

## Modèle objet - relationnel SQL99

Université de Nice Sophia-Antipolis  
Version 1.0.2 – 5/11/11  
Richard Grin

## Introduction

## Modèle objet-relationnel

- ❑ Le modèle objet-relationnel (OR) reprend le modèle relationnel en ajoutant quelques notions qui peuvent être très utiles dans certaines circonstances
- ❑ La compatibilité est ascendante : les anciennes applications relationnelles fonctionnent dans le monde OR
- ❑ La norme SQL99 (SQL3) reprend beaucoup d'idées du modèle OR

## Pourquoi étendre le modèle relationnel ? (1)

- ❑ La reconstitution d'objets complexes éclatés sur plusieurs tables relationnelles est coûteuse car elle occasionne de nombreuses jointures
- ❑ Pour échapper aux éclatements-jointures, l'OR réhabilite
  - les références qui permettent d'implanter des structures complexes
  - les attributs multivalués (tableaux, ensembles ou listes)

## Pourquoi étendre le modèle relationnel ? (2)

- ❑ Même si ce problème n'est pas inhérent au modèle relationnel, SQL92 ne permet pas de créer de nouveaux types, ce qui implique un manque de souplesse et une interface difficile avec les applications orientées objet
- ❑ L'OR (et SQL99) permet de définir de nouveaux types utilisateur simples ou complexes (*User data type*), avec des fonctions ou procédures associées comme dans les classes des langages objet

## Pourquoi étendre le modèle relationnel ? (3)

- ❑ L'OR supporte l'héritage de type pour profiter du polymorphisme et faciliter la réutilisation

## Pourquoi ne pas passer directement aux SGBD Objet ? (1)

- Le relationnel a ses avantages, en particulier
  - sa grande facilité et efficacité pour effectuer des recherches complexes dans des grandes bases de données
  - la facilité de spécifier des contraintes d'intégrité sans programmation
  - une théorie solide et des normes reconnues

R. Grin

Objet-relationnel

page 7

## Pourquoi ne pas passer directement aux SGBD Objet ? (2)

- Inertie de l'existant : de très nombreuses bases relationnelles en fonctionnement
- Manque de normalisation pour les SGBDO ; trop de solutions propriétaires
- SGBDOO moins souple que le relationnel pour s'adapter à plusieurs applications et à l'augmentation de charge
- Peu d'informaticiens formés aux SGBDO
- Le modèle OR peut permettre un passage en douceur

R. Grin

Objet-relationnel

page 8

## Nouvelles possibilités de l'OR

- Définir de nouveaux types complexes avec des fonctions pour les manipuler
- Une colonne peut contenir une collection (ensemble, sac, liste)
- Ligne considérée comme un objet, avec un identificateur (*Object Identifier* OID)
- Utilisation de références aux objets
- Extensions du langage SQL (SQL3 ou SQL99) pour la recherche et la modification des données

R. Grin

Objet-relationnel

page 9

## Les problèmes de l'OR

- Ne s'appuie pas sur une théorie solide comme le modèle relationnel
- Manque de standard de fait : implantations différentes, et encore partielles, dans les divers SGBDs

R. Grin

Objet-relationnel

page 10

## SQL99 (SQL3)

- Cette partie du cours s'appuie autant que possible sur les spécifications de SQL99
- Le langage de programmation SQL99 ajoute à SQL2 des variables et instructions de contrôle pour en faire un langage procédural complet ; ce cours ne porte pas sur ces extensions
- Les exemples concrets sont donnés dans le langage SQL de la version 10g d'Oracle ; les différences avec SQL99 seront signalées

R. Grin

Objet-relationnel

page 11

## Types définis par l'utilisateur

R. Grin

Objet-relationnel

page 12

## Nouveaux types prédéfinis

- ❑ Le relationnel objet ajoute des types prédéfinis à la norme SQL (étudiés plus loin dans le cours) :
  - référence
  - collection
  - LOB (lié aux objets de grande taille)

## Les types utilisateur

- ❑ Le développeur peut aussi créer ses propres types de données :
  - types « distincts »
  - types structurés

