

Apache O/R Mappers

Henning Schmiedehausen

henning@apache.org

ApacheCon 2006 EU, Dublin, Ireland

The logo for ApacheCon Europe 06 is located in the bottom right corner. It features the text "ApacheCon" in a white serif font, with "Europe 06" in a white sans-serif font below it. A small, colorful feather icon is positioned to the left of the word "Europe". The entire logo is set against a teal background that has a yellow and red border at the top.

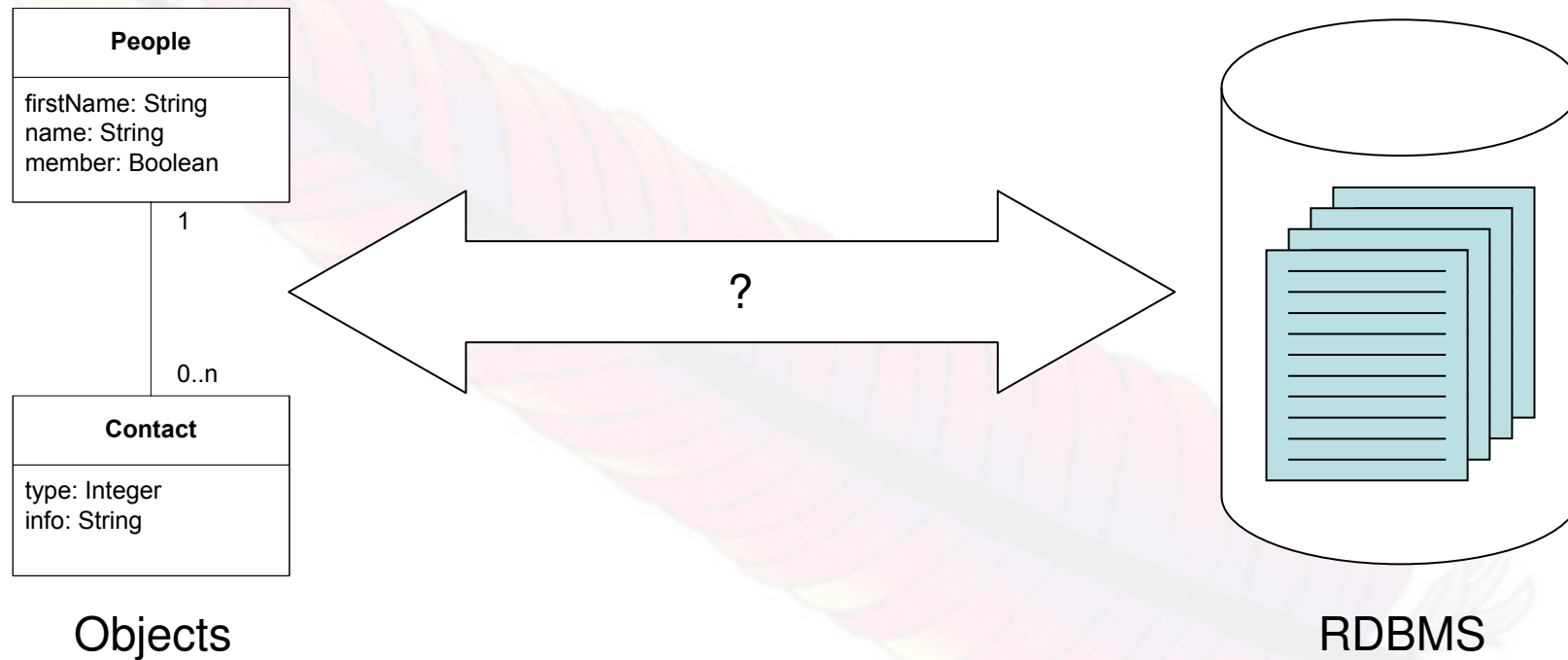
ApacheCon
Europe 06

This is a Java talk!

- Apache O/R mappers in other languages:
 - iBATIS.NET - Microsoft .NET framework
- That's all folks! The rest is Java 😊

What is O/R mapping?

The problem:



O/R mapping in a nutshell

“Object-Relational mapping (or O/RM), is a programming technique that links databases to object-oriented language concepts, creating (in effect) a ‘virtual object database’.” - Wikipedia

Do we need to talk about O/R?

- O/R mapping is „done“, isn't it?
- Hibernate is the 800lb gorilla.
- EJB3 and JDO2 will rule the market, right?
- JTA is on the horizon
- Oracle TopLink and Apple Enterprise Objects
 - (TopLink Essentials is Open Source RI for JSR-220)
- Do we need anything else?

Yes, we do!

