Introduction To ORM Frameworks

Procedural Vs. OO

Procedural Vs. OO

- Procedures are core entities
- Data exist to feed procedures execution
- Workflow relation between procedures

- Objects are core entities
- Objects consist of function and data at the same level
- Rich OO relation:
  - Composition
  - Inheritance
  - Workflow
  - …

Object Oriented Concepts

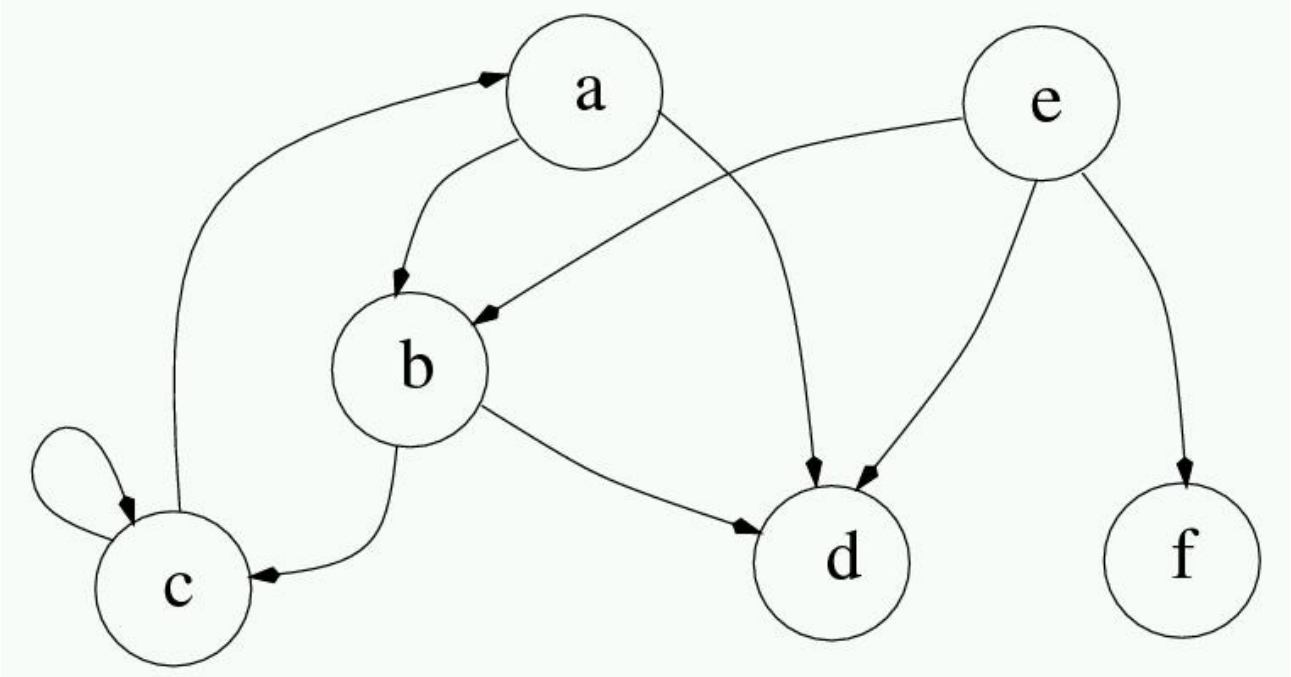
- Class & Object

Object Oriented Concepts, Cont.

- Data (fields, properties, state, attributes)
- Function (methods, procedures)

Object Oriented Concepts, Cont.

- Reference (pointer, link, complex objects graph)



Object Oriented Concepts, Cont.

- Encapsulation
  & Interfaces

Object Oriented Concepts, Cont.

