Mémento des principaux ordres SQL Oracle

SELECT [nom_table.]* | [nom_table].nom_col, [nom_table.]nom_col, ...
FROM [nom_user.]table ,...
[WHERE condition]
[CONNECT BY condition START WITH condition]
[GROUP BY expr, expr... HAVING condition]
[UNION | INTERSECT | MINUS ordre_select]
[ORDER BY expr [ASC | DESC]]
[FOR UPDATE OF nom_col [NOWAIT]]

ramène les valeurs des colonnes spécifiées de la (les) table(s) spécifiée(s), pour les lignes satisfaisant la condition de la clause WHERE

* : toutes les colonnes de la table
CONNECT BY : gère les liens hiérarchiques au sein de la table
START WITH : début d'un parcours hiérarchique
GROUP BY : regroupement suivant les différentes valeurs de la colonne ou de l'expression spécifiée
UNION : union ensembliste de deux SELECT
INTERSECT : intersection ensembliste de deux SELECT
MINUS : difference ensembliste entre deux SELECT
ORDER BY : tri suivant une colonne ou une expression ascendant (par défaut) ou descendant
FOR UPDATE OF : consultation en vue de mise à jour, éventuellement sans attente si la donnée est déjà verrouillée de manière incompatible
Exemple :

```
SELECT code, nom, ventes FROM clients
WHERE departement = 'ESSONNE'
ORDER BY nom
```

SELECT liste_colonnes INTO liste_variables
FROM nom_table [, nom_table,...]
[...]
charge des variables de réception (simples ou structurées) avec les valeurs ramenées par le SELECT
Exemple :

```
SELECT code, nom, ventes
INTO :v_code, :v_nom, :v_ventes FROM clients
```

PDATE nom_table SET nom_colonne = expression_SQL nom_colonne = (ordre_SELECT_mono_colonne) (nom_colonne, nom_colonne, ...) = ordre_SELECT [WHERE condition]

met a jour les colonnes specifiées (pour des lignes déjà existantes de la table), ces valeurs sont des constantes, ou des expressions, ou le résultat d'un SELECT.
Les lignes concernées sont determinées par la clause WHERE.
Exemple :

```
UPDATE clients
SET ventes = ventes + 12500
WHERE code = 112
```

INSERT INTO nom_table [(liste_colonnes)]
VALUES (liste_valeurs)

insère des valeurs dans les colonnes specifiées de la table ou de la vue (par défaut toutes les colonnes). Ces valeurs sont des constantes
Exemple :

```
INSERT INTO clients (code, nom)
VALUES (115,'MATHON')
```

INSERT INTO nom_table [(liste_colonnes)] ordre_select
insère des valeurs dans les colonnes specifiées de latable ou de la vue (par défaut toutes les colonnes). Ces valeurs sont le résultat d'un sous-SELECT
Exemple :

```
INSERT INTO clients (code, nom)
SELECT * FROM nouveaux_clients
```

DELETE FROM nom_table [clause_where]
supprime une ou plusieurs lignes d'une table, la totalité si aucune clause where n'est spécifiée.
Exemple :

```
DELETE FROM clients where code < 0
```

TRUNCATE TABLE nom_table [DROP STORAGE|REUSE STORAGE]

supprime toutes les lignes d'une table, en supprimant (ou gardant) les blocs physiques précédemment alloués
Exemple :

```
TRUNCATE nouveaux_clients DROP STORAGE
```

CREATE TABLE nom_table (nom_col1 type_col1 [contrainte], nom_col2 type_col2 [contrainte], ...)
[TABLESPACE nom_tbs]

[STORAGE INITIAL taille K | M NEXT taille K | M
MINEXTENTS n MAXEXTENTS n
PCTINCREASE pct]
crée une table dans le tablespace spécifié ou dans le tablespace SYSTEM par défaut avec les paramètres de stockage physique spécifiés par la clause STORAGE.
taille est un nombre qui précise la taille en KO ou MO du premier extent de la table et de l'extent suivant. n précise respectivement le nombre d'extents alloué à la création et le nombre maximum d'extents. pct précise la loi d'augmentation des extents futurs, en pourcentage
Exemple :

CREATE TABLE nouveaux_clients
(code NUMBER(4) PRIMARY KEY, nom VARCHAR(10))
TABLESPACE tbs_clients
STORAGE INITIAL 2MO NEXT 2MO PCTINCREASE 0

CREATE [OR REPLACE] VIEW [nom_proprietaire.]nom_vue [(syn_col1, syn_col2,...)]
AS SELECT ...
WITH CHECK OPTION [CONSTRAINT nom_contrainte]

