Oracle Database

Sections

Introduction

Présentation

Oracle Database est un système de gestion de base de données (SGBD) des plus employés au monde[1]. Il fût créé en 1979 et son langage de requête est nommé PL/SQL.

Installation d'Oracle

Oracle est disponible sur Windows et Linux, en plusieurs versions[2] :

- **Express Edition** (XE)Télécharger (http://www.oracle.com/technetwork/database/express-edition/downloads/index.html) : gratuite mais en version plus ancienne que la payante (ex : 11g au lieu de 12c en 2016), avec une limite de stockage quatre gigaoctets en monoprocesseur.
- **Standard Edition One** : sans limite de stockage, ni support des systèmes multiprocesseurs.
- **Standard Edition** (SE2) : gestion de clusters (*Oracle Real Application Clusters*, alias *Oracle RAC*).
- **Enterprise Edition** (EE) : aucune limite.
- **Personal Edition** (PE) : non disponible en version 12. Il s'agissait d'une sorte de EE mono-utilisateur.

Remarque : on peut télécharger et installer les versions payantes gratuitement, mais il faut décocher la case "Recevoir les mise à jour de sécurité".

Une fois téléchargé, il suffit de décompresser le ou les .zip, s'il y en a deux il faut les fusionner (dossier "database"), puis de lancer pour l'installer :

- Dans Linux, *runInstaller.sh*.
- Dans Windows, le *setup.exe*.

Script de lancement

> **Attention !**
>
> Oracle se lance ensuite automatiquement à chaque démarrage de la machine, ce qui la ralentit significativement.

Pour éviter cela :

- Dans Linux : voir /etc/init.d.
- Dans Windows : exécuter *services.msc*, puis passer les services *OracleServiceXE* et *OracleXETNSListener* en démarrage manuel. Ensuite pour lancer le service à souhait (en tant qu'administrateur), créer un script *Oracle.cmd* contenant les lignes suivantes :
  - Pour XE :

```
net start OracleServiceXE
net start OracleXETNSListener
pause
net stop OracleXETNSListener
net stop OracleServiceXE
```

  - Pour SE :

```
net start OracleServiceORCL
net start OracleDB12Home1TNSListener
pause
net stop OracleDB12Home1TNSListener
net stop OracleServiceORCL
```

Si le message "Accès refusé" survient, relancer le script avec un clic droit, en tant qu'administrateur.

Prérequis

Le serveur de base de données doit avoir au moins[3] :

1. 1 Go d'espace libre sur le disque dur pour XE, 3,5 pour SE.
2. 1 Go de RAM.
3. Windows, Linux, Oracle Solaris, ou IBM AIX.

Depuis la version 12c il faut obligatoirement un processeur 64 bits.

Références

1. http://db-engines.com/en/ranking
2. http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html
3. https://docs.oracle.com/html/B13614_01/preinsta.htm#i1067829

Voir aussi

- (anglais) Procédure d'installation (http://xanadu.cs.sjsu.edu/~tylin/classes/oracle10g_install/Oracle%20Documentation/Step0_OracleInstallation.pdf)
- (anglais) Cours officiels (https://apex.oracle.com/pls/otn/f?p=44785:24:0::::P24_CONTENT_ID,P24_PREV_PAGE:5922,24)
- (français) Tutoriels developpez.com (http://oracle.developpez.com/cours/)

Interfaces

SQL*Plus

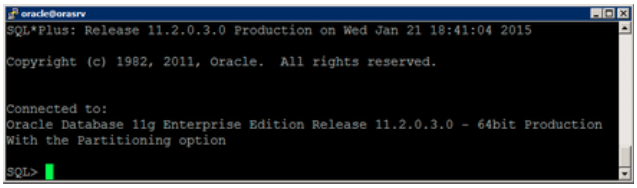
SQL*Plus est une interface de commandes fournie avec le SGBD. Sous Windows elle peut se lancer soit :

- Depuis le menu démarrer, répertoire Oracle, raccourci *Run SQL Command Line*.
- Via C:\oraclexe\app\oracle\product\11.2.0\server\bin\sqlplus.exe.
- Mais le mieux est de passer par la variable d'environnement en se connectant au SGBD en même temps. Par défaut cela se fait en console shell avec :

```
sqlplus / as sysdba
```

Sinon si vous avez déjà un compte, la syntaxe est :

```
sqlplus MonCompte/MonMotDePasse@localhost
```



La première étape est de créer un utilisateur (ex : *root*), l'autoriser à se connecter, puis lui conférer les droits administrateur (*sysdba*).

Sous 11g

```
CREATE USER root IDENTIFIED BY "MonMotDePasse";
GRANT create session to root;
GRANT sysdba to root;
```

Sous 12c

Avec la version 12c sont apparues les CDB et PDB. Ainsi la commande ci-dessus renvoie l'erreur *ORA-65096: nom utilisateur ou de rôle commun non valide*.

Il faut donc distinguer les comptes qui commencent par "c##" qui fonctionnent sur toute la CDB mais qu'il est impossible de promouvoir administrateur, de ceux propres à une PDB.

Dans ce cas nous allons utiliser le compte "sys" dont le mot de passe a été défini à l'installation.

Interface Web

Une deuxième interface est fournie avec le SGBD : l'interface web. On y accède soit :

- Pour SE 12c :
  - https://localhost:5500/em/shell
- Pour XE 11g :
  - Depuis le menu démarrer, répertoire Oracle, raccourci *Get Started*.
  - Par l'URL http://localhost:8080/apex/f?p=4950.
- Pour XE 10g :
  - Depuis le menu démarrer, *Database control*.
  - http://localhost:1158/em/console/logon/logon.

Ensuite il pour se connecter au SGBD :

- User name : *sys* (parfois *sysman* pour *system manager*)
- Password : celui fourni à l'installation.
- Connect as : *SYSDBA*.

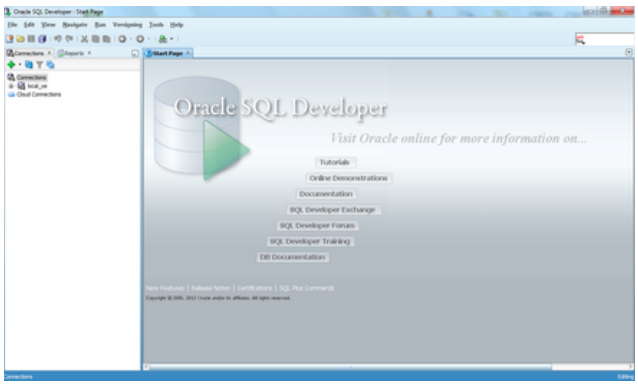
La console apparait alors, permettant de modifier la configuration de la base de données créée à l'installation (redémarrer le service, suivre l'architecture, les performances, gérer les sauvegardes...).

En cliquant sur *Application Express*, on peut éventuellement créer un nouvel utilisateur qui sera utilisé pour se connecter à Oracle. Une fois logué, celui-ci a accès à tous les outils de manipulation des données, par exemple *SQL Workshop\SQL Commands* pour entrer du code SQL.

> **Attention !**
>
> Si vous préférez créer un premier compte par les interfaces graphiques, il faudra tout de même utiliser la connexion par défaut SQL*Plus pour le `GRANT`.

Oracle SQL Developer

Oracle SQL Developer est un EDI conçu en Java. Il est fournit avec SE mais pour XE il faut télécharger son client lourd sur http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html.



Une fois installé et lancé, il faut lui configurer une connexion pour qu'il puisse s'authentifier auprès des bases de données Oracle. Remplir le compte créé par l'interface web, pour pouvoir accéder aux manipulations de données.

Pour commencer à exécuter du code, faire un clic droit sur la connexion, puis cliquer en bas sur *Ouvrir une feuille de calcul SQL*. Sinon il est possible de le travailler depuis un fichier (pour le sauvegarder) en cliquant en haut à gauche sur *Nouveau* (l'icône du plus ou CTRL + N), puis *Fichier de base de données*.

DBCA

Database Configuration Assistant (DBCA) est une interface graphique[1] disponible sur Windows ou les *nixes[2].

Hello world

Une fois l'une des consoles SQL vues précédemment lancée, il devient possible d'exécuter du PL/SQL (Procedural Language/Structured Query Language) : le langage procédural propriétaire créé par Oracle, et spécifique à sa base de données relationnelle.

```
set serveroutput on
BEGIN
  DBMS_OUTPUT.put_line ('Hello World!');
END;
/
```

Le slash dit au programme de stopper l'instruction multiligne.