- Inheritance & Specialization

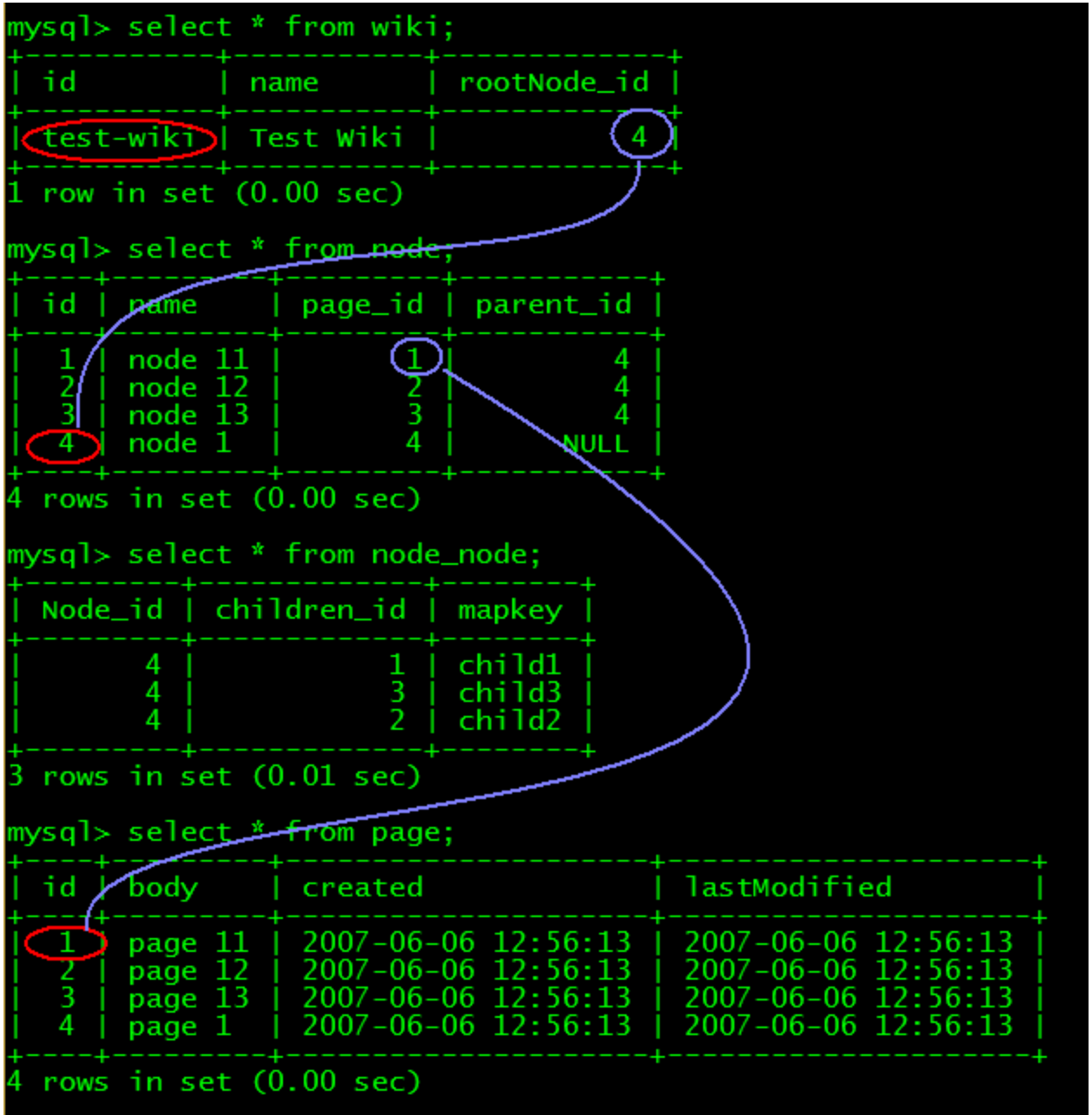
Business Logic & Domain Model

- Business Logic: What should the software do?
- Domain Model: OO implementation of BL.

