

- Pas de type BOOLEAN
- Pas de mode AUTOCOMMIT, il faut utiliser le mot-clé COMMIT;
- **Importer un CSV sous SQL Developer** : clic droit sur le dossier tables à gauche dans SQL Developer et choisir "Import data"

## SQL2

### PL/SQL

```

SELECT table_name FROM dba_tables;
SELECT table_name FROM all_tables; -- Même résultat
SELECT table_name FROM user_tables; -- Que mes tables;
SELECT * FROM ALL_OBJECTS WHERE OBJECT_TYPE IN ('FUNCTION','PROCEDURE','PACKAGE');

DROP TABLE USER CASCADE CONSTRAINTS;
CREATE TABLE USER (
  prenom varchar2(20),
  nom varchar2(20),
  specialite varchar2(20),
  dateInscription TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
  sexe char check (sexe in ('m','f')),
  constraint PK_MEMBRE primary key (prenom),
  constraint FK_MEMBRE_SPECIALITE foreign key (specialite) references SPECIALITE(intitule)
);

create sequence numerotation_projet -- Voir la cheatsheet sur PostgreSQL pour plus d'info
INCREMENT BY 1 START WITH 1 NOMAXVALUE NOCYCLE CACHE 10; -- Cache améliore les performances d'accès à la mémoire

INSERT INTO Projet VALUES (numerotation_projet.NEXTVAL, 'Dupont', TO_DATE('06052015', 'DDMMYYYY'), TO_DATE('07052015', 'DDMMYYYY'), '
  Pierre', SYSDATE); -- On peut aussi utiliser 'numerotation_projet.CURRVAL', SYSTIMESTAMP contient l'heure, en plus de SYSDATE
COMMIT;

-----FONCTION-----
CREATE OR REPLACE FUNCTION fDemande (num_produit in number) RETURN varchar
IS
qte_vendue NUMBER;
begin
  select sum(qte) INTO qte_vendue from ligne_fact lf where lf.produit = num_produit;
  IF (qte_vendue > 15) THEN
    RETURN ('forte');
  ELSIF (qte_vendue between 11 and 15) THEN
    RETURN ('moyenne');
  ELSE
    RETURN ('faible');
  END IF;
END fDemande;
/
SHOW ERRORS;

-----PROCEDURE-----
CREATE OR REPLACE PROCEDURE nb_fact(num_client IN NUMBER, nb OUT NUMBER, ca OUT NUMBER)
AS
BEGIN
  dbms_output.put_line('Hello World!');

  SELECT count(f.num) into nb FROM facture f where f.client = num_client;

  SELECT SUM(P.PRIX*LF.QTE) into ca FROM FACTURE F, PRODUIT P, LIGNE_FAC LF WHERE F.NUM = LF.FAC AND P.NUM = LF.prod AND F.cli = n_cli;
END nb_fact;
/

-----BLOC ANONYME-----
SET SERVEROUTPUT ON
DECLARE
CURSOR c_client IS SELECT num FROM client;
nb NUMBER;
ca NUMBER;
BEGIN
  For client IN c_client Loop -- ligne de requete qui contient que 'num'
    nb_fact(client.num,nb,ca);
    dbms_output.put_line('the result for client ' || client.num || ' is ' || NB || ' (nb) and ' || CA || ' (ca)'); -- concaténation
  End loop;
END;
/

-----TRIGGER-----
create or replace TRIGGER STOCK_MAJ
AFTER INSERT ON ligne_fact FOR EACH ROW
DECLARE
  NEW_STOCK NUMBER;
BEGIN
  UPDATE PRODUIT SET PRODUIT.STOCK = PRODUIT.STOCK - :new.qte WHERE PRODUIT.num= :new.produit;
  SELECT stock INTO NEW_STOCK from produit where num = :new.produit;
  if (new_stock < 5) THEN
    INSERT INTO JOURNALISATION VALUES ('Attention : rupture de stock imminente',sysdate,:new.produit, NEW_STOCK);
  END IF;

