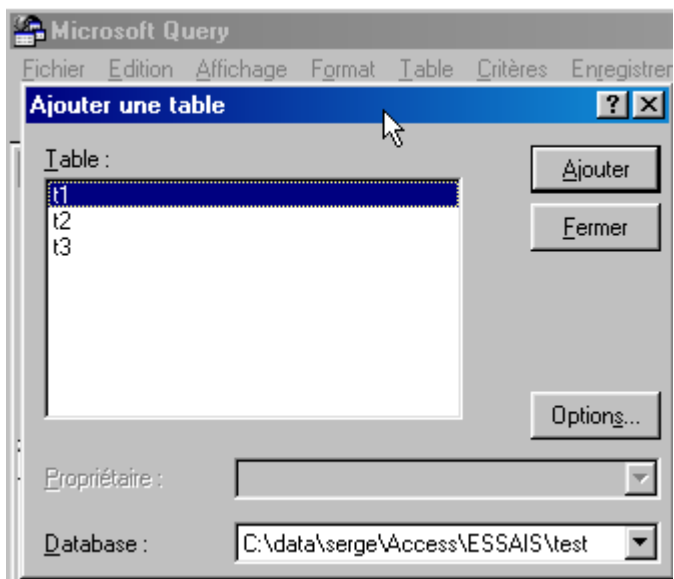
Pourquoi créer une source de données ODBC ? Beaucoup de langages de programmation sous Windows ont des fonctions d'accès aux sources de données ODBC : VB, Java, Php, Delphi,... Par ailleurs, des applications standard de Windows peuvent récupérer des données dans les sources ODBC. Prenons par exemple Excel. Lancez Excel et sur une classeur vide choisissez l'option *Données/Données externes/Créer une requête*.



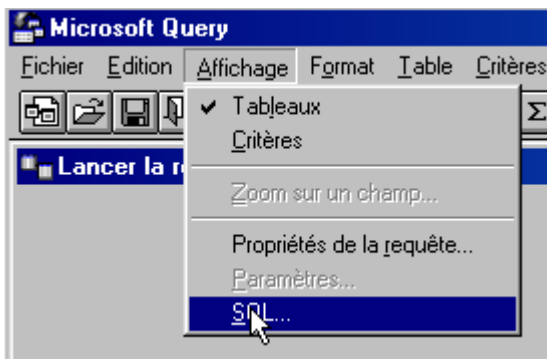
Un panneau "Choisir une source de données" apparaît. On y retrouve toutes les sources ODBC. On peut même en créer une nouvelle. Choisissons la source testODBC que nous venons de créer :



Un outil, Microsoft Query est alors lancé. Il permet de faire des requêtes sur des sources ODBC.



Nous retrouvons ci-dessus les trois tables de la base Access test.mdb. Quittons cette fenêtre avec le bouton [Fermer], puis choisissons l'option *Affichage/SQL* de Microsoft Query :



Nous avons maintenant une zone dans laquelle nous pouvons saisir une requête SQL sur la source ODBC :



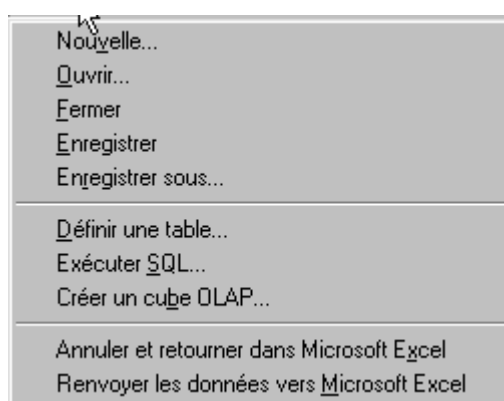
Lançons l'exécution de la requête ci-dessus par [OK]. La requête est exécutée et ses résultats affichés dans MS Query :

Lancer la requête à partir de testOD

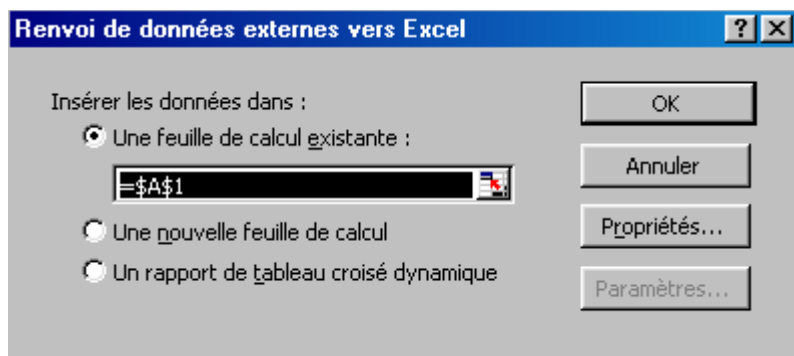
t1		
*	cola	colb
	id	

	id	cola	colb
	id1	a1	1
	id2	a2	2
	id3	a3	3
	id4	a4	4
	id5	a5	5

Nous pouvons maintenant "exporter" ces résultats vers Excel avec l'option *Fichier/Renvoyer les données vers Microsoft Excel* :



Excel redevient la fenêtre active et demande quoi faire des données qu'il reçoit :



Si nous acceptons la proposition ci-dessus, on obtient la feuille Excel suivante :

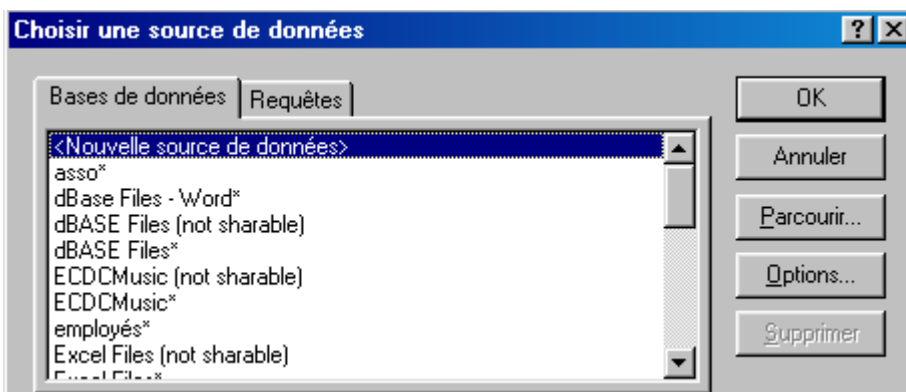
	A	B	C
1	id	cola	colb
2	id1	a1	1
3	id2	a2	2
4	id3	a3	3
5	id4	a4	4
6	id5	a5	5

7.5 Microsoft Query

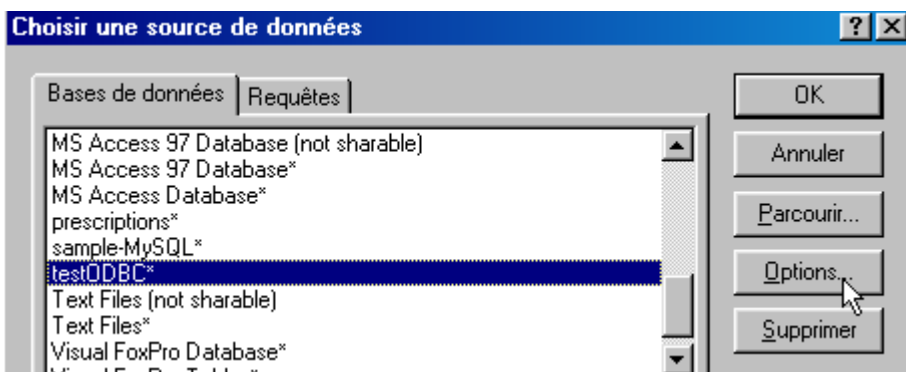
Quoique MS Query soit livré avec MS Office, il n'existe pas toujours de lien vers ce programme. On le trouve dans le dossier *Office* de MS Office sous le nom *MSQRY32.EXE*. Par exemple "*C:\Program Files\Office 2000\Office\MSQRY32.EXE*". MS Query permet d'interroger toute source de données ODBC avec des requêtes SQL. Celles-ci peuvent être construites graphiquement ou tapées directement au clavier. Comme la plupart des bases de données pour Windows fournissent des pilotes ODBC, elles peuvent donc toutes être interrogées avec MS Query. C'est souvent la solution la plus conviviale et elle a l'avantage d'être identique pour toutes les bases. Lorsque MS Query est lancé, on a l'affichage suivant :



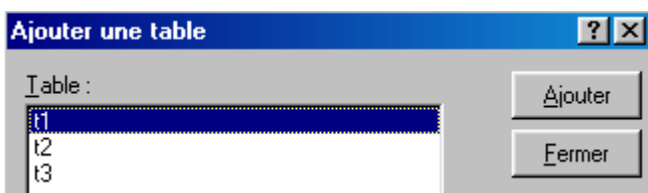
Il nous faut tout d'abord désigner la source de données ODBC qui va être interrogée par *Fichier/Nouvelle* :



On voit ci-dessus qu'on a aussi la possibilité de créer une nouvelle source de données ODBC si celle-ci n'existait pas encore. Nous avons déjà suivi ce processus pour faire d'une base Access une source de données ODBC. Le processus de création est ici identique. Choisissons de nouveau la source testODBC :



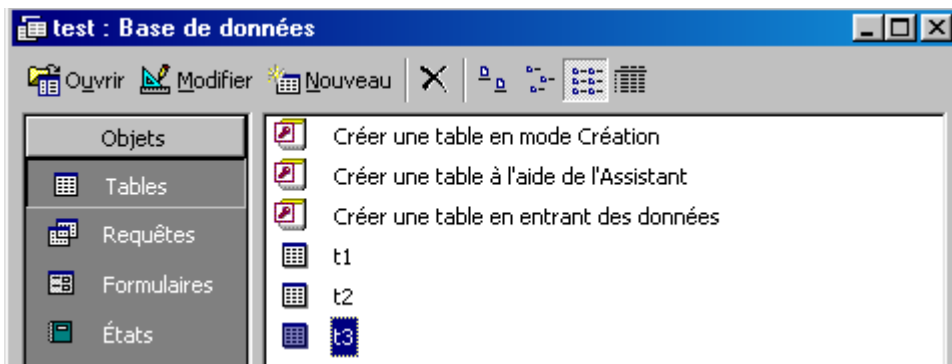
MS Query nous présente la structure de la source :



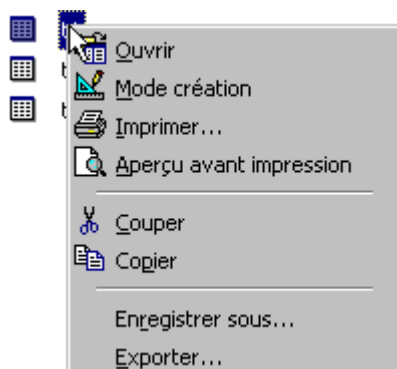
On est ramené à l'exemple précédent sur Access qu'on peut refaire ou modifier.

7.6 Échanger des données entre SGBD compatibles ODBC

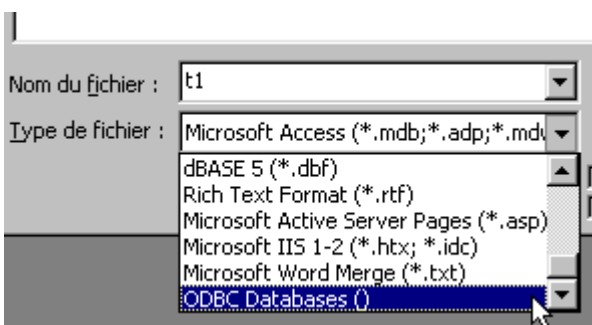
On peut échanger des données entre SGBD compatibles ODBC c.a.d. pour lesquels il existe un pilote ODBC. Il existe pour ce faire plusieurs méthodes. Nous en montrons une. Comme exemple échangeons des données entre Access et MySQL. Nous travaillerons de nouveau avec la base *test.mdb*. Lancez le SGBD MySQL si besoin est.



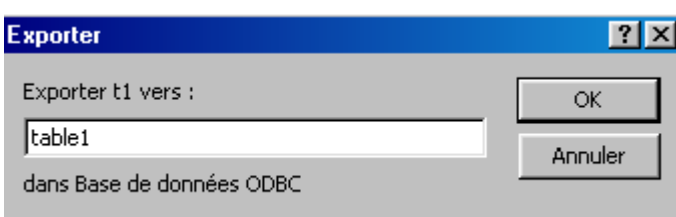
Cliquez droit sur la table *t1* et choisissez l'option *Exporter* :



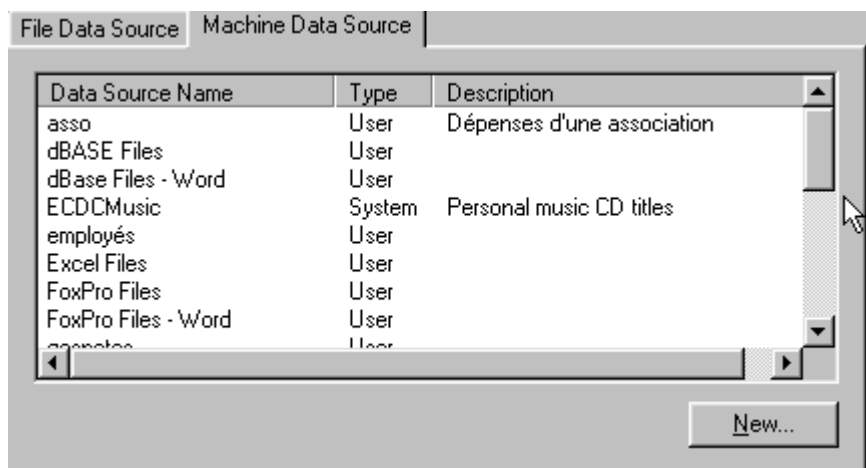
Une fenêtre demande vers quoi exporter la table *t1*. Dans type de fichier, choisissez le type "ODBC Bases", pour indiquer que vous allez l'exporter vers une source de données ODBC.



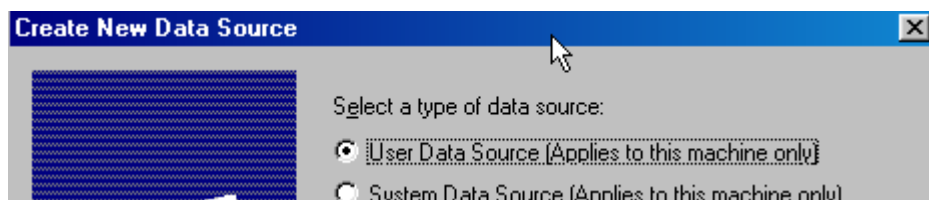
Ce choix fait, il vous est demandé vers quelle table vous voulez exporter la table *t1*. Tapez *table1* pour éviter une possible confusion avec la table *t1* de la base Access:



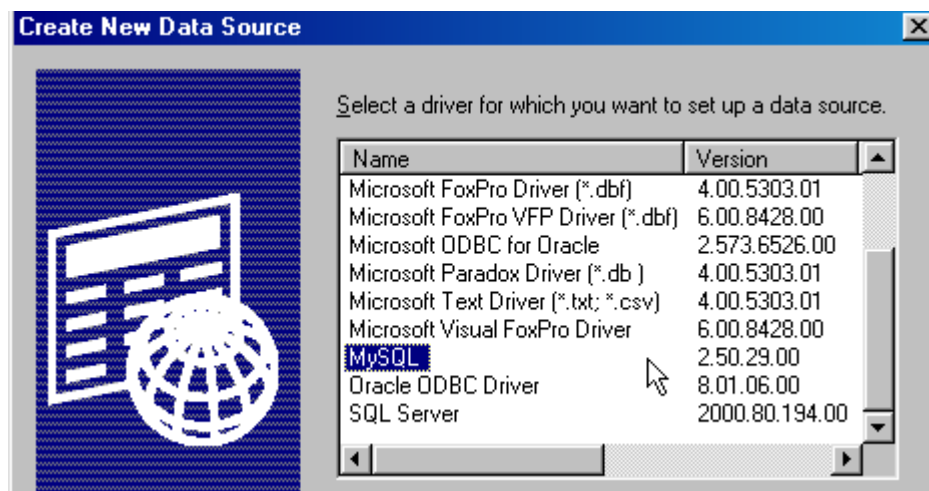
Faites [OK].



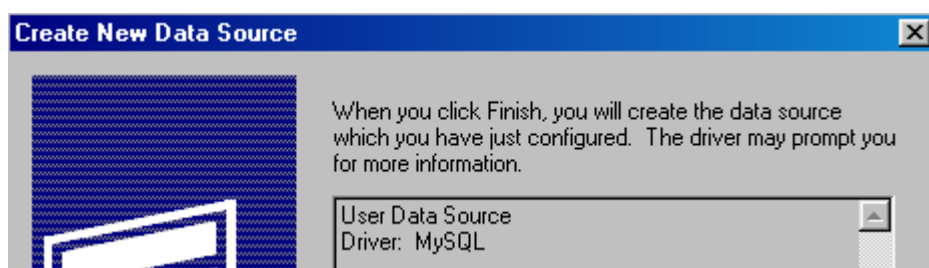
Ici, il vous est demandé de choisir la source de données ODBC où sera exportée la table *t1* d'Access sous le nom *table1*. Cette source n'existe pas encore. Faites [New] pour la créer. Démarre alors un assistant de création :



Indiquez que vous allez créer une source ODBC "utilisateur" et faites [Suivant]. Ensuite, on vous demande quel pilote ODBC votre source va utiliser. Sélectionnez le pilote MySQL :



Faites [Suivant]. Le dernier écran de l'assistant est un récapitulatif des choix faits :



Faites [Terminer]. Le pilote ODBC MySQL prend alors la main pour vous demander de configurer la source :

- lui donner un nom. Celui-ci est libre
- indiquer dans quelle base de données, la table Table1 doit être créée. Par défaut, le SGBD MySQL vient avec deux bases : *test* et *mysql* cette dernière étant réservée à l'administration des bases créées sous MySQL. Nous choisissons donc la base *test*. La ressemblance avec le nom de la base Access ou le nom de la source ODBC est ici purement fortuit.
- indiquer sous quel compte (login/mot de passe) la table doit-elle être créée.