Relational Databases Concepts

- Table & Tuple
- Primary Key
- Foreign Key
- SQL Query
- Normalization
- Transaction

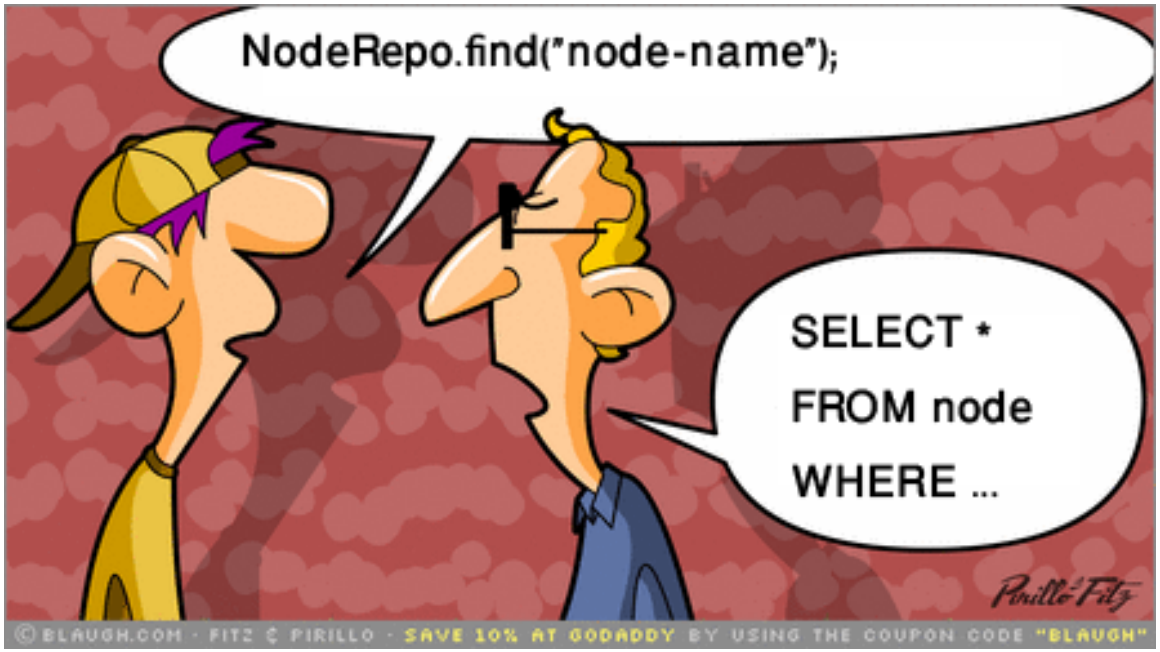
RDB Concepts, Cont.



```
mysql> select * from wiki;
+-----------+-----------+-------------+
| id        | name      | rootNode_id |
+-----------+-----------+-------------+
| test-wiki | Test Wiki |           4 |
+-----------+-----------+-------------+
1 row in set (0.00 sec)

mysql> select * from node;
+----+---------+---------+-----------+
| id | name    | page_id | parent_id |
+----+---------+---------+-----------+
|  1 | node 11 |       1 |         4 |
|  2 | node 12 |       2 |         4 |
|  3 | node 13 |       3 |         4 |
|  4 | node 1  |       4 |      NULL |
+----+---------+---------+-----------+
4 rows in set (0.00 sec)

mysql> select * from node_node;
+---------+-------------+--------+
| Node_id | children_id | mapkey |
+---------+-------------+--------+
|       4 |           1 | child1 |
|       4 |           3 | child3 |
|       4 |           2 | child2 |
+---------+-------------+--------+
3 rows in set (0.01 sec)

mysql> select * from page;
+----+---------+---------------------+---------------------+
| id | body    | created             | lastModified        |
+----+---------+---------------------+---------------------+
|  1 | page 11 | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
|  2 | page 12 | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
|  3 | page 13 | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
|  4 | page 1  | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
+----+---------+---------------------+---------------------+
4 rows in set (0.00 sec)
```

Object-Relational Impedance Mismatch

- OO does not talk SQL
  - Lots of SQL for lots of objects
  - RDBMs' dialect
  - Write/change/maintain nightmare

