Base de Données Relationnel Objet

Par: S.Selmane

Plan de cours

- Introduction : R, OO, RO
- SQL3
- Les tables imbriquées
- Les tables objets

Introduction

Si le modèle logique relationnel a prouvé sa puissance et sa fiabilité au cours des 50 dernières années, les nouveaux besoins de l'informatique industrielle ont vu l'émergence de structures de données complexes mal adaptées à une gestion relationnelle. La naissance du courant *orienté objet* et des langages associées (Java et C++ par exemple) ont donc également investi le champ des SGBD afin de proposer des solutions pour étendre les concepts du relationnel et ainsi mieux répondre aux nouveaux besoins de modélisation.

Les atouts du modèle relationnel

- Fondé sur une théorie rigoureuse et des principes simples
- Mature, fiable, performant
- Indépendance programme et données
- SGBDR : les plus utilisés, connus, maîtrisés
- SQL une implémentation standard du modèle relationnel, avec des API pour la plupart des langages de programmation
- Les SGBDR incluent des outils performants de gestion de requêtes, de générateurs d'applications, d'administration, d'optimisation, etc.

Les inconvénients du modèle relationnel

- La structure de donnée en tables est pauvre d'un point de vue de la modélisation logique
- Le *mapping* MCD vers MLD entraîne une perte de sémantique
- La manipulation de structures relationnelles par des langages objets entraîne une *impedance mismatch*, c'est à dire un décalage entre les structures de données pour le stockage et les structures de données pour le traitement (ce qui implique des conversions constantes d'un format à l'autre)
- La 1NF est inappropriée à la modélisation d'objets complexes

Les inconvénients du modèle relationnel (2)

- La normalisation entraîne la genèse de structures de données complexes et très fragmentées, qui peuvent notamment poser des problèmes de performance ou d'évolutivité
- Le SQL doit toujours être combiné à d'autres langages de programmation pour être effectivement mis en œuvre
- La notion de méthode ne peut être intégrée au modèle logique, elle doit être gérée au niveau de l'implémentation physique
- Les types de données disponibles sont limités et non extensibles

Les SGBDOO

Introduction

- Les SGBDOO ont été créés pour gérer des structures de données complexes, en profitant de la puissance de modélisation des modèles objets et de la puissance de stockage des BD classiques.

Objectifs des SGBDOO :

- Offrir aux langages de programmation orientés objets des modalités de stockage permanent et de partage entre plusieurs utilisateurs
- Offrir aux BD des types de données complexes et extensibles
- Permettre la représentation de structures complexes et/ou à taille variable

Avantages des SGBDOO :

- Le schéma d'une BD objet est plus facile à appréhender que celui d'une BD relationnelle (il contient plus de sémantique, il est plus proche des entités réelles)
- L'héritage permet de mieux structurer le schéma et de factoriser certains éléments de modélisation
- La création de ses propres types et l'intégration de méthodes permettent une représentation plus directe du domaine
- L'identification des objets permet de supprimer les clés artificielles souvent introduites pour atteindre la 3NF et donc de simplifier le schéma
- Les principes d'encapsulation et d'abstraction du modèle objet permettent de mieux séparer les BD de leurs applications (notion d'interface).

Inconvénient des SGBDOO :

- Gestion de la persistance et de la coexistence des objets en mémoire (pour leur manipulation applicative) et sur disque (pour leur persistance) complexe
- Gestion de la concurrence (transactions) plus difficile à mettre en œuvre
- Interdépendance forte des objets entre eux
- Gestion des pannes
- Complexité des systèmes (problème de fiabilité)
- Problème de compatibilité avec les SGBDR classiques

Les SGBDRO

Introduction

Les SGBDRO sont nés du double constat de la puissance nouvelle promise par les SGBDOO et de l'insuffisance de leur réalité pour répondre aux exigences de l'industrie des BD classiques. Leur approche est plutôt d'introduire dans les SGBDR les concepts apportés par les SGBDOO plutôt que de concevoir de nouveaux systèmes.

Les SGBDRO

Objectifs des SGBDRO

- Gérer des données complexes (temps, géo-référencement, multimédia, types utilisateurs, etc.)
- Rapprocher le modèle logique du modèle conceptuel
- Réduire l'*impedance mismatch*
- Réduire les pertes de performance liées à la normalisation et aux jointures

Les SGBDRO

***Définition* Modèle relationnel-objet**

- Modèle relationnel étendu avec des principes objet pour en augmenter les potentialités.
- Synonymes : Modèle objet-relationnel

Les SGBDRO

Type utilisateur

Le concept central du RO est celui de type utilisateur, correspondant à la classe en POO, qui permet d'injecter la notion d'objet dans le modèle relationnel.