Références

1. http://www.snapdba.com/2013/05/creating-an-oracle-11g-database-using-dbca-non-asm/
2. http://docs.oracle.com/cd/B16351_01/doc/server.102/b14196/install003.htm

Gestion des utilisateurs

Comptes

Création

```
CREATE USER UserTest IDENTIFIED BY MyComplexPassword DEFAULT TABLESPACE Wikibooks PASSWORD EXPIRE QUOTA UNLIMITED ON
Wikibooks;
GRANT CONNECT TO UserTest;
```

Sélection

```
SELECT * FROM ALL_USERS
```

Suppression

```
DROP USER UserTest;
```

Par ailleurs, il est possible de supprimer en cascade tous les objets associés à un utilisateur :

```
DROP USER UserTest CASCADE;
```

Rôles

Création

Un rôle peut en contenir plusieurs autres. Par exemple celui ci-après permet de se connecter en administrateur :

```
CREATE ROLE MyRole;
GRANT CONNECT TO MyRole;
GRANT DBA TO MyRole;
```

Il suffit ensuite de l'assigner à un compte utilisateur :

```
GRANT MyROLE TO UserTest;
```

Et il peut être révoqué ainsi :

```
REVOKE MyRole FROM UserTest;
```

Sélection

```
SELECT * FROM DBA_ROLES;
```

Modification

Par exemple pour changer de mot de passe :

```
ALTER ROLE MyRole IDENTIFIED BY MyPassword;
```

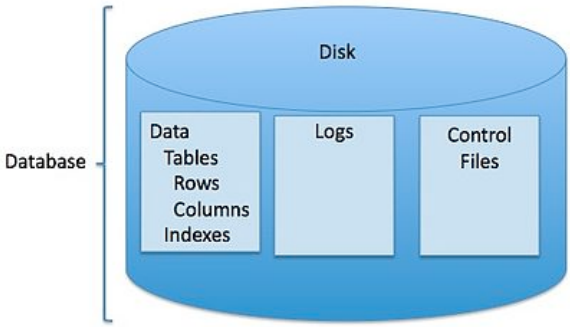
Suppression

```
DROP ROLE MyRole;
```

Tablespaces

Architecture

L'architecture Oracle ne comprend qu'une seule base par serveur[1], dans laquelle peuvent se trouver plusieurs tablespaces, équivalents des objets bases de données d'autres SGBD comme MySQL et MS-SQL, contenant des tables et procédures stockées.



Dans la version Express Windows, ces données sont stockées dans C:\oraclexe\app\oracle\oradata\XE\.

Les variables et mots-clés sont insensibles à la casse.

Créer des tablespaces

Une fois connecté, il est possible de commencer à créer directement des tables dans le tablespace par défaut (*Nom de connexion* Admin). Toutefois au préalable, on peut en créer d'autres dans des fichiers précis :

```
  CREATE TABLESPACE Wikibooks
  DATAFILE 'C:\oraclexe\app\oracle\oradata\XE\Wikibooks.dbf' size 10M reuse
  DEFAULT STORAGE (INITIAL 10K NEXT 50K MINEXTENTS 1 MAXEXTENTS 999)
  ONLINE;
```

```
Tablespace WIKIBOOKS créé(e).
```

Pour lister ceux qui existent :

```
select tablespace_name, file_name, bytes
from dba_data_files;
```

Il y en a plusieurs par défaut :

- SYSTEM : les objets du système.
- SYSAUX : depuis la version 10g, tablespace auxiliaire du précédent pour certains objets du système (ex : Statspack, Advisor, Scheduler).
- TEMP : tables temporaires pour les tris.
- UNDO : depuis la 9i, utilisé pour les transactions (commit, rollback)
- USERS : c'est le tablespace où sont créés les objets des utilisateurs par défaut.

Supprimer des tablespaces

```
DROP TABLESPACE Wikibooks;
```

```
Tablespace WIKIBOOKS supprimé(e).
```

> **Attention !**
>
> Le fichier .dbf ne disparait pas avec cette commande.

Références

1. http://www.tuto-dba-oracle.com/serveur-oracle.html

Tables

Lister les tables

Pour obtenir la liste des tables du tablespace courant :

```
SELECT owner, table_name FROM all_tables;
```

Créer des tables

En reprenant l'exemple du livre SQL :

```
CREATE TABLE client1 (nom VARCHAR(10), prenom VARCHAR(10), adresse VARCHAR(20) );
```

```
Table créée.
```

En passant par un clic droit sur les tables, *Nouvelle table...*, SQL Developer permet de générer puis exécuter cette démarche :

```
CREATE TABLE client1
( id INT NOT NULL
, nom VARCHAR2(50)
, prenom VARCHAR2(50)
, adresse VARCHAR2(255)
, CONSTRAINT client1_PK PRIMARY KEY (ID) ENABLE
) TABLESPACE Wikibooks;
```

Dans cette commande, on a précisé le tablespace dans lequel rattacher la nouvelle table avec le mot clé `TABLESPACE` dans la clause de création. Mais SQL Developer le permet en le sélectionnant dans l'interface graphique.

Modifier la structure des tables

Exemple de renommage :

```
ALTER TABLE client1 RENAME to client2
```

Ajout d'une contrainte sur les valeurs du premier champ :

```
ALTER TABLE client1 CHECK id > 1;
```

Ajout d'une clé primaire :

```
ALTER TABLE client1 ADD CONSTRAINT client1_pk PRIMARY KEY (id);
```

Retrait d'une clé primaire :

```
ALTER TABLE client1 ADD PRIMARY KEY (id) DISABLE;
```

Ajout d'une clé étrangère :

```
ALTER TABLE client1
ADD CONSTRAINT fk_client2
  FOREIGN KEY (client2_id)
  REFERENCES client2(id);
```

Supprimer des tables

```
DROP TABLE client1;
```

Insérer des lignes

```
INSERT INTO client1 (id, nom, prenom, adresse) VALUES (1, 'Croche', 'Sarah', 'Petaouchnoc');
```

```
1 ligne inséré [sic].
```

Plusieurs lignes :

```
INSERT ALL
 INTO client1 (id, nom, prenom, adresse) VALUES (2, 'Pelle', 'Sarah', 'Clochemerle')
 INTO client1 (id, nom, prenom, adresse) VALUES (3, 'Porte', 'Sarah', 'Cuges-les-Bains')
 SELECT 1 FROM DUAL;
```

```
2 lignes inséré [sic].
```

Table crée

ID	PRENOM	NOM	ADRESSE
1	Sarah	Croche	Petaouchnoc
2	Sarah	Pelle	Clochemerle
3	Sarah	Porte	Cuges-les-Bains

Lire une table

Pour accéder à sa structure :

```
desc client1;
```

```
Nom     NULL     Type
------- -------- -------------
ID      NOT NULL NUMBER(38)
NOM              VARCHAR2(10)
PRENOM           VARCHAR2(10)
ADRESSE          VARCHAR2(20)
```

> **Attention !**
>
> Si la table n'existe pas, l'erreur qui apparait est `ORA-00923: FROM keyword not found where expected`.

Pour son contenu :

```
SELECT * from client1;
```

```
ID   NOM        PRENOM     ADRESSE
-    ---------- ---------- --------------------
1    Croche      Sarah     Petaouchnoc
2    Pelle       Sarah     Clochemerle
```

Le nombre de tirets correspond à la taille du champ.

Mettre à jour des lignes

```
UPDATE client1 SET adresse = 'Cuges-les-Bains' WHERE id = 1;
```

Supprimer des lignes

```
DELETE client1 WHERE ID = 2;
```

Partitionner une table

Le partitionnement Oracle sert à diviser les données d'une table volumineuse dans plusieurs plus petites afin d'en augmenter les performances.

Range

Exemple :

```
CREATE TABLE t_range
( t1      VARCHAR2(10) NOT NULL,
  t2      NUMBER       NOT NULL,
  t3      NUMBER
)
PARTITION BY RANGE (t2)
( PARTITION part1 VALUES LESS THAN (1),
  PARTITION part2 VALUES LESS THAN (11),
  PARTITION part3 VALUES LESS THAN (MAXVALUE)
);
```

Hash

Exemple :

```
CREATE TABLE t_hash
( t1      VARCHAR2(10) NOT NULL,
  t2      NUMBER       NOT NULL,
  t3      NUMBER
PARTITION BY HASH (t2)
PARTITIONS 4
;
```

List

Exemple :

```
CREATE TABLE t_list
( ort     VARCHAR2(30) NOT NULL,
  t2      NUMBER,
  t3      NUMBER
)
PARTITION BY LIST(ort)
( PARTITION part_nord VALUES IN ('Hamburg','Berlin'),
  PARTITION part_sued VALUES IN ('Muenchen', 'Nuernberg'),
  PARTITION part_west VALUES IN ('Koeln','Duesseldorf'),
  PARTITION part_ost VALUES IN ('Halle'),
  PARTITION part_def VALUES (DEFAULT)
);
```

