Entity EJB

倪志刚
nizhigang2000@gmail.com
欢迎大家来dev2dev交流

主要内容

1. EJB历史
2. BMP实现
3. CMP实现
4. 实体EJB的性能问题与容器的价值
5. 实体EJB常见模式与性能影响

1.EJB的历史

- 1995年Sun发布JDK1.0
- 1997年Sun发布Java Web Server
- 1998年3月Sun发布EJB 1.0规范
- 1998年Weblogic Server发布
- 1999年Sun发布J2EE 1.0规范
- 2000年 6 月Sun发布EJB 2.0规范
- 2003年4月Sun发布EJB2.1规范
- 2004年发布EJB3.0草案
- 2005年发布EJB3.0第二版（预览）

1 .1EJB 2.0

- MDB消息驱动Bean
- Local接口

1.2 EJB2.1

- 计时器服务
- JAX-RPC,扩充了对Web Service的支持
- 增强了EJB QL
- 增强了消息驱动BEAN技术

1.3 EJB3.0

- 完全基于POJO的组件模型
- 简化EJB的设计，避免对组件的侵入。在3.0中，Home接口，组件接口（远程接口和本地接口），声明周期都是可选的
- 使用依赖注入
- EJB 3 .0成为一个真正的O/R Mapping工具

EntityManager非常类似Hibernate的Session为所有的实体对象提供统一的，无类型的持久化管理，新的EJB QL非常类似于Hibernate　QL，允许直接执行SQL语句

