

CS422 Principles of Database Systems Object-Relational Mapping (ORM)

Chengyu Sun
California State University, Los Angeles

The Object-Oriented Paradigm

- ◆ The world consists of objects
- ◆ So we use object-oriented languages to write applications
- ◆ We want to store some of the application objects (a.k.a. persistent objects), e.g. *accounts*, *customers*, *employees*
- ◆ So we use a Object Database?

The Reality of DBMS

- ◆ Relational DBMS are still predominant
 - Best performance
 - Most reliable
 - Widest support
- ◆ Bridge between OO applications and relational databases
 - CLI and embedded SQL
 - Object-Relational Mapping (ORM) tools

Employee – Application Object

```
public class Employee {  
  
    Integer id;  
    String name;  
    Employee supervisor;  
  
}
```

Employee – Database Table

```
create table employees (  
  
    id integer primary key,  
    name varchar(255),  
    supervisor integer references employees(id)  
  
);
```

From Database to Application

- ◆ So how do we construct an Employee object based on the data from the database?

```
public class Employee {  
  
    Integer id;  
    String name;  
    Employee supervisor;  
  
    public Employee( Integer id )  
    {  
        // access database to get name and supervisor  
        ... ..  
    }  
}
```

Problems with CLI and Embedded SQL ...

- ◆ SQL statements are hard-coded in applications

```
public Employee( Integer id ) {
    ...
    PreparedStatement p;
    p = connection.prepareStatement(
        "select * from employees where id = ?"
    );
    ...
}
```

... Problems with CLI and Embedded SQL ...

- ◆ Tedious translation between application objects and database tables

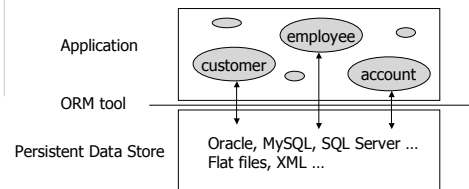
```
public Employee( Integer id ) {
    ...
    ResultSet rs = p.executeQuery();
    if( rs.next() )
    {
        name = rs.getString("name");
    }
    ...
}
```

... Problems with CLI and Embedded SQL

- ◆ Application design has to work around the limitations of relational DBMS

```
public Employee( Integer id ) {
    ...
    ResultSet rs = p.executeQuery();
    if( rs.next() )
    {
        ...
        supervisor = ??
    }
}
```

The ORM Approach



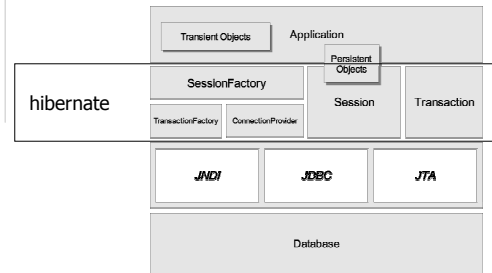
Advantages of ORM

- ◆ Make RDBMS look like ODBMS
- ◆ Data are accessed as objects, not rows and columns
- ◆ Simplify many common operations. E.g. `System.out.println(e.supervisor.name)`
- ◆ Improve portability
 - Use an object-oriented query language (OQL)
 - Separate DB specific SQL statements from application code
- ◆ Caching

Common ORM Tools

- ◆ Java Data Object (JDO)
 - One of the Java specifications
 - Flexible persistence options: RDBMS, OODBMS, files etc.
- ◆ Hibernate
 - Most popular Java ORM tool right now
 - Persistence by RDBMS only
- ◆ Others
 - http://en.wikipedia.org/wiki/Object-relational_mapping
 - http://www.theserverside.net/news/thread.tss?thread_id=29914

Hibernate Application Architecture



Setup Hibernate

- ◆ Download *hibernate core* from <http://www.hibernate.org>
- ◆ Add the following jar files to CLASSPATH
 - *hibernate-3.0\hibernate3.jar*
 - All the jar files under *hibernate-3.0\lib*
 - The JDBC driver of your DBMS

A Sample Hibernate Application

- ◆ Java classes
 - *Employee.java*
- ◆ Code to access the persistent objects
 - *EmployeeTest.java*
- ◆ O/R Mapping files
 - *Employee.hbm.xml*
- ◆ Hibernate configuration file
 - *hibernate.cfg.xml*
- ◆ (Optional) Logging configuration files
 - *Log4j.properties*

Java Classes

- ◆ Plain Java classes (POJOs) except that
 - There must be an identity field
 - Each persistent field must have a pair of getter and setter
- ◆ The *identity field* is used to uniquely identify an object
- ◆ The *getter* and *setter* for a property *xxx* follows the naming convention:
 - *getXxx()*
 - *setXxx()*

O/R Mapping Files

- ◆ Describe how class fields are mapped to table columns
- ◆ Three important types of elements in a mapping file
 - *<id>*
 - *<property>* - when the field is of simple type
 - Association - when the field is of a class type
 - *<one-to-one>*
 - *<many-to-one>*
 - *<many-to-many>*

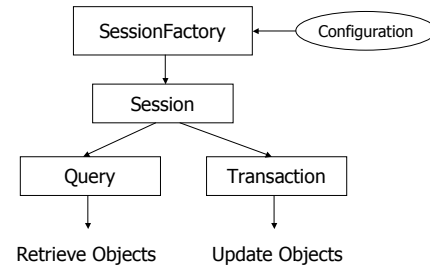
Hibernate Configuration Files

- ◆ Tell hibernate about the DBMS and other configuration parameters
- ◆ Either *hibernate.properties* or *hibernate.cfg.xml* or both
 - Sample files under *hibernate-3.0/etc*

Log4j Configuration File

- ◆ Log4j is a logging tool for Java
- ◆ Log levels
 - DEBUG, INFO, WARN, ERROR, FATAL
- ◆ Log output (appender)
 - File
 - Stdout

Access Persistent Objects



Hibernate Query Language (HQL)

- ◆ A query language that looks like SQL, but for accessing *objects*
- ◆ Automatically translated to DB-specific SQL statements
- ◆ `select e from Employee e where e.id = :id`
 - From all the Employee objects, find the one whose id matches the given value

Hibernate Query Language (HQL)

- ```
select e from Employee e where e.id = :id
```
- ◆ A query language that looks like SQL, but deals with *objects* instead of tables
    - E.g. from all the `Employee` objects, find the one whose `id` matches the given value
    - OO language-like syntax, for example `e.supervisor.name`
  - ◆ Support named query parameters
  - ◆ Automatically translated into DB-specific SQL statement

## hbm2ddl

- ◆ Generate DDL statements from Java classes and mapping files
- ◆ See `hbm2ddl.bat` in the sample code

## More About Mapping

- ◆ Map collections
- ◆ Map subclasses
  - Table per concrete class
  - Table per class hierarchy
  - Table per subclass

## O/R Mapping vs. ER-Relational Conversion

O/R Mapping                      ER-Relational Conversion

Class                      ←                      →                      Entity Set

<property>                      ←                      →                      Attribute

Association                      ←                      →                      Relationship

Subclass                      Subclass

- table per concrete class                      ←                      →                      • OO method
- table per class hierarchy                      ←                      →                      • NULL method
- table per subclass                      ←                      →                      • ER method

## More Hibernate Resource

◆ *Hibernate in Action* by Christian Bauer and Gavin King

◆ Hibernate documentation at <http://www.hibernate.org>