Interval

Exemple :

```
CREATE TABLE t_interval
( buchungs_datum  DATE NOT NULL,
  buchungs_text   VARCHAR2(100),
  betrag          NUMBER(10,2)
)
PARTITION BY RANGE (buchungs_datum)
INTERVAL(NUMTOYMINTERVAL(1, 'MONTH'))
( PARTITION p_historie VALUES LESS THAN (TO_DATE('2014.01.01', 'YYYY.MM.DD')),
  PARTITION p_2014_01  VALUES LESS THAN (TO_DATE('2014.02.01', 'YYYY.MM.DD')),
  PARTITION p_2014_02  VALUES LESS THAN (TO_DATE('2014.03.01', 'YYYY.MM.DD'))
);
```

Schémas

Un schéma est un ensemble de permissions[1] pour des tables ou procédures stockées. Le mot clé `AUTHORIZATION` permet de donner des droits à un utilisateur :

```
CREATE SCHEMA AUTHORIZATION root
  CREATE TABLE table1...
```

```
  CREATE TABLE table2...
;
```

Synonymes

Un synonyme est un alias d'un nom de table (ou d'autres objets). Il peut être utilisé pour la sécurité en masquant le nom du propriétaire de l'objet vers lequel il pointe, ou bien pour uniformiser les noms d'objets distants dans les bases de données distribuées[2].

```
CREATE SYNONYM table1 FOR client1;
SELECT * from table1;
```

Références

1. https://docs.oracle.com/cd/B12037_01/server.101/b10759/statements_6013.htm
2. https://docs.oracle.com/cd/B28359_01/server.111/b28310/views003.htm

Vues

Principe

Une vue est une table virtuelle, utilisée pour afficher le résultat d'une requête fréquente, selon des permissions précises (ex : afficher des colonnes seulement pour un utilisateur).

Créer une vue

La vue suivante n'a pas grand intérêt car elle est redondante d'une table :

```
CREATE OR REPLACE VIEW MaVue1 AS
    SELECT *
    FROM client1
```

Actualiser une vue

Les enregistrements de la vue s'actualisent automatiquement, mais si la structure d'une des tables appelées par la vue change, cette dernière renvoie une erreur. Il faut donc la mettre à jour avec :

```
ALTER VIEW MaVue1 COMPILE
```

Lister les vues

```
SELECT * FROM USER_VIEWS
SELECT * FROM ALL_VIEWS
SELECT * FROM DBA_VIEWS
```

La colonne TEXT affiche la requête SQL générant le vue.

Vues temporaires

Depuis la version 10g, il est possible de définir une vue temporaire à la volée au milieu d'un script :

Exemple :

```
WITH PremieresLignes AS
(
 SELECT id
 FROM client1
 WHERE id < 10
)

SELECT prenom, nom
FROM PremieresLignes, client1
WHERE PremieresLignes.id = client1.pl_id
```

Vues matérialisées

Les vues matérialisées sont des vues figées, c'est-à-dire le résultat d'une requête à un certain moment[1].

Créer des vues matérialisées

Création basique :

```
CREATE MATERIALIZED VIEW MV1
AS SELECT * FROM client1
```

Avec durée de rafraichissement quotidienne :

```
CREATE MATERIALIZED VIEW MV2
REFRESH FAST
START WITH SYSDATE
NEXT SYSDATE + 1
AS SELECT * FROM client1;
```

En effet, `select SYSDATE from DUAL` donne la date du jour.

Lire des vues matérialisées

```
SELECT QUERY FROM ALL_MVIEWS
WHERE MVIEW_NAME='MV1'
```

Références

1. https://docs.oracle.com/cd/B10501_01/server.920/a96567/repmview.htm

Index

Principe

Un index sert à trouver les enregistrements résultats d'une requête plus rapidement que par un parcours séquentiel.

Création

```
CREATE INDEX MonIndex1 ON client1(nom);
```

Modification

```
ALTER INDEX MonIndex1 REBUILD;
```

Suppression

```
DROP INDEX MonIndex1;
```

Lister les index

```
SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS FROM USER_INDEXES;
```

Procédures stockées

Création

Une procédure stockée est un script enregistré dans la base de données sous un nom par lequel on peut l'exécuter depuis d'autres.

Création

```
create or replace procedure nomProc(
    param1 IN date,
    param2 IN OUT date
    param3 OUT date)
is
-- declare
begin
--Instructions
end;
```

Appel

```
EXECUTE nomProc;
```

ou

```
BEGIN
 nomProc;
END;
```

Exemples

Cette section est vide, pas assez détaillée ou incomplète.

Fonctions

Création

Une fonction doit impérativement renvoyer quelque chose, par une clause RETURN.

```
create or replace function loginExist(Param1 tableName.champName%type)
return boolean
is
-- declare
	retVal boolean := true;
begin
--Instructions
	return retVal;
end;
```

Cette section est vide, pas assez détaillée ou incomplète.

Packages

Principe

Un package Oracle est un schéma regroupant divers objets tels que des types et des sous-programmes[1]. Ces modules peuvent ensuite être appelés par différents scripts.

```
CREATE OR REPLACE PACKAGE TEST_PACKAGE AS
  PROCEDURE xy;
  FUNCTION abc(p_var VARCHAR2);
  TYPE noms AS OBJECT(nom NVARCHAR2(200), prenom NVARCHAR2(200));
END TEST_PACKAGE;
```

Appel :

```
CALL TEST_PACKAGE.xy;
```

L'avantage est que si un jour on décide d'augmenter la taille des noms de famille à 1 000 caractères, il suffit de le faire dans un seul package au lieu de parcourir toutes les fonctions, procédures et triggers.

Exemple

Cette section est vide, pas assez détaillée ou incomplète.

Références

1. https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/09_packs.htm

Trigger

Syntaxe

```
create or replace trigger <triggername>
before/after insert or update or delete
on <tablename>
REFERENCING NEW AS <newROW> OLD AS <oldROW>
for each row/for each statement
when (<condition>)
DECLARE
   -- Déclaration des variables
BEGIN
   if INSERTING then
    ...
   end if;
   if UPDATING then
    ...
   end if;
   if DELETING then
    ...
   end if;
EXCEPTION
   -- Traitement en cas d'exception
END <triggername>;
```

Désignations[1] :

- `:new` : nouvelle ligne
- `:old` : ancienne ligne
- `:parent` : table parente

Exemple

Cette section est vide, pas assez détaillée ou incomplète.

Références

1. http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/triggers.htm

Séquences

Syntaxe

Une séquence est un compteur prédéfini, utilisé par exemple pour incrémenter les clés uniques.

Cela évite d'insérer des lignes en déterminant la valeur d'un ID avec la couteuse clause `select max(id)+1 from client1`.

```
CREATE SEQUENCE SEQUENCE_NAME
INCREMENT BY 1
START WITH 1
MINVALUE 1
MAXVALUE 999999
NOCYCLE / CYCLE -- Un cycle redémarre le compteur en boucle
CACHE 20
NOORDER;
```

Pour l'utiliser, on a le choix de la valeur courante ou suivante :

```
select SEQUENCE_NAME.currval from dual
select SEQUENCE_NAME.nextval from dual
```

Voir les séquences

```
select * from user_sequences;
```

Exemples

Cette section est vide, pas assez détaillée ou incomplète.

PL/SQL

PL/SQL pour *Procedural Language SQL* est un langage de programmation d'Oracle, spécialisé dans l'accès à ses bases de données. C'est donc une extension de la norme SQL.

Structure d'un programme

Le code PL/SQL est structuré en "blocs" dont la structure générale est la suivante :

```
DECLARE
    -- Partie ou l'on déclare les constantes, les variables et les curseurs
BEGIN
    -- Corps du programme
EXCEPTION
    -- Traitement des exceptions
END
```

Commentaires

```
/* Exemple de commentaire PL/SQL
   Cette forme de commentaire peut prendre plusieurs lignes
 */
-- Deuxième exemple, cette forme de commentaire est limitée à une ligne
rem Troisième exemple (sous Windows)
```

Fonctions natives

PL/SQL offre plusieurs packages prédéfini :

- DBMS_OUTPUT : utilisé pour afficher des messages pendant l'exécution du code
- DBMS_JOB : permet de lancer du code en tâche de fond
- DBMS_XPLAN : permet d'obtenir le plan d'exécution d'une commande SQL
- DBMS_SESSION
- DBMS_METADATA
- UTL_FILE : permet de gérer les fichiers sur le disque, en dehors de la base
- UTL_HTTP
- UTL_SMTP

Types du langage

Types natifs

Les principaux types de donnée en PL/SQL sont : `NUMBER`, `INTEGER`, `CHAR` (chaine de longueur fixe), `VARCHAR2` (chaine de longueur variable), `DATE`, `TIMESTAMP`, `TEXT`, etc.

Le symbole « := » est utilisé pour l'affectation d'une valeur.

