Bases de données relationnelles-objets

Guillaume Raschia — Polytech'Nantes

Date de la dernière modification : 5 janvier 2010

[Source : traduction libre des planches de Jeff. Ullman, Stanford]

Au menu

Convergence des modèles

Modèle relationnel et paradigme Objet

- Paradigme objet : types de données complexes
  - arbres, graphes, données géo-référencées, données temporelles, données multimédia, *etc.*
- Modèle relationnel : requêtes complexes
- Modèle relationnel-objet : tirer le meilleur des deux

Évolution des SGBDs

- Échec (relatif) des BDOOs : efficacité limitée
- Extension objet :
  - intégration des principes du paradigme objet
  - la relation comme abstraction fondamentale

SQL3 vs. Oracle

Définition de l'extension objet

- SQL-99 (SQL3) est le standard
- Certains SGBDs adoptent une approche différente
- Dans la suite de l'exposé : usage ponctuel des fonctions et de la syntaxe Oracle

Types utilisateur

- Un type utilisateur (*User Defined Type* ou UDT en anglais) est pour l'essentiel une définition de classe, avec une structure et des méthodes
- Deux usages :
  1. Type d'enregistrements (*rowtype*) : le schéma d'une relation
  2. Type d'attribut d'une relation

Définition de type utilisateur

CREATE TYPE <nom> AS (
    <liste de couples attribut-type>
);

- Avec Oracle :
  1. Ajout du mot-clé OBJECT : CREATE . . . AS OBJECT
  2. Symbole '/' de terminaison pour forcer le stockage du type

Définition de type utilisateur

Exemple

```
CREATE TYPE TypeBar AS (
  nom        CHAR(20),
  adr       CHAR(20)
);
CREATE TYPE TypeBière AS (
  nom        CHAR(20),
  brasserie CHAR(20)
);
```

Références

- Si T est un type utilisateur, alors REF T est le type d'une référence à T, c-à-d un pointeur vers un objet de type T
- La référence d'un objet est assimilée à un identifiant (*Object ID* ou oid en anglais)
- Contrairement aux oid, une REF est accessible, bien que sans signification

Exemple de REF

```
CREATE TYPE TypeMenu AS (
  bar         REF TypeBar,
  bière       REF TypeBière,
  prix        FLOAT
);
```

- Les objets de type TypeMenu ressemblent à ça :

@ objet BarType	@ objet BièreType	3.00

UDT comme type d'enregistrements

- Le schéma d'une relation peut être défini par un type d'enregistrements, plutôt que par une séquence de couples attribut-type
- Syntaxe :
  CREATE TABLE <nom de la relation> OF <nom du type>

Exemple : création d'une relation

```
CREATE TABLE Bars OF TypeBar {
  PRIMARY KEY (nom)};
CREATE TABLE Bières OF TypeBière {
  PRIMARY KEY (nom)};
CREATE TABLE Ventes OF TypeMenu {
  PRIMARY KEY (bar, bière),
  FOREIGN KEY ... };
}
```

- Les contraintes sont des propriétés de relations, non de types

N-uplets de relations *rowtype*

- Techniquement, la relation Bars, déclarée de type TypeBar, n'est pas un ensemble de couples (nom, adr)
- Bars est une relation unaire dont les n-uplets sont des objets avec deux champs : nom et adr
- Chaque UDT a un constructeur de type de même nom, qui enveloppe les objets de ce type

Exemple : constructeur de type

- La requête
  `SELECT * FROM Bars;`
- La réponse produit des ≪ n-uplets ≫ tels que :
  `TypeBar('Bar"oc', 'cours des 50-otages')`

Accès aux valeurs d'un *rowtype*

- Avec Oracle, la notation pointée fonctionne
- Il est néanmoins judicieux de déclarer des alias pour chaque relation dès lors que des fonctionnalités objet sont invoquées
- Exemple :

```
SELECT bb.nom, bb.adr FROM Bars bb;
```

Accès aux valeurs : approche SQL3

- Chaque attribut d'un UDT a un *get*eur et un *set*eur
  - le générateur pour A n'admet pas de paramètre : $A()$
  - le mutateur pour A admet comme paramètre la nouvelle valeur : $A(v)$

Exemple : accès aux valeurs en SQL3

- Requête avec Oracle :
  `SELECT bb.nom, bb.adr FROM Bars bb;`
- en SQL3 :

```
SELECT bb.nom(), bb.adr()
FROM Bars bb;
```

Insertion d'enregistrements *rowtype*

Avec Oracle

- Instruction INSERT usuelle
- Relation unaire qui appelle le constructeur de type
- Exemple :

```
INSERT INTO Bars VALUES (
TypeBar('Bar"oc', 'cours des 50-otages')
);
```

Insertion à la mode SQL3

1. Déclarer une variable X du type de la relation, et l'initialiser à l'aide du constructeur
2. Invoquer les mutateurs pour l'assignation des valeurs aux attributs de X
3. Insérer X dans la relation

