REVISION COMPLEMENTS

==========

- Architecture Oracle (instances, processus….)
- Vues du DD Oracle (USER_XXXX, ALL_XXXX, DBA_XXXX……)
- SQL avancé (TP SCOTT…DECODE……)
- PLSQL (TP Budget, Structure, curseurs, variables de travail, exceptions ….)
- TRIGGERS (déclencheurs)

- Fonctions (création, appel)
- Procédures (idem)
- Packages (regroupement de fonctions et procédures

RAISE_APPLICATION_ERROR (alpha, 'message')

Alpha dans l'intervalle -20999 et -20000

EXCEPTIONS

Une anomalie utilisateur peut être traitée comme une exception, pour cela il faut:

- **D'abord la déclarer comme une exception et cela comme suit:**

  DECLARE

  Nom_exception Exception;

- **Le déclencheur**

  Il doit être explicitement déclencher dans le corps du programme PL/SQL, (c.à.d entre Begin…..End) par l'ordre Raise, et cela de la façon suivante:

  IF……..Then

  Raise nom_exception;

- **La traiter dans la partie exception:**

EXECPTION

When nom_exception then traitement;Les erreurs sont de deux types:

- Des erreurs standards détectées par PL/SQL. Dans ce cas, le programme s'arrête et il y a génération d'une erreur avec un code ORAxxxxx. La solution consiste à définir une procédure.

- Des anomalies gérées par l'utilisateur.

PROCEDURES FONCTIONS

Exemple d'une fonction stockée dans une instruction SELECT (a testé sur machine):

Exp: SOIT LA table Patient (Température Number(6,2));

Créer la fonction suivante:

```
SQL>Create or replace Function Conversion(Deg_F in Number)
        Return number is deg_c number;
      Begin
        Deg_c := (5.0 / 9.0) * (Deg_f -32);
        Return Deg_c;
      End Conversion
      /
```

Résultat: Fonction créée.

EXEMPLE d'utilisation de cette fonction dans la Select suivante:

```
SQL> Select Température , Conversion(Température) From Patient;
```

Appel de procédures ou de fonctions stockées.

Concernant SQL*Plus, pour une procédure stockée ne recevant aucun argument, la commande EXECUTE s'utilise de la manière suivante:

EXECUTE nom de la procédure;

==========

EXEMPLES

---Creation de la table Etudiant---

create table etudiant (id_etu number primary key , nom varchar2(20),prenom varchar2(20), age number);

---Creation d'un sequence pour id_etu---

create sequence seq_etu start with 1 increment by 1 MAXVALUE 30;

---Creation Procedure d'insertion---

```
create or replace procedure insertEtu(pnom in varchar2,pprenom in varchar2 ,page in number)
as
begin
        insert into etudiant values (seq_etu.nextval,pnom,pprenom,page);
end insertEtu;
/
exec insertEtu('Ali','majid',22);
exec insertEtu('Brahim','Mouhcine',23);
exec insertEtu('alami','Omar',22);
exec insertEtu('Hajji','Hicham',23);
exec insertEtu('karim','rachid',21);
```

---Creation des Triggers---

----------Creation des Tables----------

/*

create table emp(nom varchar2(50),nemp number(5) primary key,fonction varchar2(30),nsup number(5),date_emb date,salaire number(10),comm number(5),nserv number(5));

create table service(numservice number(5) primary key , nom varchar2(50),lieu varchar2(30),nbemp number(5));

create table salgrade(grade number(5) primary key , losal number(10),hisal number(10));

create table t_resultat(grade number(5) primary key , losal number(10),hisal number(10));

create table t_exeption(grade number(5) primary key , losal number(10),hisal number(10),somme number(20));

alter table emp add constraint nserv foreign key(nserv) references service(numservice);

alter table service add constraint nbemp foreign key (nbemp)references salgrade(grade);

