UNIVERSITÉ LUMIÈRE LYON 2
UNIVERSITÉ DE LYON

Bases de données objets

M2 Informatique
Spécialité IDS-IIDEE
Année 2011-2012
Jérôme Darmont

http://eric.univ-lyon2.fr/~jdarmont/

Plan du cours

- Introduction
- Types d'objets et hiérarchies d'héritage
- Références et associations
- Collections
- Méthodes
- Évolutions de schéma
- Vues objet
- Dictionnaire des données
- Privilèges

Objets et bases de données

- Bases de données relationnelles
  - Paquetages
  - BLOB (*Binary Large Objects*)
  - Structures de données tabulaires uniquement
- Bases de données objets
  - Complexité et manque de base théorique du modèle de données
  - Performances limitées
  - Applications spécifiques (CAO/DAO, multimédia…)

Objets et bases de données

- Bases de données relationnelles-objets
  - Extension du modèle de données relationnel à l'aide de types de données abstraits
  - Références directes à un OID (réduction du nombre de jointures)
  - Attributs : types structurés et collections
    ⇒ perte de la 1FN (NF2 : *Non First Normal Form*)
  - Méthodes définies au niveau des types
  - Héritage entre types

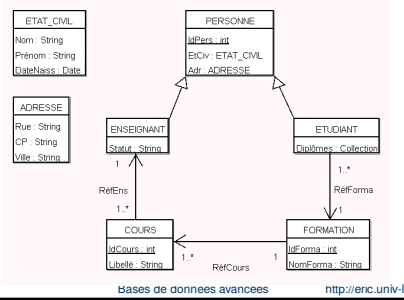
Avantages de l'approche relationnelle-objet

- Encapsulation des données des tables
- Préservation des acquis des systèmes relationnels (indépendance données/traitements, fiabilité, performance, compatibilité ascendante…)
- Extension du langage SQL (norme SQL3)
- Mise en œuvre des concepts objets (classes, héritage, méthodes)

Inconvénients de l'approche relationnelle-objet

- Modèle de données qui ne repose pas sur des principes simples / une théorie rigoureuse
- Pas de syntaxe commune de la part des éditeurs de SGBD (IBM, Oracle, PostgreSQL, SAP…)
- Migration relationnel → objet facile, mais retour en arrière complexe

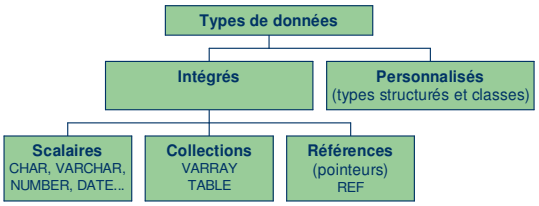
Base de données exemple



Plan du cours

- ✓ Introduction
- Types d'objets et hiérarchies d'héritage
- Références et associations
- Collections
- Méthodes
- Évolutions de schéma
- Vues objet
- Dictionnaire des données
- Privilèges

Types de données



Types d'objets personnalisés

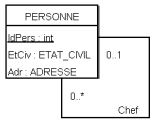
- Exemples de création de types
  - CREATE OR REPLACE TYPE T_ETAT_CIVIL AS OBJECT (
    Nom VARCHAR(50),
    Prenom VARCHAR(50),
    DateNaiss DATE)
    /
  - CREATE OR REPLACE TYPE T_ADRESSE AS OBJECT (
    Rue VARCHAR(100),
    CP CHAR(5),
    Ville VARCHAR(50))
    /

Types d'objets personnalisés

- CREATE OR REPLACE TYPE T_PERSONNE AS OBJECT (
  IdPers NUMBER(5),
  EtCiV T_ETAT_CIVIL,
  Adr T_ADRESSE)
  /

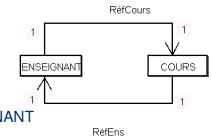
- Affichage de la structure d'un type
  - Ex. DESC T_PERSONNE
- Destruction d'un type
  - Ex. DROP TYPE T_PERSONNE;

Types récursifs



- Utilisation d'une référence
  - Ex. CREATE OR REPLACE TYPE T_PERSONNE AS OBJECT (
    IdPers NUMBER(5),
    EtCiV T_ETAT_CIVIL,
    Adr T_ADRESSE,
    chef REF T_PERSONNE)
    /