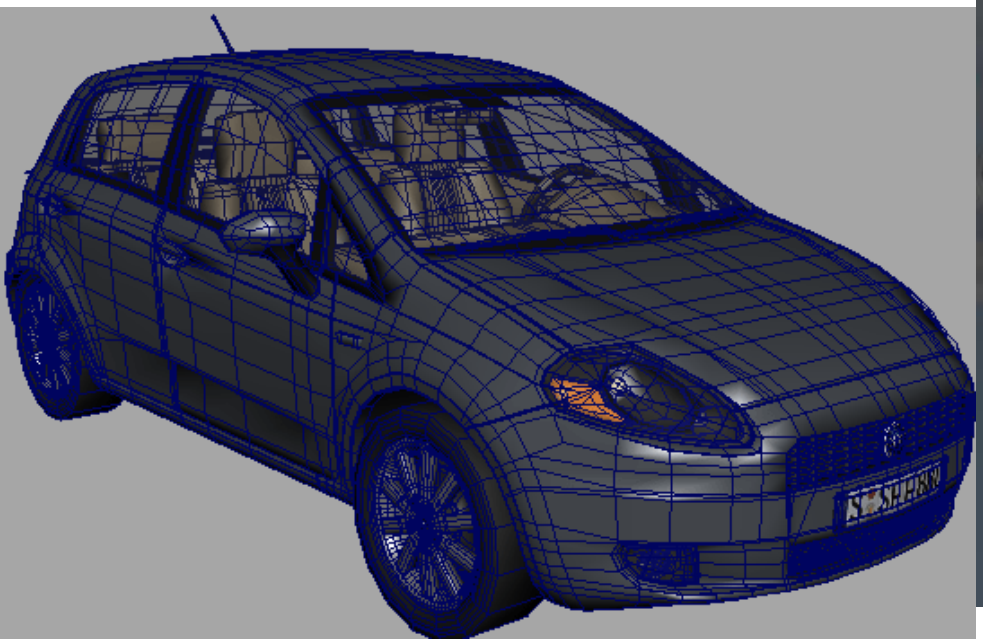


OR Impedance Mismatch, Cont.

- Moving objects between OO and RDB worlds
  - Load on demand
  - Trace and update
  - Identity

OR Impedance Mismatch, Cont.

- RDB does not support OO concepts immediately:
  - Inheritance
  - Complex, non-normalized composition
  - Encapsulation and Interface



OR Impedance Mismatch, Cont.

- Who
  - Outlive the other?
  - Is more expensive?
  - Is master/slave?

So Why Still RDB?

- High performance, Powerful
- Available, Everywhere
- Proven
  - in *theory*
  - and in *practice*
- Known in every language
- Almost no dominate replacement

Solution

- RDB+SQL:
    - JDBC (no solution!)
- RDB+ORM:
    - Hibernate (OSS, JBoss, supports EJB3)
    - JDO
    - EJB3

(JDBC/EJB3/JDO JCP Spec, supported by Sun, IBM, Oracle, JBoss and other Java friends.)

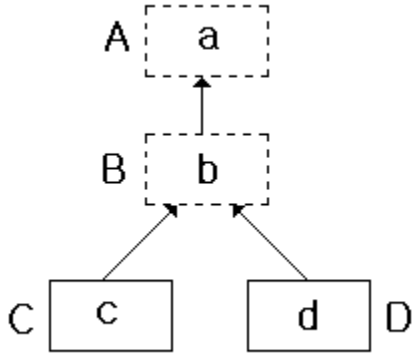
- RDB, SQL mapping:
    - iBATIS (IBM Prod.)
- OODB:
    - Db4o (OSS, Recent years tries hard to prove itself! Not dominate yet!)