2 实体EJB

- 实体Bean的本质：

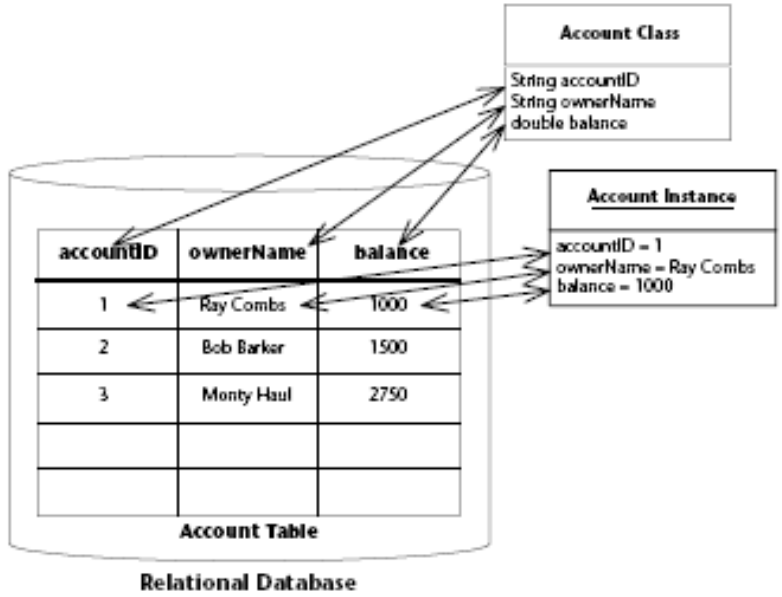
Class – Table

Bean实例－Record

- 实体Bean的类型

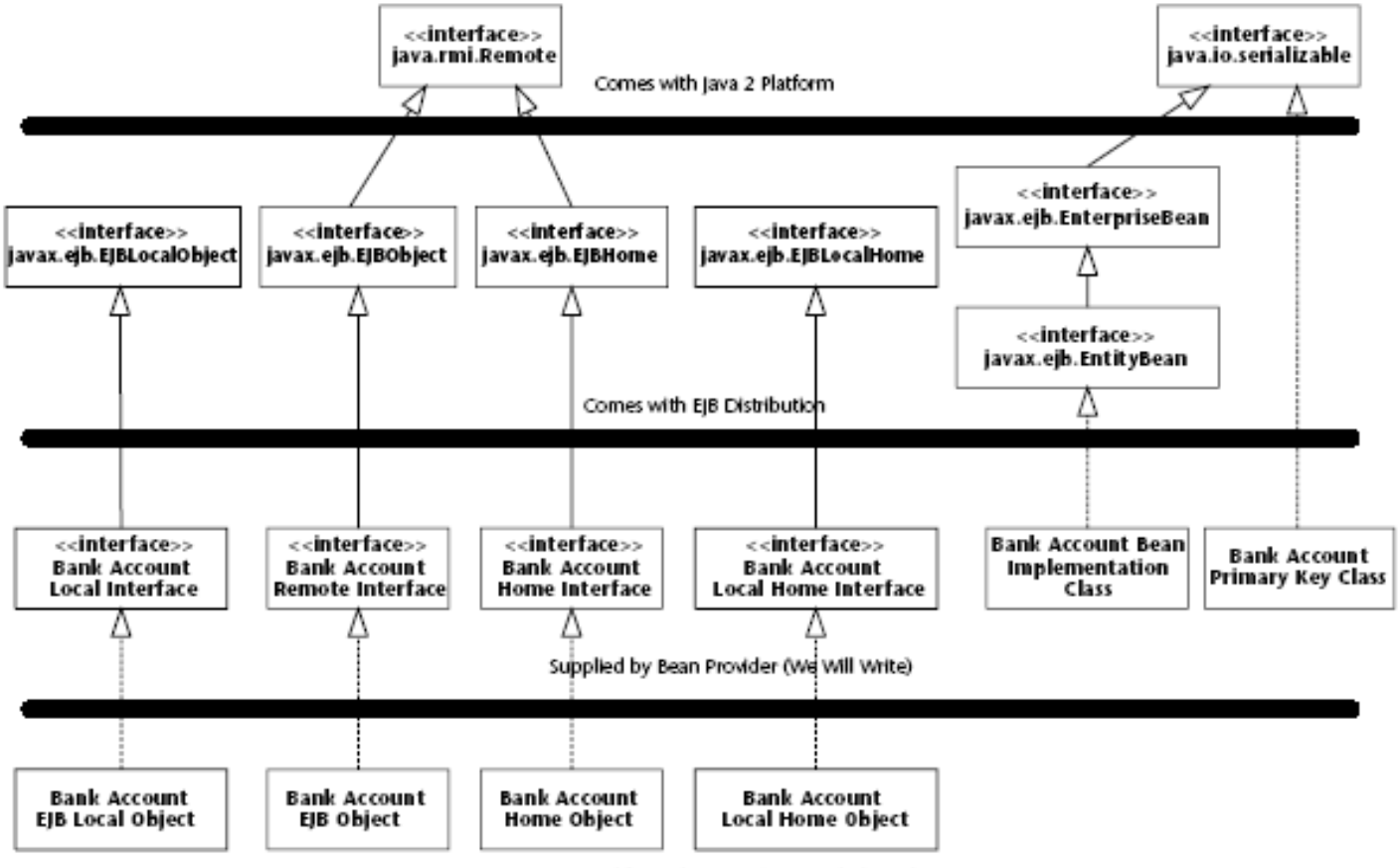
BMP（Bean管理的持久化）

CMP（容器管理的持久化）

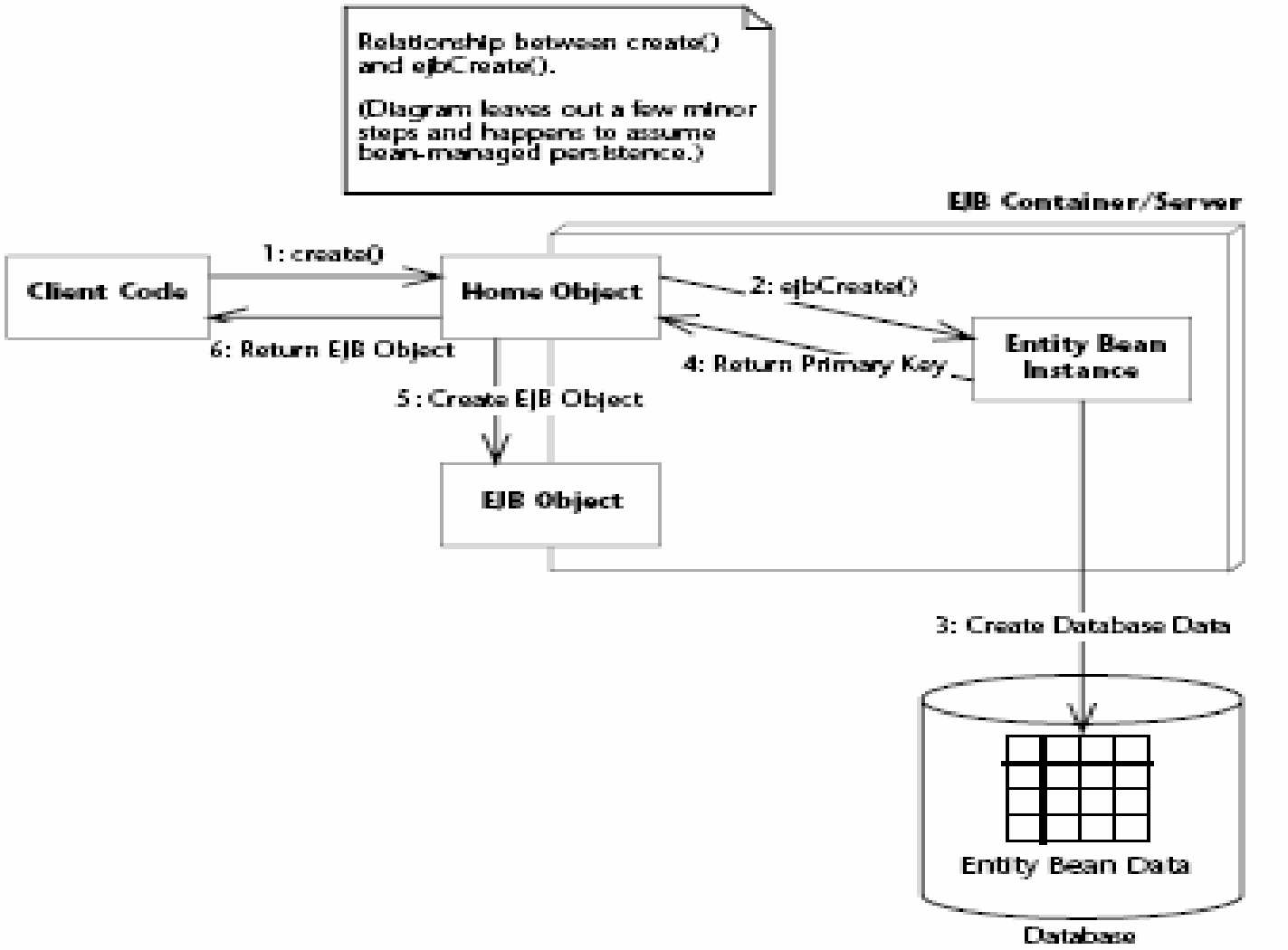


2.1BMP方法解析—继承关系

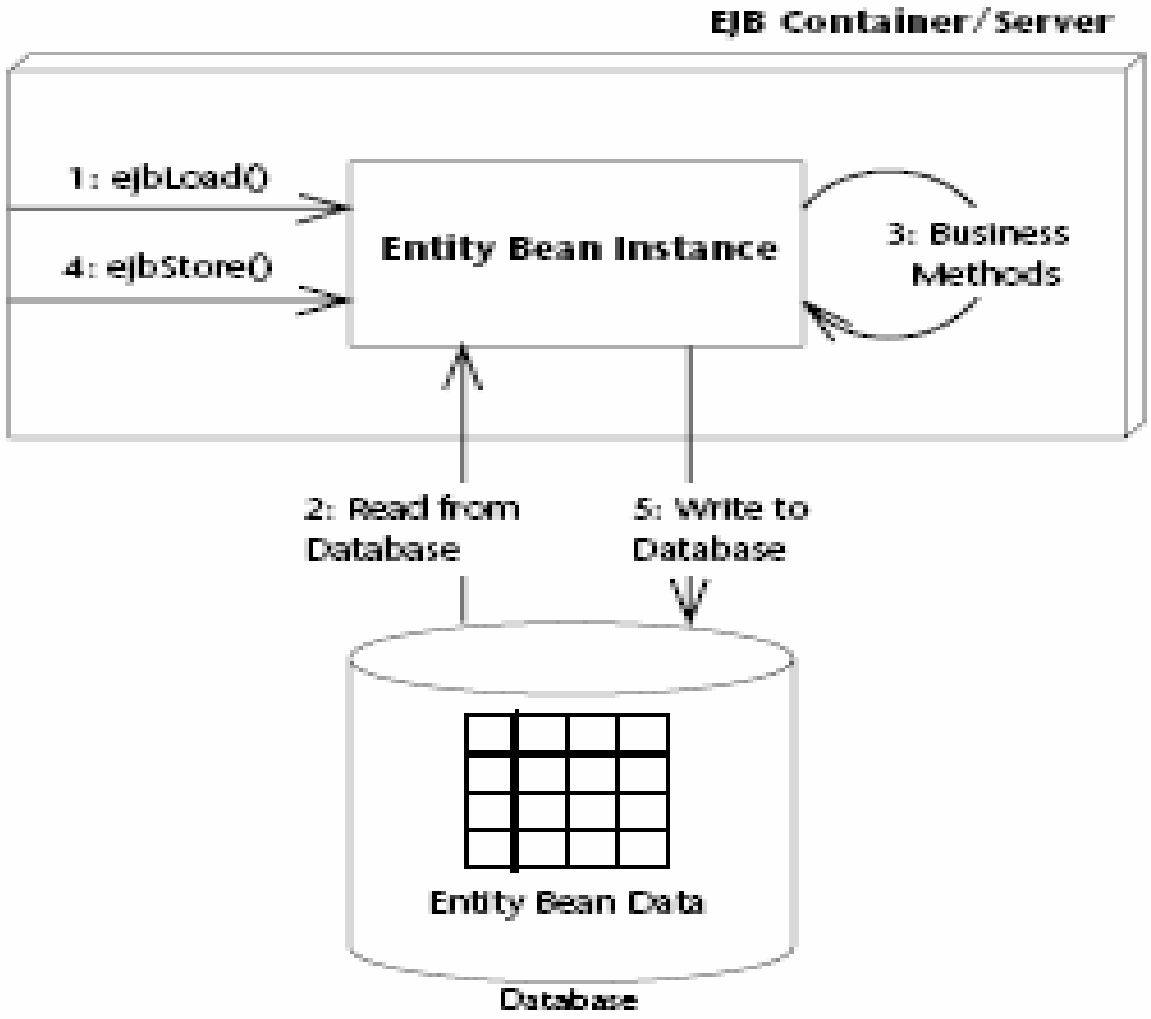
- BMP结构图



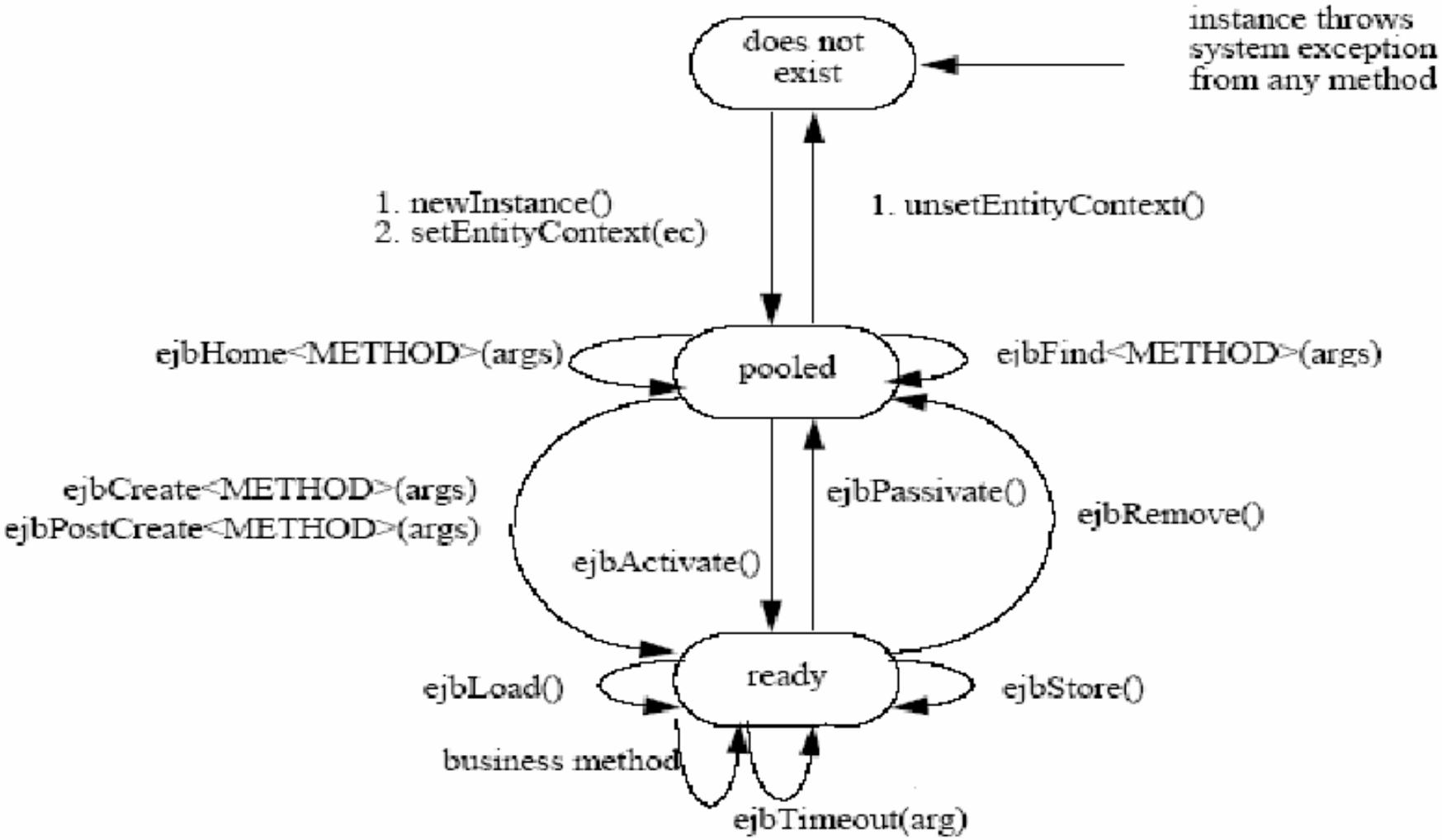
2.2 EJB远程调用序列图



2.3交互过程



2.4BMP生命周期



2.5EJB实例

- private EntityContext ctx;
- private String accountId; // also the primary Key
- private double balance;

2.5.1方法解析－ejbCreate()

- public String ejbCreate(String accountId, double initialBalance) throws CreateException
- {
- this.accountId = accountId;
- this.balance = initialBalance;
- Connection con = null;
- PreparedStatement ps = null;

2.5.2方法解析－ejbCreate()

```
try {
      con = getConnection();
      ps = con.prepareStatement("insert into ejbAccounts (id, bal) values (?, ?)");
      ps.setString(1, accountId);
      ps.setDouble(2, balance);
      if (ps.executeUpdate() != 1) {
        String error = "JDBC did not create any row";
        log(error);
        throw new CreateException (error);
      }
      return accountId;
```

2.5.3方法解析－ejbRemove()

- public void ejbRemove() {
- Connection con = null;
- PreparedStatement ps = null;
- con = getConnection();
- accountId = (String) ctx.getPrimaryKey();
- ps = con.prepareStatement("delete from ejbAccounts where id = ?");
- ps.setString(1, accountId);
- if (!(ps.executeUpdate() > 0)) {
- String error = "AccountBean (" + accountId + " not found";
- }