Types mutuellement dépendants



- Utilisation de types incomplets
  - Ex.
```
CREATE TYPE T_ENSEIGNANT
/ -- Type incomplet
CREATE TYPE T_COURS AS OBJECT (…,
        RefEns REF T_ENSEIGNANT)
/
CREATE TYPE T_ENSEIGNANT AS OBJECT(…,
        RefCours REF T_COURS)
/ -- On a complété le type incomplet
```
  - Destruction : DROP TYPE T_ENSEIGNANT FORCE;

Persistance des objets

- Objets colonnes : types structurés décrivant des champs de table(s) relationnelle(s)
- Objets lignes : stockés en tant que n-uplets d'une table objet
- Objets non persistants : utilisés en mémoire dans un programme PL/SQL

Objets colonnes

- Table relationnelle
  - Ex.
```
CREATE TABLE PERSONNE (
        IdPers NUMBER(5) PRIMARY KEY,
        EtCiv T_ETAT_CIVIL,
        Adr T_ADRESSE);
```
- Instanciation
  - Ex.
```
INSERT INTO PERSONNE VALUES (1000,
        T_ETAT_CIVIL('Darmont', 'Jérôme', '15/01/1972'),
        T_ADRESSE('5 av. P. M.France', '69676', 'Bron'));
```
- Interrogation
  - Ex.
```
SELECT IdPers, p.EtCiv.Nom, p.Adr.Ville
FROM PERSONNE p;
```


Objets lignes

- Table objet
  - Ex.
```
CREATE TABLE PERSONNE OF T_PERSONNE (
        CONSTRAINT pers_pk PRIMARY KEY (IdPers));
```
  - Baser l'identifiant objet (OID) sur la clé primaire : ajout de la clause OBJECT IDENTIFIER IS PRIMARY KEY
- Instanciation
  - Ex.
```
INSERT INTO PERSONNE VALUES (1000,
        T_ETAT_CIVIL('Darmont', 'Jérôme', '15/01/1972'),
        T_ADRESSE('5 av. P. M.France', '69676', 'Bron'));
```
- Interrogation
  - Ex.
```
SELECT IdPers, p.EtCiv.Nom, p.Adr.Ville
FROM PERSONNE p;
```


Objets non persistants

- Déclaration PL/SQL
  - Ex.
```
DECLARE
        une_personne T_PERSONNE;
```
- Instanciation
  - Ex.
```
BEGIN
une_personne := NEW T_PERSONNE (1000,
        T_ETAT_CIVIL('Darmont', 'Jérôme', '15/01/1972'),
        T_ADRESSE('5 av. P. M.France', '69676', 'Bron'));
```
- Utilisation
  - Ex.
```
une_personne.IdPers := 1111;
une_personne.Adr.Ville := 'Lyon';
DBMS_OUTPUT.PUT_LINE(une_personne.EtCiv.Nom);
```


Mise à jour d'objets persistants

- Modification
  - Ex.
```
UPDATE PERSONNE p
SET p.Adr.Ville = 'Lyon'
WHERE IdPers = 1000;
```
- Suppression
  - Ex.
```
DELETE FROM PERSONNE p
WHERE p.EtCiv.Nom = 'Darmont';
```

Chargement d'objets persistants

- Dans une requête SQL
  - Ex. SELECT VALUE(p) FROM PERSONNE p;
- Dans un bloc PL/SQL
  - Ex.

```
DECLARE
        une_personne T_PERSONNE;
BEGIN
        SELECT VALUE(p) INTO une_personne
                FROM PERSONNE p
                WHERE IdPers = 1000;
        DBMS_OUTPUT.PUT_LINE
                (une_personne.EtCiv.Nom);
END;
/
```


Héritage

- Superclasse
  - Ex.

```
CREATE TYPE T_PERSONNE AS OBJECT (
        IdPers NUMBER(5),
        T_EtCiV ETAT_CIVIL,
        T_Adr ADRESSE)
[NOT INSTANTIABLE] -- Classe abstraite (optionnel)
NOT FINAL
/
```

- Sous-classe
  - Ex.

```
CREATE TYPE T_ENSEIGNANT UNDER T_PERSONNE (
        Statut VARCHAR(20))
/
```


Substitution de types

- Objets : Un objet peut contenir une instance de tout sous-type de son propre type.
- Références : Une référence qui cible un type peut peut pointer vers une instance de tout sous-type de ce type.
- Collections : Une collection d'éléments d'un type peut contenir des instances de tout sous-type de ce type.