Les types de chaînes de caractères, il existe plusieurs familles de types de textes :

codage	nombre fixé de caractères	nombre variable de caractères
caractère	CHAR	VARCHAR2
caractère Unicode (cf À la découverte d'Unicode)	NCHAR	NVARCHAR2

Créer un type

```
CREATE TYPE ...
```

Variables Composées

Tableaux

```
DECLARE
    -- Type Tableau de chaines de 20 caractères maxi
    TYPE nom_type_tableau is TABLE OF VARCHAR(20) INDEX BY BINARY
        TYPE nom_type_tableau2d is TABLE OF nom_type_tableau INDEX BY BINARY
    -- Déclaration de la variable tab de type nom_type_tableau
    tab    nom_type_tableau ;
        matrice  nom_type_tableau2d;
    -- Variable tab2 de type nom_type_tableau initialisée avec des valeurs
    tab2 nom_type_tableau := nom_type_tableau('l1', 'l2',…) ;
BEGIN
    tab(1) := 'ligne 1' ;
    tab(2) := 'ligne 2' ;
    -- Affichage du premier élément de la variable tab, c'est à dire "ligne 1"
    DBMS_OUTPUT.put_ligne(tab(1)) ;
END ;
```

```
DECLARE
    TYPE nom_type_tableau IS VARRAY(2) OF VARCHAR2(30) ;
    tab nom_type_tableau := nom_type_tableau('l1','l2') ;
BEGIN
    DBMS_OUTPUT.put_ligne(tab(1)) ;
END ;
```

Méthodes disponibles pour/avec les variables tableau

- Tab.first
- Tab.count
- Tab.next(indice)
- Tab.last
- Tab.prior(indice)
- Tab.delete(indice)

Enregistrements

```
DECLARE
    -- Un RECORD est une variable structurée, comme une struct en C
    TYPE time_type IS RECORD
    (
        min SMALLINT,
        heure SMALLINT
    ) ;
    -- Déclararion de la variable Temps
    Temps time_type ;
BEGIN
    Temps.min := 30 ;
    Temps.heure :=13 ;
END ;
```

Structures de contrôle

Conditions

```
DECLARE
    -- Déclaration de la variable emp_rec qui sera structurée avec les mêmes champs
    -- qu'un enregistrement (%ROWTYPE) de la table ''employe''
    Emp_rec  employe%ROWTYPE ;
BEGIN
    -- Il est possible de récupérer le résultat du SELECT directement dans la variable
    -- La valeur &temp sera demandée à l'utilisateur lors de l'exécution du script dans
    -- l'outil Sql*Plus d'Oracle
    SELECT * INTO emp_rec
    FROM employe
    WHERE emp_num = &temp ;
```

```
    DBMS_output.put_line (emp_rec.nom) ;
END ;
```

```
DECLARE
    Age NUMBER(3) := &temp ;
BEGIN
    IF age < 18 THEN
        DBMS_OUTPUT.put_line('7') ;
    ELSIF age > 65 THEN
        DBMS_OUTPUT.put_line('6,5') ;
ELSE
END IF ;
END
```

Boucles

FOR

```
DECLARE
    NUM NUMBER(2) := 0
BEGIN
    FOR num IN 0..10
LOOP
        DBMS_OUTPUT.put_line(to_char(num)) ;
END LOOP ;
END ;
```

WHILE

```
LOOP
    DBMS_OUTPUT.put_line(to_char(num)) ;
    Num := num+1 ;
    EXIT WHEN num = 10 ;  // if num = 10 THEN EXIT ;
END LOOP ;
    WHILE num < 11 AND (cool OR pascool)
        LOOP
            Instructions …
END LOOP ;
```

Exemples

- Exemples 1

Mettre la date à jour dans la db

```
DECLARE
    TYPE DATE IS RECORD (
        Jour NUMBER(2),
        Mois NUMBER(2),
        Annees NUMBER(4));
TYPE DATE_SEVEN VARRAY(7) OF DATE
    DATE DATE_SEVEN
BEGIN
    DATE_NOW(1).Jour   := &temp1 ;
    DATE_NOW(1).Mois   := &temp2 ;
    DATE_NOW(1).Annees := &temp3 ;
```

- Exemples 2

Créer un record qui contient matricule, nom, prénom

```
DECLARE
    TYPE eleves IS RECORD(
        Matricule number(10),
        Nom varchar(20),
        Prenom varchar(20) )
    TYPE LesEleves TABLE OF eleves INDEX BY BINARY INTEGER ;
```

```
    Tab LesEleves ;
BEGIN
    Tab(1).Matricule := 001 ;
    Tab(1).Nom       := 'Bonjean' ;
    Tab(1).Prenom    := 'Simon' ;
```

- Exemples 3

Chercher dans une table ListeDeCourse

```
DECLARE
    TYPE Course IS RECORD(
        NumArt number(10),
        Prix varchar(20),
        NomArt varchar(20) )
    TYPE ListeDeCourses TABLE OF Course INDEX BY BINARY INTEGER ;
    LesCourses ListeDeCourse;
BEGIN
    SELECT *
INTO LesCourses(1)
FROM tCourse
    WHERE numArticle =1
```

- Exemples 4

Relevé météo

```
DECLARE
    TYPE MeteoType IS RECORD(
        temp number(4,2),
        tx varchar(2),
        lieu varchar(10) )
    TYPE tabMeteoTypeTABLE OF MeteoType INDEX BY BINARY INTEGER ;
    tabMeteo tabMeteoType;
BEGIN
    DBMS_output.putline(tabMeteo.first.lieu.tochar) ;
    DBMS_output.putline(tabMeteo.last.temp.tochar) ;
```

- Exemples 5

Créer un tableau de 10 nombres

```
DECLARE
    TYPE unAdix varray(10)
    Tab unAdix := unAdix(1,2,3,4,5,6,7,8,9,10) ;
    Compteur number(2) ;
BEGIN
    FOR Compteur in 1..10
    LOOP
        IF MOD(tab(Compteur), 2) =0 THEN
            DBMS_output.putline('C est pair') ;
        ELSE
            DBMS_output.putline('C est pas pair') ;
        END IF ;
    END LOOP
END ;
```

- Exemples 6

Augmenter de 10 % tous les logiciels Photoshop

```
DECLARE
    Intitule Logiciel.nom % type := &temp ;
BEGIN
    IF intitule = 'photoshop' THEN
        UPDATE logiciel
        SET prix = prix * 1,1
        WHERE  nom = intitule ;
    END IF ;
END;
```

- Exemples 7

Insérer un élément dans la table locaux

```
DECLARE
```

```
localType is record (n° number(1), etage varchar2(4), type varchar2(10)) ;
     // locate locaux % ROWTYPE;
locate localType ;
BEGIN
    Locate.n° := 4 ;
    Locate.etage := '2eme' ;
    Locate.type := 'linux' ;
    INSERT INTO locaux VALUES (locate.n°,locate.etage, locate.type) ;
END ;
```

- Exemples 8

Vérifie si le prix du logiciel encodé est supérieur à la moyenne

```
DECLARE
Log logiciel % ROWTYPE
Intitule Logiciel.nom % type := &temp ;
Prix2    Logiciel.prix % type ;
Moyenne Logiciel.prix % type ;
BEGIN
    SELECT prix INTO Prix2
FROM Logiciel
    WHERE nom = Intitule;
    SELECT avg(prix) INTO Moyenne
    FROM Logiciel ;
    IF prix2 > Moyenne THEN
    UPDATE Logiciel set prix = prix -100 ;
    WHERE Logiciel.nom = Intitule ;
END IF
END ;
```

Curseurs

- Explicite :

Un curseur explicite est un curseur déclaré explicitement avec le mot-clef CURSOR dans le bloc PLSQL.

```
DECLARE
  CURSOR c1 IS SELECT nom FROM EMP;
  nomEmp EMP.nom %type;
BEGIN
  OPEN c1;
  FETCH c1 INTO nomEmp;
  dbms_output.putline(nomEmp);
  --FETCH c1 INTO nomEmp;
  CLOSE c1;
END;
```

- Implicite :

Un curseur implicite est un curseur généré automatiquement par Oracle pour une commande SQL incluse généralement dans un bloc PLSQL.

```
UPDATE EMP SET sol = sol *1.1;
```

```
SELECT SUM(sol) INTO total FROM EMP WHERE deptNo = 10;
```

- Exemples :

```
DECLARE
  bonus NUMBER(8,8) := 1000;
  CURSOR sol_cur IS SELECT sol, sol + bonus nouveauSol FROM emp
  WHERE dateEmbauche < SYSDATE;
BEGIN;
```

- Explicite :

```
DECLARE
  CURSOR salleCur IS SELECT * FROM SALLE
  maSalle SALLE%ROWTYPE;
BEGIN
```

```
  FOR maSalle IN salleCur
  LOOP
     dbms_output.putline(maSalle.nSalle);
  END LOOP;
END;
```

- Implicite :

```
DECLARE
  maSalle SALLE%ROWTYPE;
BEGIN
  FOR maSalle IN (SELECT * FROM SALLE)
  LOOP
     dbms_output.putline(maSalle.nSalle);
  END LOOP;
END;
```

Exceptions

Exception prédéfinie

- Exemples 1

```
declare
begin
    insert into pilote values(1, 'CHARLIE', 'PARIS', 07);
exception
    when dup_val_on_index
    then dbms_output.put_line('Doublon');
end;
```