2.5.4方法解析－ejbPostCreate()

- public void ejbPostCreate(String accountId, double initialBalance)
- 与ejbCreate()中的参数一致，有一个ejbCreate方法，就要求有一个ejbPostCreate()方法

2.5.5方法解析－ejbLoad()

- public void ejbLoad() {
- Connection con = null;
- PreparedStatement ps = null;
- accountId = (String) ctx.getPrimaryKey();
- con = getConnection();
- ps = con.prepareStatement("select bal from ejbAccounts where id = ?");
- ps.setString(1, accountId);
- ps.executeQuery();
- ResultSet rs = ps.getResultSet();
- if (rs.next()) {
- balance = rs.getDouble(1);
- } else { … …

2.5.6EJB方法解析－ejbStore()

- public void ejbStore() {
- Connection con = null;
- PreparedStatement ps = null;
- con = getConnection();
- ps = con.prepareStatement("update ejbAccounts set bal = ? where id = ?");
- ps.setDouble(1, balance);
- ps.setString(2, accountId);
- if (!(ps.executeUpdate() > 0)) {
- String error = "ejbStore: AccountBean (" + accountId + ") not updated";
- throw new NoSuchEntityException (error);
- }
-

2.5.7EJB方法解析－ejbFindByPrimaryKey（）

- public String ejbFindByPrimaryKey(String pk)
- throws ObjectNotFoundException
- {
- Connection con = null; PreparedStatement ps = null;
- con = getConnection();
- ps = con.prepareStatement("select bal from ejbAccounts where id = ?");
- ps.setString(1, pk);
- ResultSet rs = ps.executeQuery();
- if (rs.next()) { balance = rs.getDouble(1);
- } else {……
- throw new ObjectNotFoundException (error);
- return pk;

2.5.8ＥＪＢ方法解析－ ejbFindBigAccounts()

- public Collection ejbFindBigAccounts(double balanceGreaterThan) {
- Connection con = null; PreparedStatement ps = null;
- String pk; con = getConnection();
- ps = con.prepareStatement("select id from ejbAccounts where bal > ?");
- ps.setDouble(1, balanceGreaterThan);
- ps.executeQuery();
- ResultSet rs = ps.getResultSet();
- Vector v = new Vector();
- while (rs.next()) { pk = rs.getString(1);
- v.addElement(pk); }
- return v;

2.5.9EJB方法解析—其它方法

- public void setEntityContext(EntityContext ctx) {
- log("setEntityContext called");
- this.ctx = ctx; }

- public void unsetEntityContext() {
- log("unsetEntityContext");
- this.ctx = null; }

- public void ejbActivate() {
- log("ejbActivate (" + id() + ")"); }

- public void ejbPassivate() {
- log("ejbPassivate (" + id() + ")"); }

2.5.10ＥＪＢ方法解析－EntityContext类

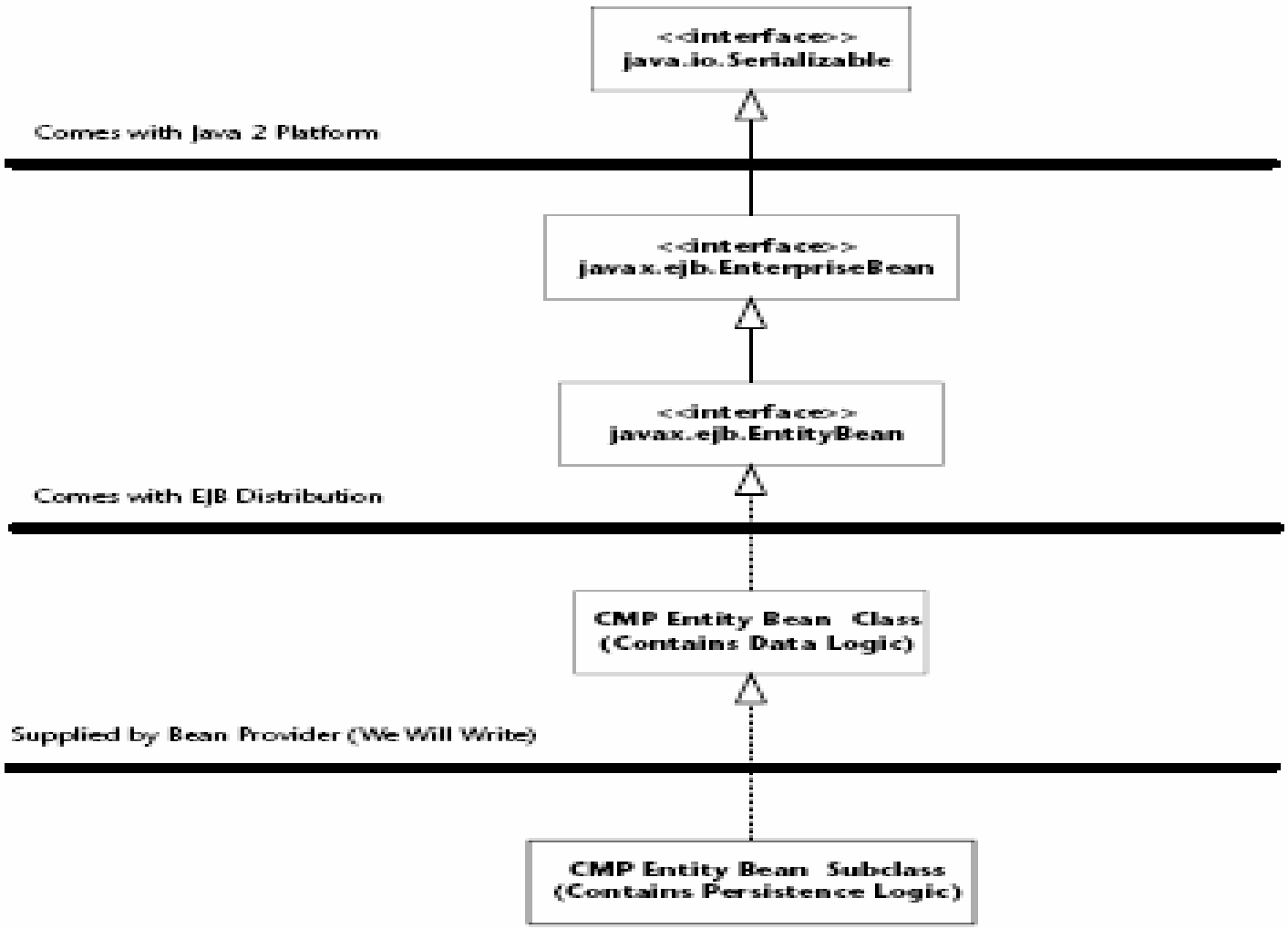
- public abstract interface EntityContext extends EJBContext {

- EJBLocalObject getEJBLocalObject() throws IllegalStateException;
- EJBObject getEJBObject() throws IllegalStateException;
- Object getPrimaryKey() throws IllegalStateException;