TDX mysql Driver default configuration

This is in public domain and comes with NO 'WARRANTY' of any kind
Enter a database and options for connect

Windows DSN name: testMYSQL

MySQL host (name or IP):

MySQL database name: test

User:

Password:

Port (if not 3306):

SQL command on connect:

Faites [OK]. La source de données s'ajoute alors à la liste des sources ODBC déjà existantes :

Select Data Source

File Data Source Machine Data Source

Data Source Name	Type	Description
gesnotes	User	
impots	User	
MQIS	User	SQL Server
MS Access 97 Database	User	
MS Access Database	User	
prescriptions	User	
sample-MySQL	User	
testMYSQL	User	
testODBC	User	Base de test

Faites [OK]. Access tente alors une connexion au SGBD MySQL avec les informations qu'on lui a données et représente même une page de connexion à MySQL qui reprend les informations données lors de la configuration de la source de données ODBC :

TDX mysql Driver connect

This is in public domain and comes with NO WARRANTY of any kind
Enter a database and options for connect

Windows DSN name:

MySQL host (name or IP):

MySQL database name:

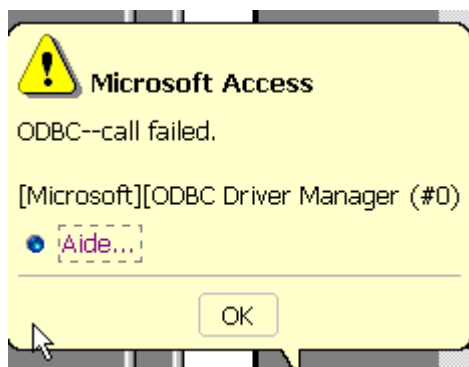
User:

Password:

Port (if not 3306):

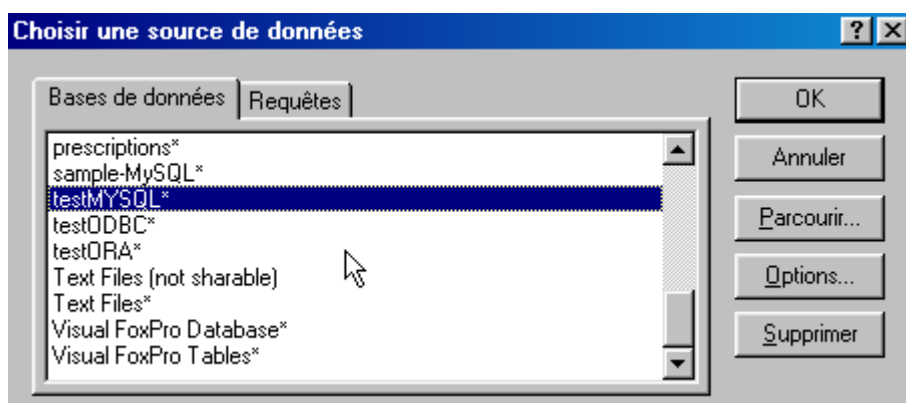
SQL command on connect:

On pourrait donc ici modifier certaines informations, notamment le login/mot de passe si celui-ci avait changé depuis la configuration initiale. Faites simplement [OK]. Et là.... on a une erreur ODBC incompréhensible.

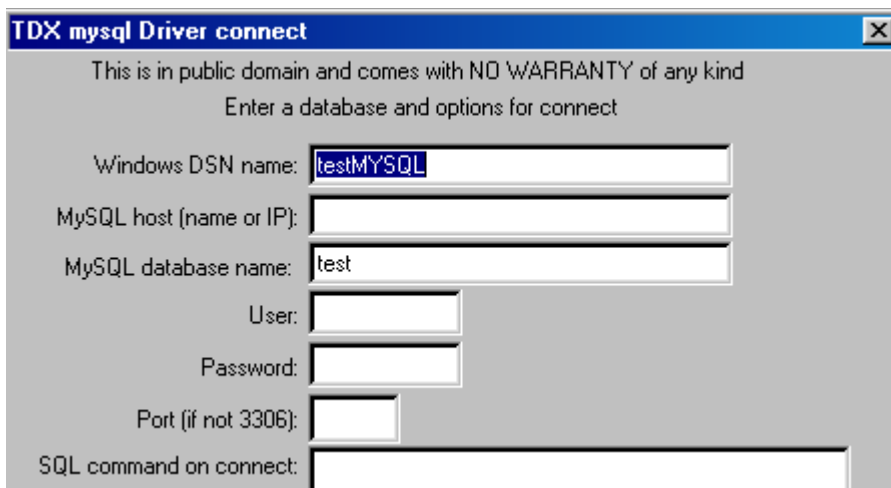


Après de multiples essais, il a fallu se rendre à l'évidence : sur la machine de test, la méthode précédente ne fonctionnait pas. Nous l'avons cependant laissée car il se peut que sur votre machine personnelle, cela marche.

Si ce n'est pas le cas, vous pouvez essayer la méthode suivante. Lancez MS Query et connectez-vous à la source de données ODBC testMYSQL qui vient d'être créée :



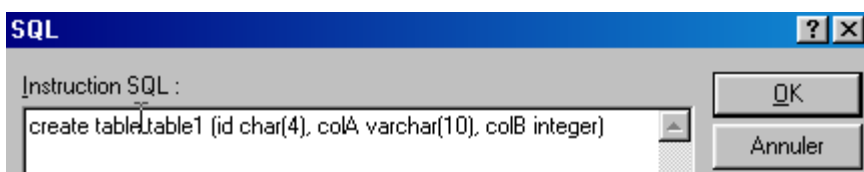
Faites [OK]. La fenêtre de connexion à la source MySQL s'affiche alors.



Faites [OK]. La liste des tables présentes dans la base test de la base *mySQL* s'affiche alors :



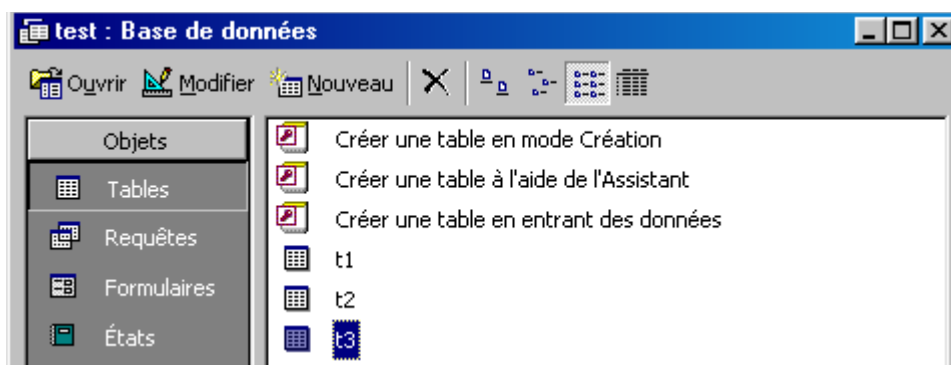
Ici, la base était vide. Ca peut être différent pour vous. Faites [Fermer] puis prenez l'option *Affichage/SQL* pour émettre la requête suivante :



Avec un client MySQL, vous pouvez vérifier la création de la table :

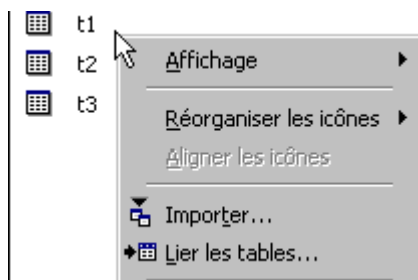
```
mysql> use test;
Database changed
mysql> show tables;
+-----+
| Tables in test |
+-----+
| table1         |
+-----+
1 row in set (0.05 sec)
```

Maintenant, revenez sous Access et sa base *test* :

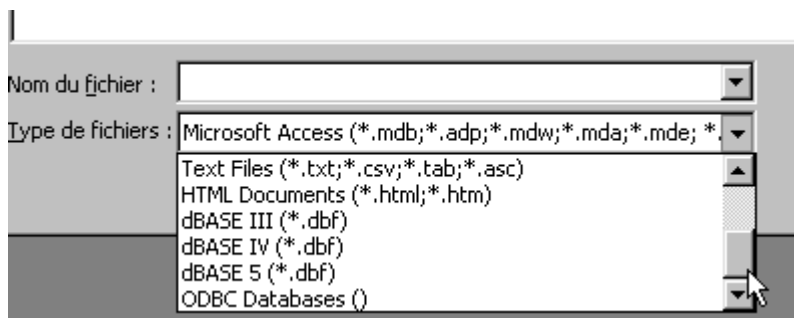


Cliquez droit sur un endroit vide de la fenêtre précédente :

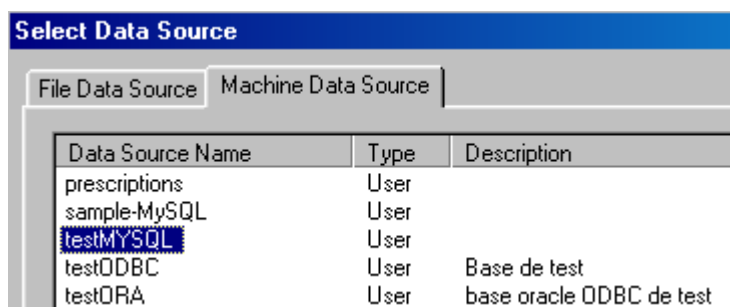
Sources de données ODBC



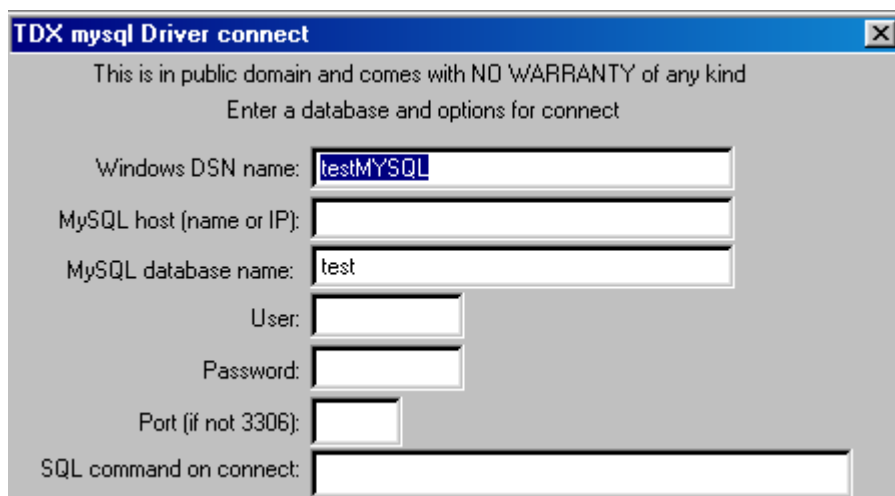
Prenez l'option [Lier les tables]. S'affiche alors la fenêtre déjà rencontrée dans l'essai d'exportation de la table t1 et qui vise à sélectionner la base dans laquelle on va choisir les tables à lier.



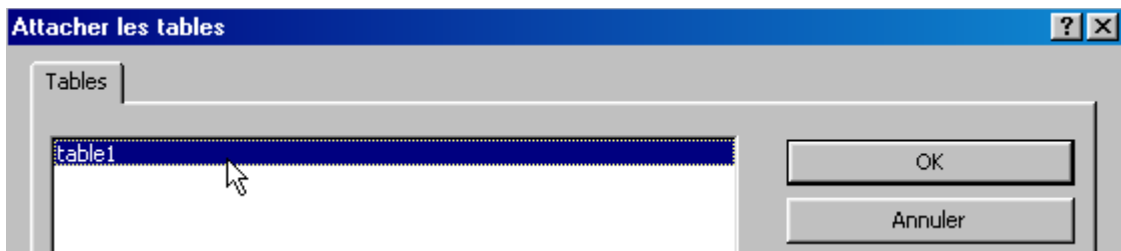
Dans *type de fichier*, choisir *ODBC Databases*. IL nous faut alors choisir la source de données ODBC :



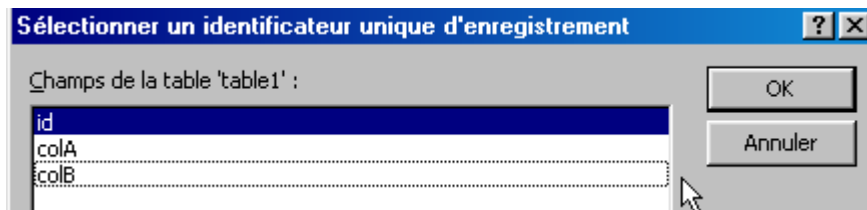
Choisir *testMYSQL*. Apparaît alors la fenêtre de connexion au SGBD MySQL :



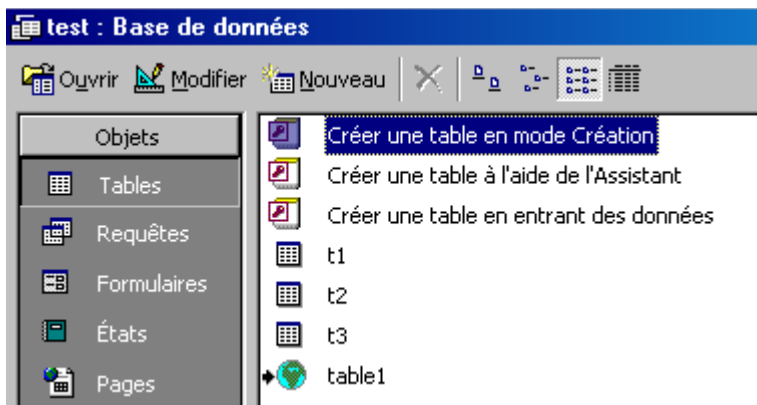
Faites [OK]. Si besoin est, modifiez auparavant le login/mot de passe. La connexion ODBC se fait et les tables de la base ODBC nous sont présentées :



Sélectionnez *table1* et faites [OK]. La liste des colonnes de *table1* est ensuite affichée. Il s'agit de choisir parmi elles celle qui doit jouer le rôle de clé primaire.



Choisissez la colonne *id* et faites [OK]. La table *table1* est alors intégrée à la base *test* en tant que table attachée :



La table *table1* ci-dessus n'est qu'une image ACCESS de la table *table1* MySQL, seule celle-ci étant réelle. Lorsqu'on modifie l'une, l'autre est modifiée également. Nous allons utiliser cette propriété pour remplir *table1* avec le contenu de *t1* par un copier/coller entre les deux tables ACCESS ci-dessus. Tout d'abord, vérifions la structure de la table attachée *table1* :

table1 : Table		
	Nom du champ	Type de données
	 id	Texte
	colA	Texte
	colB	Numérique

puis son contenu :

table1 : Table			
	id	colA	colB
			

Elle est vide. Maintenant copions toutes les lignes de *t1* pour les coller dans *table1* :

table1 : Table			
	id	colA	colB
	id1	a1	1
	id2	a2	2
	id3	a3	3
	id4	a4	4
	id5	a5	5

et vérifions que les données sont bien arrivées dans la table *mysql table1*. Cela peut se faire de plusieurs façons. Avec MS Query, après s'être connectée à la base ODBC testMySQL, on émet la requête suivante :

SQL
Instruction SQL :
select * from table1

On obtient le résultat suivant :

Lancer la requête à partir de testMySQL				
	id	colA	colB	
	id1	a1	1	
	id2	a2	2	
	id3	a3	3	
	id4	a4	4	
	id5	a5	5	

Avec le client *mysql* on émet la même requête :

```
mysql> use test
Database changed
mysql> show tables;
+-----+
| Tables in test |
+-----+
| table1         |
+-----+
1 row in set (0.02 sec)

mysql> select * from table1;
+-----+-----+-----+
| id   | colA | colB |
+-----+-----+-----+
| id1  | a1   | 1    |
| id2  | a2   | 2    |
| id3  | a3   | 3    |
| id4  | a4   | 4    |
| id5  | a5   | 5    |
+-----+-----+-----+
5 rows in set (0.05 sec)
```

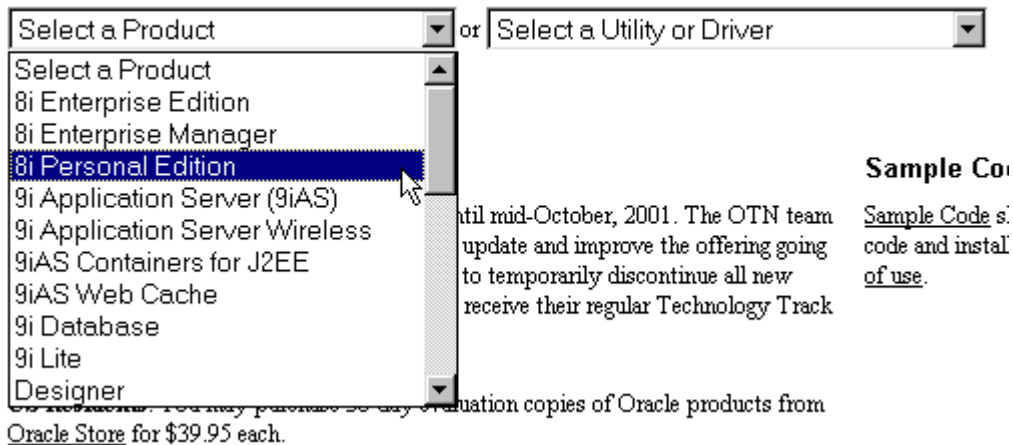
Nous avons donc pu "exporter" des données ACCESS vers une base MySQL.

