

CS3743 Introduction to Database Systems Object-Oriented and Object-Relational Databases

Dr. Weining Zhang

Department of Computer Science
University of Texas at San Antonio

October 28, 2009

Outline

- 1 Object and Object-Relational Databases
 - Motivation
 - OODB Concepts
 - ORDB Standard
 - ORDB Features in Oracle

Motivation

- Relational databases lack support for non-business database applications such as CAD/CAM, GIS, Telecom, multimedia, etc.
- New features needed by these applications include
 - Complex structure of data
 - New types of data, such as image, audio, text, video
 - Application-specific operations
 - High performance for computation-intensive applications
 - Semantic concepts such as generalization.
- These features are part of object-oriented programming

OODB vs ORDB

There are two approaches to extend DBMSs to support object-oriented concepts

- Object-Oriented Database (OODB) extends OOPL with database functionality, such as, persistent objects, indexing, concurrency control, recovery, etc.
 - OODB is based on ODMG 3.0 standard
- Object-Relational Database (ORDB) extends relational DBMS with features of object-oriented systems, such as, complex objects, abstract types, new operations
 - ORDB follows the SQL1999 standard
- OODB and ORDB share a set of core object concepts

Persistence

- Some objects can be made persistent by naming or reachability.
- **Naming** Mechanism.
 - Assign a unique persistent name to an object by a statement or a program operation
 - Retrieve the persistent object by name
- **Reachability** Mechanism.
 - Making an object reachable from some named persistent object
 - Ex. adding an object to a named persistent collection makes it persistent as well
 - Extent of a class is the set of all persistent objects of the class

Inheritance

- A class can be specified in two ways.
 - **Base class**. Specifies a type name, a set of attributes, and a set of operations
 - **Subclass**. Specifies a superclass and additional attributes and operations local to the subclass.
 - A subclass **inherits** attributes and operations of superclass
- Superclasses and subclasses form a type (class) hierarchy
- **Extent** of a type is a collection of persistent objects of that type
 - Constraint: each object in an extent corresponding to a subtype must also be in the extent corresponding to the supertype.
- **Overriding**. Subclasses may inherit method interface and provide different implementations
- Some systems allow multiple inheritance

Polymorphism

- **Polymorphism** is to allow the same operator name to be bound to two or more implementations depending on the type of objects to which the operator is applied.
 - **Early binding**. The appropriate implementation is selected at the compilation time
 - **Late (dynamic) binding**. An appropriate implementation is selected at run time (when parameters and the object are known)

OODB

- OODB systems include O_2 , Ontos, ObjectStore, Versant, and Objectivity.
- OMG (Object Management Group), established in 1989, is a software industry consortium with over 300 members
- OODB standard is the ODMG (Object Database Management Group) standard URL: www.odmg.org
 - Defines an object data model
 - ODL. A language to define OODB schema (classes, relationships, inheritance)
 - OQL. An OODB query language

ORDB

- Pioneering work started with Postgres (now Illustra) in late 80's.
- Major RDBMSs are moving into ORDBMS.
 - Products include Oracle, DB2, and Informix.
 - Many variations in functionality.
- ORDB standard defined in SQL1999 include new features.
 - User-defined Types: Row type & ADT
 - User-defined Method
 - Inheritance
 - Nested Relation

Row Type

Example

Creating row types

```
create row type StudentType( <list of attributes>)
```

Creating tables

```
create table Students of type StudentType
```

Queries Similar to querying conventional relations

- Features beyond relational model include
 - nesting, referencing, dereferencing, oid

Type Nesting

Example

Create nested row type

```
create row type AddressType(
  street char(30),
  city char(20)
)
create row type StudentType(
  sid ref(StudentType),
  name varchar(30),
  address AddressType,
  courses set(ref(CourseType))
)
```

OID and Scope

Example

Let system generate oids

```
create table Students of type StudentType
  values for sid are system generated;
```

Specify scope of a reference

```
create row type EnrollmentType (
  student ref(StudentType),
  course ref(CourseType)
  grade char(2))
create table Enrollment of type EnrollmentType
  scope for student is Students,
  scope for course is Courses;
```

Dereferencing

- Using “.” to access components of a row type
- Using “->” on a reference of a row type to access a component of the row type

Example

```
select student->name
from Enrollment
where student->address..city = 'Huston'
```

- student is a reference of StudentType
- address is of AddressType

Value Type (ADT)

Example

```
create type AddressADT (
  street char(50),
  city char(20),
  equals addrEq,
  less addrLt,
  function AddressADT(:s char(50), :c char(20))
    return AddressADT;
:a AddressADT;
begin
:a := AddressADT();
:a.street := :s;
:a.city := :c;
return :a;
end;
...)
```

Inheritance

- Inheritance is specified in terms of subtable and supertable

Example

```
create table G_Students under Students (
  advisor ref(Professors),
  thesis varchar(50),
  thesis_committee set(ref(Professors)))
```

- subtable G_Students inherits all attributes from supertable Students

Nesting and Unnesting

Example

Unnesting

```
select sid, name, c as course,
       address.street, address.city
from Students as s, s.course as c
```

Nesting

```
select sid, name, set(course) as course,
       (month, day, year) as birthdate
from flat_Students
group by sid, name, birthdate
```

Define a Row Type in Oracle

Example

Create a row type

```
create type StudentType as object (
  name char(40),
  address varchar(100),
  GPA float)
```

Create a table

```
create table Students of type StudentType
```

Query: Find John's GPA

```
select GPA from Students where name = 'John'
```

Define an ADT in Oracle

Example

Create a value type

```
create type NameType as object (
  firstname char(20),
  lastname char(20) )
```

Use a value type in a row type

```
create type StudentType as object (
  name NameType,
  address varchar(100),
  gpa float)
```

Define an ADT in Oracle (cont.)

Example

Create a table

```
create table Students of type StudentType
```

Insert an object

```
insert into Students
values ( StudentType(NameType('John', 'Simpson'),
  'San Antonio', 3.5))
```

Query: find Simpson's GPA

```
select s.gpa
from Students s
where s.name.lastname = 'Simpson'
```