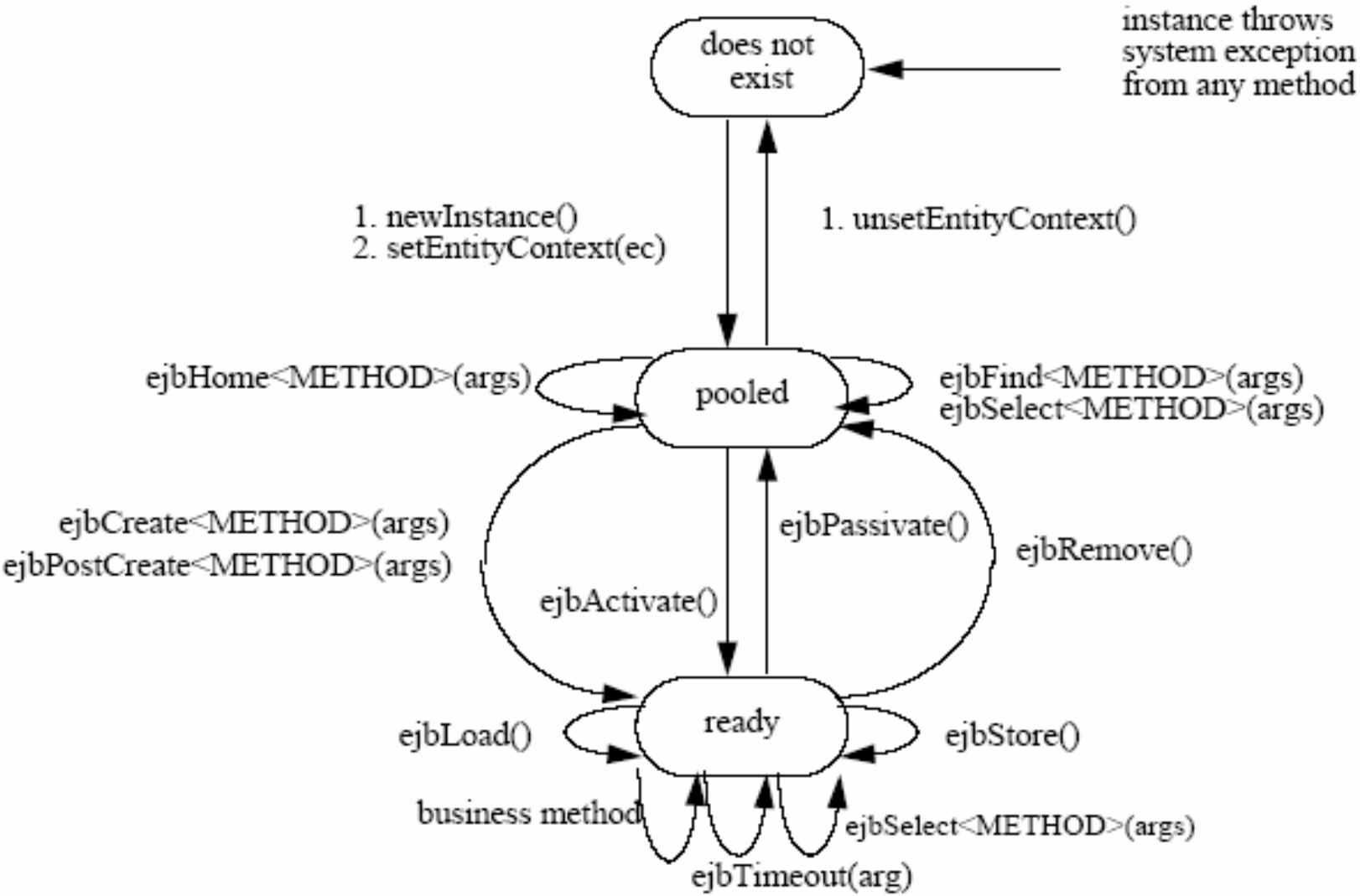
2.5.11 BMP总结

- JDBC的一种包装
- 在wls中实例Bean在编译时会由编译器自动继承一个子类，Accounts_xxxxxx_Impl，实际在运行时，会生成该类的实例

3 CMP－容器管理的持久



3.1CMP生命周期



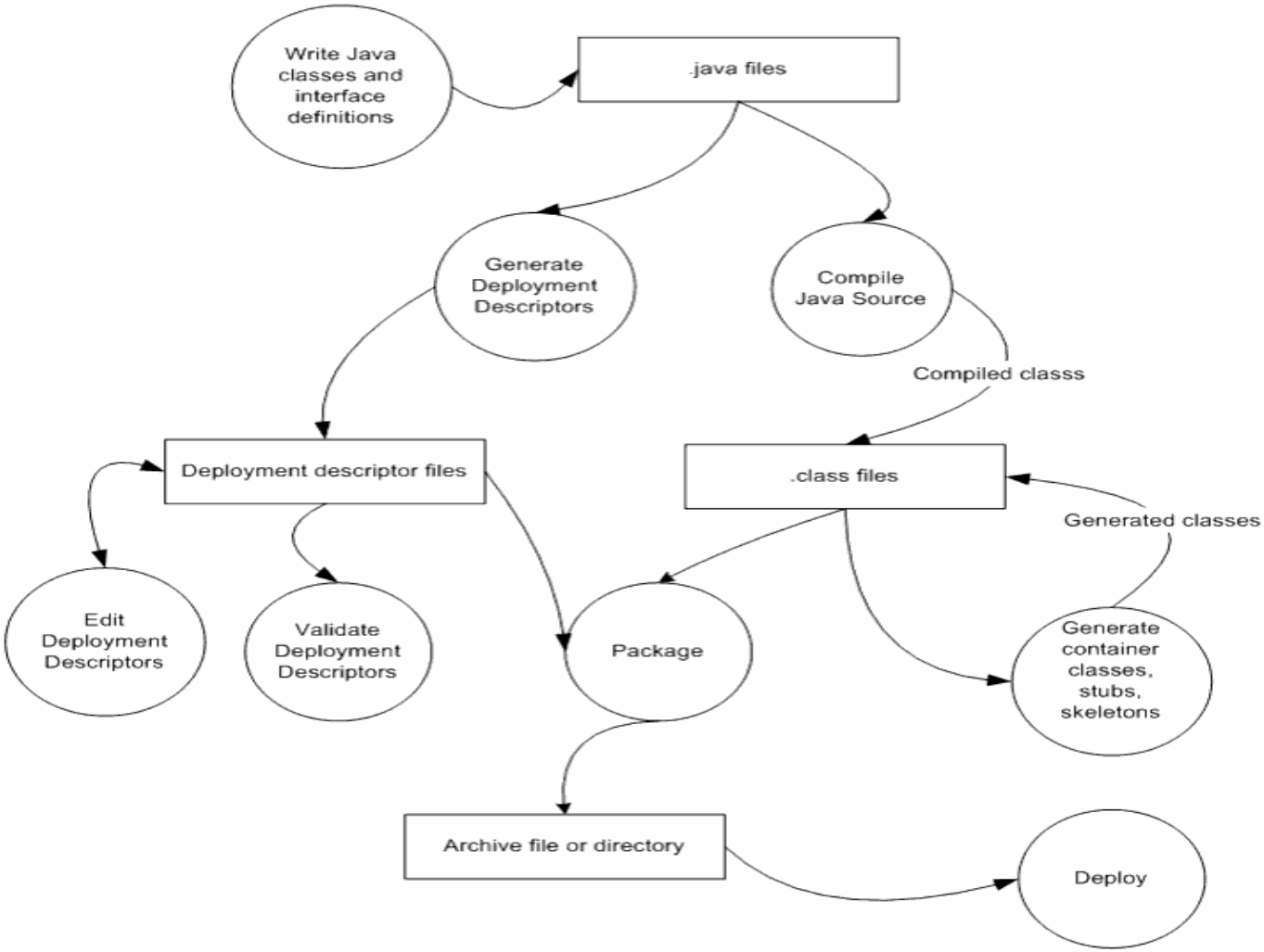
3.2ＣＭＰ解析-CMP类

- abstract public class AccountBean implements EntityBean
- abstract public String getAccountId();
- abstract public void setAccountId(String val);

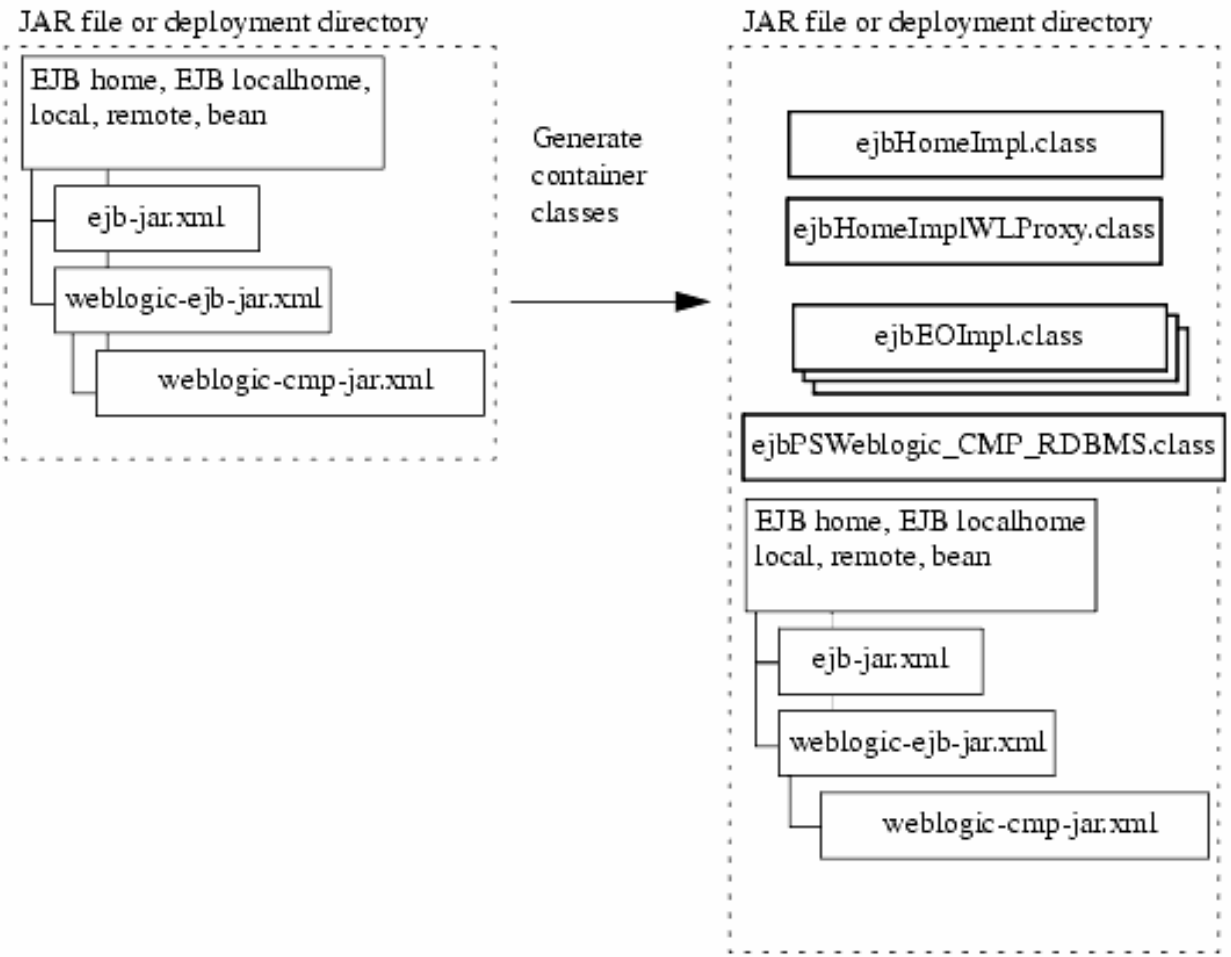
- abstract public double getBalance();
- abstract public void setBalance(double val);

- abstract public String getAccountType();
- abstract public void setAccountType(String val);