8 Installation d'Oracle sous Windows

8.1 Où trouver le produit ?

Oracle est disponible à l'URL <http://technet.oracle.com>. Suivre le lien *downloads*.

Download Oracle Products, Drivers, and Utilities.



Dans la liste des produits, plusieurs versions d'Oracle sont proposées. La *Personal Edition* semble suffisante pour un apprentissage. Il faut tout d'abord s'enregistrer auprès d'Oracle pour pouvoir faire le téléchargement. Moyennant divers renseignements qui vous seront demandés, vous obtiendrez un login/mot de passe qui vous donneront accès aux produits à télécharger.

8.2 L'installation

Une fois le téléchargement terminé, on a un fichier .zip :



Il faut environ 1 Go de libre sur le disque. Décompressons ce fichier puis lançons l'installation d'Oracle :

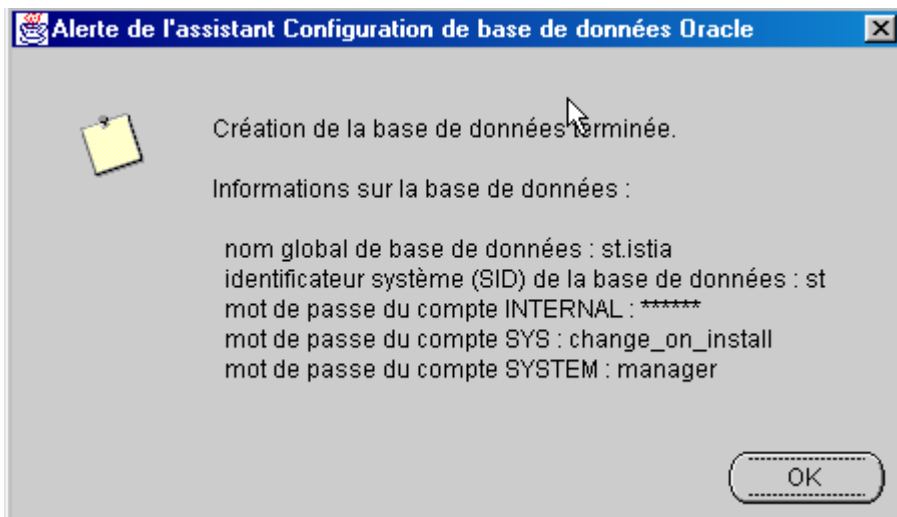


On reconnaît le panneau de bienvenue de l'application "Oracle Universal Installer", application qui sert également à l'installation d'Oracle sous Linux. On a là la même démarche. Il vous suffit donc de suivre la méthode décrite pour Linux. Nous indiquons seulement les différences importantes :

- Pour le type d'installation, choisir l'installation standard :

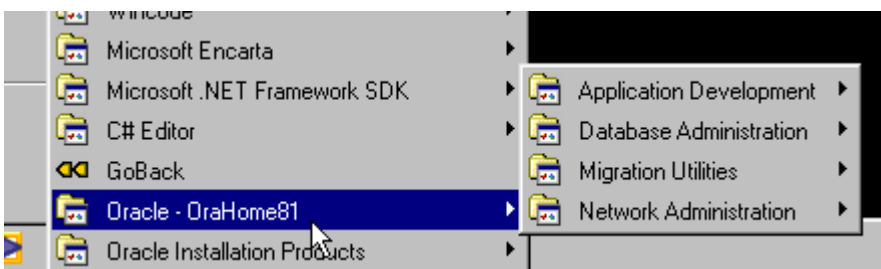


- Ensuite tout est assez automatique. Il vous faudra à un moment donner un nom à la base. Celle de cet exemple est st.istia. L'installation se termine avec l'écran suivant :



8.3 Lancer la base

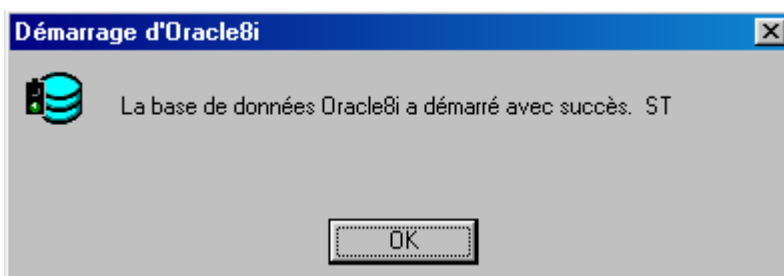
Après l'installation d'Oracle sous Windows, vous disposez dans le menu d'une entrée pour Oracle :



Oracle est lancé par l'option *Start Database* du menu *Database Administration* :

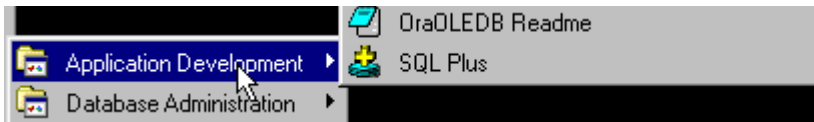


Un panneau indique si l'opération de démarrage de la base s'est bien passée.



8.4 Utiliser SQL Plus

Une fois la base lancée, l'outil SQLPLUS peut être utilisé. Il se trouve dans le menu *Application Development* d'Oracle.



SQL Plus tente une connexion à la base Oracle locale. Pour cela, il faut lui fournir un login/mot de passe. Dans un premier temps, vous pouvez utiliser le couple login/mot de passe suivant : *system/manager* qui est un compte d'administration de la base.

Une fenêtre SQL Plus s'ouvre. Dedans on va pouvoir émettre des commandes SQL et voir leur résultat.

```
SQL*Plus: Release 8.1.6.0.0 - Production on Di Aou 26 10:44:04 2001
```

```
(c) Copyright 1999 Oracle Corporation. All rights reserved.
```

```
Connecté à :  
Oracle8i Personal Edition Release 8.1.6.0.0 - Production  
With the Java option  
JServer Release 8.1.6.0.0 - Production
```

```
SQL>
```

8.5 Créer des comptes utilisateurs

Il n'est pas conseillé de travailler avec le compte *system/manager* en dehors des travaux d'administration. Sur une machine personnelle, cela n'a peut-être pas grand sens de différencier l'administrateur des autres utilisateurs de la base, le propriétaire de la machine étant probablement tout ceci à la fois. Néanmoins par souci des bonnes habitudes, nous allons créer un utilisateur banalisé *user1/user1*. Pour cela, on tape la commande SQL suivante :

```
SQL> grant connect, resource to user1 identified by user1;
```

Autorisation de privilèges (GRANT) acceptée.

Maintenant, nous pouvons quitter Sql plus avec la commande *exit* pour ensuite le relancer sous l'identité *user1/user1* :

On peut ensuite, sous SQL plus, vérifier qui on est :

```
SQL> select user from dual;
```

```
USER
```

```
-----  
USER1
```

On peut alors émettre toute commande SQL. L'outil SQL plus peut être lancé plusieurs fois. On peut donc se connecter sous différentes identités. Cela permet de tester aisément les commandes SQL qui font intervenir des droits entre utilisateurs.

Émettons quelques commandes de base :

```
SQL> select user from dual;
```

```
USER
```

```
-----  
USER1
```

```
SQL> select user from dual;
```

```
USER
```

```
-----  
USER1
```

```
SQL> create table maTable (nom varchar(30), age integer);
```

Table créée.

```
SQL> describe maTable;
```

Nom	NULL ?	Type
NOM		VARCHAR2 (30)
AGE		NUMBER (38)

```
SQL> insert into maTable (nom,age) values ('tintin',37);
```

1 ligne créée.

```
SQL> select * from maTable;
```

NOM	AGE
tintin	37

```
SQL> drop table maTable;
```

Table supprimée.

L'outil SQL plus se quitte avec la commande *exit*.

8.6 La fermeture de la base

L'option de fermeture de la base se trouve au même endroit que celle de son démarrage :



9 Installation d'Oracle sous Linux

Cette annexe montre comment installer Oracle sous Linux. Il est possible d'installer Oracle sous Windows par une procédure très analogue. Installer Oracle sur sa machine personnelle est très utile si on souhaite apprendre à administrer un système Oracle.

9.1 Installation d'Oracle 8, version 1.7.1 sur Linux RH 7.0

La société Oracle a porté son SGBD du même nom sur linux. Ce produit est disponible gratuitement à l'url <http://technet.oracle.com> en suivant le lien *downloads*. C'est un produit assez lourd à télécharger. La version décrite ici était disponible sous la forme d'un fichier .tar de 530 mo. Par ailleurs, un patch est disponible pour cette version. Il faut également le télécharger, sinon l'installation ne se fait pas correctement et le SGBD est inexploitable.

L'installation décrite est celle d'Oracle 8, version 1.7.1 sur Linux RH 7.0 disposant de l'interface KDE.

L'installation, la configuration, l'administration et l'utilisation d'Oracle 8i nécessitent de fortes compétences. Il faudrait de nombreux stages pour couvrir tous ces sujets. Ici, nous faisons une installation automatisée d'Oracle qui ne nous demande pas d'avoir des compétences d'administrateur de SGBD. Nous montrons ensuite comment lancer et arrêter la base de données et comment un utilisateur peut l'exploiter à l'aide du langage SQL.

On lira l'excellent livre de Gilles Briard aux éditions Eyrolles "Oracle 8i sous Linux" pour approfondir ses connaissances.

Passons à l'installation. Nous avons récupéré sur le site d'Oracle deux fichiers :

<i>linux81701.tar</i>	fait 530 mo environ. C'est lui qui contient tous les fichiers d'installation
<i>glbc-2.1.3-stubs.tar.gz</i>	contient les patches à passer pour que l'installation puisse se faire correctement.

Mettez ces fichiers quelque part sur votre machine linux. Si vous êtes en stage ou en tp, ils sont disponibles sur un CD. Dans la suite nous appellerons ce répertoire *<installOracle>*.

Lancez une session X en tant que *root*
\$ startx

Lancez une fenêtre terminal
menu/Utilitaires/Kconsole

Créez le groupe **dba**
\$ groupadd dba

Vérifier la présence du groupe dba dans le fichier des groupes */etc/group*
\$ grep -i dba /etc/group
dba:x:501:

Créez l'utilisateur **oracle** dans le groupe *dba*
\$ useradd -g dba oracle

Vérifiez la présence de l'utilisateur oracle dans le groupe */etc/passwd*
\$ grep -i oracle /etc/passwd
oracle:x:501:501::/home/oracle:/bin/bash

Fixez lui le mot de passe *oracle*
\$ passwd oracle
[root@localhost linux]# passwd oracle
Changing password for user oracle
New UNIX password:
BAD PASSWORD: it is based on a dictionary word
Retype new UNIX password:
passwd: all authentication tokens updated successfully

Le mot de passe *oracle* est un mauvais mot de passe parce que basé sur le login. Nous le garderons

néanmoins car il est simple à retenir

Créez un répertoire */oracle* et le donner à l'utilisateur *oracle*. Nous installerons le sgbd sous ce répertoire

```
$ mkdir /oracle
$ ls -ld /oracle
$ chown oracle:dba /oracle
$ ls -ld /oracle
drwxr-xr-x  6 oracle  dba      4096 jan  9 23:45 /oracle/
```

Allez dans le répertoire *<installOracle>* où vous avez placé les deux fichiers *linux81701.tar* et *glibc-2.1.3-stubs.tar.gz*. Désarchivez le fichier *linux81701.tar* :

```
$ tar xvf linux81701.tar
```

L'opération donne naissance à un répertoire *Disk1*.

L'installation d'oracle peut maintenant commencer mais elle doit être faite par l'utilisateur *oracle*. Pour l'instant, l'interface X a été lancée par l'utilisateur *root* et nous opérons sous cette identité. Nous quittons donc la session X courante pour en relancer une nouvelle sous l'identité *oracle*.

Quittez l'interface graphique par *Menu/Déconnexion*. Vous êtes maintenant sur une console texte, normalement celle de *root*. Ouvrez une seconde console texte (Alt-F2 pour ouvrir la console texte n° 2) et dans cette console identifiez vous avec le login/mot de passe *oracle/oracle*.

Lancez une session X

```
$ startx
```

Ouvrez une fenêtre terminal : *menu/Utilitaires/Kconsole*. Avec la commande *whoami* vous devez vérifier que vous êtes l'utilisateur *oracle*. Nous aurons parfois à émettre des commandes en tant que *root*. Pour cela, nous utiliserons une seconde fenêtre. Par la suite, nous pourrions être amenés à distinguer ces deux fenêtres terminal par leurs noms : fenêtre terminal *root*, fenêtre terminal *oracle*.

Ouvrez une seconde fenêtre terminal : *menu/Utilitaires/Kconsole*. Dans cette fenêtre vous êtes actuellement l'utilisateur *oracle*. Prenez l'identité de l'utilisateur *root* avec la commande *su*

```
$ su -
passwd : azerty
```

Vérifiez votre nouvelle identité :

```
$ whoami
root
```

Maintenant, il nous faut un gestionnaire de fichiers que vous trouverez dans *menu/répertoire home*. Celui-ci ne vous montre pas nécessairement toute l'arborescence de votre machine. Pour l'avoir, choisissez l'option *Affichage/arborescence*. Allez dans le répertoire *<installOracle>/Disk1* et cliquez sur le fichier *runInstaller*. L'installation d'Oracle commence.

Le premier écran est un écran de bienvenue.



Notez le fait que vous pourrez désinstaller les produits que vous allez installer. Faites [suivant].

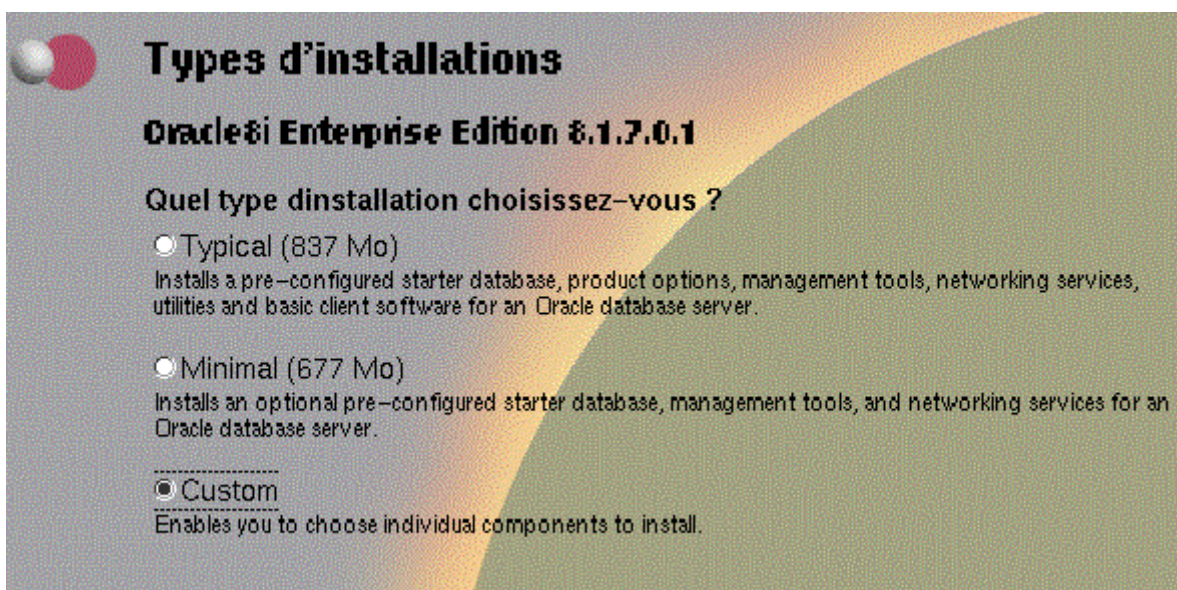


L'écran "Emplacement des fichiers" permet de préciser où seront trouvés les fichiers permettant l'installation d'Oracle (source) et où Oracle sera installé (destination). Nous placerons les fichiers sous l'arborescence `/oracle/OraHome1`. Le répertoire `/oracle` a été, rappelons-le, créé avant l'installation. Faites [suivant].