## Types « distincts »

- ❑ Ces types permettent de mieux différencier les domaines des colonnes ; ils sont formés à partir des types de base :

```
CREATE TYPE codePays as char(2);  
CREATE TYPE matricule as integer;
```

- ❑ Par exemple, pour différencier les domaines des colonnes `matricule` et `numDept`
- ❑ Ces types s'utilisent avec les mêmes instructions que le type de base sous-jacent
- ❑ Pas supporté par Oracle

## Types structurés

- ❑ Correspondent aux classes des langages objets
- ❑ Ils peuvent contenir des constructeurs, attributs (≈ variables d'instances), fonctions et procédures (≈ méthodes)
- ❑ Les membres peuvent être public, protected ou private
- ❑ Les fonctions et procédures peuvent être écrites en SQL ou en un autre langage
- ❑ Supportent l'héritage

## Création d'un type de données

- ❑ La syntaxe est semblable à celle de la création d'une table :

```
CREATE TYPE departement_type AS OBJECT  
(  
  numDept integer,  
  nomD varchar(30),  
  lieu varchar(30));
```

- ❑ Les types créés par l'utilisateur peuvent être utilisés comme les types natifs SQL (integer, varchar,...)

## Création d'un type de données

- ❑ Un type ne peut contenir de contrainte d'intégrité
- ❑ La commande « `create or replace type` » permet de redéfinir un type s'il existe déjà
- ❑ Une colonne ne peut avoir le nom d'un type ; il est donc conseillé de suffixer les noms de type avec « `_type` » ou « `_t` »

## Fonctions dans les types

```
CREATE TYPE departement_type AS OBJECT
(numDept integer,
 nomD varchar(30),
 lieu varchar(30),
 MEMBER FUNCTION getLieu RETURN varchar);
```

```
CREATE TYPE BODY departement_type AS
MEMBER FUNCTION getLieu RETURN varchar IS
begin
    return lieu;
end;
```

R. Grin

Objet-relational

page 19

## Héritage

- ❑ Les types supportent l'héritage multiple avec le mot-clé UNDER :

```
create type employe_type as object
(matr integer, nom varchar(30),
 sal numeric(8,2))
not final;
```

```
create type commercial_type
under employe_type
(comm numeric(8,2))
not final;
```

- ❑ Un type est **final** par défaut

R. Grin

Objet-relational

page 20

## Ajout d'un attribut dans un type

```
alter type employe_type
add attribute date_naissance date
cascade;
```

Propage aux tables  
déjà construites à  
partir du type

R. Grin

Objet-relational

page 21

## Ajout d'une méthode/fonction à un type

```
alter type employe_type
add member
function age return integer
cascade;
```

R. Grin

Objet-relational

page 22

## Supprimer un type

```
drop type employe_type;
```

R. Grin

Objet-relational

page 23

## Type de ligne

- ❑ SQL99 possède aussi la notion de type de ligne qui correspond aux structures du C : c'est un ensemble non encapsulé d'attributs
- ❑ Le type peut être nommé ou non

R. Grin

Objet-relational

page 24

## Type de ligne non nommé

```
❑ create table EMP
  (nomE varchar(35),
   adresse ROW(numero integer,
                rue varchar(30),...))
```

## Type de ligne nommé

```
❑ CREATE ROW TYPE adresse_t
  (numero integer,
   rue varchar(30),...)
```

❑ On peut ensuite utiliser ce type pour une déclaration d'attribut ou même pour créer une table à partir de ce type (comme pour les autres types)

## Vues du dictionnaire des données

❑ Sous Oracle :

- ❑ `USER_TYPES` pour les types (et les collections)
- ❑ `USER_TYPE_ATTRS` pour attributs des types
- ❑ `USER_TYPE_METHODS` pour les méthodes des types
- ❑ `USER_OBJECT_TABLES` pour les tables objet-relacionnelles

❑ Sous sqlplus d'Oracle :

```
describe departement_type
```

## Tables

## Création d'une table à partir d'un type

- ❑ Les données d'un type ne sont persistantes que si elles sont rangées dans une table
- ❑ On peut créer des tables comme en SQL92
- ❑ On peut aussi créer des tables à partir d'un type de données