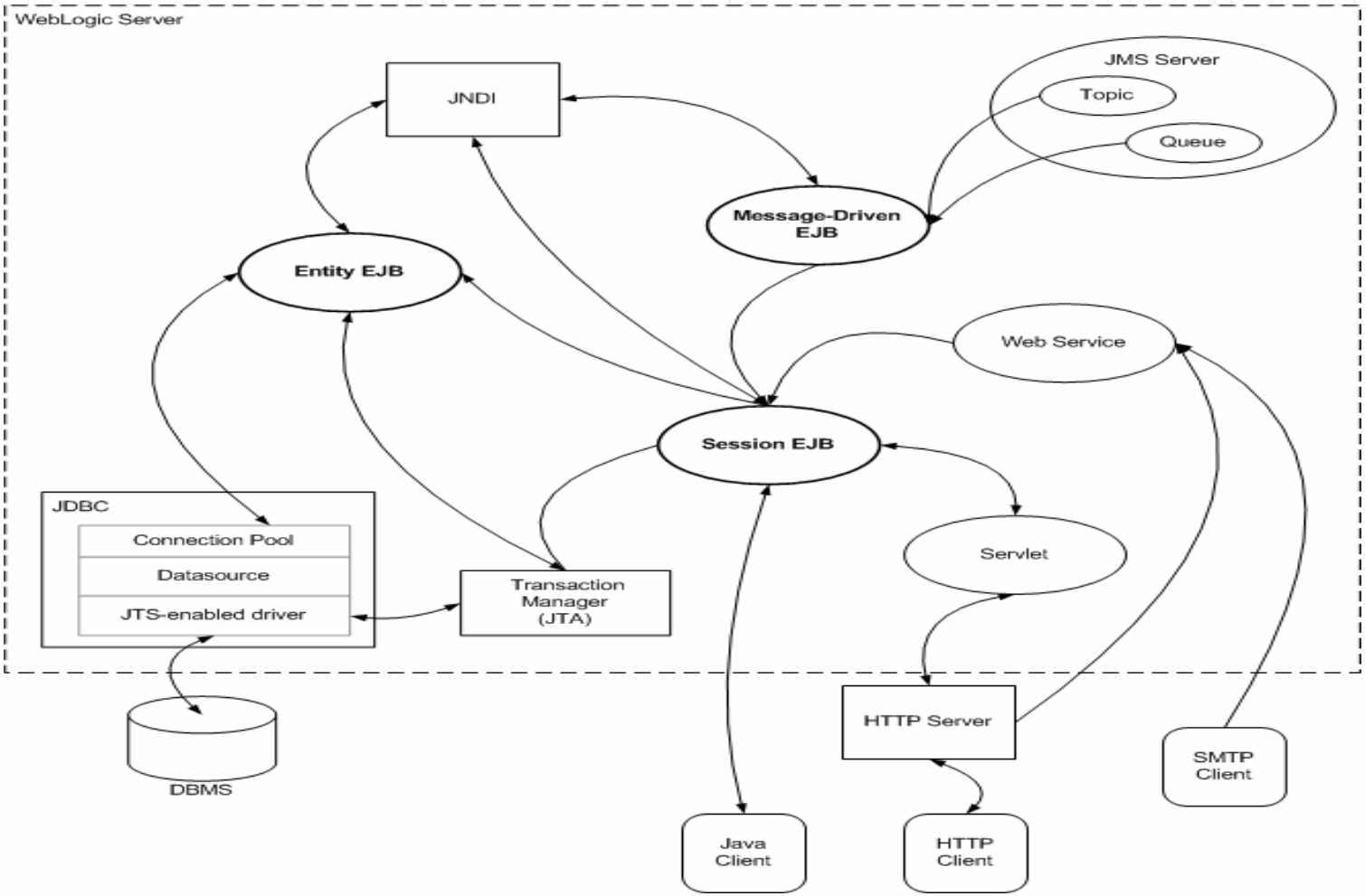
3.3 CMP解析—开发部署过程



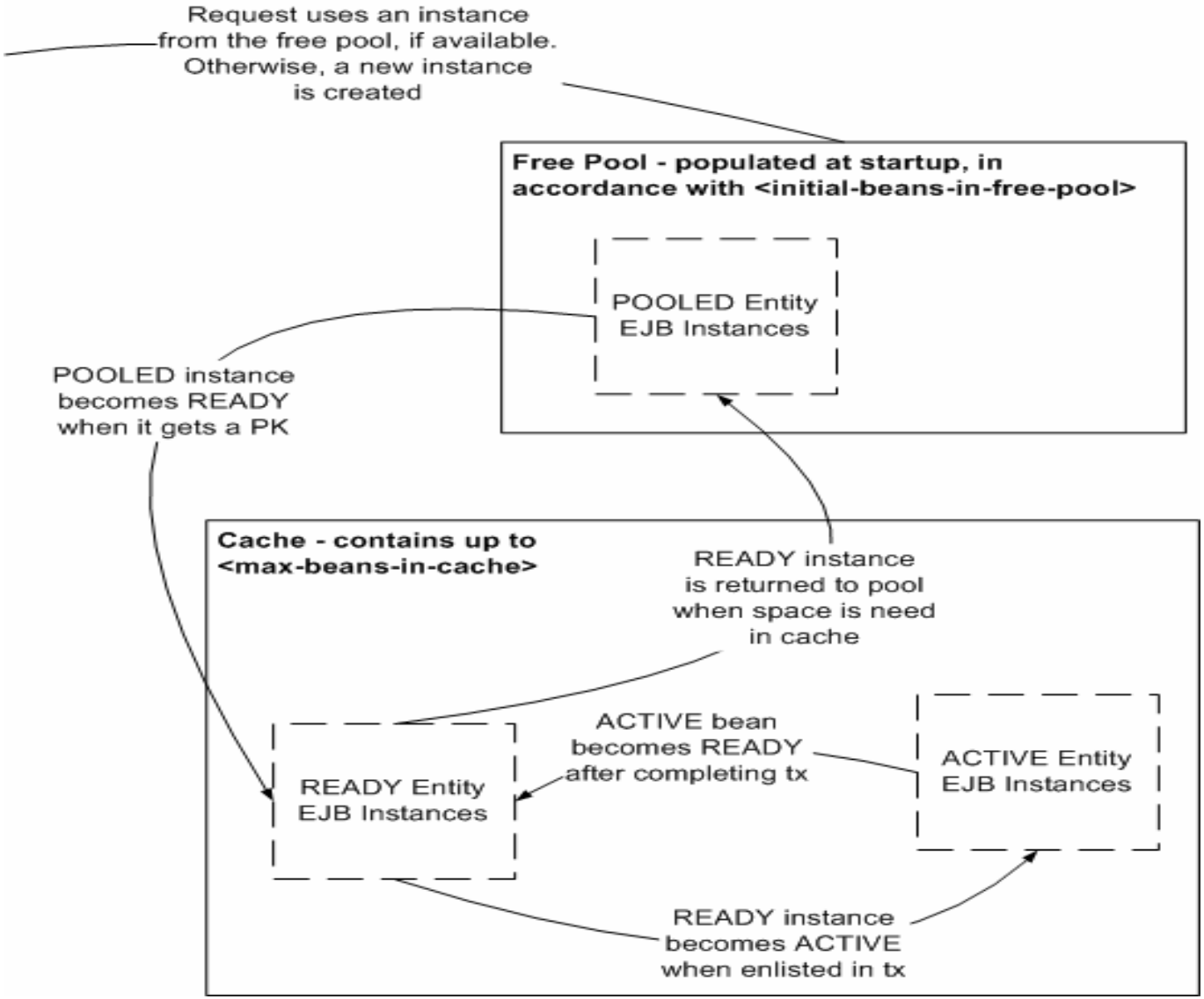
3.4 CMP在Weblogic中的结构



3.5ＥＪＢ解析－Weblogic中ＥＪＢ结构



3.7 CMP在WIs中的生命周期



3.8.1 CMP解析－XML映射文件

- <weblogic-rdbms-bean>
- <ejb-name>Accounts</ejb-name>
- <data-source-jndi-name>DataSource</data-source-jndi-name>
- <table-map>
- <table-name>ACCOUNTS</table-name>
- <field-map>
- <cmp-field>id</cmp-field>
- <dbms-column>ID</dbms-column>
- </field-map>
- <field-map>
-

3.8.2 CMP解析—XML映射文件

- <cmp-field>ownername</cmp-field>
- <dbms-column>OWNERNAME</dbms-column>
- </field-map>
- <field-map>
- <cmp-field>balance</cmp-field>
- <dbms-column>BALANCE</dbms-column>
- </field-map>
- </table-map>
- </weblogic-rdbms-bean>
- </weblogic-rdbms-jar>