- Exemples 2

Cherche l'employé n°555 et prévois le cas ou il n'existe pas.

```
declare
    employe_rec emp%rowtype;
begin
    select * into employe_rec from emp
    where emp = 555;
exception
    when no_data_found  then dbms_output.put_line('Donnée non trouvée');
    when others         then null;
end;
```

- Exemples 3

demander un nom à l'utilisateur et prévoir le cas où il inscrit trop de lettres

```
declare
    nom varchar2(5) := '&temp';
begin
    dbms_output.put_line(nom);
exception
    when value_error then dbms_output.put_line('chaine de caractères trop longue');
end;
```

Types d'exceptions prédéfinie

- invalid_cursor
- invalid_number
- no_data_found
- too_many_rows
- value_error
- zero_divide
- dup_val_on_index

Exceptions personnalisées

- Exemple 1 :

```
declare
    joueur_max exception;
    temp number(3);
begin
    select count(*) into temp from joueur
    if (temp = 100) then
        raise joueur_max;
    end if;
    insert into joueur values (1, 'test', 'test');
exception
    when dup_val_on_index   then dbms_output.put_line('Le joueur existe déjà');
    when joueur_max     then dbms_output.put_line('Nombre de joueurs max atteint');
end;
```

Oracle ignore le type booléen

Oracle ne respecte pas totalement SQL:1999 et ne dispose pas de type booléen. Ce type doit être recréé suivant deux stratégies, dont aucune ne prévaut véritablement sur l'autre.

Utilisation de CHAR

La première façon d'émuler le type booléen est d'utiliser les CHAR, notamment les char de dimension 1 : CHAR(1) 'Y'/'N'

L'inconvénient de la méthode est qu'il rend la notion dépendante de la langue utilisée. Et cela peut devenir perturbant pour un développeur récupérant un travail fait dans un autre pays.

Utilisation de NUMBER

La deuxième façon d'émuler le type booléen est d'utiliser les NUMBER, de dimension 1 : NUMBER(1) 0/1

Cette méthode n'est pas toujours sans inconvénient. Les utilisateurs de Visual Basic, par exemple, sont habitués à donner à True la valeur -1 !

Références

- http://www.developpez.net/forums/d165180/bases-donnees/oracle/oracle-8i-type-boolean/
- https://docs.oracle.com/cd/B19306_01/olap.102/b14346/dml_datatypes004.htm#CJACGBGE
- http://stackoverflow.com/questions/3726758/is-there-a-boolean-type-in-oracle-databases

Oracle ignore l'autoincrement

La fonction AUTO_INCREMENT de MySQL est très pratique pour déléguer le calcul d'un nouvel identifiant (entier) d'une ligne au système. Malheureusement, cette fonction n'existe pas pour Oracle et il faut donc passer par la création d'une séquence pour obtenir le même résultat[1].

Exemple pratique

La requête suivante illustre la création d'une table munie de l'équivalent de la fonction d'auto-incrémentation: Elle créé une sequence nommée "SQ_FOOTBALL_MATCH", commençant à 1, incrémentée automatiquement par pas de 1. Aucune valeur ne sera stockée en cache.

```
--Exemple d'autoincrement Oracle

--drop table FOOTBALL_MATCH


create table FOOTBALL_MATCH
(
  MATCH_ID NUMBER not null,
  LOCAL_ID NUMBER not null,
  VISITEUR_ID NUMBER not null,

  SCORE_LOCAL NUMBER not null,
  SCORE_VISITEUR NUMBER not null,

  --Etc... on pourrait enregistrer des informations sur l'arbitre, la date, le nombre de spectateurs

  COMMENTAIRE VARCHAR2(100)
)
;

CREATE SEQUENCE "SQ_FOOTBALL_MATCH" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START
WITH 1 NOCACHE NOORDER NOCYCLE ;

--PK
alter table FOOTBALL_MATCH add constraint PK_FOOTBALL_MATCH primary key (MATCH_ID);

-- Décommenter pour FK => D'abord créer la table pointée EQUIPE
--alter table FOOTBALL_MATCH add constraint FK_LOCAL_ID foreign key (LOCAL_ID) references EQUIPE (EQUIPE_ID);
--alter table FOOTBALL_MATCH add constraint FK_VISITEUR_ID foreign key (VISITEUR_ID) references EQUIPE (EQUIPE_ID);

COMMENT ON COLUMN FOOTBALL_MATCH.MATCH_ID IS 'Cle primaire';
COMMENT ON COLUMN FOOTBALL_MATCH.LOCAL_ID IS 'Equipe jouant à domicile';
COMMENT ON COLUMN FOOTBALL_MATCH.VISITEUR_ID IS 'Equipe visiteuse';

-- Décommenter pour Grant (le profil doit exister)
--grant select on FOOTBALL_MATCH to PROFIL_UTILISATEUR;
```

Pour insérer une nouvelle ligne dans la table ainsi créée:

```
--Exemple d'INSERT

INSERT INTO FOOTBALL_MATCH(MATCH_ID,LOCAL_ID,VISITEUR_ID,SCORE_LOCAL,SCORE_VISITEUR)
VALUES ("SQ_FOOTBALL_MATCH".nextval -- Utilisation de la séquence
,630,599 -- ID des équipes
,1,0) -- Score
```

Cette instruction va récupérer la prochaine valeur fournie par la séquence "SQ_FOOTBALL_MATCH" et proposer celle-ci comme valeur MATCH_ID de la ligne

insérée.

Quelques requêtes utiles/Dictionnaire de données

Le dictionnaire de données Oracle est un ensemble de tables système, qui contiennent les informations de fonctionnement de la base de données comme :

- Les utilisateurs
- Les tables
- Les contraintes d'intégrité
- etc.

Ces informations sur les informations sont appelées méta données.

Les informations du dictionnaire de données sont consultables par l'administrateur SYSTEM. De nombreuses vues permettent d'accéder à des contenus spécifiques (comme ceux énumérés précédemment)[2] [3].

Afficher toutes les vues du dictionnaire

La requête suivante, adaptable en commentant/décommentant les éléments souhaités, permet de lister divers éléments du dictionnaire

```
SELECT *

FROM dict

WHERE table_name LIKE
'%USER_%' --vues user
--'%DBA_%' -- 7 vues dba
--'C%' -- 10 vues débutent par la lettre C
--'G%' -- 492 vues débutent par la lettre G
--'V%' -- 618 vues débutent par la lettre V
--'V%' and comments not like 'Syno%' -- Toutes les vues en V sont des Synonymes
--1830 vues en tout

ORDER BY TABLE_NAME
```

Afficher quelques vues dynamiques utiles au DBA

La requête suivante, adaptable en commentant/décommentant les éléments souhaités, permet d'obtenir quelques informations et statistiques

```
-- Vues dynamiques utiles au DBA

SELECT *

FROM
-- V$VERSION -- Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
--V$DATABASE -- DBID NAME CREATED RESETLOGS_CHANG etc...
--V$SESSION -- 30 sessions ouvertes, multiples informations comme la machine etc...
--V$LOCK -- Infos sur les verrous
--V$SGA -- Fixed Size 2230768, Variable Size 1275069968, Database Buffers 1912602624, Redo Buffers 16932864
--V$SQL -- non accessible
V$SYSSTAT -- 628 statistiques
```

Afficher toutes les tables dynamiques

Les tables dynamiques correspondent à des zones de la mémoire SGA lorsqu'Oracle est en activité (elles disparaissent à l'arrêt de la base)

```
-- Les tables dynamiques correspondent à des zones de la mémoire SGA lorsqu'Oracle est en activité (elles disparaissent à l'arrêt de la base)
-- La vue v$fixed_table permet d'avoir toutes les tables et les vues PUREMENT dynamiques
--
-- DROITS REQUIS: SYS

SELECT *

FROM v$fixed_table

WHERE type='TABLE';
```

Travailler sur les méta-informations des tables

Le dictionnaire de données permet de collecter des informations par introspectif.

Classer les tables par nombre croissant de lignes

```
-- Classement des tables de la moins chargée en ligne à celle comptant le plus d'enregistrements

SELECT
table_name, num_rows

FROM user_tables

ORDER BY num_rows
```

Classer les tables par nombre croissant de colonnes

```
-- Classement des tables par nombre croissant de colonnes

SELECT
table_name,
count(*) as Nb_Cols

FROM user_tab_columns

WHERE
--table_name = 'POOL'
--table_name like 'FOOTBALL_CLUB_%'

GROUP BY table_name

ORDER BY Nb_Cols
```

Lister les colonnes d'une table

```
-- Liste des colonnes de la table

SELECT
column_name
--count(column_name)
```

```
FROM user_tab_columns

WHERE
table_name = 'MATCH'
--'REFEREE'
--'PLAYER'

--table_name like 'POOL%'
--table_name like 'FOOTBALL_CLUB_%'
--GROUP BY table_name
```

Quelques requêtes utiles/Paramètres système

Il est possible de lire (et même modifier) des paramètres système par simple requêtage.