- EJB 3 and JDO 2 are a bit unwieldy
 - EJB3 spec: 818 pages! (simplified: 59)
 - JDO2 spec: 332 pages
- Hibernate is LGPL
 - legal implications for your code
- JDO is loaded with politics
- EJB3 was just finalized



I just want to query some data for
my web application, dammit!

O/R projects @ Apache

- JDO 1.0 and 2.0 API (and TCK)
- ‘real’ O/R mappers
 - DB Torque
 - DB ObjectRelationalBridge - OJB
 - Cayenne - In incubation
- SQL Maps/Data mapper
 - iBATIS

A few words about JDO

- what Sun intended to be the “Java object persistence standard”
- Transparency of persistence
- Steamrolled by the success of Hibernate
- EJB 3 broke out the persistence layer (into Java Persistence API, JPA)
- JDO works with J2SE (also JPA)

JDO @ Apache

- Code donated by Sun to the ASF
- JDO 1.0 API (JSR-12)
- JDO 1.1 Reference Implementation (RI)
- JDO 1.1 TCK
- JDO 2.0 API (JSR-243)
- JDO 2.0 TCK
- no JDO 2.0 RI, it is at jpo.x.org

Common to all projects

- DB Support: MySQL, PostgreSQL, Oracle, HSQLDB, Apache Derby
- O/R mapping definition in XML
- 1:1, 1:n, m:n mappings
- native and generated primary keys
- “Half day rule”

DB Torque

- current release is 3.2 (Dec. 2005)
- forked off the Jakarta Turbine project
- generates Java and SQL code from model (data objects and peers (DAOs))
- ant, maven-1 support

DB Torque

- Pros:
 - DB Management (creation, dropping, loading)
 - XML schema support for SQL data
 - thin runtime layer
- Cons:
 - Lots of static classes, hard to extend
 - ‘heavy’ data objects (POJO support in 3.2)
 - no standards compliant API

DB ObjectRelationalBridge

- current release is 1.0.4 (Dec 2005)
- Multiple APIs: PersistenceBroker, ODMG 3.0, JDO 1.0.1, OTM
- J2EE integration: JNDI, JTA, JCA coming
- Enterprise features: locking, distributed caching, clustering
- Polymorphism, Extents, Collections
- Object Query Language - OQL

DB ObjectRelationalBridge

- Pros:
 - Standards compliance (ODMG, JDO)
 - Tool + IDE support
 - Enterprise features, scales well
 - Uses POJOs
- Cons:
 - Heavy
 - Uses private tables in the database

Cayenne

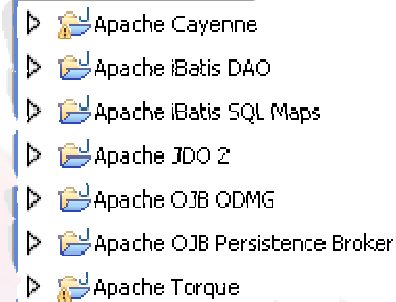
- current release is 1.1.4 (off-Apache)
- comes with a Swing GUI tool for modelling
- ant, maven support
- concepts related to WebObjects EOF
- J2EE: JNDI, Managed Transactions

Cayenne

- Pros:
 - GUI modeller included
 - Good documentation
- Cons:
 - Heavy data objects (inherit `DataObject`)
 - JTA support missing (though not needed)
 - no standards compliant API

Example code

- This talk can only scratch the concepts
- Example code is available from my homepage:



<http://henning.schmiedehausen.org/or-mappers>

Retrieving an Object by PK

- Torque:

```
public People retrieve() {  
    People user = PeoplePeer.retrieveByPK(2L);  
    return user;  
}
```

- Cayenne:

```
People user = (People) DataObjectUtils  
    .objectForPK(dataContext, People.class, 2);
```

Retrieving an Object by PK

- OJB (Persistence Broker):

```
Identity sel = broker.serviceIdentity()  
                .buildIdentity(People.class, 2);  
  
People user = (People) broker.getObjectByIdentity(sel);
```

- OJB (ODMG):

```
OQLQuery query = odmng.newOQLQuery();  
query.create("select people from " +  
            People.class.getName() + " where id=$1");  
query.bind(2L);  
People user = ((List<People>) query.execute()).get(0);
```

Caveat

- Calling `retrieve()` twice returns:
 - Torque and OJB: Two different objects
 - Cayenne: The same object

Changing an Object

- Torque:

```
People people = retrieve();  
people.setMember(true);  
people.save();
```

- Cayenne:

```
People people = retrieve();  
people.setMember(true);  
dataContext.commitChanges();
```

Changing an Object

- OJB (Persistence Broker):

```
broker.beginTransaction();  
People people = retrieve();  
people.setMember(true);  
broker.store(people);  
Broker.commitTransaction();
```