3.10 CMP的其它特性

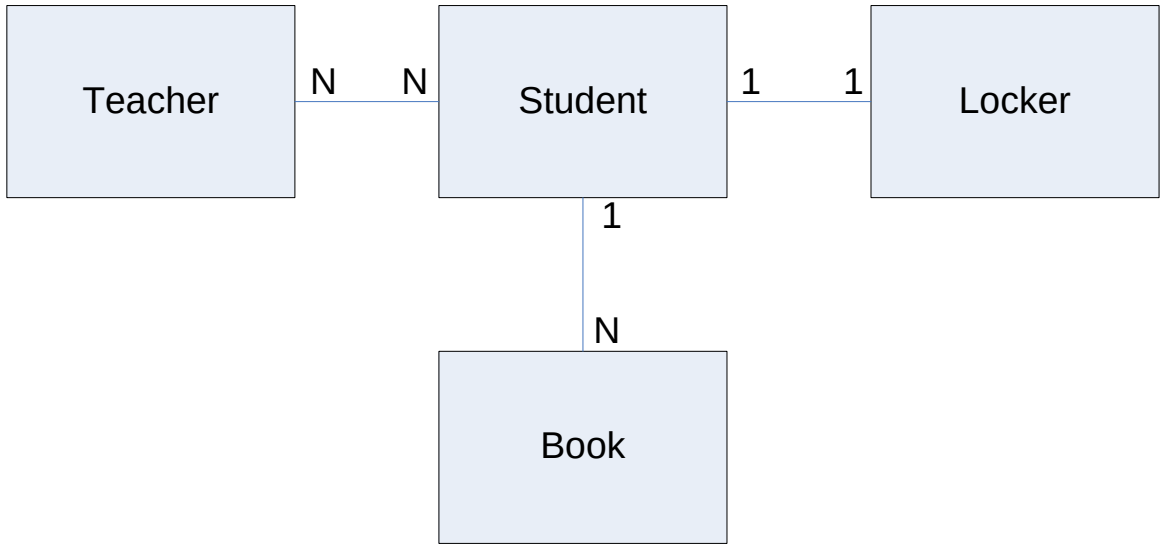
- EJB QL-- SQL
- CRM--Relation Ship

3.10.1EJB QL

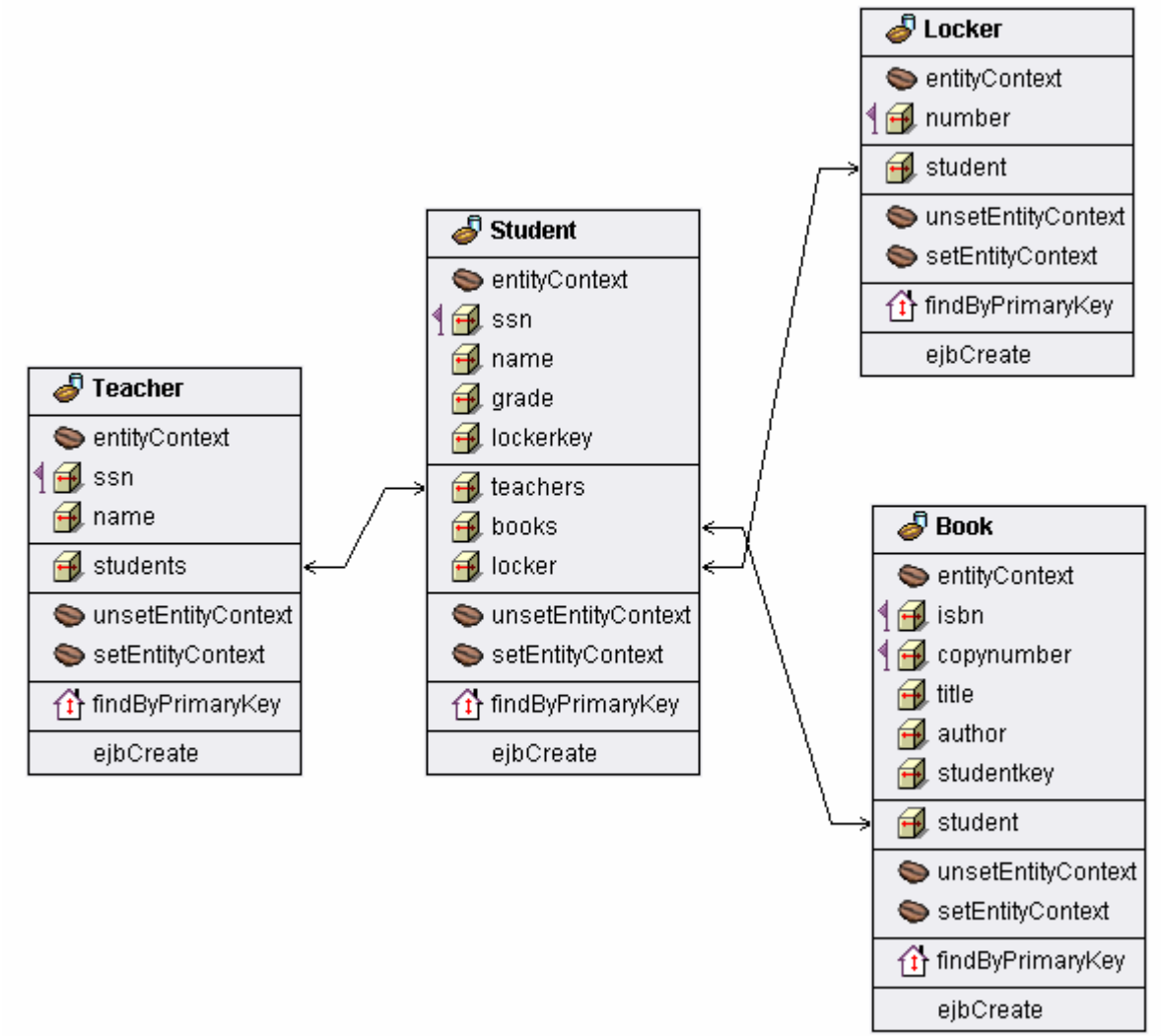
- EJB QL是基于SQL-92标准
- 可以在其中用于Finder()和Select()方法
- Select()方法不对远程公开，只针对标识实例调用,在select()中可以使用灵活的返回值，Bean实例,几个字段的集合，bean的集合

3.10.2CRM(容器管理的关系)

- 单向关系与比向关系
- 1：1映射
- 1：N映射
- N：N映射
- Cache关系
- 级联删除



3.10.3CMR



```
<weblogic-rdbms-relation>
  <relation-name>student-has-locker</relation-name>
  <weblogic-relationship-role>
   <relationship-role-name>StudentRelationshipRole</relationship-role-name>
   <relationship-role-map>
    <foreign-key-table>STUDENTCMPTABLE</foreign-key-table>
    <primary-key-table>LOCKERCMPTABLE</primary-key-table>
    <column-map>
     <foreign-key-column>LOCKERKEY</foreign-key-column>
     <key-column>NUMBER</key-column>
    </column-map>
   </relationship-role-map>
  </weblogic-relationship-role>
</weblogic-rdbms-relation>
```

```
<weblogic-rdbms-relation>
  <relation-name>student-has-books</relation-name>
  <weblogic-relationship-role>
    <relationship-role-name>BookRelationshipRole</relationship-role-name>
    <relationship-role-map>
      <foreign-key-table>BOOKCMPTABLE</foreign-key-table>
      <primary-key-table>STUDENTCMPTABLE</primary-key-table>
      <column-map>
        <foreign-key-column>STUDENTKEY</foreign-key-column>
        <key-column>SSN</key-column>
      </column-map>
    </relationship-role-map>
  </weblogic-relationship-role>
</weblogic-rdbms-relation>
```

```
<weblogic-rdbms-relation>
  <relation-name>student-map-teachers</relation-name>
  <weblogic-relationship-role>
   <relationship-role-name>StudentRelationshipRole</relationship-role-name>
   <relationship-role-map>
    <foreign-key-table>STUDENTCMPTABLE</foreign-key-table>
    <primary-key-table>TEACHERCMPTABLE</primary-key-table>
    <column-map>
     <foreign-key-column>NAME</foreign-key-column>
     <key-column>NAME</key-column>
    </column-map>
    <column-map>
     <foreign-key-column>SSN</foreign-key-column>
     <key-column>SSN</key-column>
    </column-map>
   </relationship-role-map>
  </weblogic-relationship-role>
```