Afficher les informations de version

Pour savoir quelles versions (et composants) on utilise.

```
SELECT * FROM PRODUCT_COMPONENT_VERSION;
```

Afficher les paramètres système modifiables dans la session

La requête suivante, adaptable en commentant/décommentant les éléments souhaités, permet de lister divers paramètres système et d'ordonner ceux-ci selon qu'ils soient modifiables ou pas.

```
--Paramètres système modifiables dans la session

SELECT
p.name, p.isses_modifiable, p.issys_modifiable, p.display_value

FROM v$system_parameter p

WHERE p.isses_modifiable = 'TRUE' --Oracle ne comprend pas le type booléen: il s'agit en fait d'un string !!
--WHERE name like '%param%'
--WHERE name like '%'||lower('&param')||'%';

ORDER BY p.issys_modifiable
```

Quelques requêtes utiles/Utiliser les dates

L'utilisation des dates dans une base de données réserve toujours son lot de surprises.

D'une manière générale, l'utilisation de l'égalité est une mauvaise idée pour les dates, celle-ci étant rarement vérifiée du fait de la précision des dates qui excède celle des usages courants. Mieux vaut utiliser les opérateurs de comparaison < et > et définir des intervalles.

Exemple basique de requête utilisant les dates

La requête suivante illustre l'utilisation d'une date.

```
--Exemple basique de requête utilisant les dates
SELECT
*

FROM MATCH m

WHERE m.date>=TO_DATE('31-dec-2002','dd-MON-yyyy')
--WHERE m.date>=TO_DATE('31/12/2002','DD/MM/YYYY')
```

Quelques requêtes utiles/Modifier une ligne

La mise à jour d'un enregistrement (ou plusieurs) se fait par l'intermédiaire de l'expression UPDATE.

Exemple de mise à jour

La requête suivante illustre la modification d'une ligne (A noter aussi l'utilisation d'une double concaténation).

```
--Exemple de mise à jour

UPDATE match

SET played = 'YES',
   score = (SELECT CONCAT(s.points1, CONCAT(' - ', s.points2))
         FROM score s
         WHERE s.matchId = 123)

WHERE id = 123;
```

Références

1. http://www.w3schools.com/sql/sql_autoincrement.asp
2. http://www.commentcamarche.net/contents/700-oracle-le-dictionnaire-de-donnees
3. http://didier.deleglise.free.fr/dba/dictionnaire/dict_main.htm
   - http://www.techonthenet.com/oracle/update.php
   - https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions026.htm

Quelques requêtes utiles/Analyse d'une table

Quelques fonctions utiles pour analyser les données d'une table.

Nombre de valeurs distinctes par colonnes

La requête suivante permet de déterminer le nombre de valeurs distinctes pour chaque colonne : Les ";" en fin de titre de colonne permettent une segmentation plus facile.

```
--Nombre de valeurs distinctes par colonnes

SELECT
COUNT(DISTINCT(taille)) "taille;"
,COUNT(DISTINCT(poids)) "poids;"
,COUNT(DISTINCT(equipe)) "equipe;"
,COUNT(DISTINCT(salaire)) "salaire;"
,COUNT(DISTINCT(nationalite)) "nationalite;"
--etc...

FROM joueur
```

Utilisation de fonctions

Principe : utilisation de tables temporaires, créées pour l'exemple et non persistantes. Ainsi, un copier/coller de l'exemple permet de tester immédiatement le concept et sans risque d'altération de la base utilisée par le lecteur.


Cette section est vide, pas assez détaillée ou incomplète.

Utilisation de fonctions/Fonction UNPIVOT

La méthode UNPIVOT permet de transformer des colonnes en lignes supplémentaires. Son principe est d'introduire 2 nouvelles colonnes liées, à la place des N colonnes spécifiées :

- la première colonne indique le nom de la colonne
- la deuxième colonne indique la valeur de la colonne indiquée

Cette méthode peut être utile pour réduire le nombre de colonnes, et/ou appliquer aux valeurs stockées en colonnes les traitements applicables aux lignes

Illustration

La requête suivante illustre la modification d'une ligne (à noter aussi l'utilisation d'une double concaténation).

```
-- UNPIVOT Exemple 1

with Mesure as (
   select 1 MesureID,4 Capteur1,3 Capteur2,5 Capteur3,4 Capteur4,4 Capteur5 from dual union --Pour la 1ere ligne, on précise les noms des colonnes
  select 2 , 4 , 1 , 5 , 51, 5 from dual union --Pour les lignes, on ne le refait pas
  select 3 , 4 , 3 , 5 , 9 , 4 from dual union
  select 4 , 3 , 91, 5 , 5 , 4 from dual union
  select 5 , 4 , 1 , 5 , 5 , 5 from dual
  )
select * from Mesure

--***** Décommenter une ligne parmi les suivantes. Si aucune décommentée, le résultat sera l'affichage normal de la table temporaire "Mesure" *****
--UNPIVOT (ValeurColonne FOR Colonne IN (Capteur1)) -- Trivial. Une colonne "COLONNE" à valeur unique "CAPTEUR1" ajoutée. Nb lignes : 5 =>5
--UNPIVOT (ValeurColonne FOR Colonne IN (Capteur1, Capteur2)) -- Doublement du nombre de ligne Nb lignes: 5 =>10
--UNPIVOT (ValeurColonne FOR Colonne IN (Capteur1, Capteur2, Capteur3)) -- Triplement du nombre de ligne Nb lignes: 5 =>15
--UNPIVOT (ValeurColonne FOR Colonne IN (Capteur1, Capteur2, Capteur3, Capteur4)) -- Quadruplement du nombre de ligne. Nb lignes: 5 =>20
--UNPIVOT (ValeurColonne FOR Colonne IN (Capteur1, Capteur2, Capteur3, Capteur4, Capteur5)) -- Quintuplement du nombre de ligne Nb lignes: 5 =>25
;
```

-- Résultat sans rien décommenter

MESUREID	CAPTEUR1	CAPTEUR2	CAPTEUR3	CAPTEUR4	CAPTEUR5
1	4	3	5	4	4
2	4	1	5	51	5
3	4	3	5	9	4
4	3	91	5	5	4
5	4	1	5	5	5

-- Résultat en décommenter la 1ère ligne UNPIVOT

MESUREID	CAPTEUR2	CAPTEUR3	CAPTEUR4	CAPTEUR5	COLONNE	VALEURCOLONNE
1	3	5	4	4	CAPTEUR1	4
2	1	5	51	5	CAPTEUR1	4
3	3	5	9	4	CAPTEUR1	4
4	91	5	5	4	CAPTEUR1	3
5	1	5	5	5	CAPTEUR1	4

-- Résultat en décommenter la dernière ligne UNPIVOT

MESUREID	COLONNE	VALEURCOLONNE
1	CAPTEUR1	4
1	CAPTEUR2	3
1	CAPTEUR3	5
1	CAPTEUR4	4
1	CAPTEUR5	4
2	CAPTEUR1	4
2	CAPTEUR2	1
2	CAPTEUR3	5
2	CAPTEUR4	51
2	CAPTEUR5	5
3	CAPTEUR1	4
3	CAPTEUR2	3
3	CAPTEUR3	5
3	CAPTEUR4	9
3	CAPTEUR5	4
4	CAPTEUR1	3
4	CAPTEUR2	91
4	CAPTEUR3	5
4	CAPTEUR4	5
4	CAPTEUR5	4
5	CAPTEUR1	4
5	CAPTEUR2	1
5	CAPTEUR3	5
5	CAPTEUR4	5
5	CAPTEUR5	5

25 rows selected

Cas pratique d'utilisation

Gardons de la table exemple précédente la 1ere et la dernière ligne seulement. Le but sera de recenser les colonnes pour lesquelles ces lignes ont des valeurs différentes. Pour ce faire, on va

1. d'abord transformer les colonnes en lignes avec UNPIVOT
2. Puis dénombrer les valeurs ainsi transformées.
3. Filtrer les lignes créées, de couples (colonne, valeur), qui sont présentes une seule fois (ie les colonnes dont la valeur a changé)
4. Et enfin distinguer les noms de colonnes du précédent ensemble