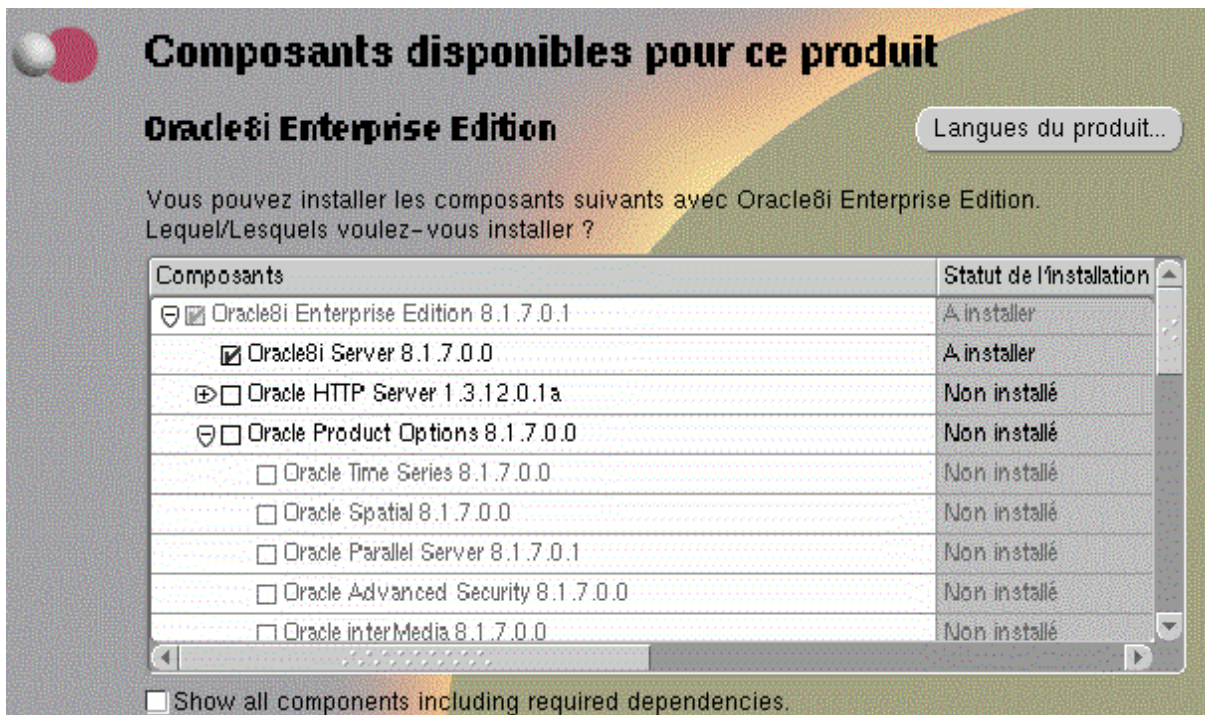


L'écran suivant "Produits disponibles" nous indique quels produits nous pouvons installer :

- *Oracle 8i Enterprise Edition* qui est le SGBD Oracle
- *Oracle 8i Management and Integration* qui regroupe des outils graphiques d'administration du SGBD
- *Oracle 8i Client* qui nous permettrait de travailler en tant que client sur une base de données Oracle située sur un ordinateur distant. Ici nous voulons installer le SGBD. Sélectionnez-le et faites [suivant].



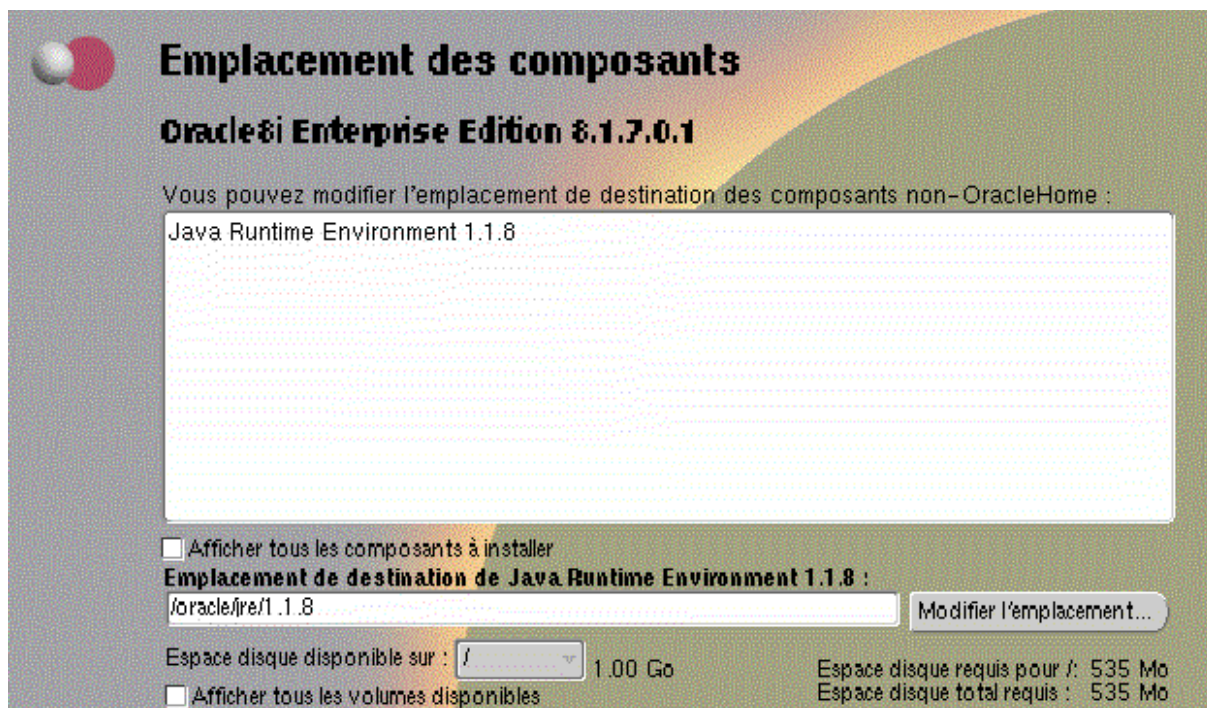
L'écran "Types d'installation" nous demande quel type d'installation nous désirons. Ici, nous prenons "Custom". Nous avons en effet un espace disque réduit et n'avons pas besoin de certains produits nécessitant des compétences que nous n'avons pas. Dans la pratique, on pourra toujours les installer ultérieurement. Choisissez donc "Custom" la et faites [suivant].



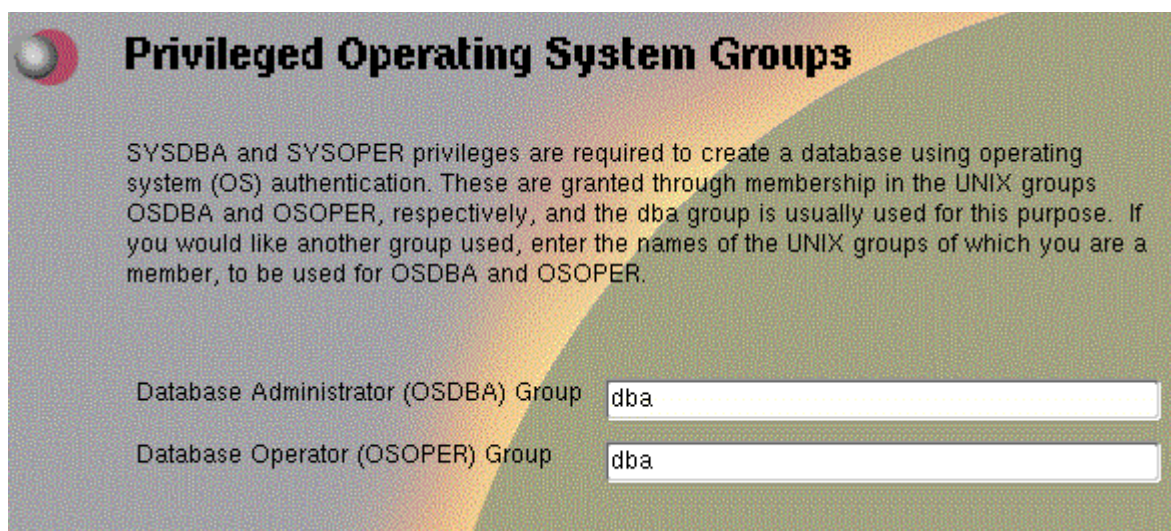
L'écran suivant "Composants disponibles" vous propose de choisir les produits à installer. Choisissez les produits suivants :

Oracle 8i server 8.1.7	SGBD Oracle
Net8 Products 8.1.7 (tout)	nécessaire pour accéder à Oracle depuis un ordinateur distant
Oracle utilities (tout)	nécessaire pour SQL plus, l'outil d'interrogation SQL utilisable dans une fenêtre terminal
Oracle Java products (tout)	permettra un accès à la base Oracle à partir d'outils Java
Oracle Configuration assistants	Assistants de configuration. Nous en utiliserons un : celui qui aide à créer une première base
Oracle installation products	
Oracle 8i Unix documentation	toujours utile

Vérifiez bien vos choix avant de faire [suivant].



Cet écran vous demande où vous voulez placer le "Java Runtime Environment". C'est un produit indépendant qui pourrait ne pas être placé dans l'arborescence d'Oracle. Acceptez l'emplacement qui vous est proposé et faites [suivant].



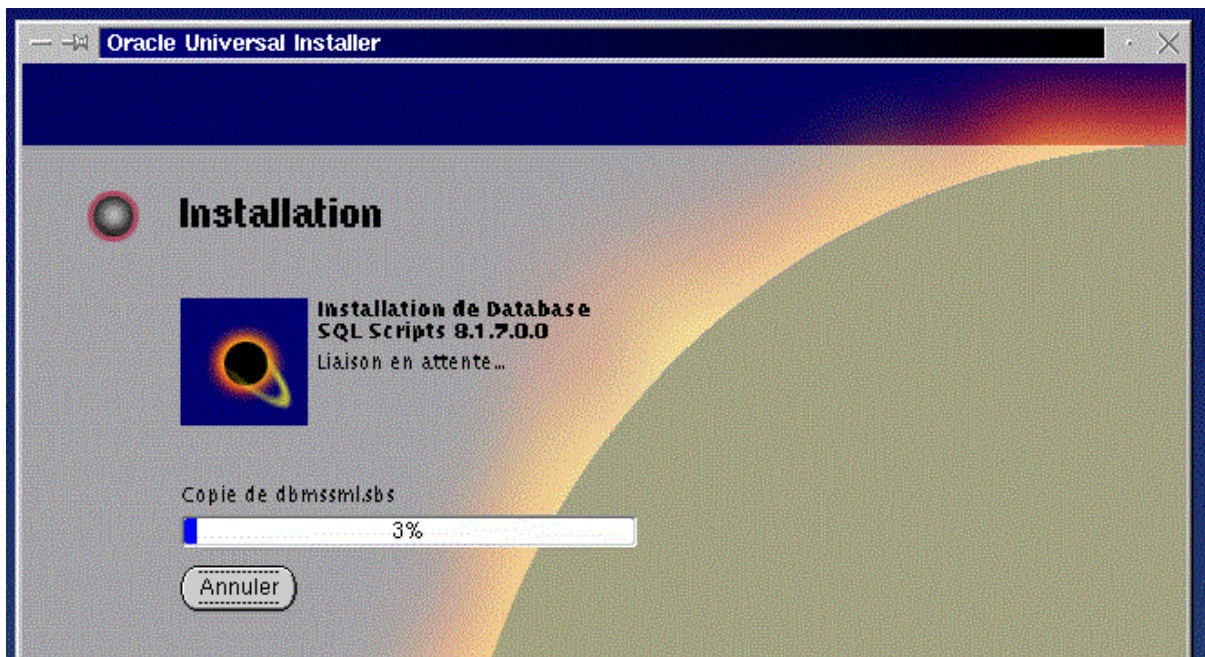
Cet écran vous demande de définir les noms des groupes d'utilisateurs autorisés à administrer la base. Acceptez le groupe dba qui vous est proposé par défaut. Il avait été créé avant l'installation et c'est le groupe de l'utilisateur oracle. Faites [suivant].



Cet écran vous demande si vous voulez créer une base de données. Normalement on répond oui. Mais ici l'installation bogue sur cette création. Il faut passer auparavant un patch, patch délivré en même temps que la version 8.1.7. d'Oracle. C'est ennuyeux, mais ça nous permettra en même temps de découvrir l'assistant de création de base. Répondez non et faites [suivant].



Cet écran fait un résumé de vos choix. Lisez-le attentivement car vous pouvez encore revenir en arrière. Notez bien l'espace disque requis et l'espace disque disponible ci-dessus. Faites [suivant].

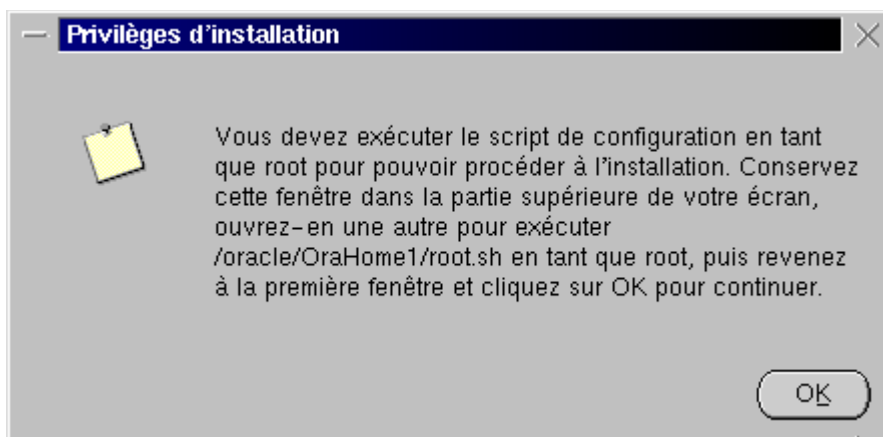


L'installation commence. Elle va durer une quinzaine de minutes. Assez rapidement, on vous demande d'exécuter un script en tant que *root*. Cela arrivera deux fois. La première fois, passez dans votre fenêtre terminal *root* et tapez la commande qui vous est proposée :

```
$ /oracle/OraHome1/orainsRoot.sh
```

Revenez à la fenêtre d'installation d'Oracle et faites "Réessayer" pour continuer.

Le même phénomène se produit vers la fin de l'installation :



Passez dans votre fenêtre terminal *root* et tapez la commande qui vous est proposée :

```
$ /oracle/OraHome1/root.sh
```



```
[root@tahe /root]# /oracle/OraHome1/root.sh
IMPORTANT NOTE: Please delete any log and trace files previously
                  created by the Oracle Enterprise Manager Intelligent
                  Agent. These files may be found in the directories
                  you use for storing other Net8 log and trace files.
                  If such files exist, the OEM IA may not restart.
/oracle/OraHome1/root.sh: -f: command not found
Running Oracle8 root.sh script...
ORACLE_SID is not set.
/oracle/OraHome1/root.sh: command substitution: line 1: unexpected EOF while loo
king for matching `''
/oracle/OraHome1/root.sh: command substitution: line 2: syntax error: unexpected
end of file
/oracle/OraHome1/root.sh: [: !=: unary operator expected
\nThe following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /oracle/OraHome1
    ORACLE_SID=

Enter the full pathname of the local bin directory: [/usr/local/bin]:
Adding entry to /etc/oratab file...
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
[root@tahe /root]# █
```

Le script signale quelques erreurs. On continue quand même en revenant sur la fenêtre précédente de l'installation et en faisant OK.



Une fois l'installation du SGBD terminée, l'assistant de configuration Net8 est lancé. Il va installer les produits qui permettront à Oracle d'avoir des clients distants.



On coche la phrase "Exécuter la configuration standard".



Ouf ! Notre première installation est terminée. Il nous faut maintenant créer une base de données mais auparavant nous devons passer les patches qui vont nous le permettre.

Rappelons que le fichier de patch à passer s'appelle *glibc-2.1.3-stubs.tar.gz*. On suppose ici qu'il a été installé lui aussi dans *<installOracle>*. Utilisez la fenêtre terminal *root*. Placez-vous dans le répertoire *<installOracle>* et décompressez le fichier de patch :

```
$ tar xvfz glibc-2.1.3-stubs.tar.gz
```

Cette décompression donne naissance à

- un exécutable appelé *setup-stubs.sh*
- un répertoire *lib* et dedans un répertoire *stubs*

Le répertoire *stubs* doit être placé dans l'arborescence d'oracle dans le répertoire */oracle/OraHome1/lib*. On procède de la façon suivante :

```
$ ls -l /oracle/OraHome1/lib
// pour vérifier la présence du répertoire lib
$ ls -l /oracle/OraHome1/lib/stubs
// pour vérifier l'absence du répertoire stubs
$ mv lib/stubs /oracle/OraHome1/lib
// pour mettre dans l'arborescence d'oracle le répertoire stubs du patch
$ ls -l /oracle/OraHome1/lib/stubs
// pour vérifier la présence du répertoire stubs dans l'arborescence d'Oracle
```

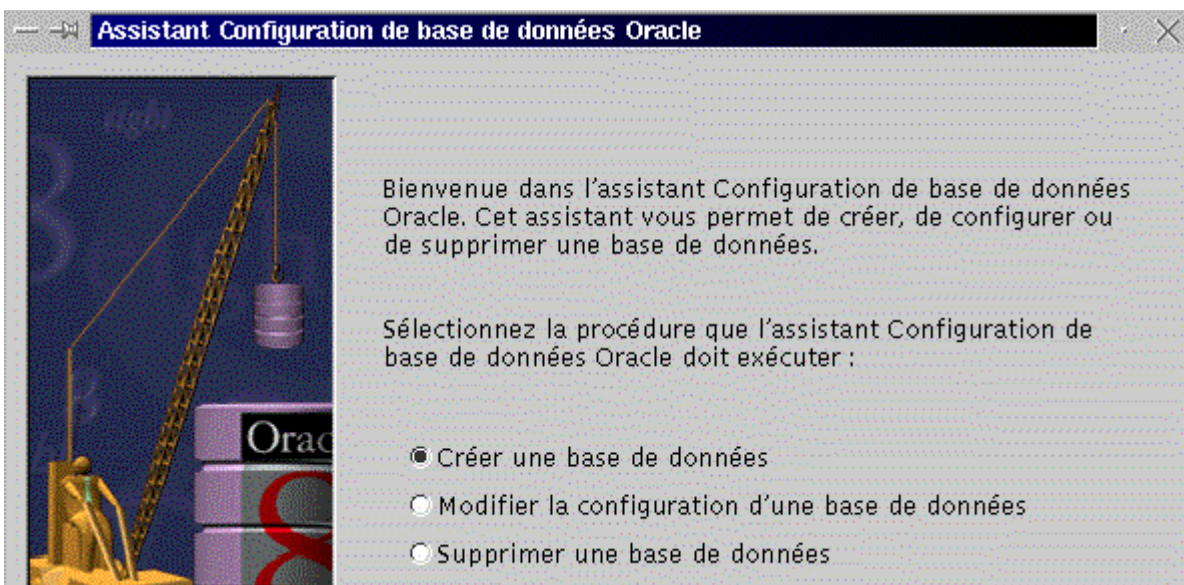
```
$ export ORACLE_HOME=/oracle/OraHome1
// variable d'environnement nécessaire au script du patch
$ ./setup-stubs.sh
// lance le programme de patch - dure quelques minutes environ
// signale ce qu'il fait et doit indiquer qu'il réussit à le faire
```

On est maintenant prêt à créer notre première base de données. Dans les faits, il m'a fallu relancer la machine linux pour faire la suite. Faites-le également si vous échouez dans ce qui suit.