## Création d'une table à partir d'un type

❑ Soit le type `employe_type` :

```
CREATE TYPE employe_type AS OBJECT
  (matricule integer,
   nom varchar(30),
   . . .
   dept integer);
```

❑ On peut créer une table à partir de ce type et indiquer des contraintes d'intégrité :

```
create table employe OF employe_type
  (primary key (matricule));
```

## Héritage de tables

- ❑ Une table peut hériter d'une ou plusieurs tables
- ❑ Pas supporté par Oracle 10g

## Création de table à partir d'un type dérivé

```
create table commerciaux of commercial_type
(constraint pk_com primary key(matr));
```

## Caractéristiques d'une table objet-relationnelle

- ❑ Une table est une table objet-relationnelle si elle a été construite à partir d'un type (`create table ... OF`)
- ❑ Les lignes de ces tables sont considérées comme des objets avec un identifiant (OID, *Object Identifier*)
- ❑ On peut utiliser des références pour désigner les lignes de ces tables (pas possible pour les autres tables)

## Vues du dictionnaire des données

- ❑ `USER_OBJECT_TABLES` pour les tables objet-relationnelles

## Insertion de données

- ❑ On ajoute des données comme avec une table normale :

```
insert into commerciaux
(matr, nom, sal, comm)
values (234, 'TITI', 3200, 600);
```

## Insertion avec constructeur

- ❑ On peut aussi utiliser le « constructeur du type » avec lequel la table a été construite :

```
insert into employe values (
employe_type(125, 'Dupond', ...));
```

- ❑ Si le type est un type utilisé par un autre type, l'utilisation du constructeur du type est obligatoire :

```
insert into employe
(matr, nom, sal, adresse)
values (1, 'Toto', 12000,
adresse_type(12, 'Victor Hugo', 'Nice'))
```

## Afficher les valeurs des types

```
select nom, e.adresse.rue
from employe e
```

## Modifications

- ❑ Utiliser la notation pointée comme en SQL92 mais avec un alias si un type est concerné :

```
update employe e
  set salaire = 12000,
      e.adresse.numero = 23
  where nom = 'Dupond';
```

- ❑ SQL99 utilise la notation « .. » pour désigner un attribut d'une colonne de type structuré :

```
update employe e
  set e.adresse..numero = 12
  where nom = 'Dupond';
```

## Appel de procédure ou fonction

- ❑ 

```
select nom, age(e)
from employe e
where age(e) < 40
```

Le « this » est passé en paramètre

- ❑ Sous Oracle :

```
select nom, e.age()
from employe e
where e.age() < 40
```

## Références

## Références

- ❑ On peut indiquer dans la définition d'un type qu'un attribut contient des références (et non des valeurs) à des données d'un autre type ; la syntaxe est « REF nom-du-type » :

```
create type employe_type as object
  (matricule integer,
   nom varchar(30),
   . . .
   dept REF dept_type);
```

## Exemple de select avec référence

- ❑ La notation pointée permet de récupérer les attributs d'un type dont on a un pointeur

- ❑ Lieu de travail des employes (avec Oracle) :

```
select nom, e.dept.lieu
from employe e
```

- ❑ En SQL99 :

```
select nom, e.dept->lieu
from employe e
```

- ❑ Attention, l'alias e est indispensable

## Insertions avec référence

```
insert into employe values (  
1230, 'Durand', ..., NULL);
```

pointeur  
NULL

```
insert into employe(matricule, nom, dept)  
select 1240, 'Dupond', REF(d)  
from dept d  
where d.numDept = 10;
```

référence vers  
le dept de  
numéro 10

R. Grin

Objet-relationalnel

page 43

## DEREF

❑ La fonction **DEREF** renvoie un objet dont on a la référence (penser à tester si la référence n'est pas NULL)

❑ Exemple :

```
select deref(dept) from emp  
where matricule = 500
```

❑ Affiche

```
DEPARTEMENT_TYPE(10,'Finances','Nice')
```

R. Grin

Objet-relationalnel

page 44

## Modification d'une référence

```
update employe  
set dept =  
(select REF(d)  
from dept d  
where numDept = 10)  
where matricule = 7500;
```

Attention, cette instruction peut très bien mettre la valeur NULL dans la colonne **dept** car le select renvoie NULL si le département de numéro 10 n'existe pas !