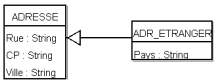
Substitution de types

- Exemple

```
INSERT INTO PERSONNE VALUES (
      T_PERSONNE(2000,
            T_ETAT_CIVIL('Durand', 'Maurice', '02/02/1902'),
            T_ADRESSE ('Place Bellecourt', '69002', 'Lyon')));

INSERT INTO PERSONNE VALUES (
      T_ENSEIGNANT(3000,
            T_ETAT_CIVIL('Grand', 'Aimé', '01/08/1962'),
            T_ADRESSE ('Quai Claude Bernard', '69007', 'Lyon'),
            'Professeur'));
```


Fonctions sur les objets colonnes



- SELECT TREAT(Adr AS T_ADR_ETRANGER)
  FROM PERSONNE;
  -- Adresses des personnes habitant à l'étranger
- SELECT VALUE(p) FROM PERSONNE p
  WHERE p.Adr IS OF(T_ADR_ETRANGER);
  -- Caractéristiques des personnes habitant à l'étranger
- SELECT p.EtCiv.Nom, SYS_TYPEID(Adr)
  FROM PERSONNE p;
  -- Nom et identifiant (nº) du type des adresses des personnes

Plan du cours

Mise en œuvre de références

- Définition
  - Ex.
```
CREATE TYPE T_COURS AS OBJECT(
        IdCours NUMBER(2),
        Libelle VARCHAR(50),
        RefEns REF T_ENSEIGNANT)
/
```
  - NB : RefEns pointe vers l'OID d'un enseignant.
- Instanciation
  - Ex.
```
CREATE TABLE COURS OF T_COURS(
        CONSTRAINT pk_cours PRIMARY KEY(IdCours));
```
  - Une référence peut aussi être une colonne d'une table relationnelle.

Mise en œuvre de références

- Insertion
  - Ex.
```
INSERT INTO COURS
VALUES(1, 'Economie', (SELECT REF(e) FROM
        ENSEIGNANT e WHERE e.IdPers = 9000));
```
  - NB : L'OID de ENSEIGNANT ne doit pas être basé sur la clé primaire pour que la référence fonctionne.
- Mise à jour
  - Ex.
```
UPDATE COURS c
SET c.RefEns = (SELECT REF(e) FROM
        ENSEIGNANT e WHERE e.IdPers = 9002)
WHERE c.IdCours = 1;
```


Jointures implicites

- Dans la clause WHERE
  - Ex.
```
SELECT c.Libelle FROM COURS c
WHERE c.RefEns.EtCiv.Nom = 'Darmont';
```
- Dans la clause SELECT
  - Ex.
```
SELECT c.RefEns.EtCiv.Nom
FROM COURS c
WHERE c.Libelle = 'Économie';
```
- Déréférencement
  - Ex.
```
SELECT DEREF(RefEns) FROM COURS
WHERE IdCours = 1;
```


Substitution de références

- Exemple

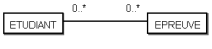
```
CREATE TYPE T_REF_PERS(Ref REF T_PERSONNE)
/
CREATE TABLE INDEX_PERS OF T_REF_PERS;

INSERT INTO INDEX_PERS VALUES
      (SELECT REF(p) FROM PERSONNE p
      WHERE p.IdPers = 1000);

INSERT INTO INDEX_PERS VALUES
      (SELECT REF(e) FROM ENSEIGNANT e
      WHERE e.IdPers = 9000);
```


Associations

- Associations 1-N
  - Ex. COURS-ENSEIGNANT
- Associations M-N



  - Ex.
```
CREATE TABLE NOTATION (
        RefEtu REF T_ETUDIANT,
        RefEpr REF T_EPREUVE,
        Note NUMBER(4,1));
```


Intégrité référentielle

- Cohérence parent-enfant
  - Ex. La suppression d'un enseignant auquel des cours sont rattachés doit être impossible.
- Instanciation
  - Ex.
```
CREATE TABLE COURS OF T_COURS(
        CONSTRAINT pk_cours PRIMARY KEY(IdCours),
        CONSTRAINT fk_cours_ref_ens1
            RefEns REFERENCES ENSEIGNANT);
```