L'exemple sera plus parlant :

```
-- UNPIVOT Exemple utilisation

with Mesure as (
  select 1 MesureID,4 Capteur1,3 Capteur2,5 Capteur3,4 Capteur4,4 Capteur5 from dual union -- 1ere ligne
  select 5 , 4 , 1 , 5 , 5 , 5 from dual -- dernière ligne
  )

, Denombrement as (
   select COLONNE, VALEURCOLONNE, count(*) nombre from Mesure -- On ne conserve que les 2 colonnes du couple (colonne, valeur), plus un dénombrement
    UNPIVOT (ValeurColonne FOR Colonne IN (Capteur1, Capteur2, Capteur3, Capteur4, Capteur5)) -- On converti en lignes l'intégralité des colonnes, sauf la PK
  group by COLONNE, VALEURCOLONNE
)
```

```
--select * from Mesure -- Décommenter cette ligne pour voir les 2 lignes comparées
--select * from Denombrement -- Décommenter cette ligne pour voir le dénombrement des valeurs
--select * from Denombrement where nombre=1 -- Décommenter cette ligne pour voir le filtrage du dénombrement, ie les colonnes variantes
select distinct COLONNE from Denombrement where nombre=1 -- Il suffit juste de recenser les colonnes variantes
;
```

-- Résultat en décommentant le premier select (« select * from Mesure »)

MESUREID	CAPTEUR1	CAPTEUR2	CAPTEUR3	CAPTEUR4	CAPTEUR5
1	4	3	5	4	4
5	4	1	5	5	5

-- Résultat en décommentant le deuxième select (« select * from Denombrement »)

COLONNE	VALEURCOLONNE	NOMBRE
CAPTEUR2	3	1
CAPTEUR5	4	1
CAPTEUR2	1	1
CAPTEUR1	4	2
CAPTEUR3	5	2
CAPTEUR5	5	1
CAPTEUR4	4	1
CAPTEUR4	5	1

-- Résultat en décommentant le dernier select (« »)

COLONNE
CAPTEUR5
CAPTEUR2
CAPTEUR4

Généralisation de la requête précédente

Bien sûr, pour un exemple aussi simple que celui pris, il serait plus simple de recenser par soi même les colonnes variantes. Mais si la table considérée faisait 300 colonnes, avec des noms complexes, alors un recensement à la main serait extrêmement fastidieux et probablement source d'erreurs !

La démarche idéale supposerait d'agir en deux temps :

1. Tout d'abord, lister les colonnes de la table cible
2. Copier coller le résultat de la requête précédente en lieu et place de "Capteur1, Capteur2, Capteur3, Capteur4, Capteur5"

Utilisation de fonctions/fonction LISTAGG

La méthode LISTAGG permet de concaténer les valeurs d'une colonne, en gérant la séparation (typiquement, avec des "," qui est le choix par défaut). Cette fonction est disponible depuis la version Oracle 11g

Illustration

La requête suivante illustre l'utilisation de cette fonction.

```
with Groupe as (
  select 'Claire' prenom, 'F' sexe, 25 age from dual union --Pour la 1ere ligne, on précise les noms des colonnes
  select 'Jean-Sebastien', 'M', 32 from dual union --Pour les lignes, on ne le refait pas
  select 'Marie', 'F', 23 from dual union
  select 'Kevin', 'M', 19 from dual union
  select 'Natacha', 'F', 31 from dual
  )

--select * from Groupe;
SELECT LISTAGG(prenom, ', ') WITHIN GROUP (ORDER BY prenom) "Membres féminins du groupe" FROM Groupe WHERE sexe='F';
--SELECT LISTAGG(prenom, ', ') WITHIN GROUP (ORDER BY age DESC) "Membres trentenaires du groupe" FROM Groupe WHERE age>30;
```

-- Résultat du 1er select (« select * from Groupe »)

PRENOM	SEXE	AGE
Claire	F	25
Jean-Sebastien	M	32
Kevin	M	19
Marie	F	23
Natacha	F	31

-- Résultat du 2eme select :

```
Membres féminins du groupe
--------------------------------------------------------------------
Claire, Marie, Natacha
```

-- Résultat du 3eme select :

```
Membres trentenaires du groupe
--------------------------------------------------------------------
Jean-Sebastien, Natacha
```

Cas pratique d'utilisation

La requête suivante permet de lister les colonnes de type 'NUMBER' d'une table 'MA_TABLE', en concaténant les noms

```
SELECT LISTAGG(column_name, ', ') WITHIN GROUP (ORDER BY column_name) "numberTypeColumns"
FROM user_tab_columns WHERE table_name = 'MA_TABLE' and data_type like 'NUMBER%';
```

Cette requête peut s'avérer utile pour utiliser la fonction UNPIVOT

Utilisation de fonctions/fonction DECODE

La fonction DECODE correspond à une fonction (IF, EQUALS, THEN, ELSE), voire (IF,THEN,ELSIF,ELSIF,...,ELSIF,ELSE) dans son utilisation avec plus de quatre arguments.

Plusieurs utilisations sont possibles, selon le nombre de cas de figure prévues :

Nombre de tests	Nombre d'arguments	Syntaxe
1	4	DECODE (Expression, Egalité, BlocTrue, BlocFalse)
2	6	DECODE (Expression, Egalité1, BlocTrue1, Egalité2, BlocTrue2, BlocFalse)
3	8	DECODE (Expression, Egalité1, BlocTrue1, Egalité2, BlocTrue2, Egalité3, BlocTrue3, BlocFalse)
etc... N	4+2*(N-1)	DECODE (Expression, Egalité1, BlocTrue1, ..., EgalitéN, BlocTrueN, BlocFalse)

Exemple d'utilisation

La requête suivante illustre l'utilisation de cette fonction.

```
--Exemple d'utilisation de la fonction DECODE

WITH Mesure AS (
  SELECT 1 MesureID,4 Capteur1,3 Capteur2,5 Capteur3,4 Capteur4,4 Capteur5 from dual union --Pour la 1ere ligne, on précise les noms des colonnes
  SELECT 2 , 4 , 1 , 5 , 51, 5 FROM DUAL UNION --Pour les lignes, on ne le refait pas
  SELECT 3 , 4 , 3 , 5 , 9 , 4 FROM DUAL UNION
  SELECT 4 , 3 , 91, 5 , 5 , 4 FROM DUAL UNION
  SELECT 5 , 4 , 1 , 5 , 5 , 5 FROM DUAL
  )
--SELECT * FROM Mesure

SELECT
MesureID
,DECODE(Capteur1, 1, 'Un', 2, 'Deux', 3, 'Trois', 4, 'Quatre', 5, 'Cinq', 'Plus que cinq') "Capteur du Jardin"
,DECODE(SIGN(Capteur2-9), 1, 'NOTABLE', 'RAS') "Capteur du parking" -- DECODE( SIGN(A-B), 1, "A>B", "A>=B") pratique pour un choix binaire
,DECODE(Capteur3, 1, 'Un', 2, 'Deux', 3, 'Trois', 4, 'Quatre', 5, 'Cinq', 'Plus que cinq') "Capteur de la Cuisine"
,DECODE(TRUNC(Capteur4/5), 0, 'Plage faible', 1, 'Plage moyenne', 'Plage forte') "Capteur du toit" -- DECODE( TRUNC(val/PAS), 0, "val entre 0 et PAS", 1, "val entre PAS et 2*PAS", "val>2*PAS") pour un choix à N plages
,DECODE(Capteur5, 1, 'Un', 2, 'Deux', 3, 'Trois', 4, 'Quatre', 5, 'Cinq', 'Plus que cinq') "Capteur du Grenier"
FROM Mesure
;
```

-- Résultat en décommentant "SELECT * FROM Mesure"

MESUREID	CAPTEUR1	CAPTEUR2	CAPTEUR3	CAPTEUR4	CAPTEUR5
1	4	3	5	4	4
2	4	1	5	51	5
3	4	3	5	9	4
4	3	91	5	5	4
5	4	1	5	5	5

-- Résultat sans rien décommenter

MESUREID	Capteur du Jardin	Capteur du parking	Capteur de la Cuisine	Capteur du toit	Capteur du Grenier

1	Quatre	RAS	Cinq	Plage faible	Quatre
2	Quatre	RAS	Cinq	Plage forte	Cinq
3	Quatre	RAS	Cinq	Plage moyenne	Quatre
4	Trois	NOTABLE	Cinq	Plage moyenne	Quatre
5	Quatre	RAS	Cinq	Plage moyenne	Cinq

Sauvegardes et restaurations

Rman

Cette section est vide, pas assez détaillée ou incomplète.

Dataguard

Cette section est vide, pas assez détaillée ou incomplète.

Bases de données multimédia

Description

Oracle Multimedia est une suite de services fournie avec Oracle Database (sauf la version Express où on ne peut pas l'ajouter[1]) depuis la version 8 (en 1997), pour gérer des bases de données multimédia.

Elle est constituée d'un package ORDSYS ("ORD" pour *object-relational data*) permettant la gestion des objets multimédia dans la base[2]. Il comprend plusieurs classes[3] :

- ORDMultimedia : superclasse abstraite stockant les attributs et méthodes communs aux classes ORDAudio, ORDImage, et ORDVideo[4].
- ORDAudio : stockage des caractéristiques des sons.
- ORDDoc : stockage des caractéristiques des documents hétérogènes.
- ORDImage : stockage des caractéristiques des images.
- ORDVideo : stockage des caractéristiques des vidéos.
- ORDSource : stockage des sources multimédia dans des BLOB de la base, ou des BFILE accessibles en HTTP[5].
- DICOM (Digital Imaging and Communications in Medicine[6]).