```

```
END;  
/  
-----Afficher le contenu de toutes les tables-----  
SET SERVEROUTPUT ON  
DECLARE  
BEGIN  
  FOR t IN (SELECT table_name FROM all_tables WHERE owner <> 'SYS' and data_type LIKE '%CHAR%') LOOP  
    dbms_output.put_line(SELECT * FROM t.table_name);  
  END LOOP;  
  
END;  
/
```

## SQL3 - Relationnel objet

PL/SQL

```

CREATE OR REPLACE TYPE Client AS OBJECT (
    num INTEGER,
    nom varchar2(30),
    prenom varchar2(30),
    adresse varchar2(50),
    date_nais date,
    tel varchar2(30),
    sexe char
);
/

-----Héritage-----
CREATE TYPE sous-type UNDER sur_type ( déclarations ou surcharges );
-----

CREATE TABLE tClient OF Client (
    primary key (num),
    check (sexe in ('m', 'f'))
);
/

CREATE OR REPLACE TYPE LigneFacture AS OBJECT (
    qte integer,
    prod REF Produit, -- prod est un OID d'un objet Produit
    prod2 Produit -- prod EST l'objet
);
/

CREATE OR REPLACE TYPE RefLigneFacture AS OBJECT (
    ligneFacture REF LigneFacture
);
/

CREATE OR REPLACE TYPE ListeRefLigneFacture AS TABLE OF RefLigneFacture; -- Une table de REFERENCES vers des objets LigneFacture
CREATE OR REPLACE TYPE ListeLigneFacture AS TABLE OF LigneFacture; -- Une table d'objets
/

CREATE OR REPLACE TYPE Facture AS OBJECT (
    num integer,
    date_etabli date,
    fk_client REF Client,
    ligness ListeLigneFacture, -- Contient une collection de LigneFacture
    member function total return integer, -- Déclaration d'une fonction, implémentée plus bas
    member function quantite return integer
);
/

CREATE TABLE tFacture OF Facture (
    PRIMARY KEY (num),
    SCOPE FOR (fk_client) is tClient -- tClient est le nom de la table que l'on vise
)

NESTED TABLE lignes STORE AS table_lignes; -- On pourra faire Facture.lignes.COLONNE

CREATE OR REPLACE TYPE BODY Facture IS
    MEMBER FUNCTION total RETURN integer
    IS
        vtotal integer;
    BEGIN
        select sum(lf.qte * lf.prod.prix) INTO vtotal FROM tFacture F, table(f.ligness) lf WHERE f.num = self.num;
        RETURN vtotal;
    END total;
    MEMBER FUNCTION quantite RETURN integer
    IS
        vquantite integer;
    BEGIN
        select sum(lf.qte) INTO vquantite FROM tFacture F, table(f.ligness) lf WHERE f.num = self.num;
        RETURN vquantite;
    END quantite;
END;
/

-----
DECLARE
ref_client_1 ref CLIENT;
ref_client_2 ref CLIENT;
ref_produit_1 ref PRODUIT;
ref_produit_2 ref PRODUIT;
ref_produit_3 ref PRODUIT;
BEGIN
SELECT REF(c) into ref_client_1
from tClient c
WHERE c.num = 1;
-- ...
INSERT INTO tFacture (num, date_etabli, clientt, ligness)
VALUES (
    6, SYSDATE, ref_client_1, ListeLigneFacture(
        LigneFacture(3, ref_produit_1),

```

```
LigneFacture(2, ref_produit_2)
)
);
END;
/
```

## Niveau logique

### Objet et référence d'objet

**Un OID ne peut pas être une FK, utiliser string par exemple**

UN OBJET CONTIENT UN OBJET

Type typBureau : <nom : string, numero : integer>

Type typIntervenant : <nom : string, bureau : typBureau> – Composition entre bureau et intervenant, CONTIENT L'OBJET. Il ne faudra pas créer de table pour Bureau

UN OBJET REFERENCE UN OBJET

Bureau de typBureau(#nom) – Table

Type typIntervenant2 : <nom : string, bureau =>o Bureau> – Bureau est une TABLE et non un TYPE, ici l'objet EST REFÉRENCÉ

### Tables imbriquées (collections)

Type RefBureau : <bureau =>o Bureau>

Type ListRefBureau : collection de <RefBureau> – Collection de références

Type ListBureau : collection de <typBureau> – Collection d'objets