Intégrité référentielle

- Cohérence enfant-parent
  - Ex. L'insertion d'une référence vers un enseignant inexistant doit être impossible.
- Instanciation
  - Ex.
```
CREATE TABLE COURS OF T_COURS(
    CONSTRAINT pk_cours PRIMARY KEY(IdCours),
    CONSTRAINT fk_cours_ref_ens1
        RefEns REFERENCES ENSEIGNANT,
    CONSTRAINT fk_cours_ref_ens2
        CHECK (RefEns IS NOT NULL));
```


Plan du cours

Tables imbriquées

- Définition : collection non-ordonnée et non limitée en taille
- Stockage : dans une table séparée
- Création d'un type TABLE
  - Ex.
```
CREATE TYPE TAB_COURS AS
    TABLE OF T_COURS
/
```


Tableaux prédimensionnés

- Définition : collection ordonnée et limitée en taille
- Stockage : dans un attribut BLOB de la table
- Création d'un type VARRAY
  - Ex.
```
CREATE TYPE TAB_DIPLOMES AS
    VARRAY(10) OF VARCHAR(20)
/
```


Stockage : table relationnelle

- Exemple

```
CREATE TABLE ENSEIGNANT (
    IdPers NUMBER(5),
    EtCiV T_ETAT_CIVIL,
    Adr T_ADRESSE,
    Statut VARCHAR(20),
    ListeCours TAB_COURS,
    CONSTRAINT pk_ens PRIMARY KEY(IdPers))
    NESTED TABLE ListeCours
        STORE AS TABLE_IMBRIQUEE_COURS;
```


Stockage : table objet

- Type
  - Ex.
```
CREATE TYPE T_ENSEIGNANT AS OBJECT (
    IdPers NUMBER(5), EtCiV T_ETAT_CIVIL,
    Adr T_ADRESSE, Statut VARCHAR(20),
    ListeCours TAB_COURS)
/
```
- Table
  - Ex.
```
CREATE TABLE ENSEIGNANTS OF T_ENSEIGNANT
    (CONSTRAINT pk_ens PRIMARY KEY(IdPers))
    NESTED TABLE ListeCours
        STORE AS TABLE_IMBRIQUEE_COURS;
```

Mises à jour

- Ajout d'un élément parent
  - Ex. INSERT INTO ENSEIGNANT VALUES (1000,
    T_ETAT_CIVIL('Darmont', 'Jerome', '15/01/1972'),
    T_ADRESSE ('5 av. P. M.France', '69676', 'Bron'),
    'PR',
    TAB_COURS(T_COURS(1, 'BD'), T_COURS(2, 'Web')));
- Ajout dans la collection
  - Ex. INSERT INTO TABLE
    (SELECT ListeCours FROM ENSEIGNANT
    WHERE IdPers = 1000)
    VALUES (T_COURS(3, 'UNIX'));

Mises à jour

- Modification
  - Ex. UPDATE TABLE
    (SELECT ListeCours FROM ENSEIGNANT
    WHERE IdPers = 1000) t
    SET t.Libelle = 'WEB'
    WHERE t.Libelle = 'Web';
- Modification d'éléments
  - Ex. UPDATE TABLE
    (SELECT ListeCours FROM ENSEIGNANT
    WHERE IdPers = 1000) t
    SET VALUE(t) = T_COURS(10, 'Bases de données')
    WHERE t.IdCours = 1;

Mises à jour

- Suppression
  - Ex. DELETE FROM TABLE
    (SELECT ListeCours FROM ENSEIGNANT
    WHERE IdPers = 1000) t
    WHERE t.IdCours = 10;
- Note : La clause TABLE n'est pas utilisable avec les tableaux VARRAY, qui ne peuvent être directement modifiés qu'avec PL/SQL.

Méthodes PL/SQL

- EXISTS(i) retourne TRUE si le $i^{ème}$ élément de la collection existe.
- COUNT retourne le nombre d'éléments dans la collection.
- LIMIT retourne la taille maximum de la collection (NULL pour les tables imbriquées).
- EXTEND(n) augmente la taille de la collection de n éléments. EXTEND est équivalent à EXTEND(1).

Méthodes PL/SQL

- TRIM(n) supprime n éléments en fin de collection (la taille de la collection est diminuée). TRIM est équivalent à TRIM(1).
- DELETE(i), DELETE effacent le $i^{ème}$ élément et tous les éléments de la collection, respectivement (tables imbriquées).
- FIRST et LAST retournent l'indice du premier et du dernier élément de la collection, resp.
  NB : FIRST=1 et LAST=COUNT pour un tableau.
- PRIOR(n) et NEXT(n) retournent l'indice de l'élément précédent et suivant de l'élément d'indice n, resp.

