What is an object-relational DBMS?
- Keeps relation as its fundamental abstraction, but throws in some object-oriented ideas
- Compare with an object-oriented DBMS, which uses class as the fundamental abstraction and tacks on relation as one of many types

Motivations for object-relational DBMS:
- Support structures more complex than just “flat tables”
- Allow DBMS to deal with specialized types—URL’s, images, videos, etc.—with their own specialized methods
- Support specialized methods even on conventional relational data

Current state of the standard:
- Most major relational DBMS vendors now call their products object-relational
- There is a great deal of variation in object-relational functionalities among current products and the SQL3 standard
- We will cover basic ideas from SQL3, but use the syntax of Oracle 8

SQL3 Object Support
- **Row types**: for tuples in relations
  - Can have references to objects of row types
- **Column types (ADT’s)**: for values of attributes
  - Can have methods

Oracle 8 Object Support

Object Types

While SQL3 has row and column types, Oracle 8 uses *object types* for both

Example: StudentType, CourseType, and TakeType

```sql
CREATE TYPE StudentType AS OBJECT (
    SID INTEGER, name CHAR(30), age INTEGER, GPA FLOAT
); /
```
CREATE TYPE CourseType AS OBJECT (CID CHAR(10), TITLE VARCHAR(100));
/
CREATE TYPE TakeType AS OBJECT (studentRef REF StudentType, courseRef REF CourseType);
/
→ In Oracle, type definitions must be followed by / in order to get them to compile

Object Types As Row Types

Example: Student, Take, and Course tables

CREATE TABLE Student OF StudentType;
CREATE TABLE Course OF CourseType;
CREATE TABLE Take OF TakeType;

Values of Object Types

Each object type has a type constructor of the same name

Example: insert Bart into Student

INSERT INTO Student VALUES(123, 'Bart', 10, 3.5);

→ It works, but it is not very “object-oriented”

→ Instead, use the type constructor:

INSERT INTO Student
VALUES(StudentType(123, 'Bart', 10, 3.5));

Example: insert CS145 into Course

Example: insert the fact that Bart takes CS145

INSERT INTO Take VALUES(123, 'CS145'); /* WRONG! */
INSERT INTO Take VALUES (/* WRONG! */
(TakeType(REF(StudentType(123, 'Bart', 10, 3.5)),
REF(CourseType('CS145', 'Intro to DB'))));

→ The referenced object must “live” in a table!

→ In Oracle, whenever object types are involved, it is a good practice to assign a tuple variable to every table in FROM—things might not always work without tuple variables

Jun Yang 2 CS145 Spring 1999
Dereferencing

Use “.”
Example: names of students taking CS145

What if we want the entire object being referenced?
  • Okay to SELECT a reference, but it is just some gibberish value
    Example: all information about CS145 students (not quite)

  → Use DEREF operator
    Example: all information about CS145 students

Object Types As Column Types

Example: NameType for student names

CREATE TYPE NameType AS OBJECT (    firstName CHAR(20), lastName CHAR(20)    );
/
CREATE TYPE StudentType AS OBJECT (    SID INTEGER, name NameType, age INTEGER, GPA FLOAT    );
/

Example: again, insert Bart into Student

Example: find Simpsons’ average GPA

Methods

  • Methods are the real reason why object-relational is more than just nested structures in relations
  • Declare in CREATE TYPE statement
  • Define in CREATE TYPE BODY statement
  • Methods in Oracle are written in PL/SQL
Example: a method to compute initials for names

    CREATE TYPE NameType AS OBJECT (
        firstName CHAR(20), lastName CHAR(20),
        MEMBER FUNCTION initials RETURN VARCHAR,
        PRAGMA RESTRICT_REFERENCES(initials, WNDS, WNPS)
    );

    CREATE TYPE BODY NameType AS
        MEMBER FUNCTION initials RETURN VARCHAR IS
            BEGIN
                RETURN SUBSTR(SELF.firstName, 1, 1) ||
                    SUBSTR(SELF.lastName, 1, 1);
            END;
        END;

    PRAGMA declares initials to be WNDS, “write no database state”,
    and WNPS, “write no package state”
    – Necessary if we want to use initials in queries
    • A method can access a special tuple variable SELF, which refers to
      the object in which the method is applied
    • A method may take arguments
      – Follow the method name by a list of argument declarations en-
        closed in parentheses, like in a PL/SQL procedure

Example: initials of students taking CS145

    • Again, use “.” to invoke methods
    • Parentheses are required even if the method takes no arguments

Order Methods

One method can be declared as the ORDER method for a type

    • This method must return less than 0, 0, or greater than 0, if SELF is
      less than, equal to, or greater than the argument object
    • This method would allow the type to participate in WHERE clauses
      involving =, <=, etc., and in ORDER BY sorting

Example: order NameType objects

    CREATE TYPE NameType AS OBJECT (...
        ORDER MEMBER FUNCTION compare
            (other IN NameType) RETURN INTEGER,
            PRAGMA RESTRICT_REFERENCES
                (compare, WNDS, WNPS, RNPS, RNDS)
    );

CREATE TYPE BODY NameType AS
...
ORDER MEMBER FUNCTION compare
  (other IN NameType) RETURN INTEGER IS
BEGIN
  IF (SELF.lastName < other.lastName) THEN
    RETURN -1;
  ELSIF (SELF.lastName > other.lastName) THEN
    RETURN 1;
  ELSIF (SELF.firstName < other.firstName) THEN
    RETURN -1;
  ELSIF (SELF.firstName > other.firstName) THEN
    RETURN 1;
  ELSE RETURN 0;
  END IF;
END;
END;
/

Example: all CS145 students, sorted by name