ORM Features

- Mapping OO concepts to RDB concepts
- Relationship navigation
  - Eager/Lazy loading
- Trace & update changes
  - Lazy update
- Map identities
- Query language, CRUD API
- **Transparency**
- Transaction

ORM, Mapping OO Concepts to RDB Concepts

- Inheritance
  - Table per Hierarchy
    - Lost Memory
    - No Modularity
  - Table per Class
    - Join, lost time
  - Table per Concrete Class
    - Redundancy



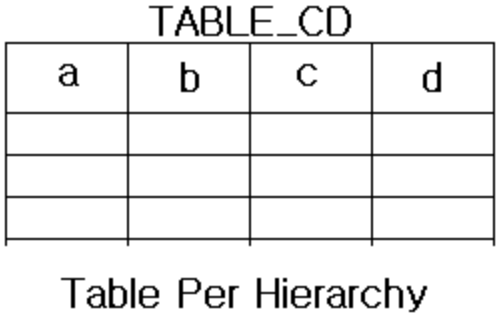


Table Per Hierarchy

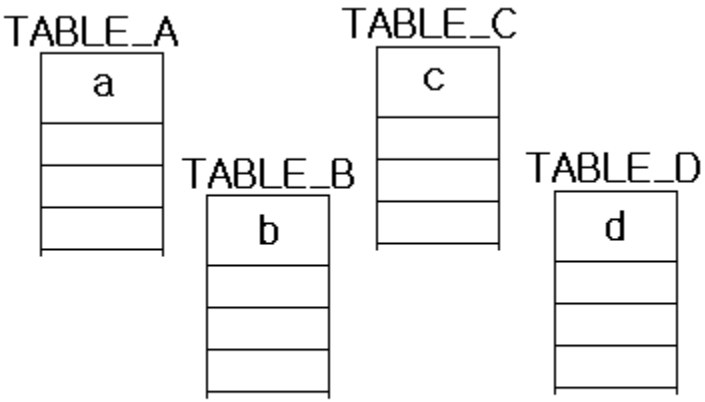


Table Per Class

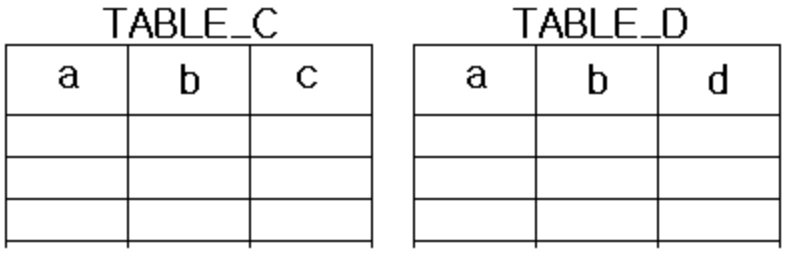


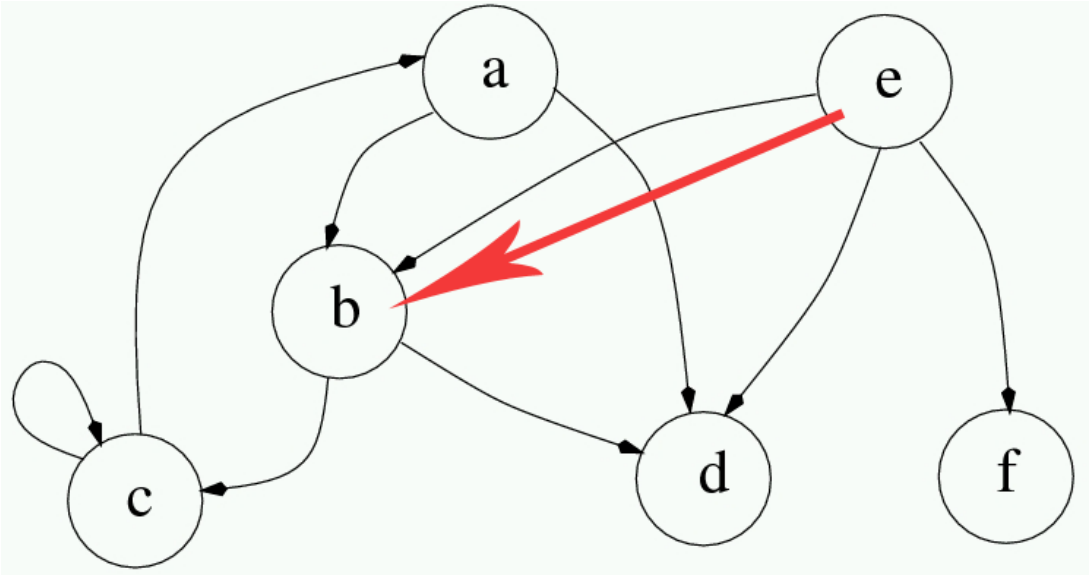
Table Per Concrete

ORM, Mapping OO Concepts to RDB Concepts, Cont.

- Composition
  - Collections (Indexed)
  - Value/Embedded Objects
- Bi-directional reference