Exemple : insertion en SQL3

- Extrait d'une procédure stockée ; i.e. en PSM, disposant de la variable nvBar

```
SET nvBar = TypeBar();
nvBar.nom('Bar"oc');
nvBar.adr('cours des 50-otages');
INSERT INTO Bars VALUES (nvBar);
```

UDT comme type d'attribut

- Un UDT peut être
  - le type d'un attribut de relation (CREATE TABLE)
  - le type d'un champ d'une autre déclaration d'UDT
- Le nom de l'UDT est utilisé pour désigner le type de l'attribut

Exemple : type d'attribut

```
CREATE TYPE TypeAdr AS (
  rue       CHAR(30),
  ville     CHAR(20),
  cp        INT
);
CREATE TABLE Buveurs (
  nom       CHAR(30),
  adr      TypeAdr,
  préférée TypeBière
);
```

Accès aux champs avec Oracle

- $A.F$ permet d'accéder au champ F de l'objet qui est la valeur de l'attribut A
- Néanmoins, il faut utiliser un alias r pour la relation R qui contient l'attribut A : $r.A.F$

Exemple : accès aux champs avec Oracle

- Syntaxe incorrecte :
  `SELECT préférée.nom FROM Buveurs`
- Syntaxe incorrecte :
  `SELECT Buveurs.préférée.nom FROM Buveurs`
- Syntaxe correcte :

```
SELECT bv.préférée.nom
FROM Buveurs bv;
```

Suivre des références en SQL3

- $A \longrightarrow F$ est correct si
  1. A est de type REF T
  2. F est un champ d'objets de type T
- $A \longrightarrow F$ désigne la valeur du champ F de l'objet pointé par A

Exemple : suivi de REF

- Rappel : Ventes est une relation *rowtype* de type TypeMenu(bar,bière,prix)
- bar et bière sont des REF d'objets de type TypeBar et TypeBière
- Requête : quelles sont les bières servies au Bar'oc ?

```
SELECT vt.bière()->nom
FROM Ventes vt
WHERE vt.bar()->nom = 'Bar"oc';
```

- Générateurs pour l'accès aux « attributs » puis notation fléchée pour l'accès aux champs

Suivi de REF avec Oracle

- Mécanisme implicite de la notation pointée
- Ingrédients : une REF, un point et un champ de l'objet référencé
- Exemple :

```
SELECT vt.bière.nom
FROM Ventes vt
WHERE vt.bar.nom = 'Bar"oc';
```

Opérateur DEREF de Oracle

Motivation

- Lister les bières (objets) vendues au Bar'oc :

```
SELECT vt.bière
FROM Ventes vt
WHERE vt.bar.nom = 'Bar"oc';
```

- Instruction correcte, mais vt.bière est une REF, donc illisible

Usage de DEREF

- Pour le rendu des objets de type TypeBière, on utilise :

```
SELECT DEREF(vt.bière)
FROM Ventes vt
WHERE vt.bar.nom = 'Bar"oc';
```

- Extrait du résultat :

```
TypeBière('Jupiler', 'Piedboeuf')
```

Les méthodes avec Oracle

- Au-delà de la structure des classes (types), on peut définir des opérations appelées méthodes
- Étude de la syntaxe Oracle

Définition de méthodes avec Oracle

- Déclaration dans CREATE TYPE
- Définition dans la construction CREATE TYPE BODY
  - Usage de la syntaxe PL/SQL
  - La variable SELF désigne l'objet courant

Exemple : déclaration de méthode

- Ajout de la méthode prixEnDollar au TypeMenu

```
CREATE TYPE TypeMenu AS OBJECT (
  bar      REF TypeBar,
  bière    REF TypeBière,
  prix     FLOAT,
  MEMBER FUNCTION prixEnDollar( taux IN FLOAT )
    RETURN FLOAT,
  PRAGMA RESTRICT_REFERENCES( prixEnDollar, WNDS )
);
/
```

- MEMBER FUNCTION est une méthode au sens de Oracle
- *Write No Database State* : lecture seule (pour les transactions)

Définition de méthode à la mode Oracle

- Construction dédiée
  CREATE TYPE BODY <nom du type> AS
  <définition de méthodes = définition de procédures PL/SQL
  avec MEMBER FUNCTION à la place de PROCEDURE>
  END ;
  /

Exemple : définition de méthode

```
CREATE TYPE BODY TypeMenu AS
  MEMBER FUNCTION
    prixEnDollar( taux FLOAT ) RETURN FLOAT IS
      BEGIN
        RETURN taux * SELF.prix;
      END;
END;
/
```

- Inutile de préciser le mode (IN)
- Parenthèses à partir d'un paramètre au moins

De l'usage des méthodes

- De façon traditionnelle, à l'aide d'une notation pointée et d'un objet
- Exemple

```
SELECT vt.bière.nom, vt.prixEnDollar( 1.44 )
FROM Ventes vt
WHERE vt.bar.nom = 'Bar"oc';
```