Les apports principaux des types utilisateurs sont :

- La gestion de l'imbrication (données structurées et collections)
- Les méthodes et l'encapsulation
- L'héritage et la réutilisation de définition de types
- L'identité d'objet et les références physiques

Les SGBDRO

Tables imbriquées et tables objets

Le modèle RO apporte deux nouveautés distinctes et indépendantes à ne pas confondre :

- Les tables imbriquées (*nested model*) qui permettent de dépasser les limites de la première forme normale et de limiter la fragmentation de l'information ;
- Les tables objets qui permettent d'ajouter les identifiants d'objets (OID), les méthodes et l'héritage aux tables classiques.
- Avec un SGBDRO on peut ne pas utiliser ces deux extensions (on reste alors en relationnel), utiliser l'une des deux ou bien les deux conjointement.

Introduction au SQL3 et aux types utilisateurs

Objectifs

- Connaître les caractéristiques principales du modèle relationnel-objet
- Connaître l'extension du LDD pour la déclaration de type (CREATE TYPE)

Types utilisateurs

- Type de données créé par le concepteur d'un schéma relationnel-objet, qui encapsule des données et des opérations sur ces données. Ce concept est à rapprocher de celui de classe d'objets.
- Synonymes : Type de données utilisateur, Type de données abstrait

Types utilisateurs

```
Type nom_type : <
   attribut1:typeAttribut1,
   attribut2:typeAttribut2,
   ...
   attributN:typeAttributN,
   =methode1:(paramètres) typeRetourné1,
   =methode2:(paramètres) typeRetourné2,
   =...
   =methodeN:(paramètres) typeRetournéN
>
```

Les types des attributs (ou des méthodes) d'un type peuvent également être des types utilisateurs. C'est ce principe qui permet l'imbrication de données relationnelles.

Déclaration des types utilisateurs en SQL3 (extension au LDD)

```
CREATE TYPE nom_type AS OBJECT (
   nom_attribut1 type_attribut1
   ...
   MEMBER FUNCTION nom_fonction1 (parametre1 IN|OUT type_parametre1, ...) RETURN
type_fonction1
   ...
) [NOT FINAL];
/
CREATE TYPE BODY nom_type
   IS
   MEMBER FUNCTION nom_fonction1 (...) RETURN type_fonction1
      IS
      BEGIN
      ...
      END ;
   MEMBER FUNCTION nom_fonction2 ...
      ...
   END ;
END ;
/
```

Les Types VARRAY ou TABLE

Syntaxe:

- CREATE TYPE <nom-type> AS (VARRAY <longueur> | TABLE) OF <nom-type2>;
- Exemple
- Create TYPE Prénom as Varray (3) of varchar(10) ;
- Create TYPE Telephone as Table of varchar(10);

Héritage de type

```
CREATE TYPE sous_type UNDER sur_type (
   //Déclarations spécifiques ou surcharges
) ;
```

Pour être héritable, un type doit être déclaré avec la clause optionnelle NOT FINAL.

Les tables imbriquées dans le modèle relationnel-objet

- Type typBureau : <centre:char, batiment:char, numero:int>

- Type typListeTelephones : collection de <entier>

- Type typSpecialite : <domaine:char, specialite:char>

- Type typListeSpecialites : collection de <typSpecialite>

tIntervenant (#nom:char, prenom:char, bureau:typBureau, ltelephones:typListeTelephones, lspecialites:typListeSpecialites)

Exemple

pknom	prenom	bureau			liste-telephones	listes-specialites	
Crozat	Stéphane	centre	batiment	numero		Domaine	Technologie
		PG	K	256	0687990000	BD	SGBDR
					0912345678	Doc	XML
					0344231234	BD	SGBDRO
Vincent	Antoine	centre	batiment	numero		Domaine	Technologie
		R	C	123	0344231235	IC	Ontologie
					0687990001	Base de données	SGBDRO

```
CREATE OR REPLACE TYPE typBureau AS OBJECT (
centre char(2),
batiment char(1),
numero number(3)
);
/
CREATE OR REPLACE TYPE typListeTelephones AS TABLE OF number(10);
/
CREATE OR REPLACE TYPE typSpecialite AS OBJECT (
domaine varchar2(15),
technologie varchar2(15)
);
/
CREATE OR REPLACE TYPE typListeSpecialites AS TABLE OF
typSpecialite;
/
```

```
CREATE TABLE tIntervenant (
pknom varchar2(20) PRIMARY KEY,
prenom varchar2(20) NOT NULL,
bureau typBureau,
ltelephones typListeTelephones,
lspecialites typListeSpecialites
)
NESTED TABLE ltelephones STORE AS tIntervenant_nt1,
NESTED TABLE lspecialites STORE AS tIntervenant_nt2;
```