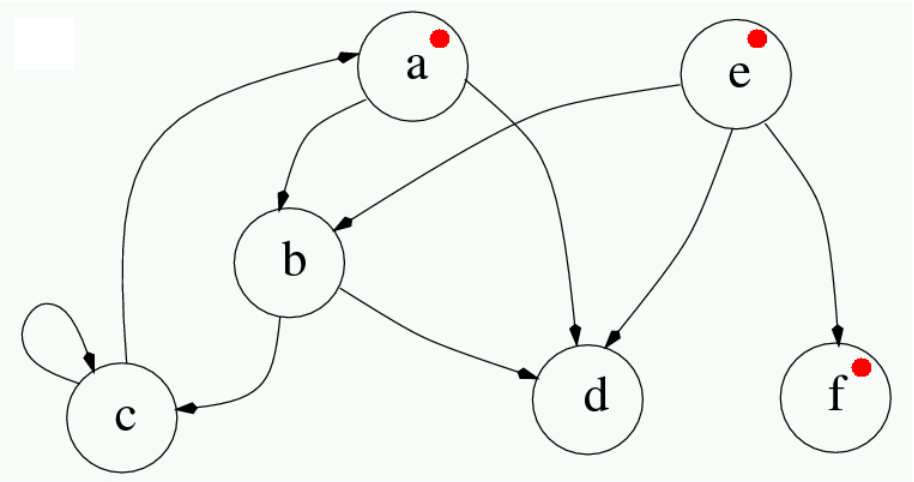
ORM, Relationship Navigation

- Lazy Loading
  - Delay loading ‘e’ if it’s rare to load ‘b’ if ‘e’ is loaded
  - Avoid memory waste for unnecessary objects
- Eager Loading
  - Load ‘b’ with ‘e’ if it’s common to load ‘b’ if ‘e’ is loaded
  - Avoid delays for loading objects



ORM, Trace, Update, Identities

- Trace loaded objects
- Preserve relation between objects and corresponding RDB data
  - Return same object for request for same data
  - Apply changes to object to same data in RDB
- Knows which object should be update



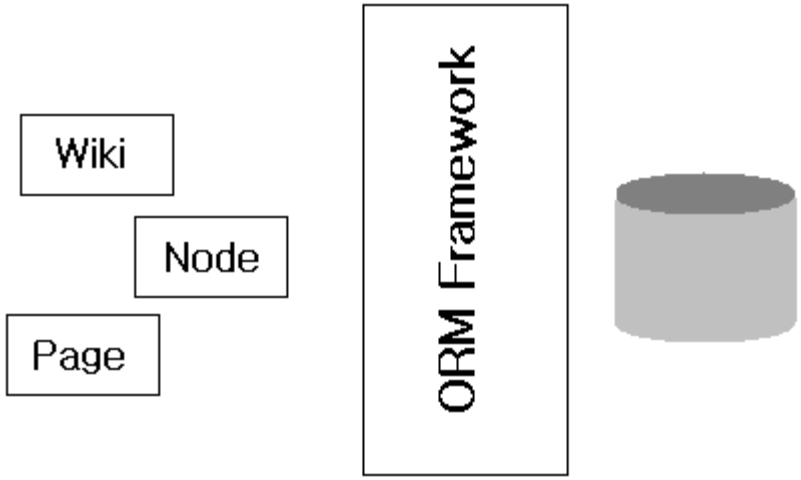
a	
b	
c	
d	
e	
f	

ORM, Query Language, CRUD API

- Create, Read, Update, Delete Objects in/from/to RDB
- Query lang for search for *objects* stored in RDB
- Support SQL, convert SQL result to objects