- OJB (ODMG):

```
Transaction tx = odmng.newTransaction();  
tx.begin();  
People people = retrieve();  
people.setMember(true);  
tx.commit();
```



And Now...
...for Something Completely
Different

iBatis

- current release is 2.1.7 (Jan. 2006)
- Data mapper framework
- maps SQL queries to Java objects
- supports JNDI and JTA

- optional DAO (data access objects) layer (supports SQLMAP, Hibernate, JDBC)

A mapped query

```
<select id="getPeople" parameterClass="people"
        resultMap="peopleResult">
  select * from people
  <dynamic prepend="where">
    <isNotNull prepend="and" property="id">
      people_id = #id#
    </isNotNull>
  </dynamic>
</select>
```

Query examples

- Querying a single object

```
People select = new People();  
select.setId(2L);  
People user = (People)  
    sqlMap.queryForObject("getPeople", select);
```

- Querying a list

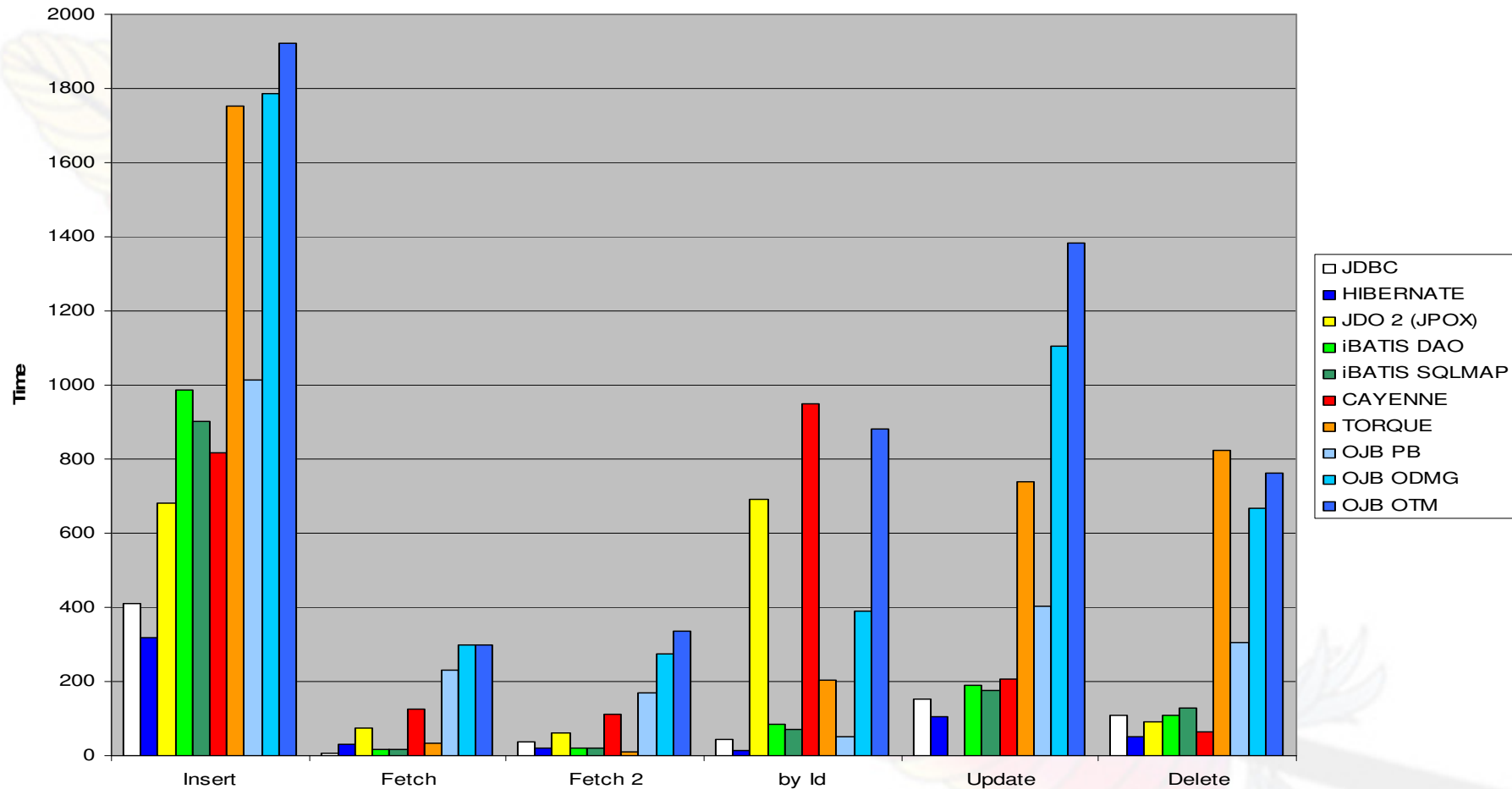
```
List select = sqlMap.queryForList("getPeople", null);
```

Some numbers...

