Création de table

```
CREATE TABLE produit (
        id_produit INTEGER NOT NULL PRIMARY KEY,
        nom VARCHAR2(30 CHAR) NOT NULL,
        quantite INTEGER DEFAULT '0',
        id_categorie INTEGER NOT NULL CONSTRAINT fk_produit_id_categorie REFERENCES
categorie (id_categorie),

        ou

        CONSTRAINT pk_produit PRIMARY KEY(id_produit),
        CONSTRAINT fk_produit_id_categorie FOREIGN KEY (id_categorie) REFERENCES
categorie (id_categorie)
);
```

-Contraintes d'attribut : NULL, NOT NULL, UNIQUE, PRIMARY KEY
-Contraintes de valeurs : CHECK(expression_boolean)
-Contraintes référentielle :

REFERENCES table_père(attribut-père) ON DELETE CASCADE : Déclenchera la suppression des fils, la suppression sera toujours acceptée

ON DELETE SET NULL : La clé étrangère des fils sera mise à NULL sous réserve d'une contrainte d'intégrité l'empêchant

Recopie d'une table

```
CREATE TABLE agence AS SELECT * FROM nomtable;
```

Renomination de tables(Vues/Sequences)

```
RENAME anciennom TO nouveaunom;
```

ATTENTION
- TRANSFERT DES CONTRAINTES D 'INTÉGRITÉ
- LES VUES ET OBJETS TRAÎTEMENT SUR L 'ANCIENNE DOIVENT ÊTRE RECRÉÉS.

Changement de structure de table

- Ajout de colonne

```
ALTER TABLE nomtable ADD(nomcol TYPCOL[DEFAULT valeur],);
```

-Renomination de colonne

```
ALTER TABLE nomtable RENAME COLUMN nomcol1 TO nomcol2;
```

-Modification de colonne

```
ALTER TABLE nomtable MODIFY(nomcol TYPE);
```

-Ajouter les contraintes

```
ALTER TABLE agence ADD CONSTRAINT pk_agence PRIMARY KEY(num_agence);
ALTER TABLE compte ADD CONSTRAINT fk_agence_compt FOREIGN KEY(num_agence)
REFERENCES agence(num_agence);
ALTER TABLE emprunt ADD CONSTRAINT c_montant CHECK(montant < 50000);
```

ALTER TABLE compte ADD CONSTRAINT uniq_client_agence UNIQUE(num_agence, num_client) NOVALIDATE;

-contraintes non référentielles
ALTER TABLE nomtable DROP CONSTRAINT nomcontrainte;
ALTER TABLE nomtable DROP PRIMARY KEY;
ALTER TABLE nomtable DROP UNIQUE(attr1, att2, …);

-contraintes référentielles
ALTER TABLE nomtable DROP CONSTRAINT nomcontrainte CASCADE;
ALTER TABLE nomtable DROP PRIMARY KEY CASCADE;

-changment du statut des contraintes
ALTER TABLE nomtable DISABLE[VALIDATE | NOVALIDATE] CONSTRAINT nomcontrainte [CASCADE];
ALTER TABLE nomtable ENABLE[VALIDATE | NOVALIDATE] CONSTRAINT nomcontrainte [EXCEPTIONS INTO table_except];

Création d'une séquence

CREATE SEQUENCE nomsequence START WITH valeurinit;

SELECT nomsequence.NEXTVAL FROM DUAL;

SELECT nomsequence.CURRVAL FROM DUAL;

Création de vue

CREATE VIEW nomvue AS SELECT….;

•OBJET VIRTUEL (PAS DE DONNÉES STOCKÉES MAIS TEXTE SQL STOCKÉ DANS LA MÉTABASE)

•S 'INTERROGE COMME UNE TABLE MAIS RESTRICTIONS EN MISE À JOUR

•PERMET DE SIMPLIFIER LES QUESTIONS ET DE FAIRE ÉVOLUER LE SCHÉMA

Destruction d'un objet de la base

DROP TABLE nomtable;
DROP TALBE nomtable CASCADE CONTRAINTS;
DROP VIEW nomvue;
DROP SEQUENCE nomsequence;