Insert

```
INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
VALUES ('Crozat', 'Stéphane', typBureau('PG','K',256),
typListeTelephones (0687990000,0912345678,0344231234),
typListeSpecialites (typSpecialite ('BD','SGBDR'), typSpecialite('Doc','XML'),
typSpecialite('BD','SGBDRO'))
);

INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
VALUES ('Vincent', 'Antoine', typBureau('R','C',123),
typListeTelephones (0344231235,0687990001),
typListeSpecialites (typSpecialite ('IC','Ontologies'), typSpecialite('BD','SGBDRO'))
);
```

Select (enregistrement imbriqué)

SELECT pknom, prenom, i.bureau.centre FROM tIntervenant i;

PKNOM	PRENOM	BUREAU.CENTRE
Crozat	Stéphane	PG
Vincent	Antoine	R

Select (collection de scalaires imbriquée)

```
SELECT i.pknom, t.*
FROM tIntervenant i, TABLE(i.ltelephones) t
```

PKNOM	COLUMN_VALUE
Crozat	687990000
Crozat	912345678
Crozat	344231234
Vincent	344231235
Vincent	687990001

Select (collection d'enregistrements imbriquée)

```
SELECT i.pknom, s.*
FROM tIntervenant i, TABLE(i.lspecialites) s
```

PKNOM	DOMAINE	TECHNOLOGIE
Crozat	BD	SGBDR
Crozat	Doc	XML
Crozat	BD	SGBDRO
Vincent	IC	Ontologies
Vincent	BD	SGBDRO

Le modèle imbriqué

Le principe du modèle imbriqué est qu'un attribut d'une table ne sera plus seulement valué par une unique valeur scalaire (principe de la 1NF), mais pourra l'être par un vecteur (enregistrement) ou une collection de scalaires ou de vecteurs, c'est à dire une autre table.

Synonyme : nested model

Table imbriquée

Table relationnel-objet dont chaque attribut peut être défini pour contenir :

- une variable scalaire,
- ou un enregistrement,
- ou une collection de scalaires,
- ou une collection enregistrements.

Exemple

```
CREATE OR REPLACE TYPE typBureau AS OBJECT (
centre char(2),
batiment char(1),
numero number(3)
);
/

CREATE TABLE tIntervenant (
pknom varchar2(20) PRIMARY KEY,
prenom varchar2(20) NOT NULL,
bureau typBureau
);
```

pknom	prenom	Bureau		
Crozat	Stéphane	centre	batiment	numero
		PG	K	256
Vincent	Antoine	centre	batiment	numero
		R	C	123

Insertion d'enregistrements imbriqués

```
INSERT INTO tIntervenant (pknom, prenom, bureau)
VALUES ('Crozat', 'Stéphane', typBureau('PG','K',256));

INSERT INTO tIntervenant (pknom, prenom, bureau)
VALUES ('Vincent', 'Antoine', typBureau('R','C',123));
```

Accès aux attributs des enregistrements imbriqués

SELECT pknom, prenom, i.bureau.centre FROM tIntervenant i;

PKNOM	PRENOM	BUREAU.CENTRE
Crozat	Stéphane	PG
Vincent	Antoine	R

Les collections imbriquées

- Une collection est un type de données générique défini afin de supporter un ensemble d'objets (scalaires ou enregistrements).
- Synonymes : Collection d'objets

```
CREATE OR REPLACE TYPE typListeTelephones AS TABLE OF number(10);
/

CREATE TABLE tIntervenant (
pknom varchar2(20) PRIMARY KEY,
prenom varchar2(20) NOT NULL,
ltelephones typListeTelephones
)
NESTED TABLE ltelephones STORE AS tIntervenant_nt1;
```

```
CREATE OR REPLACE TYPE typSpecialite AS OBJECT (
domaine varchar2(50),
technologie varchar2(50)
);
/
CREATE OR REPLACE TYPE typListeSpecialites AS TABLE OF typSpecialite;
/
CREATE TABLE tIntervenant (
pknom varchar2(20) PRIMARY KEY,
prenom varchar2(20) NOT NULL,
lspecialites typListeSpecialites
)
NESTED TABLE lspecialites STORE AS tIntervenant_nt2;
```