3.10.4CMR编程限制

- 直到调用了ejbPostCreate()方法，才能设置CMR

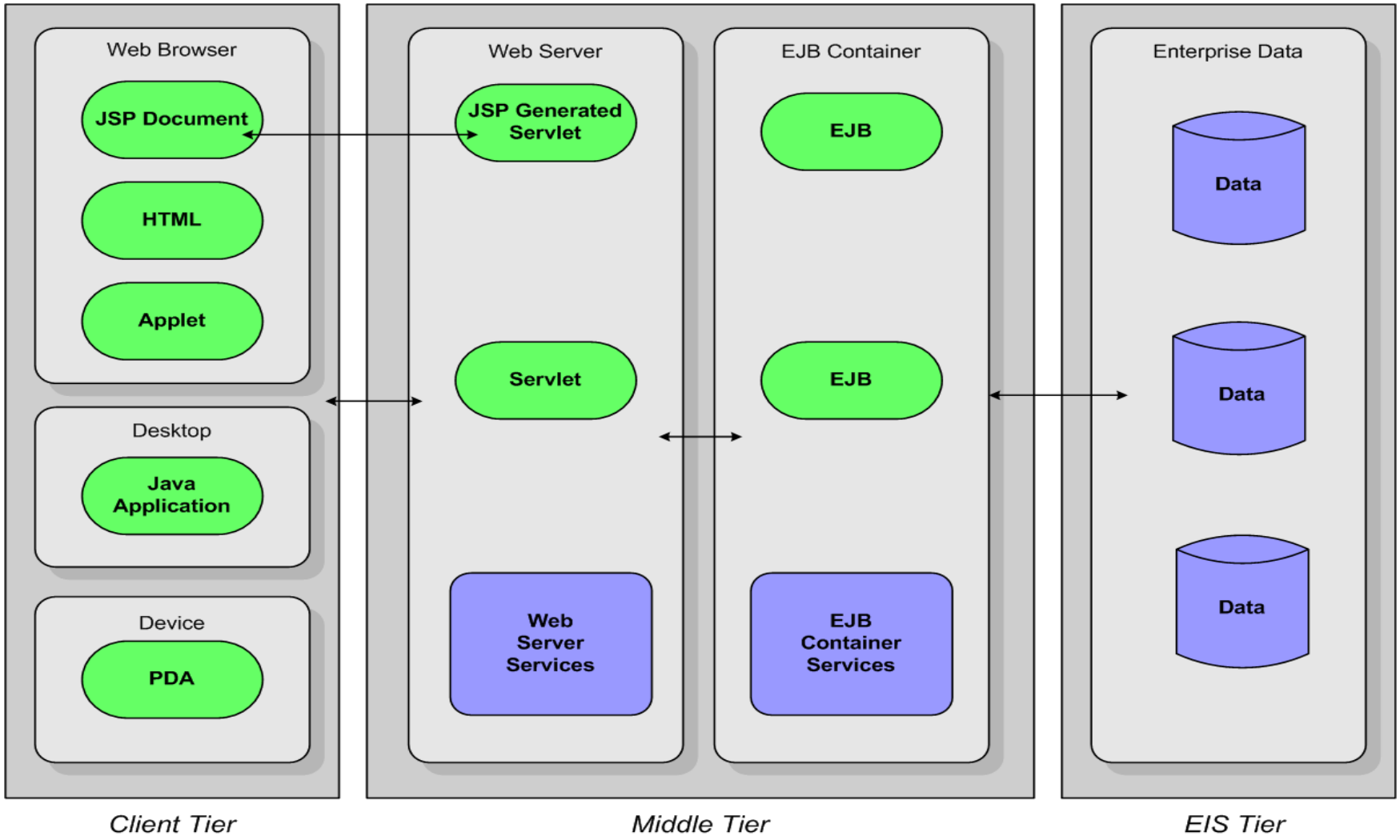
3.11 BMP Vs CMP

- 推荐使用CMP
- CMP使用了容器的优化技术
- 合理的缓存算法
- 封装了JDBC的访问
- 在取数据与更新数据时只针对必要的更新的数据

3.12使用原则

- 能用ＣＭＰ的地方尽量用ＣＭＰ
- 复杂多变的查询，用ＢＭＰ

4.典型J2EE构架



4.1实体Bean的作用

- O/R映射
- 数据缓存
- 声明性事务
- 数据一致性管理
- 安全性
- 分布式

4.2 EntityBean的优势

- 大型企业如电信，银行，电力，证券，等一些高端的应用
- Why?
- 安全，稳定性至上
- 集群环境的分布式计算功能
- 有成功的案例可参照
- 有成熟的开发团队支持
- 有庞大的公司支持
- 简洁的分布式事务处理
- 降低对开发人员素质的要求
- 有成熟的开发工具，可以降低开发难度，简化开发，如workshop,Jbuilder等。

4.3实体Bean面临的挑战

- 用户对实体Bean的理解不够深入
- 各种开源软件带来的影响，如Hibernate，Spring等
- 项目的成本控制

4.4性能杀手

- 表象：

1. EJB
2. JDBC

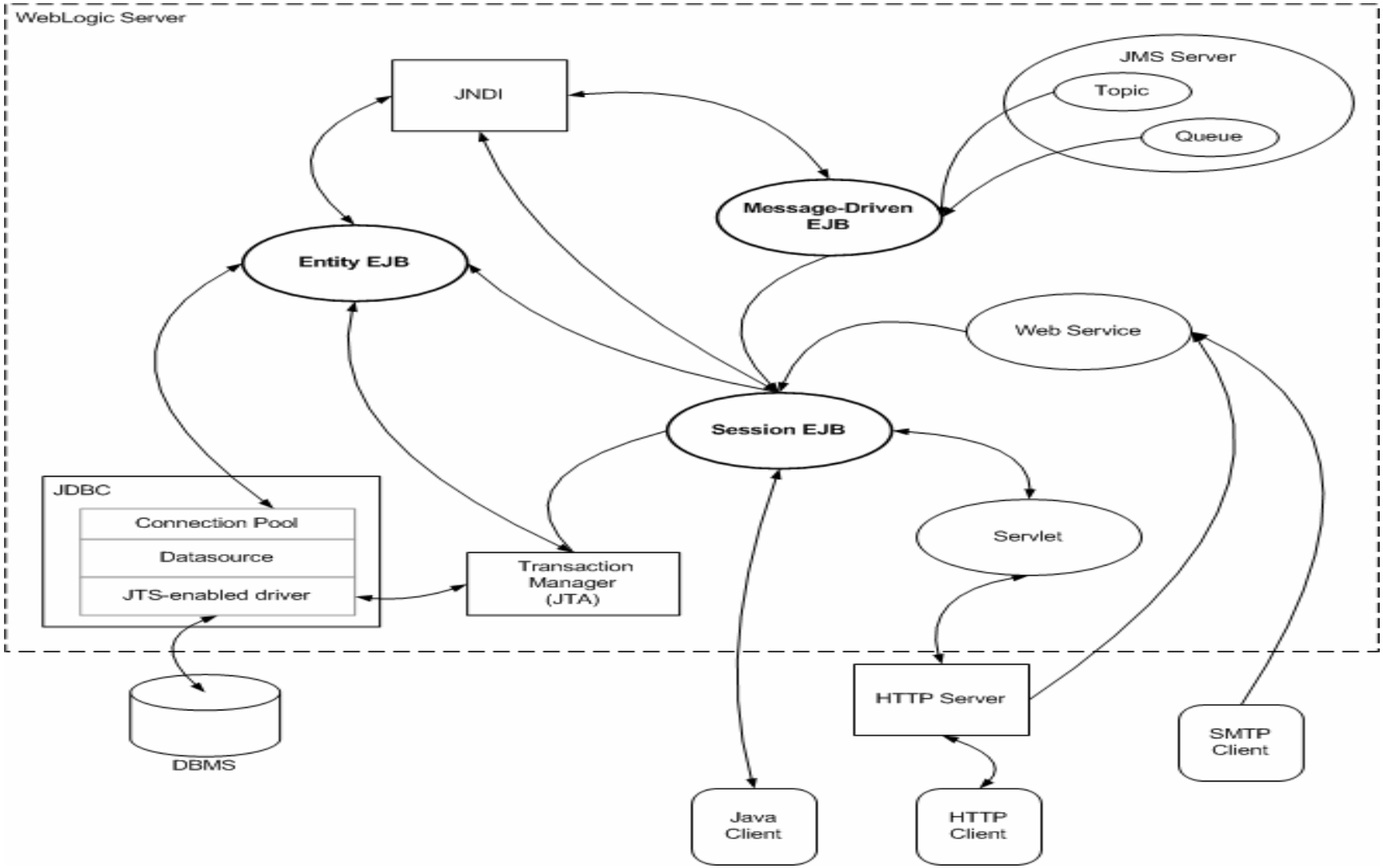
- 本质的性能消耗：

1. 序列化
2. 带宽不是瓶颈，而是CPU，非序列化设计比序列化要快１０倍
3. ＪＮＤＩ查找
4. I/O是不是？

4.5EntityBean 性能

- 性能好　Vs 性能差
- ＥＪＢ设计者的观点：笨重，强侵入性，概念混淆

4.6 J 2 E E 程序的性能消耗图



4.7 EJB容器的价值

- 设计价值：应该为开发者封装事务，组件生命周期，多线程，集群，分布式等低级API，让开发者专心关注于业务逻辑本身
- **EntityBean容器的价值**

1. 缓存：数据从数据库映射到内存，从内存中取数据比从数据库或者是另一个虚拟机上快10到50倍，不是只快了20%－50%
2. 缓存：J2EE中没有清晰的缓存提交机制，而是将其留给了最终用户和编程人员
3. 缓存算法：缓存大小，缓存命中率，缓存有效期，在内存中做越多的工作，就越少需要序列化，序列化率和缓存命中率成反比关系
4. 必要的数据库存取：只有需要的时候才从数据库中取，只有经过修改的数据才会存到数据库，

4.8 EJB容器的其它价值

- JTS/JTA：Java Transaction Service/Java Transaction Architecture
- 分布式事务：可以同时连接多个事务服务器进行两阶段提交

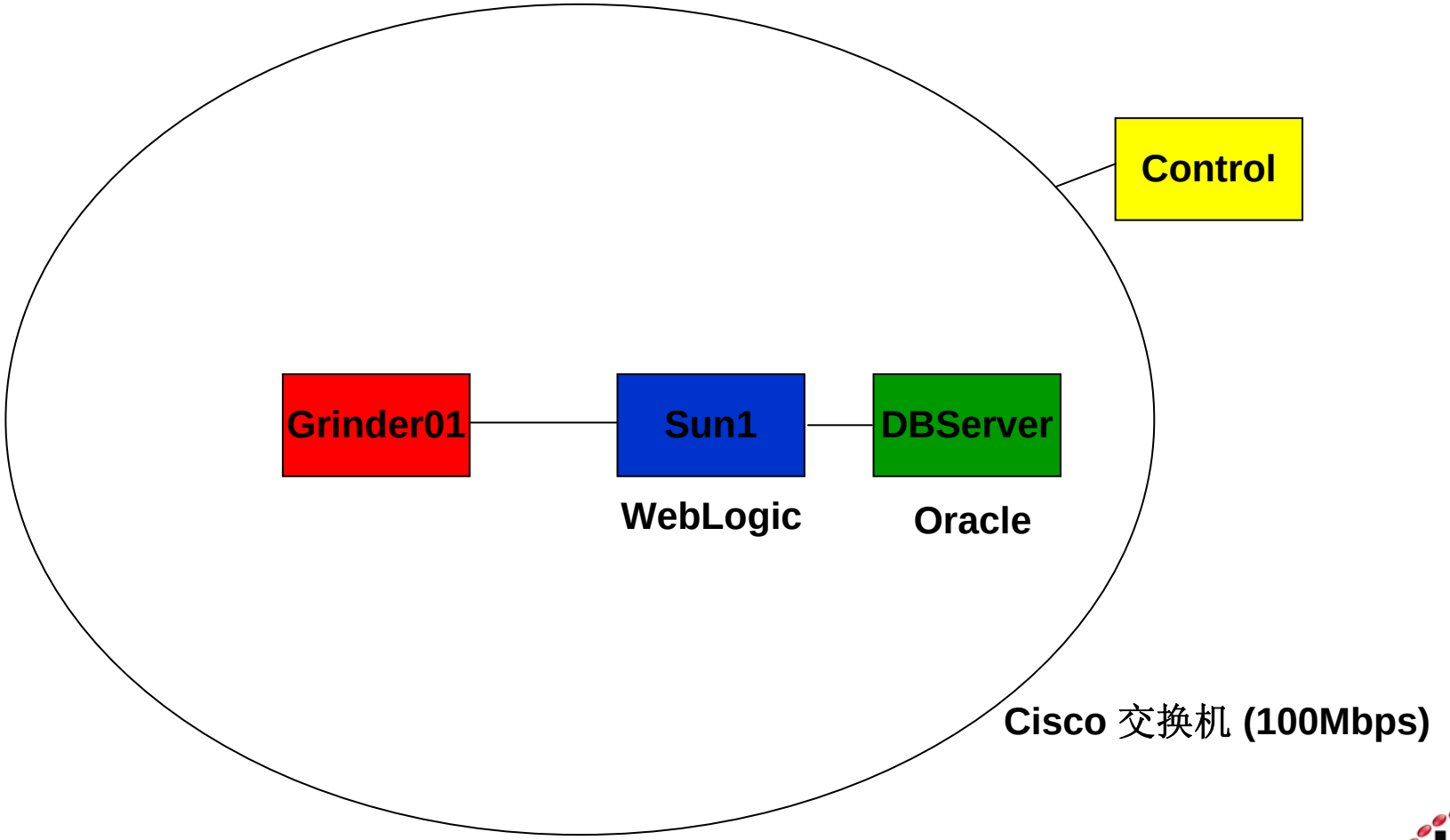
4.9获得性能根本途径

- 减少ＲＭＩ
- 减少序列化
- 减少ＪＮＤＩ查找

5实体ＥＪＢ相关模式分类

- 层次构架模式
- 层间数据传输模式
- 事务模式

5.1测试环境—硬件环境


Control
Grinder01
Sun1
DBServer
WebLogic
Oracle
Cisco 交换机 (100Mbps)