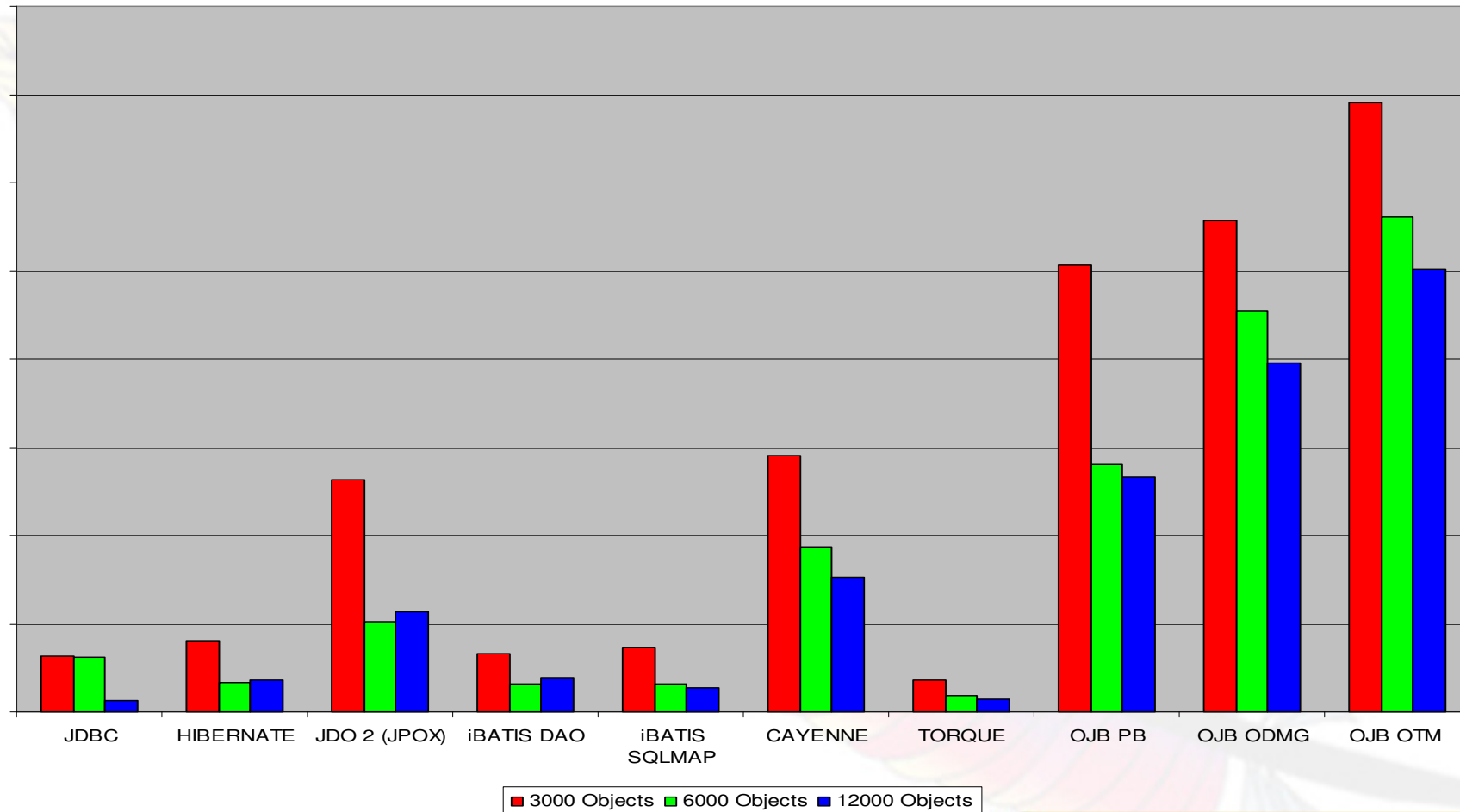
- Modified Benchmark from OJB
- Uses HSQL DB, multithreaded
- Tests insert, retrieve, update, delete

- No hard numbers, just gives a feel about mapper performance
- Does not test any advanced feature

Mapping by Product



Fetch time/Object



Which one to use?

- This table is highly subjective! YMMV!

When	What
Torque, OJB, Cayenne	...if your Java is better than your SQL
iBATIS	...if your SQL is better than your Java / .NET
OJB, iBATIS, maybe Cayenne	...if you need J2EE integration
Torque	...if you need results fast



Any questions?



Thanks for your attention!