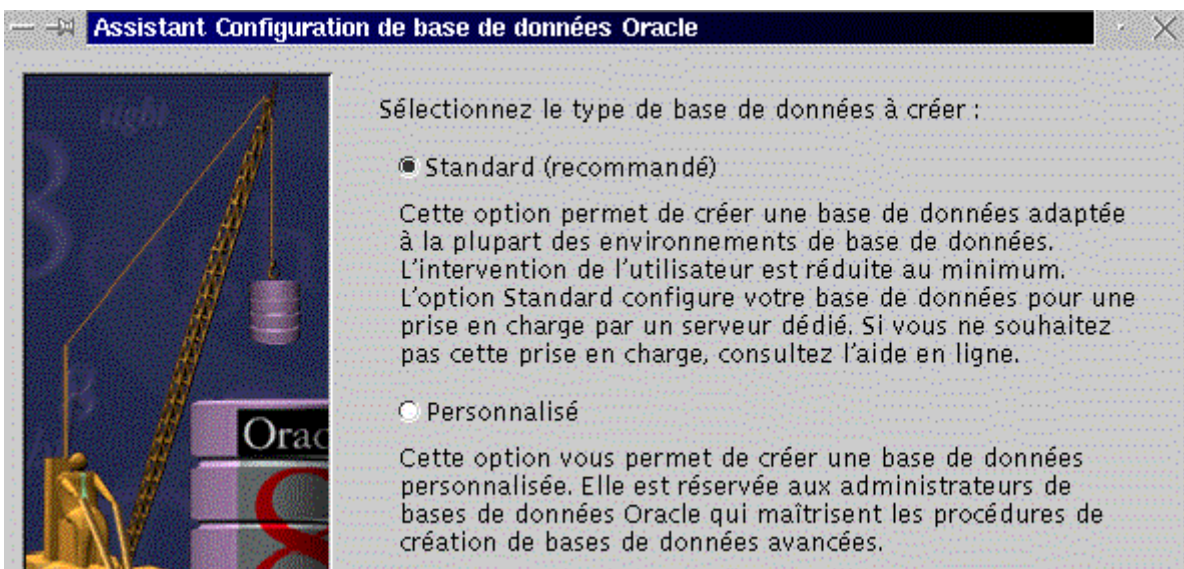
Nous utilisons maintenant l'assistant de création de base de données qui se trouve dans `/oracle/OraHome1/bin`. Revenez dans la fenêtre terminal `oracle`. C'est en effet l'utilisateur `oracle` qui doit faire cette création. Lancez l'assistant de création :

```
$ /oracle/OraHome1/bin/dbassist
// il est arrivé que cette commande ne réponde pas. Relancez votre machine linux si c'est le cas.
```

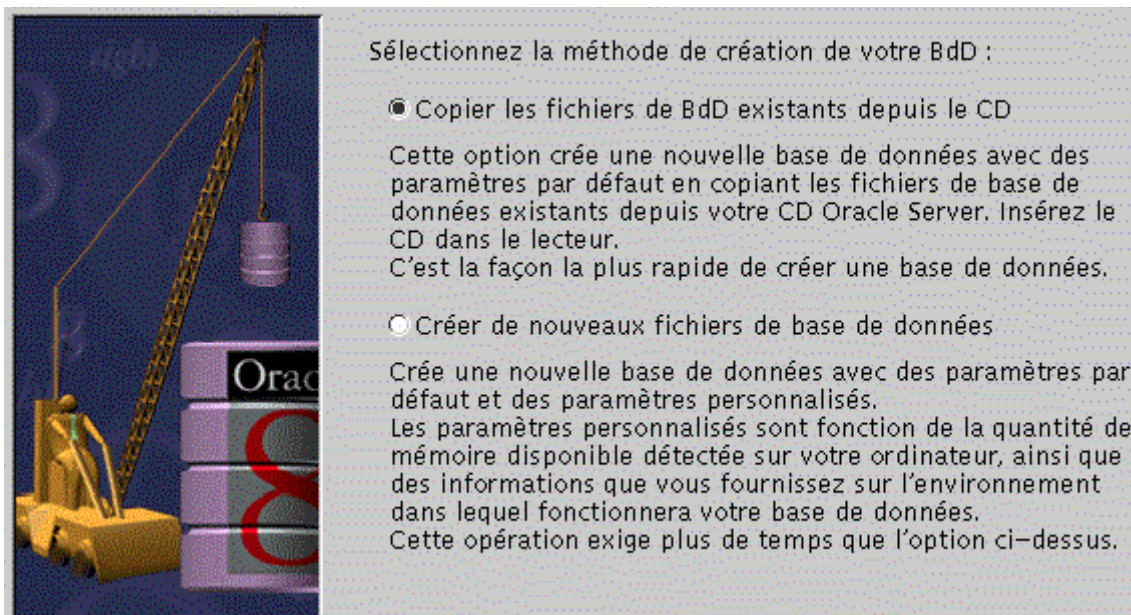
Vous obtenez l'écran suivant:



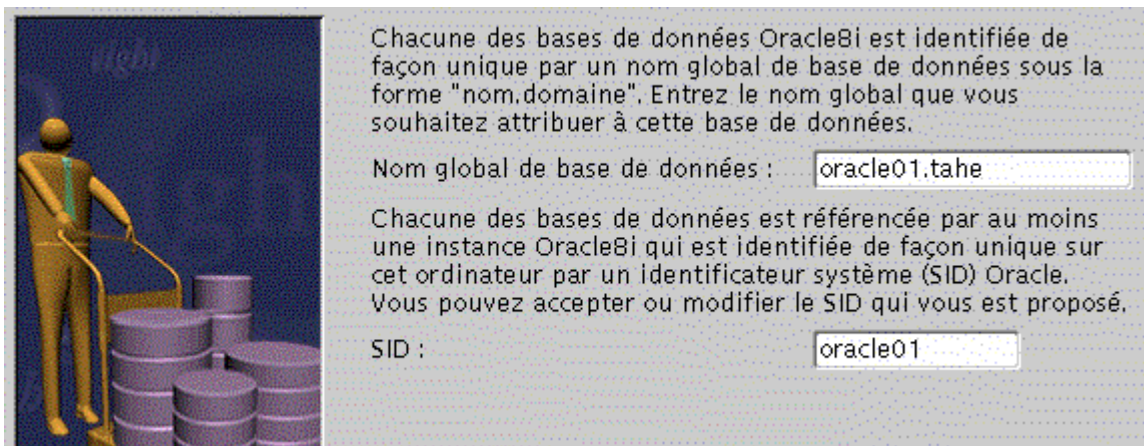
Ici nous voulons créer notre première base de données. Faites [suivant].



Nous n'avons pas assez de compétences pour personnaliser la création de la base. Nous prenons l'option standard qui va faire un certain nombre de choix pour nous. Une fois les compétences acquises, un administrateur peut toujours revenir configurer différemment sa base avec l'assistant (cf écran précédent).



Pour aller vite, nous prenons la première option. Les fichiers seront pris en fait, non pas sur un CD mais dans l'arborescence `<installOracle>` à partir de laquelle vous avez installé Oracle. A noter qu'il est intéressant de garder cette arborescence si vous avez l'intention d'installer des produits que vous n'avez pas installés la première fois.

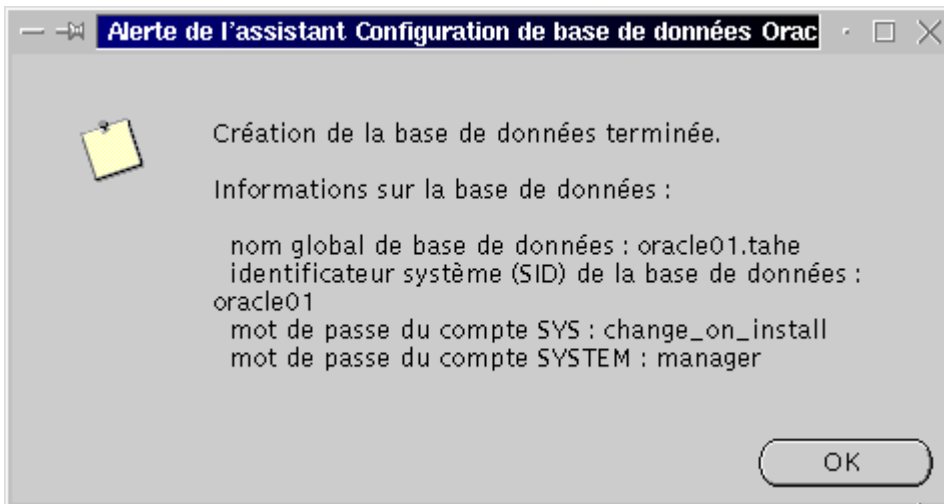


Cet écran sert à identifier la base que vous créez. Cette identification a deux composantes :

- **Nom global de base de données** est de la forme **nom.domaine** où *nom* est le nom que vous donnez à votre base et *domaine* est le domaine dans lequel elle opère. Prenez par exemple le nom `oracle01.nomMachine` où *nomMachine* est le nom de votre machine (`oracle01.linux-31` par exemple, si votre machine s'appelle `linux-31`).
- **SID** : le SGBD peut gérer plusieurs bases à qui on donne un SID (System Identification) unique. Par défaut, l'installateur prend le premier terme du Nom global de base de données que vous avez choisi. Le SID sera ultérieurement mémorisé dans une variable Unix `ORACLE_SID`. Acceptez ce choix et faites [suivant]



La création de la base est lancée et va durer entre 10 et 15 mn.



Notez les informations ci-dessus. Elles vous seront utiles.

L'installation d'oracle a été faite et nous avons maintenant à notre disposition une base. Nous pouvons désormais travailler avec. Nous continuons à travailler dans la fenêtre terminal *oracle* pour émettre des commandes unix. Tout d'abord, un certain nombre de variables d'environnement doivent être fixées à l'aide du script */usr/local/bin/oraenv*.

```
[oracle@tahe bin]$ . /usr/local/bin/oraenv
ORACLE_SID = [oracle] ? oracle01
```

Le script demande le SID de la base à utiliser. Si on suit ce qui a été fait ici, elle s'appelle *oracle01*. Ce qui est en italique & gras ci-dessus est la réponse qui a été tapée par l'utilisateur. Attention à la commande, le point est important (point espace */usr/local/bin/oraenv*) . Les outils nécessaires à la suite sont dans */oracle/OraHome1/bin*. On se place sur ce répertoire qui contient beaucoup d'exécutables liés à Oracle.

```
[oracle@tahe bin]$ cd /oracle/OraHome1/bin
[oracle@tahe bin]$ ls -l
```

Le gestionnaire qui nous permet de lancer/arrêter la base s'appelle **svrmgrl**. On le lance

```
[oracle@tahe bin]$ ./svrmgrl
```

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.1 - Production

JServer Release 8.1.7.0.1 - Production

SVRMGR>

svrmgrl est un outil interactif exécutant les commandes tapées au clavier. La liste est disponible avec la commande *help* :

SVRMGR> help

Pour lancer la base, il nous faut nous connecter avec le login *internal* :

SVRMGR> connect internal
Connected.

Nous lançons la base avec la commande *startup*.

SVRMGR> startup
ORA-01081: cannot start already-running ORACLE - shut it down first

Ici, on nous dit que la base est déjà lancée. Cette séquence a été exécutée immédiatement après la création de la base *oracle01*. Celle-ci est donc active. Ce ne sera en général pas le cas. Nous suivons le conseil et arrêtons la base avec la commande *shutdown* :

SVRMGR> shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.

Nous relançons la base avec la commande *startup*.

SVRMGR> startup
ORACLE instance started.
Total System Global Area 72704160 bytes
Fixed Size 73888 bytes
Variable Size 55681024 bytes
Database Buffers 16777216 bytes
Redo Buffers 172032 bytes
Database mounted.
Database opened.

Cette fois-ci, c'est bon. Nous quittons le programme *svrmgrl* :

SVRMGR> exit
Server Manager complete.

La base *oracle01* est active. Nous souhaitons maintenant créer des tables et les interroger. Pour exploiter la base avec le langage SQL nous utilisons le programme *sqlplus* situé au même endroit que le programme précédent *svrmgrl*. Nous lançons *sqlplus* qui va nous demander de nous identifier avec un login/mot de passe. Par défaut, l'installation a créé un utilisateur d'Oracle (pas d'Unix) nommé *system* et de mot de passe *manager*. Cet utilisateur est un administrateur de la base. Il vous faudra changer son mot de passe dès que possible.

[oracle@tahe bin]\$ **./sqlplus**

SQL*Plus: Release 8.1.7.0.0 - Production on Tue May 1 17:54:39 2001

(c) Copyright 2000 Oracle Corporation. All rights reserved.

Enter user-name: system
Enter password: **manager**

Connected to:

Oracle8i Enterprise Edition Release 8.1.7.0.1 - Production
JServer Release 8.1.7.0.1 - Production

Nous créons un utilisateur *dupont* de mot de passe *dupont* et lui donnons le droit de travailler avec la base :

```
SQL> grant connect,resource to dupont identified by dupont;  
Grant succeeded.
```

Nous quittons *sqlplus* :

```
SQL> exit
```

Nous nous reconnectons à la base en tant qu'utilisateur *dupont* cette fois avec des droits moindres que l'administrateur *system* :

```
[oracle@tahe bin]$ ./sqlplus
```

SQL*Plus: Release 8.1.7.0.0 - Production on Tue May 1 17:56:48 2001

(c) Copyright 2000 Oracle Corporation. All rights reserved.

```
Enter user-name: dupont  
Enter password: dupont
```

```
Connected to:  
Oracle8i Enterprise Edition Release 8.1.7.0.1 - Production  
JServer Release 8.1.7.0.1 - Production
```

Nous créons une table puis vérifions sa création :

```
SQL> create table t1 (nom varchar(20), age number(3));  
Table created.
```

```
SQL> describe t1;
```

Name	Null?	Type
NOM		VARCHAR2(20)
AGE		NUMBER(3)

```
SQL> exit
```

Nous arrêtons la base. Notons ici que nous sommes toujours sous l'identité Unix *oracle*.

```
[oracle@tahe bin]$ ./svrmgrl
```

Oracle Server Manager Release 3.1.7.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.7.0.1 - Production
JServer Release 8.1.7.0.1 - Production

```
SVRMGR> connect internal  
Connected.
```

```
SVRMGR> shutdown  
Database closed.  
Database dismounted.  
ORACLE instance shut down.
```

Oracle est maintenant installé et opérationnel. Si vous ne connaissez pas le langage SQL, vous pouvez suivre l'initiation faite dans ce document dans la partie consacrée au SGBD MySQL.

A l'ISTIA, Oracle est installé sur une machine linux accessible aux étudiants via *telnet*. Une fois connecté, l'étudiant utilise le produit *sqlplus* pour gérer ses propres tables de données Oracle. Chaque étudiant ayant un cours SQL est enregistré dans la

base des utilisateurs d'Oracle. S'il a une machine linux personnelle, l'étudiant peut installer Oracle dessus et accéder ainsi à l'administration du SGBD. Le couple Oracle/Linux devient ainsi un formidable outil de formation.

Le répertoire *<installOracle>* occupe beaucoup de Mo. Il peut être intéressant de le supprimer. Il peut être aussi intéressant de le garder si vous avez de l'espace disque libre. En effet, ici nous n'avons pas installé tous les produits Oracle. Le répertoire *<installOracle>* pourrait donc servir à de futures installations. Pour détruire le répertoire Oracle, émettez la commande suivante dans une fenêtre terminal où vous êtes *root* :

[rm -rf <installOracle>]

ou plus prudemment, dans une session X lancée par *root*, utilisez le gestionnaire de fichiers pour supprimer *<installOracle>*.

10 Installer MySQL sous Linux

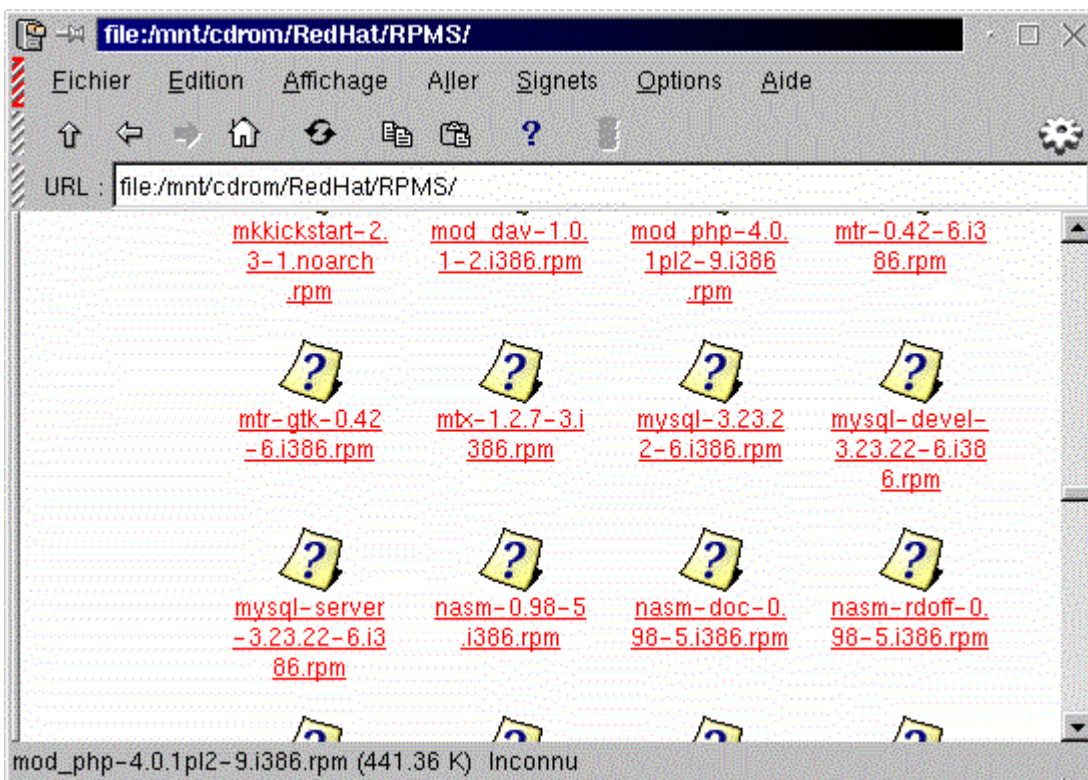
10.1 Introduction

MySQL est une base de données relationnelle gratuite distribuée par la société TCX. Bien que gratuite, elle est performante et permet de s'initier à bon compte à la gestion de bases de données relationnelles et à leur interrogation avec le langage SQL (Structured Query Language). MySQL est couramment utilisée sur les systèmes linux en conjonction avec le langage PHP et le serveur Web Apache pour créer des sites Web dont le contenu est tiré de bases de données.