ORM, Transparency

- Domain model objects should act as if they are not persistence object
- Use Repository interfaces and implementation to talk to ORM framework



ORM, Transaction

- An API to do atomic jobs

```
startTransaction();
…
If( everyThingIsOk){
   commit();
}else{
   rollback();
}
```

ORM, Caching

- Cache loaded objects even if they are not needed any more to benefit from request for them in near future

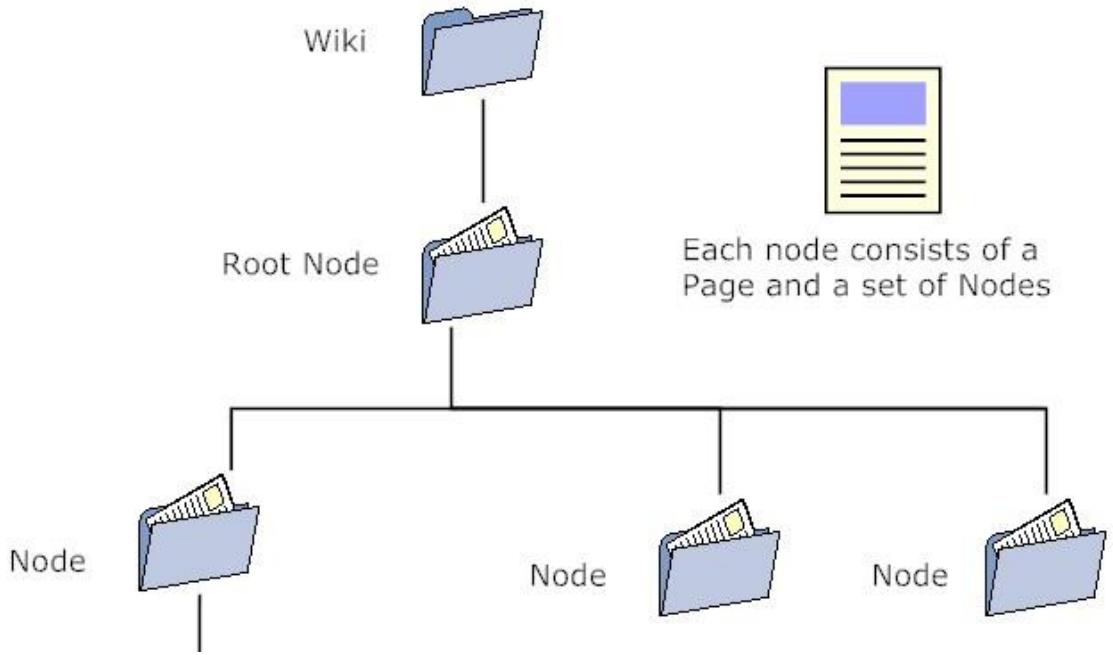
ORM Benefits

- Productivity
  - OOP
  - No SQL
- Performance
  - Cache Objects
  - Lazy/Eager Loading
  - Lazy Updating
- Portability
  - Different SQL dialects

When SQL?