Insertion d'enregistrements et de collections imbriqués

```
INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
VALUES ('Crozat', 'Stéphane',
typBureau('PG','K',256),
typListeTelephones (0687990000,0912345678,0344231234),
typListeSpecialites (typSpecialite ('BD','SGBDR'), typSpecialite('Doc','XML'),
typSpecialite('BD','SGBDRO'))
);
INSERT INTO tIntervenant (pknom, prenom, bureau, ltelephones, lspecialites)
VALUES ('Vincent', 'Antoine', typBureau('R','C',123),
typListeTelephones (0344231235,0687990001),
typListeSpecialites (typSpecialite ('IC','Ontologies'), typSpecialite('BD','SGBDRO'))
);
```

Sélection dans les collections imbriquées

```
SELECT i.pknom, t.*
FROM tIntervenant i, TABLE(i.ltelephones) t
```

PKNOM	COLUMN_VALUE
Crozat	687990000
Crozat	912345678
Crozat	344231234
Vincent	344231235
Vincent	687990001

Accès à une collection imbriquée d'enregistrement

```
SELECT i.pknom, s.*
FROM tIntervenant i, TABLE(i.lspecialites) s
```

PKNOM	DOMAINE	TECHNOLOGIE
Crozat	BD	SGBDR
Crozat	Doc	XML
Crozat	BD	SGBDRO
Vincent	IC	Ontologies
Vincent	BD	SGBDRO

Accès aux colonnes d'une table imbriquée

```
SELECT i.pknom, t.COLUMN_VALUE, s.domaine
FROM tIntervenant i, TABLE(i.ltelephones) t,
TABLE(i.lspecialites) s
```

PKNOM	COLUMN_VALUE	DOMAINE
Crozat	687990000	BD
Crozat	687990000	Doc
Crozat	687990000	BD
Crozat	912345678	BD
Crozat	912345678	Doc
Crozat	912345678	BD
Crozat	344231234	BD
Crozat	344231234	Doc
Crozat	344231234	BD
Vincent	344231235	IC
Vincent	344231235	BD
Vincent	687990001	IC
Vincent	687990001	BD

Définition de tables objets

- Une table peut être définie en référençant un type de données plutôt que par des instructions LDD classiques. On parle alors de table objet.
- Synonymes : table-objet, table d'objets
- Les enregistrements d'une table-objet peuvent être identifiés par un OID

```
CREATE OR REPLACE TYPE typIntervenant AS OBJECT(
pknom varchar2(20),
prenom varchar2(20)
);
/
CREATE TABLE tIntervenant OF typIntervenant (
PRIMARY KEY(pknom),
prenom NOT NULL
);
```

Les OID (identificateurs d'objets)

Le modèle relationnel-objet permet de disposer d'identificateurs d'objet (OID).

Une table objet dispose d'un OID pour chacun de ses tuples.

Caractéristiques

- L'OID est une référence unique pour toute la base de données qui permet de référencer un enregistrement dans une table objets.
- L'OID est une référence physique (adresse disque) construit à partir du stockage physique de l'enregistrement dans la base de données

Avantages

- Permet de créer des associations entre des objets sans effectuer de jointure (gain de performance).
- Fournit une identité à l'objet indépendamment de ses valeurs (clé artificielle).
- Fournit un index de performance maximale (un seul accès disque).
- Permet de garder en mémoire des identificateurs uniques dans le cadre d'une application objet, et ainsi gérer la persistance d'objets que l'on ne peut pas garder en mémoire, avec de bonnes performances (alors que sinon il faudrait exécuter des requêtes SQL pour retrouver l'objet).

Inconvénient

- Plus de séparation entre le niveau logique et physique.
- L'adresse physique peut changer si le schéma de la table change (changement de la taille d'un champ par exemple)
- Manipulation des données plus complexes, il faut un langage applicatif au dessus de SQL pour obtenir, stocker et gérer des OID dans des variables.

Références entre enregistrements avec les OID

```
CREATE OR REPLACE TYPE typCours AS OBJECT(
pkannee number(4),
pknum number(2),
titre varchar2(50),
type char(2),
refintervenant REF typIntervenant,
debut date,
);
/
```

Insertion de références par OID (INSERT)

- Pour faire une référence avec un OID, il est nécessaire de récupérer l'OID correspondant à l'enregistrement que l'on veut référencer.
- L'OID est accessible grâce à la syntaxe REF en SQL3.

```
INSERT INTO table1 (champs1, ..., clé_étrangère)
SELECT 'valeur1', ...,  REF(alias)
FROM table2 alias
WHERE clé_table2='valeur';
```

Insertion de références par OID (INSERT)

- La référence par OID permet la navigation des objets sans effectuer de jointure, grâce à la syntaxe suivante :

```
SELECT c.pkannee, c.pknum, c.refintervenant.pknom
 FROM tCours c
```

PKANNEE	PKNUM	REFINTERVENANT.PKNOM
2003	1	CROZAT
2003	2	CROZAT