Exemple :
CREATE VIEW martin.vue_annuaire
AS SELECT nom,no_tel FROM annuaire
WHERE departement = 91

create index

CREATE [UNIQUE | BITMAP] INDEX [nom_prpriétaire.]nom_index
ON [nom_propriétaire.]nom_table(nom_col1 [,nom_col2,...] [ASC | DESC]
)
[TABLESPACE nom_tablespace]
[STORAGE (parametres_de_stockage)]

CREATE UNIQUE INDEX idx_clients
ON clients(no_client)
TABLESPACE tbs_idx

CREATE [PUBLIC] SYNONYM [nom_proprietaire.]nom_synonyme
FOR [nom_proprietaire.]nom_objetcrée un synonyme public (visible par tous, créé par le DBA en général) ou privé (appartenant à un user particulier) sur un objet. Ceci permet d'éviter le préfixage par le nom du propriétaire ensuite. Le d'avoir accès à un synonyme ne dispense pas d'avoir des droits sur l'objet référencé.
Exemple :

CREATE PUBIC SYNONYM annuaire FOR devel1.annuaire

Create sequence

```
CREATE SEQUENCE [nom_propriétaire.]nom_compteur
[START WITH n]
[INCREMENT BY n]
[MAXVALUE n | NOMAXVALUE]
[MINVALUE n | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE n | NOCACHE]
[ORDER | NOORDER ]
CREATE SEQUENCE seq_no_client
START WITH 10
INCREMENT BY 10
NOMAXVALUE
```

```
CREATE [OR REPLACE] PROCEDURE nom_procedure
( nom_arg1 type_arg1 IN|OUT|IN OUT, ... )
IS | AS bloc_PL/SQL
```

crée une procédure avec des paramètres éventuels d'entrée et/ou de sortie
Exemple :

```
CREATE PROCEDURE maj_ventes
(code_client IN NUMBER, montant_vente IN NUMBER)
AS BEGIN
UPDATE clients SET ventes = ventes + montant_vente
WHERE code = code_client;
END;
```

```
CREATE [OR REPLACE] FUNCTION nom_fonction
( nom_arg1 [IN] type_arg1 , ... )
RETURN type_val_retour AS|IS bloc_PL/SQL
```

crée une fonction, en fait, une procédure retournant une valeur
Exemple :

```
CREATE FUNCTION lit_ventes
(code_client NUMBER)
RETURN NUMBER
IS
montant NUMBER (10,2);
BEGIN
SELECT ventes INTO montant
FROM clients
WHERE code = code_client;
RETURN (montant);
END;
```

CREATE [OR REPLACE] PACKAGE nom_package
IS|AS specification_package
crée une spécification de package, les traitements à effectuer seront définis dans le PACKAGE BODY
Exemple :

```
CREATE PACKAGE va_et_vient AS
PROCEDURE ajoute_client (code NUMBER, nom VARCHAR);
PROCEDURE supprime_client (code NUMBER);
END va_et_vient;
```

CREATE [OR REPLACE] TRIGGER nom_trigger
BEFORE|AFTER DELETE|INSERT|UPDATE [OF nom_col]
ON nom_table
REFERENCING OLD variable|NEW variable
[FOR EACH ROW]
[WHEN (condition)] bloc_PL/SQL

crée un trigger au niveau de la base activable automatiquement lors d'une mise à jour

Exemple :

```
CREATE TRIGGER verif_code
BEFORE INSERT OF code ON appli.clients
FOR EACH ROW
DECLARE dernier_code NUMBER;
BEGIN
SELECT max(code) INTO dernier_code FROM clients;
IF (:new.code < dernier_code)
THEN raise_application_error
                (-99999,'code client erronne')
END_IF;
END;
```

DROP TABLE nom_table [CASCADE CONSTRAINTS]

supprime une table et éventuellement les contraintes....
Exemple :

```
DROP TABLE archive_clients
```

DROP PROCEDURE | FUNCTION | PACKAGE | TRIGGER
nom_procedure|nom_fonction|nom_package|nom_trigger
supprime une procédure, une fonction, un package ou un trigger
Exemple :

DROP PROCEDURE test

DROP CLUSTER|VIEW | INDEX | SEQUENCE | [PUBLIC] SYNONYM
[nom_proprietaire.]nom_objet
Exemple

DROP INDEX martin.idx_client

RENAME nom_ancien_objet TO nom_nouveau_objet

renomme une table, une vue ou un synonyme
Exemple :

RENAME sauvegarde_clients TO clients

LOCK TABLE nom_table
IN SHARE|ROW SHARE|EXCLUSIVE|ROW EXCLUSIVE MODE
[NOWAIT]

verrouille explicitement la table dans un mode particulier. Si NOWAIT est précisé le système rendra la main à l'utilisateur, dans le cas où la donnée est déjà verrouillée dans un mode incompatible
Exemple :

LOCK TABLE clients IN EXCLUSIVE MODE
NOWAIT