R. Grin

Objet-relationalnel

page 45

## Contrainte NOT NULL

❑ Pour éviter le problème de l'exemple précédent, il faut ajouter la contrainte NOT NULL sur la colonne dept :

```
dept REF dept_type NOT NULL
```

R. Grin

Objet-relationalnel

page 46

## Contrainte sur les références

- ❑ Le type « REF dept\_type » restreint le type référencé mais pas la table référencée
- ❑ Une telle référence peut référencer une valeur de n'importe quelle colonne de table qui a le type dept\_type
- ❑ La clause SCOPE restreint la colonne référencée ; elle peut être ajoutée lors de la définition d'une table

R. Grin

Objet-relationalnel

page 47

## Exemple de SCOPE

```
dept REF dept_type  
scope is dept_table
```

indique que dept référencera une ligne de la table dept\_table (et pas une ligne d'une autre table créée à partir du type dept\_type)

R. Grin

Objet-relationalnel

page 48

## Référence perdue

- ❑ SCOPE ne suffit pas pour imposer une contrainte stricte
- ❑ Ainsi la référence peut être « pendante » (*dangling*) ou « perdue », c'est-à-dire ne pas correspondre à une ligne existante, si la ligne référencée au départ a été ensuite supprimée

R. Grin

Objet-relationnel

page 49

## References

- ❑ Pour éviter les références perdues il faut remplacer la contrainte SCOPE par une contrainte REFERENCES que l'on ajoute dans la table associée au type (comme avec les tables relationnelles) :

```
constraint r_emp_dept
foreign key(dept)
references dept_table
```

R. Grin

Objet-relationnel

page 50

## Collections

R. Grin

Objet-relationnel

page 51

## Types de collections

- ❑ Pour représenter une colonne multivaluée, on peut utiliser les collections ou les tableaux :
  - tableaux de taille fixe (array)
  - ensembles, au sens mathématiques ; pas de doublons (set)
  - sacs, avec des doublons (bag ou multiset)
  - listes, ordonnées et indexées par un entier (list)
- ❑ D'autres types de collections peuvent être ajoutées par les SGBD

R. Grin

Objet-relationnel

page 52

## Exemple de collection

```
create type employe_type
(matricule integer,
 nom varchar(30),
 prenom LIST(varchar(15)),
 enfants SET(personne),
 . . .);
```

R. Grin

Objet-relationnel

page 53

## Utilisation d'une collection

- ❑ On peut utiliser une collection comme une table en la faisant précéder par le mot-clé TABLE :

```
select nom from employe E
where nom in
(select *
 from TABLE(E.prenoms))
```

- ❑ On peut aussi faire afficher une collection comme un tout :

```
select nom, prenom from employe
```

R. Grin

Objet-relationnel

page 54

## Les collections avec Oracle 10g

- Oracle 10g n'offre que 2 types de collections :
  - table imbriquée (NESTED TABLE) qui est une collection non ordonnée et non limitée en nombre d'éléments
  - tableau prédimensionné (VARRAY) qui est une collection d'éléments de même type, ordonnée et limitée en taille

R. Grin

Objet-relationalnel

page 55

## Tables imbriquées

- Une table relationnelle (pas nécessairement OR) peut contenir une ou plusieurs tables imbriquées
- Pas étudié dans ce cours ; se reporter au manuel Oracle pour plus de précisions

R. Grin

Objet-relationalnel

page 56

## Tableaux dimensionnés

- Un VARRAY est une collection ordonnée et limitée en nombre, d'éléments d'un même type
- On peut imbriquer plusieurs tableaux dimensionnés en utilisant des pointeurs sur des tableaux

R. Grin

Objet-relationalnel

page 57

## Exemple de VARRAY

```
create type telephones_type as  
VARRAY(3) OF varchar(10)
```

```
create type personne_type as object  
(nom varchar(30),  
telephones telephones_type)
```

```
insert into personne (nom, telephones)  
values('Dupond',  
telephones_type('0492077987',  
'0492074567'))
```

R. Grin

Objet-relationalnel

page 58

## Référence

- Programmer objet avec Oracle de Christian Soutou Vuibert

R. Grin

Objet-relationalnel

page 59