Interrogation

- Requêtes de désemboîtement (pseudojointure)
  - Ex. SELECT e.EtCiv.Nom, e.EtCiv.Prenom,
    e.Statut, c.Libelle
    FROM ENSEIGNANT e, TABLE(e.ListeCours) c;
- Extraction d'une collection
  - Ex. SELECT c.Libelle FROM TABLE
    (SELECT ListeCours
    FROM ENSEIGNANT
    WHERE IdPers = 1000) c;

Interrogation

- Curseur imbriqué
  - Ex.
```
SELECT e.EtCiv.Nom, e.EtCiv.Prenom,
       CURSOR(SELECT c.Libelle
              FROM TABLE (ListeCours) c)
FROM ENSEIGNANT e;
```
  - Liste nom, prénom, puis tous les cours de cet enseignant, puis enseignant suivant, etc.
  - Résultat non assimilable dans une table.

Collections de références

- Types (2 étapes)
  - Ex.
```
CREATE TYPE T_REF_COURS AS
    OBJECT(RefCours REF T_COURS)
/
```
  - Ex.
```
CREATE TYPE TAB_COURS AS
    TABLE OF T_REF_COURS
/
```
- Table : pas de changement

Problèmes d'intégrité référentielle

- Associations 1-N et M-N possibles
- Contraintes :
  - On ne doit pas pouvoir affecter un OID inexistant à la référence. Possible sous Oracle.
  - Dans une association 1-N, chaque référence est unique. Impossible sous Oracle.
  - On ne doit pas pouvoir supprimer un objet d'une table s'il est pointé par une référence. Impossible sous Oracle.

⇒ Programmation de déclencheurs

Collections multiniveaux

- Exemple de création des types
  -
```
CREATE TYPE TAB_NOTES AS
  TABLE OF NUMBER(4,1)
/
```
  -
```
CREATE TYPE T_EPREUVE AS
  OBJECT(RefCours REF T_COURS, Notes TAB_NOTES)
/
```
  -
```
CREATE TYPE TAB_EPREUVES AS TABLE OF T_EPREUVE
/
```
  -
```
CREATE TYPE T_ETUDIANT AS
  OBJECT (IdPers NUMBER(5), EtCiV T_ETAT_CIVIL,
  Adr T_ADRESSE, Resultats TAB_EPREUVES)
/
```


Collections multiniveaux

- Exemple de création de table

```
CREATE TABLE ETUDIANT OF T_ETUDIANT
    (CONSTRAINT pk_etu PRIMARY KEY(IdPers))
    NESTED TABLE Resultats STORE AS TI1_RES
        (NESTED TABLE Notes STORE AS TI2_NOTES);
```


Plan du cours

Caractéristiques des méthodes sous Oracle

- Encapsulation non automatique
- Surcharge possible
- Trois catégories de méthodes
  - Membre (MEMBER) : s'applique à un objet
  - Statique (STATIC) : s'applique à un type (encapsulation)
  - Constructeur (CONSTRUCTOR) : instanciation
- Appel : Requête, autre méthode, PL/SQL, programme externe

Déclaration

Exemple

```
CREATE TYPE T_COURS AS OBJECT(
        IdCours NUMBER(2),
        Libelle VARCHAR(50),
        RefEns REF T_ENSEIGNANT,
        MEMBER PROCEDURE ChangeLib (NouvLib VARCHAR),
        STATIC FUNCTION NbCours RETURN NUMBER,
        CONSTRUCTOR FUNCTION T_COURS (
                IdC NUMBER,
                Lib VARCHAR,
                IdE NUMBER) RETURN SELF AS RESULT)
/
```


Implémentation

Exemple

```
CREATE TYPE BODY T_COURS AS

MEMBER PROCEDURE ChangeLib (NouvLib VARCHAR) IS
BEGIN
        UPDATE les_cours
                SET Libelle = NouvLib
                WHERE IdCours = SELF.IdCours;
END;

-- À suivre
```


Implémentation

```
STATIC FUNCTION NbCours RETURN NUMBER IS
        n INTEGER;
BEGIN
        SELECT COUNT(*)
                INTO n
                FROM les_cours;
        RETURN n;
END;

-- À suivre
```