Il existe des versions de MySQL aussi bien pour Linux que pour Windows. On trouve également pour cette base des pilotes ODBC et JDBC qui permettent à des applications écrites dans des langages divers (C, C++, Java, VB, Perl, ...) d'avoir accès au contenu d'une base MySQL.

Où trouver MySQL ?

Pour Linux RedHat, il existe des paquetages RPM pour MySQL qu'on trouvera sur le site officiel de MySQL <http://www.mysql.com>. Ces RPM sont également disponibles sur le CD n° de la RH 7.0.



Ci-dessus, on voit trois fichiers RPM concernant MySQL :

```
MySQL-3.23.22-6.i.386.rpm  
    // pour interroger le serveur MySQL  
MySQL-devel-3.23.22-6.i.386.rpm  
    // pour développer avec MySQL  
MySQL-server-3.23.22-6.i.386.rpm  
    // le serveur MySQL
```

La version peut bien sûr changer. Sur le site officiel de MySQL, on trouvera de la documentation au format PDF. Il faut la télécharger car elle s'avère indispensable dès qu'on veut comprendre comment gérer les bases de données MySQL.

10.2 Installation de MySQL

Récupérez les fichiers précédents, soit sur le CD soit sur le site officiel et copiez-les par exemple dans le répertoire */tmp*. Installez les différents RPM téléchargés.

```
[cd /tmp]
[ls -l]
// les fichiers RPM doivent être présents
[rpm -i mysql-3.23.22-6.i386.rpm]
[rpm -i mysql-devel-3.23.22-6.i386.rpm]
[rpm -i mysql-server-3.23.22-6.i386.rpm]
// installent le gestionnaire de bases MySQL (mysql-server), un client permettant de travailler sur ces bases
(mysql), et des outils nécessaires au développement d'applications utilisant les bases MySQL (mysql-devel).
```

10.3 Lancement du serveur mysqld

Comme beaucoup de services sous RedHat, il existe des scripts de lancement et d'arrêt du service *mysqld* :

```
[/etc/rc.d/init.d/mysqld start] pour lancer le service
[/etc/rc.d/init.d/mysqld stop] pour arrêter le service
[/etc/rc.d/init.d/mysqld restart] pour enchaîner arrêt/démarrage
```

Lançons donc le service *mysqld* :

```
[root@tahe /root]# /etc/rc.d/init.d/mysqld start
Initializing MySQL database           [ OK ]
Starting MySQL:                       [ OK ]
```

10.4 Utilisation du client MySQL

Nous présentons ici un court tutoriel d'utilisation de la base MySQL. L'utilisateur est invité à lire la documentation PDF du produit. Au besoins, lancer le service *mysqld* si ce n'est déjà fait.

```
[ps -aux | grep -i mysql]
// pour vérifier la présence du démon mysqld
mysql      497  0.1  1.7 10780 1132 ?        SN    07:00   0:00 /usr/sbin/mysqld
mysql      500  0.0  1.7 10780 1132 ?        SN    07:00   0:00 /usr/sbin/mysqld
mysql      501  0.0  1.7 10780 1132 ?        SN    07:00   0:00 /usr/sbin/mysqld
```

```
[mysql]
// appelle le programme client mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 3.22.27
```

Type 'help' for help.

```
// les commandes tapées au clavier seront maintenant exécutées par le
// programme client mysql. Les commandes sont archivées et récupérables avec les
// flèches [HAUT] et [BAS] du clavier.
// les commandes se terminent par le caractère ;
// lorsque mysql affiche le caractère ->, c'est que le ; n'a pas été tapé
// il faut alors continuer la commande - la saisie se terminera par
// l'mission du caractère ; de fin de commande.
```

```
mysql> show databases;
// affiche les bases de données déjà créées
+-----+
| Database |
+-----+
| mysql    |
| test     |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> create database commerce;
// crée une base commerce
Query OK, 1 row affected (0.00 sec)
```

```
mysql> show databases;
```

```

// affiche la nouvelle base
+-----+
| Database |
+-----+
| commerce |
| mysql    |
| test     |
+-----+
3 rows in set (0.00 sec)

mysql> use commerce;
// on fait de la nouvelle base la base courante
Database changed

mysql> show tables;
// affiche les tables de la base courante
Empty set (0.00 sec)

mysql> create table articles (code char(4), nom char(20), prix decimal(10,2),
-> stock integer);
// crée une table articles
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
// vérification
+-----+
| Tables in commerce |
+-----+
| articles            |
+-----+
1 row in set (0.00 sec)

mysql> create table clients (code char(4), nom char(20));
// crée une table clients
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+-----+
| Tables in commerce |
+-----+
| articles            |
| clients             |
+-----+
2 rows in set (0.00 sec)

mysql> create table ventes (codeClient char(4), codeArticle char(4), quantite integer);
// table des ventes
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+-----+
| Tables in commerce |
+-----+
| articles            |
| clients             |
| ventes              |
+-----+
3 rows in set (0.00 sec)

mysql> describe articles;
// affiche la structure de la table articles
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| code  | char(4)       | YES  |     | NULL    |       |
| nom   | char(20)      | YES  |     | NULL    |       |
| prix  | decimal(10,2) | YES  |     | NULL    |       |
| stock | int(11)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> describe clients;
// structure de la table clients
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| code  | char(4)       | YES  |     | NULL    |       |
| nom   | char(20)      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

```
mysql> describe ventes;
// structure de la table ventes
+-----+-----+-----+-----+-----+-----+
| Field      | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| codeClient | char(4) | YES  |     | NULL    |       |
| codeArticle | char(4) | YES  |     | NULL    |       |
| quantite   | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> insert into articles (code,nom,prix,stock) values ('a001','article1',1000,100);
// insertion d'une ligne dans la table articles
Query OK, 1 row affected (0.00 sec)

mysql> insert into articles (code,nom,prix,stock) values ('a002','article2',2000,200);
mysql> insert into articles (code,nom,prix,stock) values ('a003','article3',3000,300);
mysql> insert into articles (code,nom,prix,stock) values ('a004','article4',4000,400);
mysql> insert into articles (code,nom,prix,stock) values ('a005','article5',5000,500);

mysql> select * from articles;
// affiche le contenu de la table articles
+-----+-----+-----+-----+
| code | nom      | prix    | stock |
+-----+-----+-----+-----+
| a001 | article1 | 1000.00 | 100   |
| a002 | article2 | 2000.00 | 200   |
| a003 | article3 | 3000.00 | 300   |
| a004 | article4 | 4000.00 | 400   |
| a005 | article5 | 5000.00 | 500   |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> insert into clients (code,nom) values ('c001','client1');
// ajoute une ligne dans la table clients
Query OK, 1 row affected (0.00 sec)

mysql> insert into clients (code,nom) values ('c002','client2');
mysql> insert into clients (code,nom) values ('c003','client3');

mysql> select * from clients;
// contenu de la table clients
+-----+-----+
| code | nom      |
+-----+-----+
| c001 | client1  |
| c002 | client2  |
| c003 | client3  |
+-----+-----+
3 rows in set (0.01 sec)

mysql> insert into ventes (codeClient, codeArticle, quantite) values ('c001','a003',10);
// le client c001 a acheté 10 articles a003
Query OK, 1 row affected (0.00 sec)

// on fait d'autres insertions dans la table ventes
mysql> insert into ventes (codeClient, codeArticle, quantite) values ('c001','a002',5);
mysql> insert into ventes (codeClient, codeArticle, quantite) values ('c002','a002',8);
mysql> insert into ventes (codeClient, codeArticle, quantite) values ('c002','a001',2);
mysql> insert into ventes (codeClient, codeArticle, quantite) values ('c003','a001',20);
mysql> insert into ventes (codeClient, codeArticle, quantite) values ('c003','a005',12);

mysql> select * from ventes;
// les ventes effectuées
+-----+-----+-----+
| codeClient | codeArticle | quantite |
+-----+-----+-----+
| c001       | a003       | 10       |
| c001       | a002       | 5        |
| c002       | a002       | 8        |
| c002       | a001       | 2        |
| c003       | a001       | 20       |
| c003       | a005       | 12       |
+-----+-----+-----+
6 rows in set (0.00 sec)

// quelques interrogations

mysql> select nom, prix from articles;
// seulement certaines colonnes
+-----+-----+

```

```
| nom      | prix      |
+-----+-----+
| article1 | 1000.00   |
| article2 | 2000.00   |
| article3 | 3000.00   |
| article4 | 4000.00   |
| article5 | 5000.00   |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select nom, prix from articles order by prix desc;
// idem dans l'ordre décroissant des prix
```

```
+-----+-----+
| nom      | prix      |
+-----+-----+
| article5 | 5000.00   |
| article4 | 4000.00   |
| article3 | 3000.00   |
| article2 | 2000.00   |
| article1 | 1000.00   |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select nom, prix, stock from articles where stock<300 order by prix desc;
// idem mais seulement avec les lignes où stock<300
```

```
+-----+-----+-----+
| nom      | prix      | stock |
+-----+-----+-----+
| article2 | 2000.00   | 200   |
| article1 | 1000.00   | 100   |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select nom, prix, stock from articles where stock<300 and prix<2000 order by prix desc;
// idem mais on a rajouté la condition prix<2000
```

```
+-----+-----+-----+
| nom      | prix      | stock |
+-----+-----+-----+
| article1 | 1000.00   | 100   |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> select * from ventes order by quantite;
// les ventes par ordre croissant des quantités
```

```
+-----+-----+-----+
| codeClient | codeArticle | quantite |
+-----+-----+-----+
| c002       | a001       | 2        |
| c001       | a002       | 5        |
| c002       | a002       | 8        |
| c001       | a003       | 10       |
| c003       | a005       | 12       |
| c003       | a001       | 20       |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select articles.nom,ventes.codeClient,ventes.quantite from ventes,articles
-> where articles.code=ventes.codeArticle
-> order by quantite;
```

```
// idem mais avec les noms des clients - jointure entre 2 tables
```

```
+-----+-----+-----+
| nom      | codeClient | quantite |
+-----+-----+-----+
| article1 | c002       | 2        |
| article2 | c001       | 5        |
| article2 | c002       | 8        |
| article3 | c001       | 10       |
| article5 | c003       | 12       |
| article1 | c003       | 20       |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select articles.nom article ,ventes.codeClient client,ventes.quantite from ventes, articles
-> where articles.code=ventes.codeArticle
-> order by quantite;
```

```
// idem mais avec des alias pour les colonnes (article, client)
```

article	client	quantite
article1	c002	2
article2	c001	5
article2	c002	8
article3	c001	10
article5	c003	12
article1	c003	20

6 rows in set (0.00 sec)

```
mysql> select articles.nom article, clients.nom client, ventes.quantite
-> from ventes, articles, clients
-> where ventes.codeArticle=articles.code
-> and ventes.codeClient=clients.code
-> order by quantite;
```

// on rajoute le nom du client

article	client	quantite
article1	client2	2
article2	client1	5
article2	client2	8
article3	client1	10
article5	client3	12
article1	client3	20

6 rows in set (0.01 sec)

// quelques mises à jour

```
mysql> select nom, prix from articles;
// état actuel des prix
```

nom	prix
article1	1000.00
article2	2000.00
article3	3000.00
article4	4000.00
article5	5000.00

5 rows in set (0.00 sec)

```
mysql> update articles set prix=prix*1.1;
```

// augmente les prix de 10%

Query OK, 5 rows affected (0.00 sec)

Rows matched: 5 Changed: 5 Warnings: 0

```
mysql> select nom, prix from articles;
```

// vérification

nom	prix
article1	1100.00
article2	2200.00
article3	3300.00
article4	4400.00
article5	5500.00

5 rows in set (0.00 sec)

```
mysql> update articles set prix=prix+1000 where prix>3000;
```

// une hausse pour les articles dont le prix est >3000

Query OK, 3 rows affected (0.00 sec)

Rows matched: 3 Changed: 3 Warnings: 0

```
mysql> select nom, prix from articles;
```

// vérification

nom	prix
article1	1100.00
article2	2200.00
article3	4300.00
article4	5400.00
article5	6500.00

```

5 rows in set (0.01 sec)

mysql> delete from articles where nom='article4';
// suppression article 4
Query OK, 1 row affected (0.00 sec)

mysql> select nom, prix from articles;
// vérification
+-----+-----+
| nom      | prix    |
+-----+-----+
| article1 | 1100.00 |
| article2 | 2200.00 |
| article3 | 4300.00 |
| article5 | 6500.00 |
+-----+-----+
4 rows in set (0.00 sec)