Attributs

ORDAudio[7]	**ORDDoc**[8]	**ORDImage**[9]	**ORDVideo**[10]
description	source	source	description
source	format	height	source
format	mimeType	width	format
mimeType	contentLength	contentLength	mimeType
comments	comments	fileFormat	comments
encoding		contentFormat	width
numberOfChannels		compressionFormat	height
sampleSize		mimeType	frameResolution
compressionType			frameRate
audioDuration			videoDuration
			numberOfFrames
			compressionType
			numberOfColors
			bitRate

Utilisation

```
CREATE TABLE MesImages (
    id  INTEGER PRIMARY KEY,
    image   ORDSYS.ORDImage
);
```

Cette section est vide, pas assez détaillée ou incomplète.

Oracle HTTP Server Télécharger (http://www.oracle.com/technetwork/middleware/webtier/downloads/index.html) permet d'exécuter des requêtes PL/SQL depuis un navigateur.

Références

1. *(en)* Managing Oracle Multimedia Installations (http://www.comp.dit.ie/btierney/Oracle11gDoc/appdev.111/b28415/ap_instl_upgrd.htm)
2. http://fildz.developpez.com/tutoriel/oracle-java/ordimage/
3. *(en)* Common Methods and Notes for Oracle Multimedia Object Types (https://docs.oracle.com/cd/E11882_01/appdev.112/e10776/ch_comref.htm#AIVUG3000) sur *Oracle.com*
4. *(en)* Common Methods and Notes for Oracle Multimedia Object Types (https://docs.oracle.com/html/A67296_01/im_mmref.htm#998184) sur *Oracle.com*
5. (en) *Oracle 10g Developing Media Rich Applications*, 8 avril 2011 (lire en ligne (https://books.google.fr/books?id=-dbeFbCswAYC&pg=PA60&dq=ORDSYS++ORDAudio+ORDDoc+ORDImage+ORDVideo+ORDSource&hl=fr&sa=X&ved=0ahUKEwj7z9jC5OPKAhVHMhoKHXsYCr8Q6AEIJjAB#v=onepage&q=ORDSYS%20%20ORDAudio%20ORDDoc%20ORDImage%20ORDVideo%20ORDSource&f=false))

6. *(en)* Medical Imaging and Communication (https://docs.oracle.com/database/121/IMDCM/ch_intro.htm#IMDCM1100) sur *Oracle.com*
7. *(en)* ORDAudio (https://docs.oracle.com/cd/B19306_01/appdev.102/b14297/ch_audref.htm) sur *Oracle.com*
8. *(en)* ORDDoc (https://docs.oracle.com/cd/B28359_01/appdev.111/b28414/ch_docref.htm) sur *Oracle.com*
9. *(en)* ORDImage (https://docs.oracle.com/cd/B28359_01/appdev.111/b28414/ch_imgref.htm) sur *Oracle.com*
10. *(en)* ORDVideo (https://docs.oracle.com/cd/B28359_01/appdev.111/b28414/ch_vidref.htm) sur *Oracle.com*

- *(en)* Multimedia User's Guide (http://docs.oracle.com/database/121/IMURG/toc.htm) sur *Oracle.com*

Bases de données spatiotemporelles

Données spatiales

Lors du typage des champs, certains représentent des objets graphiques, et sont donc considérés comme étant de catégorie "Spatial" (cf. base de données spatiales). Par conséquent, ils se manipulent par des requêtes différentes que pour le texte.

Sous Oracle, c'est implémenté depuis la version 7 dans une extension de la version *Enterprise Edition* Télécharger (http://www.oracle.com/technetwork/database/options/spatialandgraph/downloads/index-093371.html), fournissant des objets avec des préfixes SDO pour *Spatial Data Option*[1].

Objets

Pour stocker les objets spatiaux, on utilise le type de champs *SDO_GEOMETRY*.

Ainsi que sept méthodes pour le manipuler[2] :

1. Get_Dims
2. Get_GType
3. Get_LRS_Dim
4. Get_WKB
5. Get_WKT
6. ST_CoordDim
7. ST_IsValid

Plus des opérateurs de requête[3] :

1. SDO_FILTER : liste les objets interagissant avec la cible.
2. SDO_JOIN : jointure spatiale.
3. SDO_NN (pour *nearest neighbor*) : renvoie le voisin le plus proche de la cible.
4. SDO_NN_DISTANCE : la distance avec le voisin le plus proche.
5. SDO_RELATE : liste les objets interagissant d'une certaine façon.
6. SDO_WITHIN_DISTANCE : dit si deux objets sont à moins d'une certaine distance l'un de l'autre.

Données spatiotemporelles

On utilise un prédicat pour prévoir le mouvement des objets stockés[4]. Toutefois les bases de données spatiotemporelles nécessitent malgré tout des mises à jour fréquentes.

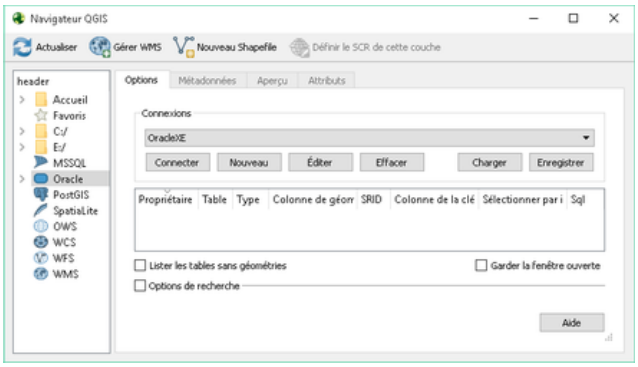
Indexation

Les modes d'indexation choisis par Oracle pour les données spatiales sont l'arbre R[5], l'arbre Q, et le Z-order[6].

Liaison avec des SIG

Pour représenter les données en base sur des cartes, on utilise un système d'information géographique (SIG). Par exemple :

- GRASS GIS[7]
- QGIS



Configuration QGIS des liaisons avec les bases de données (dont Oracle).

Si le logiciel a besoin d'une source de données ODBC pour accéder aux bases Oracle :

1. Lancer %windir%\system32\odbcad32.exe.
2. Ajouter une source de données système. Le driver Oracle peut être choisi dans la liste si le SGBD est installé.
3. Remplir le champ *Nom du service TNS* avec le nom situé dans le fichier C:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN\tnsnames.ora.
4. Puis inscrire le mot de passe de la connexion crée dans SQL*Plus.

Exemples

- http://download.oracle.com/otndocs/products/spatial/pdf/au_melbourne06_start.pdf
- http://download.oracle.com/otndocs/products/spatial/pdf/GeocodingInOracleUsing_HERE_MapContent.pdf

Cette section est vide, pas assez détaillée ou incomplète.

Références

1. *(en)* Spatial Developer's Guide (http://docs.oracle.com/cd/E11882_01/appdev.112/e11830/toc.htm) sur *Oracle.com*
2. *(en)* SDO_GEOMETRY Object Type (https://docs.oracle.com/cd/B19306_01/appdev.102/b14255/sdo_objrelschema.htm#i1004087) sur *Oracle.com*
3. *(en)* Spatial Operators (https://docs.oracle.com/cd/B19306_01/appdev.102/b14255/sdo_operat.htm) sur *Oracle.com*
4. *(en)* Authorizing Access to Dynamic Spatial-Temporal Data (http://www.oracle.com/technetwork/articles/vanvelden-spatial-095837.html) sur *Oracle.com*
5. *(en)* Spatial Concepts (https://docs.oracle.com/html/A88805_01/sdo_intr.htm) sur *Oracle.com*
6. *(en)* ZOrder Method (http://docs.oracle.com/cd/E20213_01/doc/win.112/e17727/dcmethods006.htm) sur *Oracle.com*
7. *(en)* Geographic Resources Analysis Support System (GRASS): More Than a Mapping Tool (http://www.oracle.com/technetwork/articles/mitasova-grass-092663.html) sur *Oracle.com*

Débogage

AUTOTRACE

Cette section est vide, pas assez détaillée ou incomplète.

Une erreur s'est produite lors de l'opération demandée

Listener refused the connection with the following error: ORA-12505, TNS:listener does not currently know of SID given in connect descriptor Code fournisseur 12505

Un processus est manquant, relancer les services Oracle.

Si cela ne change rien, vérifier le contenu de *listener.ora* avant de relancer le service *listener*.

Sinon, désinstaller puis réinstaller le logiciel.

SP2-0734: commande inconnue au début de "..." - le reste de la ligne est ignoré

Essayer d'entourer la commande par `BEGIN` et `END`.

Récupérée de « https://fr.wikibooks.org/w/index.php?title=Oracle_Database/Version_imprimable&oldid=506167 »

La dernière modification de cette page a été faite le 8 février 2016 à 21:52.