Méthodes de comparaison en SQL3

- Chaque UDT T peut définir 2 méthodes singulières :
  - x.EQUAL(y) : renvoie VRAI si $x = y$
  - x.LESSTHAN(y) : renvoie vrai si $x < y$
- Un seul paramètre de type T
- Support de la comparaison $=$, $<$, $\geq$, etc. des objets de type T dans le clause WHERE et pour le tri (ORDER BY)

Méthodes de comparaison avec Oracle

- Désignation explicite d'une méthode quelconque d'un UDT comme méthode de comparaison
- La méthode de comparaison renvoie une valeur négative, nulle ou positive selon que l'objet SELF est inférieur, égal ou supérieur à l'objet passé en paramètre

Exemple : déclaration de méthode de comparaison

- Ordre lexicographique sur le nom

```
CREATE TYPE TypeBar AS OBJECT (
  nom      CHAR(20),
  adr      CHAR(20),
  ORDER MEMBER FUNCTION avant( bar2 IN BarType )
    RETURN INT,
  PRAGMA RESTRICT_REFERENCES( avant, WNDS, RNDS,
    WNPS, RNPS )
);
/
```

- *Read/Write No Database State/Package State*
- Un *package* est une collection de procédures et variables qui peuvent partager des valeurs

Exemple : définition de méthode de comparaison

```
CREATE TYPE BODY TypeBar AS
  ORDER MEMBER FUNCTION
    avant( bar2 TypeBar ) RETURN INT IS
  BEGIN
    IF SELF.nom < bar2.nom THEN RETURN -1;
    ELSIF SELF.nom = bar2.nom THEN RETURN 0;
    ELSE RETURN 1;
    END IF;
  END;
END;
/
```

Tables imbriquées avec Oracle

- Une valeur d'attribut peut être une relation
- Création d'un type S à partir d'un UDT T
  CREATE TYPE S AS TABLE OF T ;
- Une valeur de S est une relation *rowtype* (type T)

Exemple : Type de table imbriquée

```
CREATE TYPE TypeBière AS OBJECT (
  nom      CHAR(20),
  sorte    CHAR(10),
  couleur CHAR(10)
);
/
CREATE TYPE TypeTableDeBières AS
  TABLE OF TypeBière;
/
```

Suite de l'exemple

- Utilisation du type TypeTableDeBières dans la relation Brasseries qui regroupe l'ensemble des bières pour une brasserie donnée dans un seul n-uplet

```
CREATE TABLE Brasseries (
  nom     CHAR(30),
  adr    CHAR(50),
  bières TypeTableDeBières
);
```

Stockage des relations imbriquées

- Oracle ne stocke pas chaque relation interne de façon séparée
- Il existe une relation R dans laquelle tous les n-uplets de toutes les relations internes d'un attribut A donné sont enregistrés
- Déclaration : NESTED TABLE A STORE AS R

Exemple : stockage des relations imbriquées

```
CREATE TABLE Brasseries (
  nom     CHAR(30),
  adr    CHAR(50),
  bières TypeTableDeBières
)
NESTED TABLE bières STORE AS TableDeBières;
```

- L'instruction SQL se termine après la construction NESTED TABLE

Interroger une table imbriquée

- Les valeurs des attributs de type table imbriquée peuvent être affichées (clause SELECT)
- Elles sont néanmoins enveloppées dans 2 constructeurs de type :
  1. pour la relation
  2. pour le type des n-uplets dans la relation

Exemple : interroger une table imbriquée

- Trouver les bières brassées par Anheuser-Busch :

```
SELECT bières FROM Brasseries
WHERE nom = 'Anheuser-Busch';
```

- Extrait du résultat :

```
TypeTableDeBières(
  TypeBière( 'Bud', 'lager', 'blonde' ),
  TypeBière( 'Lite', 'malt', 'pale' ),...
)
```

Requête sur une table imbriquée

- Conversion d'une table imbriquée en une relation ordinaire grâce à l'opérateur THE(.)
- La relation induite est invocable dans la clause FROM

Exemple : usage de THE

- Trouver les *Ale* brassées par Anheuser-Busch :

```
SELECT bs.nom
FROM THE(
  SELECT bières
  FROM Brasseries
  WHERE nom = 'Anheuser-Busch'
) bs
WHERE bs.sorte = 'ale';
```

Relation vers table imbriquée

- Toute relation ayant le nombre et le type requis d'attributs peut être promue valeur de table imbriquée
- Utilisation de CAST(MULTISET(...) AS <type>)

Exemple : CAST

- On considère la relation Bières(bière, brasserie), où bière est de type TypeBière et brasserie une chaîne de caractère (*string*)
- On souhaite insérer dans la relation Brasseries un nouvel enregistrement avec la brasserie Pete's Brewing Co

Suite de l'exemple

```
INSERT INTO Brasseries VALUES (
  'Pete"s', 'Palo Alto',
  CAST(
    MULTISET(
      SELECT bs.bière
      FROM Bières bs
      WHERE bs.brasserie = 'Pete"s'
    ) AS TypeTableDeBières
  )
);
```