Requêtes en SQL

SELECT nomcol1, nomcol2,…. FROM nomtab1,nomtab2,..
WHERE [jointures]
GROUP BY nomcol1, nomcol2,…
HAVING [condition sur les groupes]

-fonctions
SUM(attr_number)
COUNT(attr) ou COUNT(*)
COUNT(DISTINCT attr)
AVG(attr_number)
MIN(attr)
MAX(attr)

Mise à jour

INSERT INTO nomtable VALUES(…, …, …);
INSERT INTO nomtable(att1, att2, …) VALUES (val1, val2, …);
UPDATE nomtable SET att1 = atten1, ….. WHERE …;
DELETE FROM nomtable WHERE …;

Types d'attributs

-Type caractères
CHAR(n), VARCHAR2(n), NCHAR(n), NVARCHAR2(n), CLOB, NCLOB, LONG

-Type date
DATE: siècle, année, mois, jours, minutes, secondes
TIMESTAMP(fsec) : Idem date avec fractions de secondes
TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH : différence entre 2 moments avec une précision mois/année
INTERVAL DAY TO SECOND : différence entre 2 moments avec une précision de fraction de secondes

-Type number
NUMBER(X, Y) : réel avec X chiffres au total, Y après la virgule
NUMBER(X) : entier de longueur X

-Type binaire
BLOB, BFILE, RAW, LONG RAW

Fonctions de conversion explicites

TO_CHAR(att_date, 'format_date')
TO_CHAR(att_number, 'format_number')
TO_NUMBER('att_char')
TO_DATE(att_char, 'format_date')

Fonctions standards

-Attributs caractères
UPPER(attr_char), LOWER(attr_char), INITCAP(attr_char), SUBSTR(attr_char, pos, lg), LENGTH(attr_char)

INITCAP—将字符串每个单词首字母均变为大写 例: INITCAP('this is a test')=' This Is A Test''

-Attributs date
ADD_MONTH(attr_date, attr_number)
LAST_DAY(attr_date)
ROUND(attr_date, 'YEAR')
ROUND(attr_date, 'MONTH')

-Autres fonctions
NVL(attr1, expr)
DECODE(attr, expr1, res1, expr2, res2,….resdef)

PL/SQL

```
set serveroutput on  - - Pour afficher la sortie standard
DECLARE
        un_int INTEGER;   // FLOAT, NUMBER, VARCHAR2, DATE…
BEGIN
        un_int := 10;
        dbms_output.put_line('valeur: ' || un_int );
EXCEPTION
        - - Ici le traitement en cas d'echec
END;
```

```
- IF condition THEN
        instruction
ELSIF
        instruction
ELSE
        instruction
END IF;
```

```
- CASE variable
WHEN valeur THEN
        instruction
WHEN valeur THEN
        instruction
ELSE
        instruction
END CASE;
```

```
- FOR un_int IN valeur1….valeurn LOOP
        instruction
END LOOP;

- WHILE condition LOOP
        instruction
END LOOP;
```

Types dérivés %TYPE et %ROWTYPE
une_description une_colone%TYPE;
un_produit une_table%ROWTYPE;

Curseurs

Variable prédéfinies:
        SQL%ROWCOUNT : nombre de lignes affectées par le dernier UPDATE ou DELETE
        SQL%FOUND : booléen vrai quand au moins une ligne a été affectée par la dernière instruction
        SQL%NOTFOUND : booléen vrai quand aucune ligne n'a été affectée.