Implémentation

```
CONSTRUCTOR FUNCTION T_COURS (IdC NUMBER,
    Lib VARCHAR, IdE NUMBER) RETURN SELF AS RESULT IS
BEGIN
        SELF.IdCours := IdC;
        SELF.Libelle := Lib;
        SELECT REF(e) INTO SELF.RefEns
                FROM ENSEIGNANT e
                WHERE e.IdPers = IdE;
        RETURN;
END;

END;
/
```


Appel

Exemple

```
DECLARE
        un_cours T_COURS;
        nb_cours INTEGER;
        nouveau_cours T_COURS;
BEGIN
        SELECT VALUE(c) INTO un_cours FROM COURS c
                WHERE c.IdCours=1;
        un_cours.ChangeLib('Informatique');
        nb_cours := T_COURS.NbCours();
        nouveau_cours := NEW T_COURS(5, 'Statistiques', 1000);
        INSERT INTO COURS VALUES(nouveau_cours);
END;
/
```

Surcharge

- Définition multiple d'une méthode
  - Au sein d'une même classe
  - Dans une sous classe
- Exemple 1

```
CREATE TYPE T_COURS AS OBJECT(
        -- (…)
        STATIC PROCEDURE Ajout(IdC Number,
                Lib VARCHAR, IdE NUMBER),
        STATIC PROCEDURE Ajout(Lib VARCHAR,
                IdE NUMBER))
/
```


Surcharge

```
CREATE TYPE BODY T_COURS AS
    STATIC PROCEDURE Ajout(IdC Number,
                Lib VARCHAR, IdE NUMBER) IS
    BEGIN
        INSERT INTO COURS VALUES (IdC, Lib, (SELECT
          REF(e) FROM ENSEIGNANT e WHERE e.IdPers = IdE));
    END;
    STATIC PROCEDURE Ajout(Lib VARCHAR, IdE NUMBER) IS
        new_id INTEGER;
    BEGIN
        SELECT MAX(IdCours)+1 INTO new_id FROM COURS;
        INSERT INTO COURS VALUES (new_id, Lib, (SELECT
          REF(e) FROM ENSEIGNANT e WHERE e.IdPers = IdE));
    END;
END;
/
```


Surcharge

- Exemple 2

```
CREATE TYPE T_PERSONNE AS OBJECT (
        -- (…)
        NOT FINAL MEMBER PROCEDURE Affiche)
/

CREATE TYPE T_ENSEIGNANT UNDER T_PERSONNE (
        -- (…)
        OVERRIDING MEMBER PROCEDURE Affiche)
/
```


Comparaison d'objets : Fonction MAP

- Exemple de définition

```
CREATE TYPE T_FRACTION AS OBJECT (
        numerateur INTEGER,
        denominateur INTEGER,
        MAP MEMBER FUNCTION Calcul RETURN REAL)
/

CREATE TYPE BODY T_FRACTION AS
        MAP MEMBER FUNCTION Calcul RETURN REAL IS
        BEGIN
                RETURN numerateur / denominateur;
        END;
END;
/
```


Comparaison d'objets : Fonction MAP

- Exemple d'utilisation

```
DECLARE
        f1 T_FRACTION;
        f2 T_FRACTION;
BEGIN
        f1 := NEW T_FRACTION (10, 4);
        f2 := NEW T_FRACTION (249, 100);
        IF f1 > f2 THEN
                -- (…)
        END IF;
END;
/
```


Comparaison d'objets : Fonction ORDER

- Utilisation pour des objets trop complexes pour être traités par MAP
- Une seule fonction MAP ou ORDER par classe
- Exemple de définition

```
CREATE TYPE T_FRACTION AS OBJECT (
        num INTEGER,
        den INTEGER,
        ORDER MEMBER FUNCTION Compare (f T_FRACTION)
                RETURN INTEGER)
/
```

Comparaison d'objets : Fonction ORDER

```
CREATE TYPE BODY T_FRACTION AS
      ORDER MEMBER FUNCTION Compare (f T_FRACTION)
            RETURN INTEGER IS
      BEGIN
            IF (f.num/f.den) < (SELF.num/SELF.den) THEN
                  RETURN –1;
            ELSIF (f.num/f.den) > (SELF.num/SELF.den) THEN
                  RETURN 1;
            ELSE
                  RETURN 0;
            END IF;
      END;
END;
/
```


