Revision MLBDA SQL3

creation schema

- utilisation de `create type nom_type as object` pour creer plusieur
- utilisation `create type nom_table as table of nom_type;` pour creer une table de ce type
    - pour creer un table de ref : create type nom_table as table of **ref** nom_type

```
create type T_entre_dans as object (
    quoi      ref T_Piece_Composite,
    quantite  number
);
@compile

-- cardinalit? de l'association : 1-N
create type T_entre_dans_plusieurs as table of T_entre_dans;
@compile

create type T_Piece as object (
    nom         varchar2(30),
        entre_dans  T_entre_dans_plusieurs
) not final; -- dire quelle peut etre heriter
@compile

create type T_Piece_base under T_Piece ( -- under heritage
    est_en ref   T_Matiere,
    member function volume return number
) not final not instantiable ;
@compile
```

instanciation des tables

- syntaxe : `create table nom_table of nom type;`
- si y a des sous tableaux utilisation de `nested table attribut_contenant_soustab as un nom`

```
create table LesMatieres of T_Matiere;
create table LESPIECESB of T_PIECE_BASE nested table entre_dans store as
t1;
create table LesPiecesC of T_PIECE_COMPOSITE nested table contient_pieces
store as t2 nested table  entre_dans store as t3;
```

insertion

- syntaxe : `insert into nom_table values(constructeur(att1,att2))`

```
insert into LesMatieres values (T_Matiere('bois' , 10, 2));
insert into LesMatieres values (T_Matiere('fer' , 5, 3));
insert into LesMatieres values (T_Matiere('ferrite' , 6, 10));
```

```
--d) ajouter la ref d'alice dans les "presents" de la 3eme seance MLBDA
insert into table (
    select s.prensents  -- ici c'est un singleton , son type est element
de son relation ,ici sont type est Etudiants (pluriels)
    from lesUE u, table(u.contenu)s
    where u.nomU = 'MLBDA' and s.numero = 3
    ) values (select ref(a) from LesEtudiants a where a.nom = 'Alice')
```

- exemplee avec utilisation de methode

```
create or replace procedure p1 as
    BOIS REF T_MATIERE;
    FER REF T_MATIERE;
    FERRITE REF T_MATIERE;
    plateau REF T_PIECE;
    pied REF T_PIECE;
    clou REF T_PIECE;
    tab REf T_PIECE;
    boule REF T_PIECE;
    canne ref T_PIECE;
Begin
select ref(m) into BOIS from LesMatieres m where m.nom = 'bois';
select ref(m) into FER from  LesMatieres m where m.nom = 'fer';
select ref(m) into FERRITE from LesMatieres m where m.nom ='ferrite';



insert into LesPiecesB values (T_CYLINDRE('canne', null, BOIS, 2, 30));
insert into lesPiecesB values(T_PARAL('plateau' , null ,BOIS, 1,100,80));
insert into lesPiecesB values(T_SPHERE('boule',null,FER, 30));
insert into LESPIECESB values(T_SPHERE('pied' , null , BOIS, 30));
insert into LEspiecesB values(T_CYLINDRE('clou' , null , FER, 1, 20));
insert into LesPiecesB values(T_CYLINDRE('aimant' , null , FERRITE, 2,
5));


select ref (p) into plateau from LesPiecesB p where p.nom = 'plateau';
select ref (p) into pied from LesPiecesB p where p.nom = 'pied';
select ref (p) into clou from LesPiecesB p where p.nom = 'clou';
select ref (p) into boule from LesPiecesB p where p.nom = 'boule';
select ref (p) into canne from LesPiecesB p where p.nom = 'canne';


insert into  LesPiecesC values('table', null , 100,
```

```
T_CONTIENT_PLUSIEURS(T_CONTIENT(plateau,1), T_CONTIENT(pied,4),
T_CONTIENT(clou,12)));
select ref (p) into tab from LesPiecesC p where p.nom = 'table';

insert into  LesPiecesC values('billard', null , 10,
T_CONTIENT_PLUSIEURS(T_CONTIENT(tab,1), T_CONTIENT(boule,3),
T_CONTIENT(canne,2) ) );
end;
/
@compile
begin
p1;
end;
/
show errors
```

- remarque: possibilite d'inserer dans un sous tableau

```
insert into table (select logements
from LesProprietaires where nom='Max Dupont') -- logement attribut
contenant un sous tableau
select ref(l) from LesLogements l
where adresse='App 3 - 1, rue de Paris, Gentilly';
```

update des donnes

- syntaxe :

```
update nom_table
set attribut = a renouveler  --peut etre requete sql
where condition
```

- exemple :

```
update LesLogements
set proprietaire = (select ref(p) from LesProprietaires p
where p.nom='Max')
where adresse='App 3 - 1, rue de Paris, Gentilly';
```

```
 -- autre solutio avec update
 update table (select u.contenu
 From lesUE u,
 where u.nomU = 'MLBDA'
 )set presents= Etudiants(select ref(a) from LesEtudiants a where a.nom =
'Alice') where numero = 3 -- alice est la seule presente avec cette
```

```
requete, on aura besoin de fair un union des restes pour avoir tous les
etudiants presents
 -- le syntaxe s.presents = s.presents union Etudiants n'est pas supporter
```

les methodes

- des fonction qui peuvent etre appeler directement par un type d'objet
- mettre la signature de methode dans les schema
    - signature : `overriding member function nom_function return type_retour`
- `bulk collect into resultat` pour mettre le resultat du requete dans le variable
- syntaxe :

```
--d)
create or replace type BODY Etudiant as
    member function notes return EnsC1 is
    res EnsC1 --variable affecte le resultat de la fonction
    begin
        select C1(u.code,c.note)
        bulk collect into res -- on utilise bulk collect pour stocker le
resultat dans une variable
        from table(self.contrat) u , table(value(u).noteInscrits) c
        where c.etu = ref(self)

        return res ;
    end;
    --tout autre methodes de etudiant
end;

create or replace type BODY T_cube as -- le type qu'on veut ajouter une
methode
    overriding member function volume return number is
    res number; --les variable
    begin
        res := self.cote * self.cote * self.cote;
        return res;
    end;
end;


create or replace type body T_Agence as
member function getLocataires(dateloc Date) return Ens_Locataires is
resultat Ens_Locataires;
begin
select value(l).locataire bulk collect into resultat
from table(self.logements) v, table (value(v).locations) l where
value(l).debut <= dateloc and value(l).fin >= dateloc; return resultat;
end;
end;
```

Requete SQL3

- les fonction = count , sum , max ,min , avg
- top k: pour avoir les k premiere lignes
- syntax :

```
select
from
where
group by  -- group par attribut,
having
order by
```

- pour prendre un table dans une table

```
select value(e).nomE
from lesUE u , table(u.contenu) s, table(s.presents) e
where u.nomU = 'MLBDA' and s.numero = 3
```

- si on a une reference vers un autre obj et on veut un attribut dans cette objet on utilise `value(ref).att`
- quand est ce que un variable est de type ref , on a besoin de utiliser value() pour extraire les attributs
- le point fait un dereferencement de maniere implicite
- deref prend en argument une expression et renvoie un type unite
- Rmq : value (variable) est une fonction identite le type retourne est egale au type de l'argument

```
Select value(m).nom
From LesDepartements d, table(d.services) s, table(value(s).membres) m
Where d.nom = 'Informatique';
```