CS422 Principles of Database Systems
Databases and Object-Oriented Paradigm

Chengyu Sun
California State University, Los Angeles

Book Example

- ISBN, Title, AuID, AuName, AuAddr, AuPhone
  - A book may have multiple authors
  - An author may have multiple phone numbers

OO Design

class Book {
    String ISBN;
    String title;
    Author authors[];
}
class Author {
    int id;
    String name;
    String address;
    String phones[];
}

Relational Design

<table>
<thead>
<tr>
<th>Books</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISBN</td>
<td>Title</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ISBN</td>
<td>AuID</td>
</tr>
</tbody>
</table>

Relational vs. OO

- Relational
  - Atomic types: numbers, string ...
  - Efficient implementation
- OO
  - Atomic types, composite types, collection types, reference types, and subtypes
  - Model real world well

Object-Relational Model

- Brings OO features into the relational model
  - Structured types for attributes
  - Methods
  - References
  - Identifiers for tuples (OIDs)
**OO in PostgreSQL – Array Types**

create table authors (
   id int primary key,
   name varchar(64),
   phones char(14)[4]
);

- Arrays in PostgreSQL are *dynamic*

**Access Array Types**

-- access array elements by index
select * from authors where phones[1] like '(323)%';

-- access array elements by range
select phones[1:3] from authors;

-- access any/all elements
select * from authors where '(323) 343-1111' = any(phones);

**OO in PostgreSQL – Composite Types**

CREATE TYPE beer_price AS (
   beer varchar(50),
   price decimal(10,2)
);

- Composite types can be used in functions
- Composite types cannot be used as attribute type – no *nested relations*
  - feature to be supported in 8.0

**OO in PostgreSQL – Inheritance**

CREATE TABLE product (
   pid int primary key,
   brand varchar(32),
   desc text,
   quantity int,
   price decimal(10,2)
);

CREATE TABLE cpu (
   model varchar(16),
   frequency decimal(4,2),
   rpm int,
   ) INHERITS (product);

CREATE TABLE hdd (CREATE TABLE product (pid int primary key,
   capacity int,
   rpm int)
   ) INHERITS (product);

**OO in PostgreSQL – Other**

- Object identifiers (OID)
  - Primary keys for systems tables
  - System column in user created tables
- User-defined types and associated operations
  - in C

**Object-Oriented Model**

- What are we restricted by relations? All we want is *Persistent Objects!*
  - Object Definition Language (ODL)
    - Object Query Language (OQL)
- Unfortunately, OO DBMS have not been very successful
  - Too much was invested in relational DBMS
  - Possible performance issues
Hibernate

 Hibernate is a persistence service that stores Java objects in relational databases - or provides an object oriented view of existing relational data. — www.hibernate.org

Hibernate Architecture

Person Example – Simple JDBC Version

- Duplicated design efforts
- Going back and forth between SQL and Java

Person Example – Hibernate Version

// get the tuple
Person p = (Person) query.iterate().next();

// increment age
p.setAge( p.getAge()+1 );

// update tuple
Transaction tx = session.beginTransaction();
session.save( p );
tx.commit();

Hibernate – The Big Picture

- Classes
- OO ⇒ Relational
- O/R mapping ⇒ Schema
- Connections
- Query language

Persistent Classes

- Bean-style classes
  - Default constructor
  - Persistent fields (properties)
    - getXYZ(), isXYZ()
    - setXYZ()
- Accessors and mutators do not have to be public
### Object-Relational Mapping
- Classname.hbm.xml
- Automatically generate the mapping from a compiled class
  - java
  - net.sf.hibernate.tool.class2hbm.MapGenerator
  - Classname
  - MapGenerator is in hibernate-extension package
- "There is no way to produce a full Hibernate mapping without extra input from the user."

### Schema Generation
- java
  - net.sf.hibernate.tool.hbm2ddl.SchemaExport
  - Classname.hbm.xml
- ... tables with the same names will be dropped

### Connection Configuration
- hibernate.properties and/or hibernate.cfg.xml
  - Database related information
  - Connection pool related information

### Query Language – HQL
- Very SQL-like
- Instances of a class rather than tuples of a relation
  - Fields and sub-fields
  - Path expressions
  - Polymorphic queries

### Additional Readings
- Chap 4 of the Textbook
- JDBC Tutorial and API reference
- Hibernate documentation and API reference