Comparaison d'objets : Fonction ORDER

- Exemple d'utilisation

```
DECLARE
      f1 T_FRACTION;
      f2 T_FRACTION;
BEGIN
      f1 := NEW T_FRACTION (10, 4);
      f2 := NEW T_FRACTION (249, 100);
      IF f1.Compare(f2) > 0 THEN
            -- (…)
      END IF;
END;
/
```


Plan du cours

- ✓ Introduction
- ✓ Types d'objets et hiérarchies d'héritage
- ✓ Références et associations
- ✓ Collections
- ✓ Méthodes
- – Évolutions de schéma
- – Vues objet
- – Dictionnaire des données
- – Privilèges

Opérations possibles

- Ajout/suppression d'attributs
- Ajout/suppression de méthodes
- Augmentation (taille, précision) d'un attribut
- Modification des caractéristiques d'héritage (FINAL) et de persistance (INSTANTIABLE)

Éléments dépendants

- Tables objets du type / Tables relationnelles contenant un attribut du type
- Type référençant la table / Sous-type
- Programmes PL/SQL utilisant le type
- Index référençant un attribut modifié du type
- Vues créées à partir du type ou d'une table dépendant du type

Marche à suivre

- Modifier la structure du type
  ALTER TYPE…
- Recompiler le corps du type
  CREATE OR REPLACE TYPE BODY…
- Rendre les tables conformes
  ALTER TABLE…
- Recompiler les programmes PL/SQL

Exemples

- Attributs
  - ALTER TYPE T_COURS DROP ATTRIBUTE RefEns CASCADE INCLUDING TABLE DATA;
  - ALTER TYPE T_COURS ADD ATTRIBUTE (NumEns NUMBER(5)) CASCADE INCLUDING TABLE DATA;
  - ALTER TYPE T_COURS MODIFY ATTRIBUTE (IdCours NUMBER(5)) CASCADE INCLUDING TABLE DATA;
- Si l'option INCLUDING TABLE DATA n'est pas utilisée
  - ALTER TABLE COURS DROP UNUSED COLUMNS;
  - ALTER TABLE COURS UPGRADE INCLUDING DATA;

Exemples

- Méthodes
  - ALTER TYPE T_COURS DROP MEMBER PROCEDURE ChangeLib (NouvLib VARCHAR);
  - ALTER TYPE T_COURS ADD STATIC PROCEDURE Suppr(IdC NUMBER);
- Recompilation du corps
  - CREATE OR REPLACE TYPE BODY T_COURS AS…
  - Suppression du code de la procédure ChangeLib
  - Ajout du code de la procédure Suppr
  - Les autres procédures/fonctions restent inchangées

Exemples

- Hiérarchie d'héritage
  - ALTER TYPE T_COURS NOT INSTANTIABLE;
    Possible si une table ne dépend pas du type
  - ALTER TYPE T_COURS INSTANTIABLE CASCADE INCLUDING TABLE DATA;
    Toujours possible, n'affecte pas les tables
  - ALTER TYPE T_COURS FINAL CASCADE INCLUDING TABLE DATA;
    Possible si le type n'a pas de sous-type
  - ALTER TYPE T_COURS NOT FINAL CASCADE INCLUDING TABLE DATA; -- *Substitution interdite ou* CASCADE CONVERT TO SUBSTITUTABLE;

Plan du cours

Intérêt

- Même utilité que les vues relationnelles
  - Simplification de l'accès aux données
  - Sauvegarde indirecte de requêtes complexes
  - Présentation des données variable selon les utilisateurs (sécurité)
  - Support de l'indépendance logique
- Migration facilitée vers la technologie objet
  - Utilisation de méthodes sur des données tabulaires
  - Interfaçage aisé avec des langages orientés objets

Type structure de vue

- Exemple

```
CREATE TYPE T_ENS_COMPLET AS OBJECT (
      IdPers NUMBER(5),
      EtCiv T_ETAT_CIVIL,
      Adr T_ADRESSE,
      Statut VARCHAR(20))
/
```

Création de vue objet

- Exemple

```
CREATE VIEW VUE_ENS OF T_ENS_COMPLET
WITH OBJECT IDENTIFIER(IdPers) AS
      SELECT p.IdPers, EtCiv, Adr, Statut
      FROM PERSONNE p, ENSEIGNANT e
      WHERE p.IdPers = e.IdPers;
```