- Performance
  - Handwritten SQLs
  - Special SQL queries
  - Special RDBMS features
- Limited SQL
  - Organization policy or DBA
  - Legacy RDBMSs or RDBs
- No OO, i.e. Procedural style
- Create an ORM framework

Case Study: Simple Wiki



Case Study: Simple Wiki, Cont.

```java
@Entity
public class Wiki {
    @Id
    public String id;

    @Column
    public String name;

    @OneToOne
    public Node rootNode;
}
```

```java
@Entity
public class Page {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    long id;

    @OneToOne(mappedBy = "page")
    public Node node;

    @Column
    public String body;

    @Column
    public Date created;

    @Column
    public Date lastModified;
}
```

```java
@Entity
public class Node {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    long id;

    @Column
    public String name;

    @OneToOne
    public Page page;

    @OneToOne
    public Node parent;

    @org.hibernate.annotations.CollectionOfElements
    public Map<String, Node> children;
}
```

Case Study: Simple Wiki, Cont.

```
Transaction transaction = null;
Session session = MySessionFactory.getInstance().getCurrentSession();
try {
    transaction = session.beginTransaction();

    Page page1 = new Page("page 1", new Date(), new Date());
    Node node1 = new Node("node 1", page1);

    Page page11 = new Page("page 11", new Date(), new Date());
    Node node11 = new Node(node1, "node 11", page11);
    Page page12 = new Page("page 12", new Date(), new Date());
    Node node12 = new Node(node1, "node 12", page12);
    Page page13 = new Page("page 13", new Date(), new Date());
    Node node13 = new Node(node1, "node 13", page13);

    node1.children = new HashMap<String, Node>();
    node1.children.put("child1", node11);
    node1.children.put("child2", node12);
    node1.children.put("child3", node13);

    Wiki wiki = new Wiki(wikiId, "Test Wiki", node1);

    session.save(page11); session.save(node11);
    session.save(page12); session.save(node12);
    session.save(page13); session.save(node13);
    session.save(wiki); session.save(page1);
    session.save(node1);

    transaction.commit();
} catch (HibernateException e) {
    if (transaction != null && transaction.isActive())
        transaction.rollback();
}
```

Case Study: Simple Wiki, Cont.

```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | varchar(255) | NO   | PRI | NULL    |       |
| name        | varchar(255) | YES  |     | NULL    |       |
| rootNode_id | bigint(20)   | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| name      | varchar(255) | YES  |     | NULL    |                |
| parent_id | bigint(20)   | YES  | MUL | NULL    |                |
| page_id   | bigint(20)   | YES  | MUL | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)

+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Node_id     | bigint(20)   | NO   | PRI | NULL    |       |
| children_id | bigint(20)   | NO   | UNI | NULL    |       |
| mapkey      | varchar(255) | NO   | PRI | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| body         | varchar(255) | YES  |     | NULL    |                |
| created      | datetime     | YES  |     | NULL    |                |
| lastModified | datetime     | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Case Study: Simple Wiki, Cont.

```
mysql> select * from wiki;
+-----------+-----------+-------------+
| id        | name      | rootNode_id |
+-----------+-----------+-------------+
| test-wiki | Test Wiki |           4 |
+-----------+-----------+-------------+
1 row in set (0.00 sec)

mysql> select * from node;
+----+---------+---------+-----------+
| id | name    | page_id | parent_id |
+----+---------+---------+-----------+
|  1 | node 11 |       1 |         4 |
|  2 | node 12 |       2 |         4 |
|  3 | node 13 |       3 |         4 |
|  4 | node 1  |       4 |      NULL |
+----+---------+---------+-----------+
4 rows in set (0.00 sec)

mysql> select * from node_node;
+---------+-------------+--------+
| Node_id | children_id | mapkey |
+---------+-------------+--------+
|       4 |           1 | child1 |
|       4 |           3 | child3 |
|       4 |           2 | child2 |
+---------+-------------+--------+
3 rows in set (0.01 sec)

mysql> select * from page;
+----+---------+---------------------+---------------------+
| id | body    | created             | lastModified        |
+----+---------+---------------------+---------------------+
|  1 | page 11 | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
|  2 | page 12 | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
|  3 | page 13 | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
|  4 | page 1  | 2007-06-06 12:56:13 | 2007-06-06 12:56:13 |
+----+---------+---------------------+---------------------+
4 rows in set (0.00 sec)
```