// on quitte
mysql> quit
Bye

```

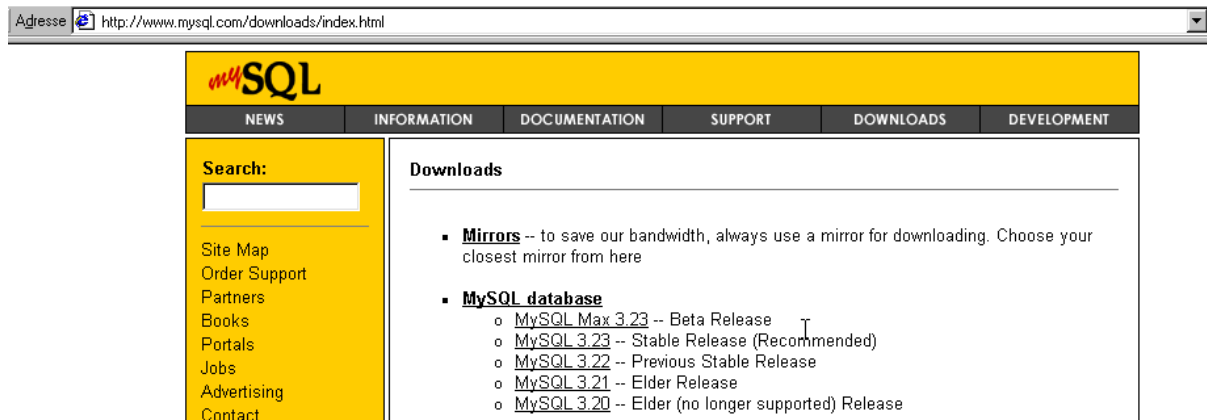
10.5 Poursuivre

Il est maintenant temps de lire la documentation de MySQL au format PDF que vous trouverez sur les sites tels que <http://www.mysql.com> et ses sites miroirs.

11 Installation de MySQL sous Windows 9x

11.1 Où trouver MySQL

On trouvera MySQL à l'URL <http://www.mysql.com/downloads/index.html> :



En suivant ci-dessus par exemple le lien MySQL 3.23 on arrive à la page suivante qui propose des versions de MySQL pour différents OS, dont Win32 :

Standard binary (tarball) distributions:

- [MacOS X Server \(PowerPC\) \[apple-rhapsody5.5-powerpc\]](#)
- [HP-UX 10.20 \[hp-hpux10.20-hppa1.1\]](#)
- [IBM AIX \(PowerPC\) \[ibm-aix4.3.3.0-rs6000\]](#)
- [Linux \(Intel libc6 systems\) \[pc-linux-gnu-i386\]](#)
- [SCO OpenServer \(Intel\) \[pc-sco3.2v5.0.6-i386\]](#)
- [SCO UnixWare \(Intel\) \[sco-sysv5uw7.1.0-i386-lcudk70\]](#)
- [SCO UnixWare \(Intel\) \[sco-sysv5uw7.1.0-i386\]](#)
- [SGI Irix \[sgi-irix6.5-mips\]](#)
- [Sun Solaris \(Sparc\) \[sun-solaris2.7-sparc\] \(Use GNU Tar\)](#)
- [Sun Solaris \(Sparc\) \[sun-solaris2.8-sparc\] \(Use GNU Tar\)](#)
- [FreeBSD ELF \(Intel\) \[unknown-freebsdelf4.3-i386\]](#)
- [Linux \(Alpha\) \[unknown-linux-gnu-alphaev6\]](#)
- [Linux \(Sparc\) \[unknown-linux-gnu-sparc\]](#)
- [Windows 95/98/NT/2000 \(Intel\)](#)

Le lien *Window 95/98/NT/2000* permet de récupérer un fichier zippé de MySQL. Utile également est un driver ODBC pour MySQL. Les applications Windows peuvent travailler avec n'importe quelle base de donnée si celle-ci a un driver ODBC. Chaque constructeur de BD fournit en général ce pilote. Celui de MySQL s'appelle **MyODBC** et peut être trouvé à l'URL <http://www.mysql.com/downloads/api-myodbc.html>.

La documentation sur MySQL peut être trouvée à l'URL <http://www.mysql.com/documentation>.

MySQL Manual

- [Searchable, with user comments](#)
- [HTML, one page per chapter](#)
- [HTML, all on one page](#)
- [Download alternate formats \(HTML, PDF, PS, Texinfo\)](#)
- [Dynamic FAQ \(courtesy of SupportWizard.com\)](#)

En suivant le lien *alternate formats*, on arrive à la page permettant d'obtenir la documentation au format PDF pour impression ou format HTML pour une aide en ligne.

Alternate formats

HTML

- [MySQL Manual](#) (one page per chapter, tarball)
- [MySQL Manual](#) (all on one page, tarball)

PDF (Adobe Acrobat Reader)

- [MySQL Manual](#) (download by holding down the shift-key when you click)

MyODBC for Windows

Read the latest [Release Notes](#) for MyODBC 2.50.38.

- [MyODBC 2.50.38 for Windows95/98 \(full setup\)](#)
- [MyODBC 2.50.38 for NT/2000 \(full setup\)](#)
- [MyODBC 2.50.38 Windows95/98 and NT/2000 \(only myodbc.dll and myodbc2.dll\)](#)
- [Source for MyODBC 2.50.38 for Windows95/98 or NT/2000](#) Note that you need the [3.23.xx client distribution](#) to compile this! You can also use the 3.23.xx clients with earlier mysqld servers.

Au final, on a au moins les trois fichiers suivants avec des noms qui peuvent être différents selon les versions du moment.

Nom	Taille
 manual.pdf	2 022 Ko
 myodbc-2.50.29-win95.zip	1 520 Ko
 mysql-shareware-3.22.34-win.zip	5 166 Ko

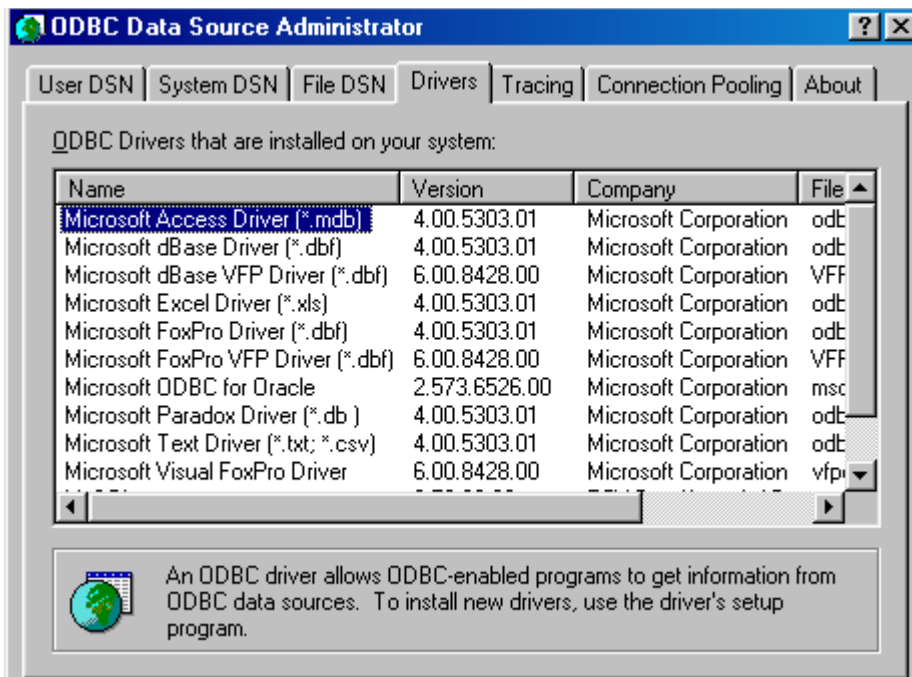
11.2 Installation du pilote MyODBC

Sous Windows, un pilote ODBC (Open DataBase Connectivity) sert à masquer aux applications les particularités des bases de données. Ainsi une application Windows pourra utiliser l'interface standard des pilotes ODBC sans se préoccuper de la base de données qui est derrière. Celle-ci peut changer, l'application elle n'aura pas à être réécrite. Cette souplesse se paie par une moindre performance vis à vis des pilotes écrits spécialement pour la base de données.

Vous pouvez voir la liste des pilotes ODBC déjà installés sur votre machine par *Démarrer/Paramètres/Panneau de configuration*.



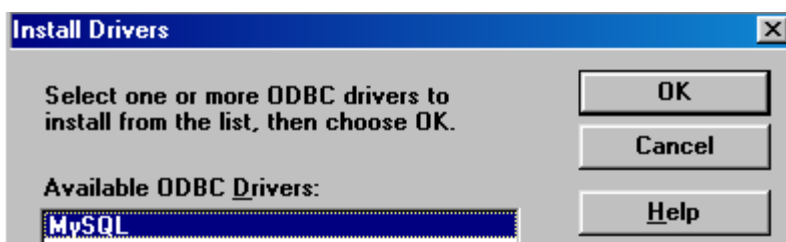
L'une des icônes du Panneau de configuration est *ODBC Data Sources*. C'est l'outil de configuration des bases de données ODBC c'est à dire des bases ayant un pilote ODBC. Lorsqu'on ouvre cette application, on obtient un classeur à plusieurs pages dont celui des pilotes ODBC :



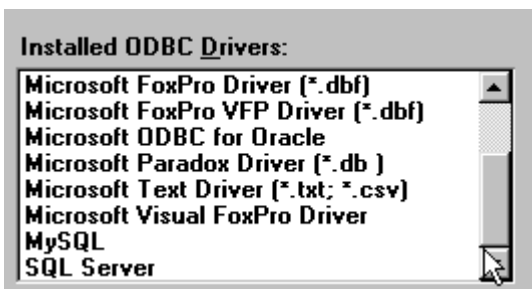
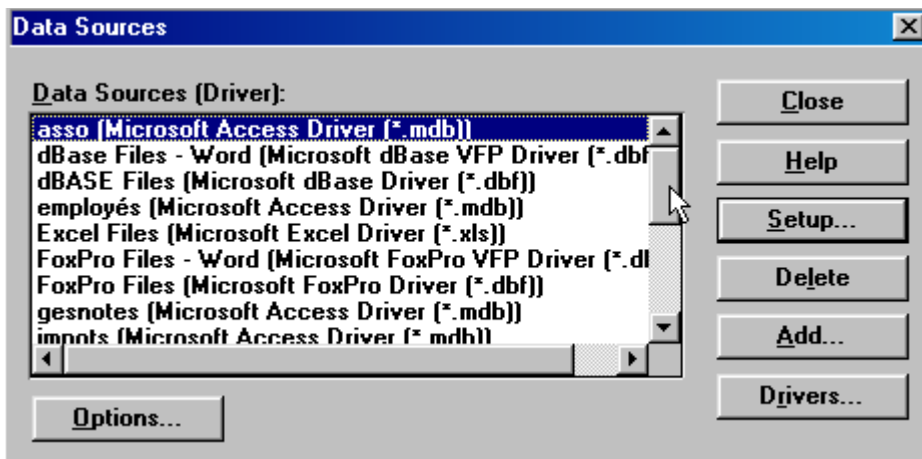
Ci-dessus, vous avez la liste des pilotes ODBC installés sur une machine Windows 9x. Pour installer le pilote ODBC pour la base MySQL, décompressez le fichier *MyODBCxxx.zip* que vous avez récupéré plus haut et lancez l'installation du pilote :



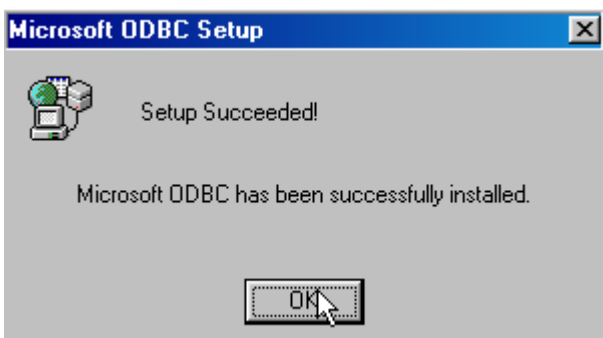
Faites [Continue].



Sélectionnez MySQL et faites [OK]. Le pilote s'installe. Ensuite un écran est présenté afin de sélectionner une source de données ODBC. Une source de données ODBC est une base de données accessible par un pilote ODBC à laquelle on a donné un nom "public" via lequel la base sera manipulée. Cela évite à une application d'avoir à connaître l'emplacement physique de la base. Nous n'allons pas configurer de source ODBC maintenant mais vous pouvez avoir la curiosité d'utiliser le bouton [Drivers] ci-dessous et découvrir qu'il existe maintenant u pilote ODBC pour MySQL.



Terminez l'installation avec le bouton [Close] des boîtes de dialogue :

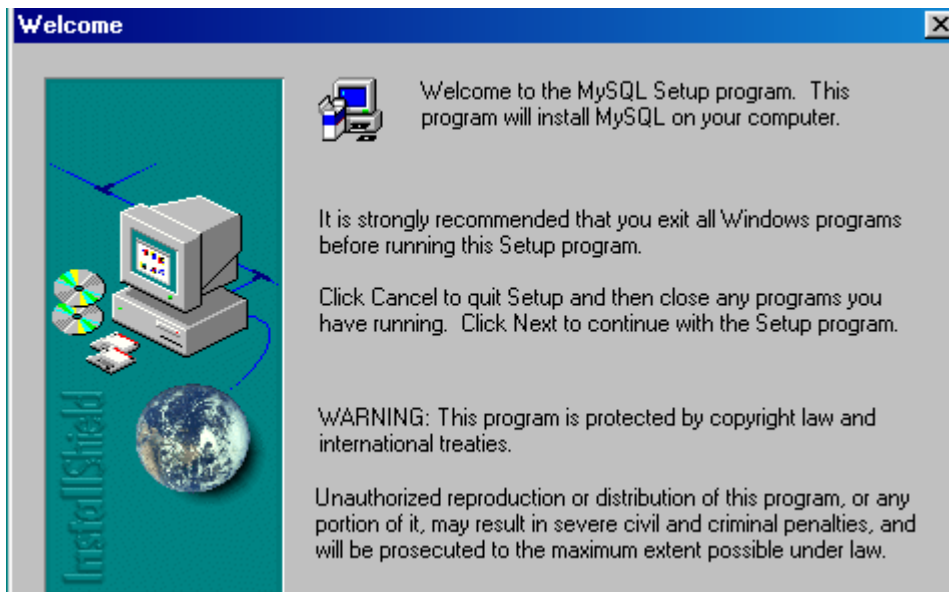


L'installation du pilote MyODBC n'est pas nécessaire à l'installation de MySQL. On peut travailler avec cette base sans ce pilote. Celui-ci est néanmoins très utile. Il permet par exemple

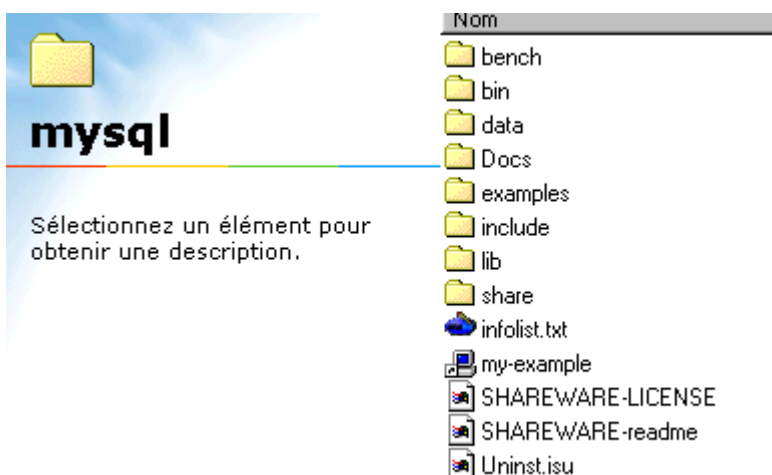
- d'interroger une base de données mySQL avec Microsoft Query
- d'échanger des données entre mySQL et d'autres applications windows telles Access ou Excel.

11.3 Installation de mySQL

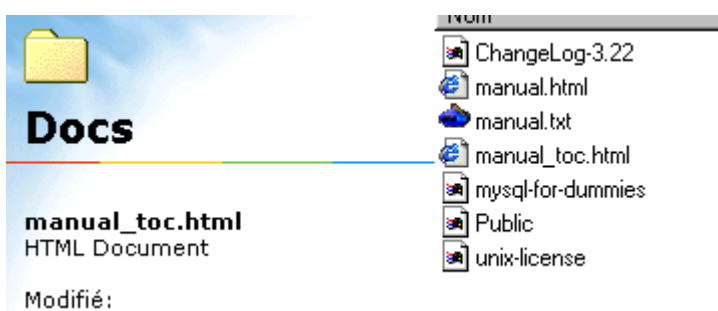
On décompresse le .zip de MySQL et on lance l'installation :



On a là une installation classique. Prendre l'installation typique (*typical*) qui est la plus simple. MySQL sera installé sous C:\mysql par défaut. Voici un exemple de ce dossier après installation :



On pourra dans un premier temps aller dans le répertoire Docs :



Le fichier *manual_toc.html* est une table des matières d'une aide HTML.

MySQL Reference Manual for version 3.23.11-alpha.

- [1 General Information about MySQL](#)
 - [1.1 What is MySQL?](#)
 - [1.2 About this manual](#)
 - [1.2.1 Conventions used in this manual](#)
 - [1.3 History of MySQL](#)
 - [1.4 Books about MySQL](#)
 - [1.5 The main features of MySQL](#)
 - [1.6 How stable is MySQL?](#)
 - [1.7 Year 2000 compliance](#)
 - [1.8 General SQL information and tutorials](#)
 - [1.9 Useful MySQL-related links](#)
- [2 MySQL mailing lists and how to ask questions or report errors \(bugs\)](#)
 - [2.1 The MySQL mailing lists](#)
 - [2.2 Asking questions or reporting bugs](#)
 - [2.3 How to report bugs or problems](#)
 - [2.4 Guidelines for answering questions on the mailing list](#)

On trouve dans cette table des matières une section concernant MySQL pour windows :

- [4.12 Win32 notes](#)
 - [4.12.1 Installing MySQL on Win32](#)
 - [4.12.2 Starting MySQL on Win95 / Win98](#)
 - [4.12.3 Starting MySQL on NT](#)
 - [4.12.4 Running MySQL on Win32](#)
 - [4.12.5 Connecting to a remote MySQL from Win32 with SSH](#)
 - [4.12.6 MySQL-Win32 compared to Unix MySQL](#)

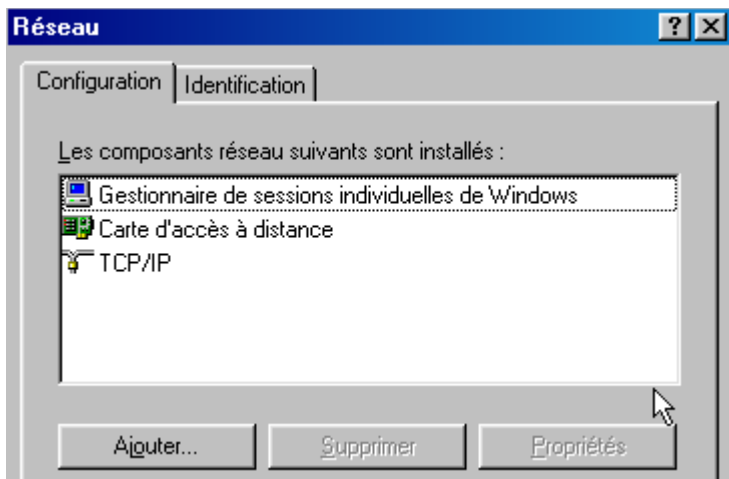
On y apprend que pour que MySQL fonctionne sur une machine Windows il faut que l'environnement TCP-IP ait été installé sur cette machine. Nous décrivons maintenant cette installation si elle a n'a pas été faite sur votre machine :

11.4 Installation de l'environnement TCP-IP sur une machine Windows

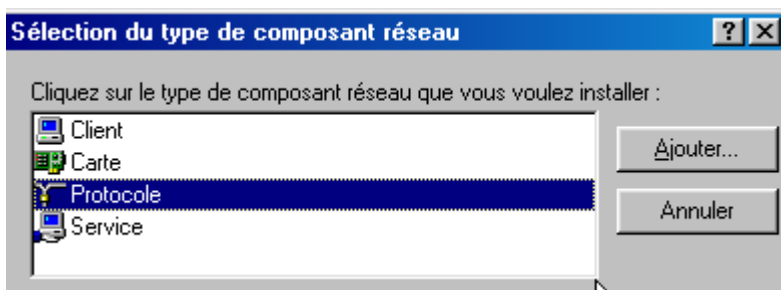
Prenez l'option *Démarrer/Paramètres/Panneau de configuration* du menu :



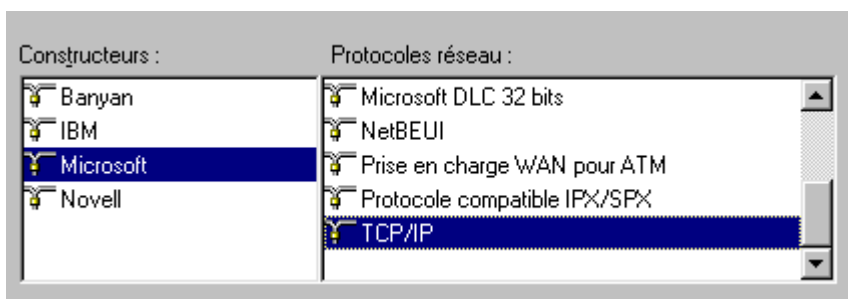
Prenez l'option réseau.



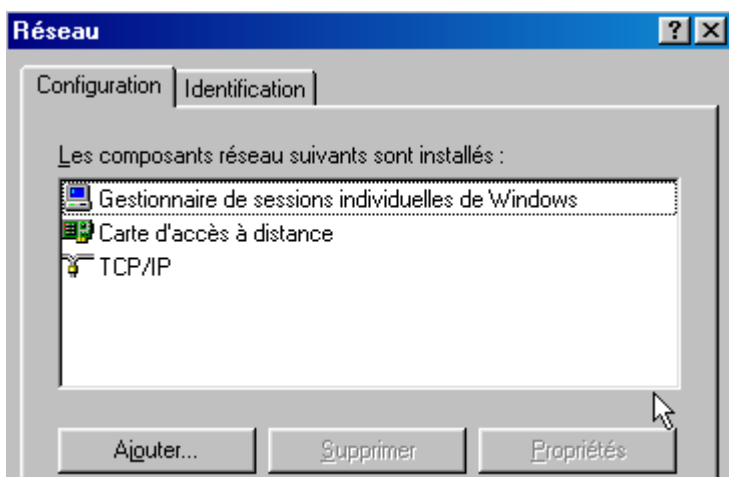
Sur la machine de cet exemple, l'environnement TCP/IP a été installé. Si ce n'est pas le cas sur votre machine, prenez l'option [Ajouter] :



Sélectionnez [Protocole] puis [Ajouter] pour ajouter un protocole réseau à votre machine. Dans le panneau suivant, sélectionnez le protocole TCP/IP de Microsoft :



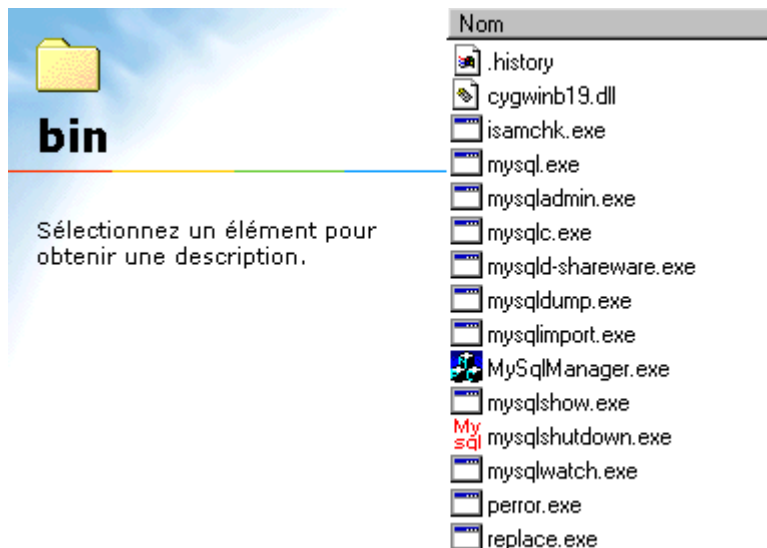
Vous devez avoir maintenant le panneau du début avec de plus le protocole TCP/IP :



Il ne vous reste plus qu'à faire [OK] et suivre les instructions. Vous aurez besoin du CD de Windows 9x.

11.5 Lancer et tester MySQL

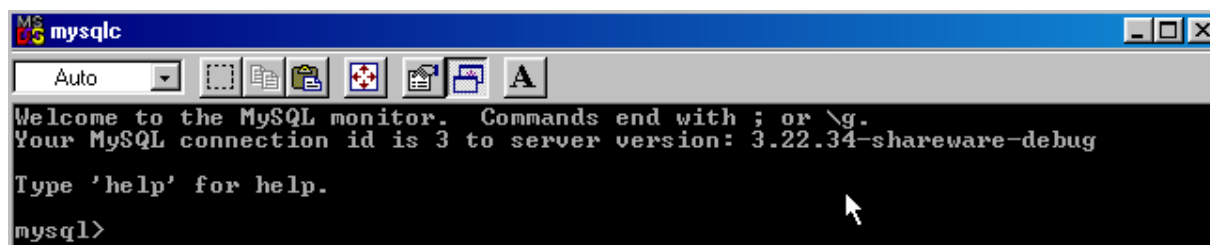
L'exécutable du serveur MySQL se trouve dans le dossier *bin* du répertoire d'installation :



Ici, il s'agit de *mysqld-shareware.exe*. Ce nom peut changer. Consultez la documentation du dossier *Docs*. On peut lancer le serveur MySQL en double-cliquant sur son exécutable ou bien en tapant son nom dans une fenêtre Dos. Cette dernière solution permet de passer des paramètres au programme si besoin est. Lorsque le serveur se lance, il affiche une fenêtre Dos qui se ferme aussitôt et c'est tout. En fait, le serveur travaille en "tâche de fond" et n'a pas d'interface visible. On peut s'assurer de sa présence en faisant Ctrl-Alt-Suppr ce qui a pour effet de présenter la liste des tâches actives. Vous devriez voir *Mysqld-shareware* dans la liste. Un autre moyen de s'assurer de la présence active du serveur MySQL est de l'interroger. Nous utiliserons ici, un outil en ligne *mysqlc.exe* qui permet de taper des commandes sql. On pourrait utiliser aussi le client *mysql.exe* qui fait la même chose. *mysqlc.exe* est une amélioration de *mysql.exe* qui permet une mémorisation des commandes émises ce qui permet de les retrouver pour les rejouer.

11.6 Client ligne mysqlc

Double-cliquez sur l'exécutable *mysqlc.exe*. Une fenêtre Dos s'ouvre.



Vous pouvez maintenant taper toute commande mysql dont des commandes SQL, par exemple `show databases` qui donne la liste des bases de données gérées par MySQL :