Vues objets modifiables

- Exemple

```
CREATE TRIGGER CASCADE INSERT
INSTEAD OF INSERT ON VUE_ENS FOR EACH ROW
BEGIN
      INSERT INTO PERSONNE
          VALUES (:NEW.IdPers, :NEW.EtCiv, :NEW.Adr);
      INSERT INTO ENSEIGNANT
          VALUES (:NEW.IdPers, :NEW.Statut);
END;
/
```


Vues contenant une collection

- Exemple

```
CREATE TYPE T_DIPLOME AS OBJECT(Libelle VARCHAR(20))
CREATE TYPE TAB_DIPLOME AS TABLE OF T_DIPLOME
CREATE TYPE T_ETU_DIP AS OBJECT
      (NumPers NUMBER(5), Diplomes TAB_DIPLOME)

CREATE VIEW VUE_ETU  OF T_ETU_DIP
WITH OBJECT IDENTIFIER(NumPers) AS
      SELECT e.Numpers, CAST( MULTISET(
            SELECT Libelle FROM DIPLOME d
            WHERE d.NumPers = e.NumPers) AS TAB_DIPLOME
            ) AS Diplomes
      FROM ETUDIANT e;
```


Plan du cours

- ✓ Introduction
- ✓ Types d'objets et hiérarchies d'héritage
- ✓ Références et associations
- ✓ Collections
- ✓ Méthodes
- ✓ Évolutions de schéma
- ✓ Vues objet
- – Dictionnaire des données
- – Privilèges

Types, tables objets, collections

- USER_TYPES (TYPE_NAME, SUPERTYPE_NAME, FINAL, INSTANTIABLE…)
- USER_TYPE_VERSIONS (TYPE_NAME, VERSION#…)
- USER_OBJECT_TABLES (TABLE_NAME, OBJECT_ID_TYPE, TABLE_TYPE, NESTED…)
- USER_COLL_TYPES (TYPE_NAME, COLL_TYPE, UPPER_BOUND, ELEM_TYPE_NAME…)

Attributs, méthodes

- USER_TYPE_ATTRS (TYPE_NAME, ATTR_NAME, ATTR_TYPE_NAME, LENGTH, PRECISION, INHERITED…)
- USER_TYPE_METHODS (TYPE_NAME, METHOD_NAME, METHOD_TYPE, FINAL, INSTANTIABLE, OVERRIDING, INHERITED…)
- USER_METHOD_PARAMS (TYPE_NAME, METHOD_NAME, PARAM_NAME, PARAM_MODE, PARAM_TYPE_NAME…)
- USER_METHOD_RESULTS (TYPE_NAME, METHOD_NAME, RESULT_TYPE_NAME…)

Plan du cours

- ✓ Introduction
- ✓ Types d'objets et hiérarchies d'héritage
- ✓ Références et associations
- ✓ Collections
- ✓ Méthodes
- ✓ Évolutions de schéma
- ✓ Vues objet
- ✓ Dictionnaire des données
- – Privilèges

Droits d'invocation

- Exécution des méthodes membres
  - Avec les privilèges du créateur du type
    CREATE TYPE nom_type AUTHID DEFINER
    AS OBJECT(…)
    /
  - Avec les privilèges de l'utilisateur courant
    CREATE TYPE nom_type AUTHID CURRENT_USER
    AS OBJECT(…)
    /

Privilèges objets

- EXECUTE
  - Définition d'une table objet du type
  - Définition d'un attribut du type dans une table relationnelle
  - Déclaration d'une variable ou d'un paramètre du type
  - Invocation des méthodes du type
- UNDER
  - Création de sous-types
- Transmission des privilèges
  - Commande GRANT *privilège* ON *objet* TO *utilisateur*
    Ex. GRANT EXECUTE, UNDER
    ON COURS TO PUBLIC
    WITH GRANT OPTION;

Synonymes de types

- Création
  - CREATE SYNONYM TypeCours FOR darmont.T_COURS;
- Modification
  - CREATE SYNONYM TypeCours FOR darmont.T_COURS2;
- Renommage
  - RENAME TypeCours TO CoursType;
- Suppression
  - DROP SYNONYM CoursType;

Plan du cours

- ✓ Introduction
- ✓ Types d'objets et hiérarchies d'héritage
- ✓ Références et associations
- ✓ Collections
- ✓ Méthodes
- ✓ Évolutions de schéma
- ✓ Vues objet
- ✓ Dictionnaire des données
- ✓ Privilèges