***/**

----------PACKAGE Avec des Procedures D'inserction et de suppression----------

/*

```
create or replace package table_exe1 as

        procedure insert_emp(nom varchar2,nemp number,fonction varchar2,nsup
number,date_emb date,salaire number,comm number,nserv number );

        procedure insert_service(numservice number, nom varchar2,lieu varchar2,nbemp
number);

        procedure insert_salgrade(grade number, losal number,hisal number);


        procedure supp_emp(num number);

        procedure supp_service(num number);

        procedure supp_salgrade(num number);

end table_exe1;

/
```

```
create or replace package body table_exe1 as

        procedure insert_emp(nom varchar2,nemp number,fonction varchar2,nsup
number,date_emb date,salaire number,comm number,nserv number ) is

                begin

                        insert into emp
values(nom,nemp,fonction,nsup,date_emb,salaire,comm,nserv);

                end insert_emp;
```

```
        procedure insert_service(numservice number, nom varchar2,lieu varchar2,nbemp
number) is
                begin
                        insert into service values(numservice,nom,lieu,nbemp);
                end insert_service;

        procedure insert_salgrade(grade number, losal number,hisal number) is
                begin
                        insert into salgrade values(grade,losal,hisal);
                end insert_salgrade;

        procedure supp_emp (num number) is
                begin
                        delete from emp where nemp = num;
                end supp_emp;

        procedure supp_service(num number) is
                begin
                        delete from service where numservice = num;
                end supp_service;

        procedure supp_salgrade(num number) is
```

```
                begin
                        delete from salgrade where grade = num;

                end supp_salgrade;

end table_exe1;
/
*/


----------Execution des Procedures d'insertions pour les Tables----------

/*
exec table_exe1.insert_emp('',,'',,,,,);
exec table_exe1.insert_service(,'','',);
exec table_exe1.insert_salgrade(,,);
exec table_exe1.supp_emp();
exec table_exe1.supp_service();
exec table_exe1.supp_salgrade();


--http://localhost:5560/isqlplus/
*/

----------Creation Des Curseurs----------

/*
raise_application_error(num,"jklhlkjnkl")
```

```
DECLARE
CURSOR  InserCur  IS
select * from salgrade where grade=3;
t_salg salgrade%ROWTYPE;
        begin
                open InserCur;
                loop
                        FETCH InserCur into t_salg;
                        EXIT  WHEN  InserCur%NOTFOUND;
                        insert into t_resulta values(t_salg.grade,t_salg.losal,t_salg.hisal);

                end loop;
                close InserCur;
        end;
/



declare
cursor InserSal is
select * from salgrade where grade=2;
t_salg salgrade%rowtype;

        begin
                open InserSal;
                loop
                        fetch InserSal into t_salg;
                        exit when InserSal%NOTFOUND;
                        insert into t_resultat values(t_salg.grade,t_salg.losal/11,t_salg.hisal/11);
```

```
                end loop;
        close InserSal;
        end;
/



--Exeption--


declare
cursor ExeptionSal is
select * from salgrade;
somme number(10);
t_salg salgrade%rowtype;
        begin
                open ExeptionSal;
                loop
                        fetch ExeptionSal into t_salg;
                        exit when ExeptionSal%NOTFOUND;
                        somme:=t_salg.losal+t_salg.hisal;
                        if somme>=400
                                then
                                        raise_application_error(-20055,'erreur');
                                else
                                        insert into t_exeption
values(t_salg.grade,t_salg.losal,t_salg.hisal,somme);
                        end if;


                end loop;
        close ExeptionSal;
```

end;

/

***/**

----------Affectation/Suppresion des privilége systémes pour l'utilisateur ----------

/*

grant create table create any table to mouh with admin option;

grant drop any table lock any table to mouh with admin option;

revoke create table create any table from mouh;

revoke connect,resource from mouh;

***/**

----------Affectation / Suppression des priviléges Objetc pour un utilisateur----------

/*

connect user1/user1@orcl2

grant select on salgarde to scott;

grant select on salgarde to mouh;

grant insert on salgarde to scott;

grant update (hisal) on salgrade to mouh with grant option;

connect scott/pwd@orcl2

select * from user1.salgrade;

insert into user1.salgrade values(6,500,250);

connect mouh/mouh@orcl2

select * from user1.salgrade;

update user1.salgrade set hisal=999 where grade=6;

update user1.salgrade set losal=111 where grade=6; --impossible puisque mouh na ps l'autorisation de modification sur la colune losal

grant update (hosal) on user1.salgrade to scott; -- impossible puisque user1 na donné ke l'autorisation de distribution sur le chmps (hisal)

grant update (hisal) on user1.salgrade to scott;

connect scott/pwd@orcl2

update user1.salgrade set losal=0909 where grade=6; --impossible puisque mouh il a l'autorisation de distribution des droit juste pour le champs (hisal)

update user1.salgrade set hisal=0000 where grade=6;

connect mouh/mouh@orcl2

revoke update (hisal) on user1.salgrade to scott;

----------Visualiser les autorisations affectés------

connect system/manager@orcl2

desc sys.dba_col_privs

desc sys.dba_tab_privs

select * from sys.dba_col_privs where table_name='SALGRADE';

select * from sys.dba_tab_privs where table_name='SALGRADE';

*/

----------Creation/Suppression/Affectation des roles----------

/*

connect system/manager@orcl2

grant create role to user1;

connect user1/user1@orcl2

--Création--

create role R_manager identified by password;

create role R_informatique ;

--Affectation des priviléges a des roles--

grant connect,resource to R_manager;

grant create user,create role, create profile to R_informatique;

--Affectation d'un role a un role--

grant R_manager to R_informatique;

--Affectation d'un role a un utilisateur--

grant R_manager to scott with admin option;

grant R_informatique to user1 with admin option;

--visualiser les information sur les role crée--

desc dba_roles;

select * from dba_roles;

--modification des parametre d'un role--

alter role R_informatique identified by password;

--Suppression--

drop role R_informatique;

***/**

----------Creation Des Profiles----------

/*

----Exe 1----

--Q1 : un Trigger qui affiche un message 'Opération terminer' pour chaque insertion/suppression/modification

```
create or replace trigger Alert
after delete or insert or update on etudiant
for each row
begin
        dbms_output.enable(1000000);
        dbms_output.put_line('Opération Términer');
end ;
/
exec insertEtu('ttt','ttt',17);
--Désactiver / activer
 ALTER TRIGGER Alert DISABLE;
 ALTER TRIGGER Alert ENABLE;
```

==========

Vues du dictionnaire de données :

• DBA_ROLES : Tous les rôles qui existent dans la base de données

• DBA_ROLE_PRIVS : Rôles accordés à des utilisateurs et à des rôles

• ROLE_ROLE_PRIVS : Rôles accordés à des rôles

• DBA_SYS_PRIVS : Privilèges système accordés à des utilisateurs et à des rôles

• ROLE_SYS_PRIVS : Privilèges système accordés à des rôles

• ROLE_TAB_PRIVS : Privilèges objet accordés à des rôles

• SESSION_ROLES : Rôles activés par l'utilisateur