```
mysql> show databases;
+-----+
| Database |
+-----+
| impots   |
| levacher |
| mysql    |
| personnes|
| test     |
+-----+
5 rows in set (0.04 sec)

mysql> _
```

Vous pouvez reprendre ici l'exemple donné pour MySQL/Linux. On quitte l'application cliente *mysql* avec la commande *quit* :

```
mysql> exit
```

11.7 Administrer le serveur MySQL

Ce document n'a pas pour but d'apprendre l'administration d'une base MySQL, mais simplement d'en installer une pour faire des requêtes SQL. Il serait cependant utile de se plonger dans l'administration de MySQL, ce SGBD étant relativement répandu sur des machines de production, essentiellement des machines Linux. Un outil pour administrer MySQL sous Windows est *mysqladmin.exe* dans le répertoire BIN de l'installation.

Dans une fenêtre DOS, tapez :

```
C:\MYSQL\BIN>mysqladmin
```

```
C:\MYSQL\BIN\MYSQLA~1.EXE Ver 8.2 Distrib 3.22.34, for Win95/Win98 on i586
TCX Datakonsult AB, by Monty
This software comes with NO WARRANTY: see the file PUBLIC for details.
```

Administer program for the mysqld demon

Usage: C:\MYSQL\BIN\MYSQLA~1.EXE [OPTIONS] command command....

```
-#, --debug=...      Output debug log. Often this is 'd:t:o,filename`
-f, --force          Don't ask for confirmation on drop database; with
                    multiple commands, continue even if an error occurs

-?, --help          Display this help and exit
-C, --compress       Use compression in server/client protocol
-h, --host=#         Connect to host
-p, --password[=...] Password to use when connecting to server
                    If password is not given it's asked from the tty
-W, --pipe          Use named pipes to connect to server
-P, --port=...       Port number to use for connection
-i, --sleep=sec      Execute commands again and again with a sleep between
-r, --relative       Show difference between current and previous values
                    when used with -i. Currently works only with
                    extended-status
-R, --relativert     Same as --relative above, but output is printed
                    vertically.

-s, --silent         Silently exit if one can't connect to server
-S, --socket=...     Socket file to use for connection
-t, --timeout=...    Timeout for connection to the mysqld server
-u, --user=#         User for login if not current user
-v, --verbose        Write more information
-V, --version        Output version information and exit
-w, --wait[=retries] Wait and retry if connection is down
```

Default options are read from the following files in the given order:

```
C:\WINDOWS\my.ini c:\my.cnf
```

The following groups are read: mysqladmin client

The following options may be given as the first argument:

```
--print-defaults    Print the program argument list and exit
--no-defaults       Don't read default options from any options file
--defaults-file=#    Only read default options from the given file #
```


Where command is a one or more of: (Commands may be shortened)

create databasename	Create a new database
drop databasename	Delete a database and all its tables
extended-status	Gives an extended status message from the server
flush-hosts	Flush all cached hosts
flush-logs	Flush all logs
flush-status	Clear status variables
flush-tables	Flush all tables
flush-privileges	Reload grant tables (same as reload)
kill id,id,...	Kill mysql threads
password new-password	Change old password to new-password
ping	Check if mysqld is alive
processlist	Show list of active threads in server
reload	Reload grant tables
refresh	Flush all tables and close and open logfiles
shutdown	Take server down
status	Gives a short status message from the server
variables	Prints variables available
version	Get version info from server

En lisant le texte précédent, on y découvre deux commandes intéressantes :

<code>mysqladmin ping</code>	pour vérifier que le serveur MySQL est actif
<code>mysqladmin shutdown</code>	pour arrêter que le serveur MySQL

1 L'ENVIRONNEMENT SOLPLUS D'ORACLE.....	2
1.1 SYNTAXE DES COMMANDES SQL	3
1.2 SYNTAXE DES COMMANDES SQLPLUS.....	3
1.3 QUELQUES COMMANDES SQLPLUS.....	3
1.3.1 SORTIE DE SQLPLUS.....	3
1.4 EXÉCUTER UNE COMMANDE SYSTÈME.....	4
1.4.1 GESTION DU BUFFER SQL.....	5
1.4.2 SAUVEGARDE ET RÉCUPÉRATION DU BUFFER.....	6
1.4.3 EXÉCUTION DU BUFFER SQL.....	7
1.4.4 GESTION DES FICHIERS DE COMMANDES.....	7
1.5 CONCLUSION.....	8
2 INTRODUCTION AU LANGAGE SQL.....	9
2.1 PRÉLIMINAIRES.....	9
2.2 LES TYPES DE DONNÉES D'ORACLE.....	9
2.3 LES CHAÎNES DE CARACTÈRES.....	9
2.4 LES NOMBRES.....	9
2.4.1 LES DATES.....	10
2.4.2 LES DONNÉES NULLES.....	10
2.4.3 LES CONVERSIONS DE TYPE.....	10
2.5 CRÉATION D'UNE TABLE.....	10
2.6 AFFICHER LA STRUCTURE D'UNE TABLE.....	11
2.7 REMPLISSAGE D'UNE TABLE.....	11
2.8 CONSULTATION DE LA TABLE.....	12
2.8.1 INTRODUCTION.....	12
2.8.2 AFFICHAGE DES LIGNES VÉRIFIANT UNE CONDITION.....	13
2.8.3 AFFICHAGE DES LIGNES SELON UN ORDRE DÉTERMINÉ.....	15
2.9 SUPPRESSION DE LIGNES DANS UNE TABLE.....	17
2.10 MODIFICATION DU CONTENU D'UNE TABLE.....	17
2.11 MISE À JOUR DÉFINITIVE D'UNE TABLE.....	18
2.12 MISE EN FORME DE L'AFFICHAGE ÉCRAN.....	20
2.12.1 CONTRÔLE DE L'AFFICHAGE DES COLONNES.....	20
2.12.2 FORMAT D'UNE COLONNE NUMÉRIQUE OU TEXTE.....	20
2.12.3 AFFICHAGE D'UNE COLONNE DE DATES.....	22
2.13 CRÉATION D'UNE TABLE À PARTIR D'UNE AUTRE TABLE.....	23
2.14 OBTENIR LA LISTE DES TABLES CRÉÉES.....	24
2.15 AJOUT DE LIGNES DANS UNE TABLE EN PROVENANCE D'UNE AUTRE TABLE.....	25
2.16 CHANGER LE NOM D'UNE TABLE.....	26
2.17 SUPPRESSION D'UNE TABLE.....	26
2.18 MODIFICATION DE LA STRUCTURE D'UNE TABLE.....	27
2.19 LES VUES.....	28
2.19.1 CRÉATION D'UNE VUE.....	28
2.19.2 MISE À JOUR D'UNE VUE.....	29
2.19.3 OBTENIR LA LISTE DES VUES.....	30
2.19.4 SUPPRIMER UNE VUE.....	30
2.20 UTILISATION DE FONCTIONS DE GROUPE.....	31
2.21 LA PSEUDO-COLONNE ROWNUM.....	33
2.22 MÉMORISER ET IMPRIMER LES COMMANDES SQL ET LEURS RÉSULTATS.....	33
3 LES EXPRESSIONS DU LANGAGE SQL.....	35
3.1 INTRODUCTION.....	35
3.2 EXPRESSIONS AVEC OPÉRATEUR.....	35
3.2.1 LES EXPRESSIONS À OPÉRANDES DE TYPE NUMÉRIQUE.....	35
3.2.2 LES EXPRESSIONS À OPÉRANDES DE TYPE CARACTÈRES.....	37
3.2.3 LES EXPRESSIONS À OPÉRANDES DE TYPE DATE.....	39
3.2.4 EXPRESSIONS À OPÉRANDES BOOLÉENS.....	41
3.3 LES FONCTIONS PRÉDÉFINIES D'ORACLE.....	42
3.3.1 FONCTIONS À PARAMÈTRES DE TYPE NUMÉRIQUE.....	42
3.3.2 FONCTIONS À PARAMÈTRES DE TYPE CHAÎNE DE CARACTÈRES.....	43
3.3.3 FONCTIONS DE CONVERSION.....	44
3.3.4 FONCTIONS DE PARAMÈTRES DE TYPE DATE.....	44

3.3.5 FONCTIONS À PARAMÈTRES DE TYPE VARIABLE.....	45
3.3.6 FONCTIONS DIVERSES.....	46
4 APPROFONDISSEMENT DU LANGAGE SQL.....	47
4.1 INTRODUCTION.....	47
4.2 LA COMMANDE SELECT.....	51
4.2.1 REQUÊTE MULTI-TABLES.....	51
4.2.2 LA JOINTURE ENTRE DEUX TABLES.....	51
4.2.3 REQUÊTES IMBRIQUÉES.....	54
4.2.4 REQUÊTES CORRÉLÉES.....	57
4.2.5 CRITÈRES DE CHOIX POUR L'ÉCRITURE DU SELECT.....	57
4.3 EXTENSIONS DE SYNTAXE.....	58
4.4 GESTION DE L'ACCÈS CONCURRENT AUX DONNÉES.....	59
4.4.1 LES PRIVILÈGES D'ACCÈS AUX TABLES ET VUES.....	60
4.4.2 SUPPRESSION DES PRIVILÈGES ACCORDÉS.....	61
4.4.3 LES TRANSACTIONS.....	62
4.4.4 LECTURE COHÉRENTE.....	63
4.4.5 CONTRÔLE PAR DÉFAUT DES ACCÈS CONCURRENTS.....	64
4.4.6 CONTRÔLE EXPLICITE DES ACCÈS CONCURRENTS.....	65
4.5 GESTION DES PERFORMANCES.....	65
4.5.1 LES INDEX.....	65
4.5.2 CRÉATION D'UN INDEX.....	66
4.5.3 OBTENIR LA LISTE DES INDEX.....	66
4.5.4 ABANDON D'UN INDEX.....	67
4.5.5 CONSEILS.....	67
4.6 LE DICTIONNAIRE DES DONNÉES.....	67
5 CONCLUSION.....	70
ANNEXES.....	71
6 SQL AVEC ACCESS.....	72
6.1 CRÉER UNE BASE.....	72
6.2 GÉNÉRER DES REQUÊTES SQL.....	74
7 SOURCES DE DONNÉES ODBC.....	77
7.1 PILOTES ODBC.....	77
7.2 SOURCES DE DONNÉES ODBC.....	77
7.3 CRÉER UNE SOURCES DE DONNÉES ODBC.....	78
7.4 UTILISER UNE SOURCE DE DONNÉES ODBC.....	80
7.5 MICROSOFT QUERY.....	83
7.6 ÉCHANGER DES DONNÉES ENTRE SGBD COMPATIBLES ODBC.....	84
8 INSTALLATION D'ORACLE SOUS WINDOWS.....	92
8.1 OÙ TROUVER LE PRODUIT ?.....	92
8.2 L'INSTALLATION.....	92
8.3 LANCER LA BASE.....	94
8.4 UTILISER SQL PLUS.....	94
8.5 CRÉER DES COMPTES UTILISATEURS.....	95
8.6 LA FERMETURE DE LA BASE.....	96
9 INSTALLATION D'ORACLE SOUS LINUX.....	97
9.1 INSTALLATION D'ORACLE 8, VERSION 1.7.1 SUR LINUX RH 7.0.....	97
10 INSTALLER MYSQL SOUS LINUX.....	113
10.1 INTRODUCTION.....	113
10.2 INSTALLATION DE MYSQL.....	113
10.3 LANCEMENT DU SERVEUR MYSQLD.....	114
10.4 UTILISATION DU CLIENT MYSQL.....	114
10.5 POURSUIVRE.....	119
11 INSTALLATION DE MYSQL SOUS WINDOWS 9X.....	120

<u>11.1</u> OÙ TROUVER MySQL.....	120
<u>11.2</u> INSTALLATION DU PILOTE MyODBC.....	121
<u>11.3</u> INSTALLATION DE MySQL.....	123
<u>11.4</u> INSTALLATION DE L'ENVIRONNEMENT TCP-IP SUR UNE MACHINE WINDOWS.....	125
<u>11.5</u> LANCER ET TESTER MySQL.....	127
<u>11.6</u> CLIENT LIGNE MYSQLC.....	127
<u>11.7</u> ADMINISTRER LE SERVEUR MySQL.....	128