—Curseur explicite

```
DECLARE
        CURSOR curseur IS SELECT * FROM participant;
        ligne participant%ROWTYPE;
BEGIN
        OPEN curseur;
        LOOP
```

```
            FETCH curseur INTO ligne;
            EXIT WHEN curseur%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(ligne.id_participant || ':' || ligne.nom);
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Trouve' || ':' || curseur%ROWCOUNT);
        CLOSE curseur;
END;
—Curseur paramétré
DECLARE
        CURSOR c_enchere(un_id_produit INTEGER) IS SELECT * FROM enchere
        WHERE id_produit = un_id_produit;
BEGIN
        FOR ligne IN c_enchere(1) LOOP
            DBMS_OUTPUT.PUT_LINE('enchere de' || ligne.montant || 'par membre' ||
ligne.id_encherisseur);
        END LOOP;
END;
```

• Exceptions

Exceptions prédéfinies

Nom	Code	Signification
CURSOR_ALREADY_OPEN	-6511	Le curseur a déjà été ouvert
DUP_VAL_ON_INDEX	-1	Tentative d'insérer un doublon dans une colonne unique
TIMEOUT_ON_RESOURCE	-51	Temps d'exécution maximal atteint
TOO_MANY_ROWS	-1422	Un SELECT INTO renvoie plus d'une ligne
NO_DATA_FOUND	100	Un SELECT INTO ne renvoie aucune ligne
VALUE_ERROR	-6502	Opération arithmétique qui échoue, ou conversion d'une chaine de caractères en nombre impossible

```
EXCEPTION
        WHEN OTHERS/NO_DATA_FOUD THEN
            RAISE_APPLICATION_ERROR(-20001, 'msg'); //doivent être entre -20000 et
-209999
```

• Trigger(Déclencheurs)

Ex. normaliser le nom des employés

:OLD(cas de UPDATE et DELETE)
:NEW(cas de UPDATE et INSERT)

```
CREATE OR REPLACE TRIGGER personne_nom_normalise_trg
BEFORE/AFTER INSERT OR UPDATE OR DELETE OF colonne1, colonne2
ON personne
FOR EACH ROW
```

```
WHEN(NEW.nom IS NOT NULL)
BEGIN
        :NEW.nom = INITCAP(TRIM(REGEXP_REPLACE(:NEW.nom, ' +', ' ')));
END;
```

Les champs auto-incrémentés

```
CREATE SEQUENCE enchere_id_enchere_seq;

CREATE OR REPLACE TRIGGER enchere_id_enchere_trg
BEFORE INSERT
ON enchere
FOR EACH ROW
BEGIN
        SELECT enchere_id_enchere_seq.NEXTVAL INTO :NEW.id_enchere FROM DUAL;
END;
```

Désactiver/ Activer un Déclencheur

```
ALTER TRIGGER un_trg DISABLE/ENABLE;
ALTER TRIGGER nom_tab DISABLE/ENABLE ALL TRIGGERS;

—Déclencheurs sur vue
CREATE OR REPLACE TRIGGER nom_trg
INSTEAD OF INSERT ON nom_vue
REFERENCING
        OLD AS old_row
        NEW AS new_row
DECLARE
```

Procédures et fonctions stockées

```
—Ex Fonction

CREATE OU REPLACE FUNCTION nom_fonc(chaine IN VARCHAR2)
RETURN VARCHAR2
AS
        resultat VARCHAR2(255);
BEGIN
        resultat := TRIM(chaine);       //enlever les espaces de debut et fin
        resultat := regexp_replace(resultat, ' +', ' ');  // remplacer les espaces multiples par un seul
        resultat := INITCAP(resultat);          //capitaliser la chaine
        resultat := regexp_replace(resultat, 'De ', 'de '); //remettre en minuscules les "De" et "D' "
        resultat := regexp_replace(resultat, 'D"', 'd"');
RETURN resultat;
END;

SELECT nom_fonc('    maNuel   de          oLiVeirA       ') FROM dual;
```

—Ex Procédure

```
CREATE OU REPLACE PROCEDURE afficher_proc(un_id_participant IN INTEGER)
AS
        nb INTERGER;
BEGIN
        SELECT COUNT(*) INTO nb FROM produit WHERE id_vendeur = un_id_participant;
        DBMS_OUT.PUT('Nb de produit proposes par ' || un_id_participant || ':' || nb);
        SELECT COUNT(*) INTO nb FROM enchere WHERE id_encherisseur =
un_id_participant;
        DBMS_OUT.PUT(' - Nb d'encheres ' || nb);
END;

set serveroutput on;
CALL afficher_proc(2);
```