5.3测试脚本 1

- grinder.**test0**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test0**.parameter.post=**search-a.dat**
- grinder.**test1**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test1**.parameter.post=**search-b.dat**
- grinder.**test2**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test2**.parameter.post=**search-c.dat**
- grinder.**test3**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test3**.parameter.post=**search-d.dat**
- grinder.**test4**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test4**.parameter.post=**search-e.dat**

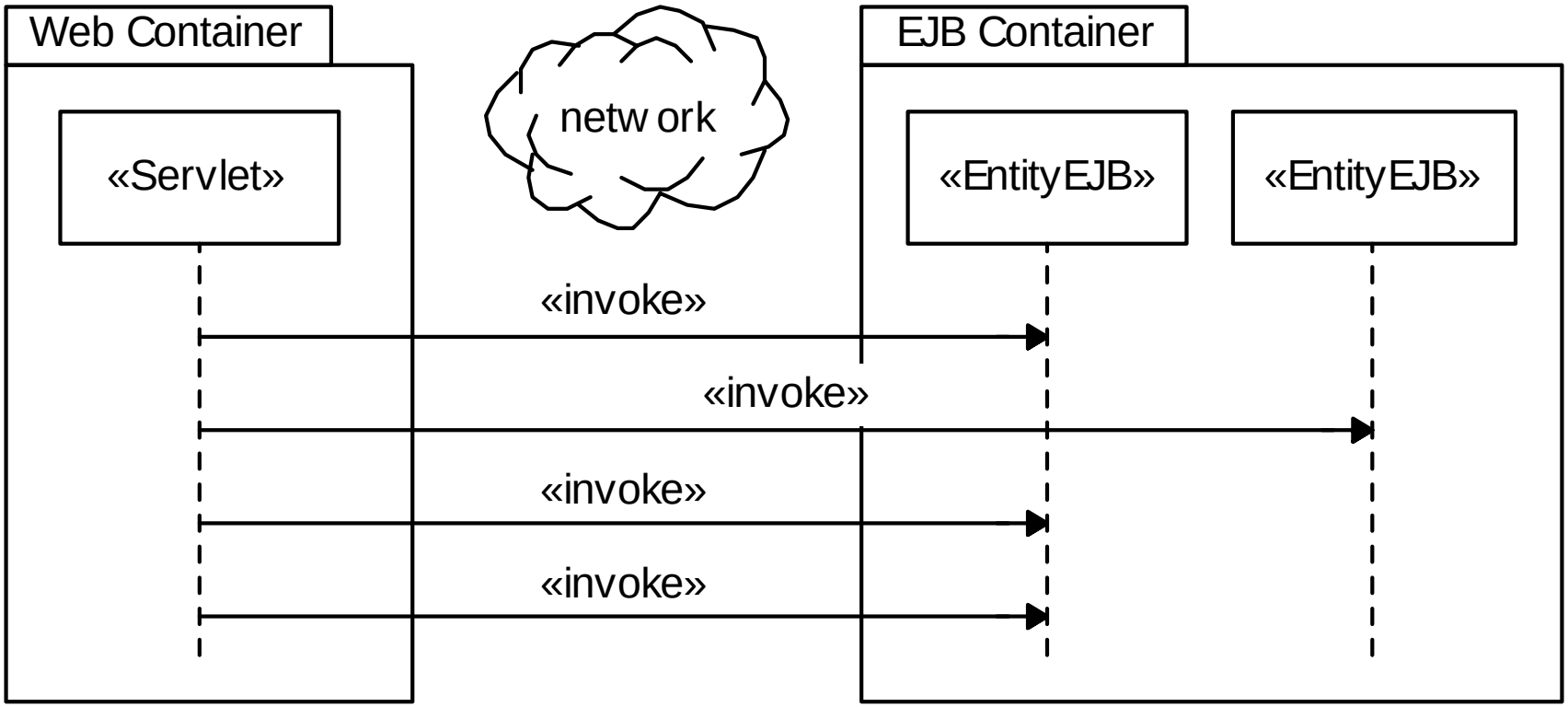
No think time!

5.4测试脚本 2

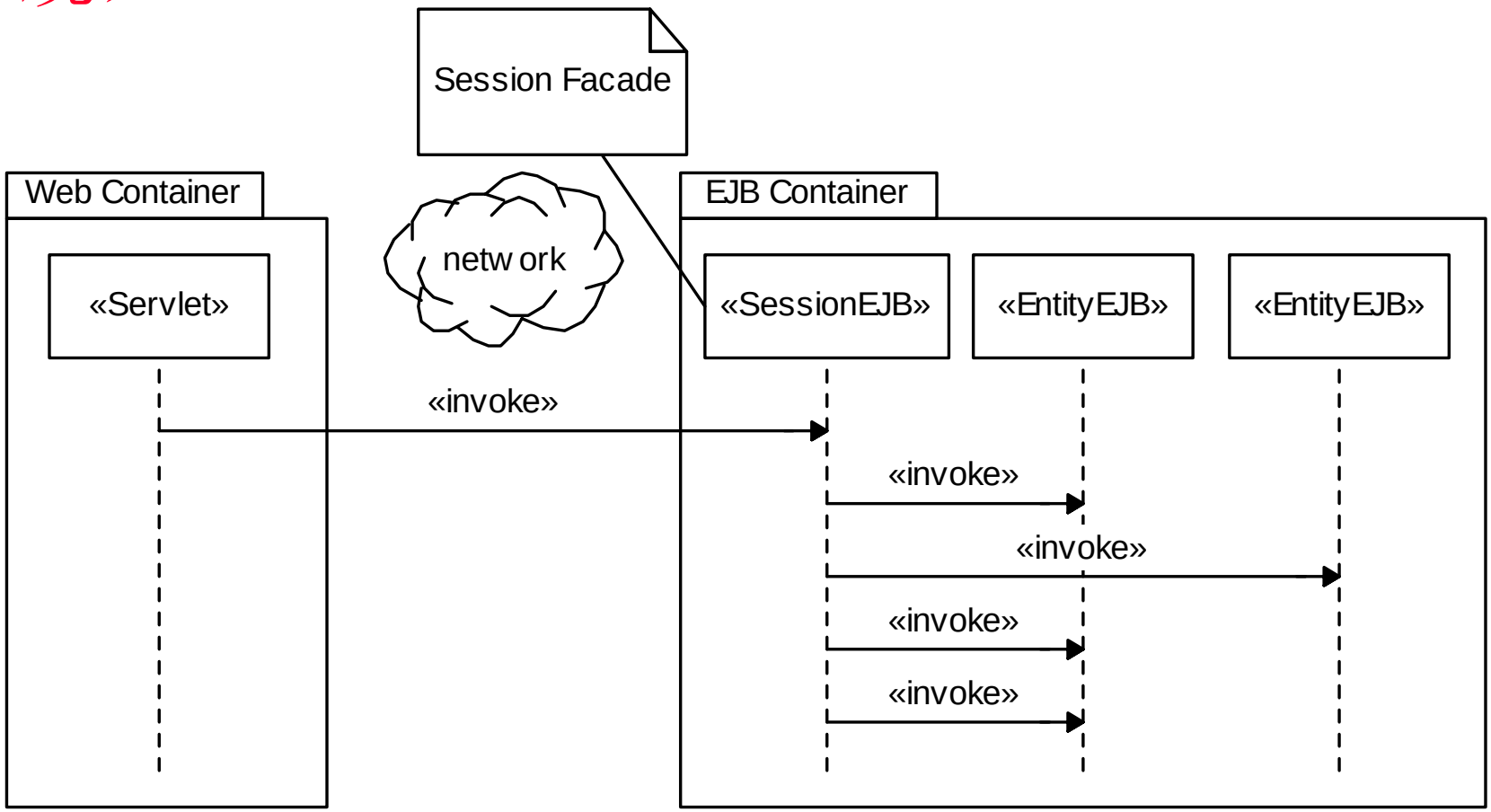
- grinder.**test5**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test5**.parameter.post=**search-f.dat**
- grinder.**test6**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test6**.parameter.post=**search-g.dat**
- grinder.**test7**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test7**.parameter.post=**search-h.dat**
- grinder.**test8**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test8**.parameter.post=**search-i.dat**
- grinder.**test9**.parameter.url=http://sun2:7001/jazzcat/Dispatcher
- grinder.**test9**.parameter.post=**search-j.dat**
- search-a.dat:
- test_id=FacadeOffTest&**search_string=a**&fname=&lname=&abbrev=&query=true

No think time!

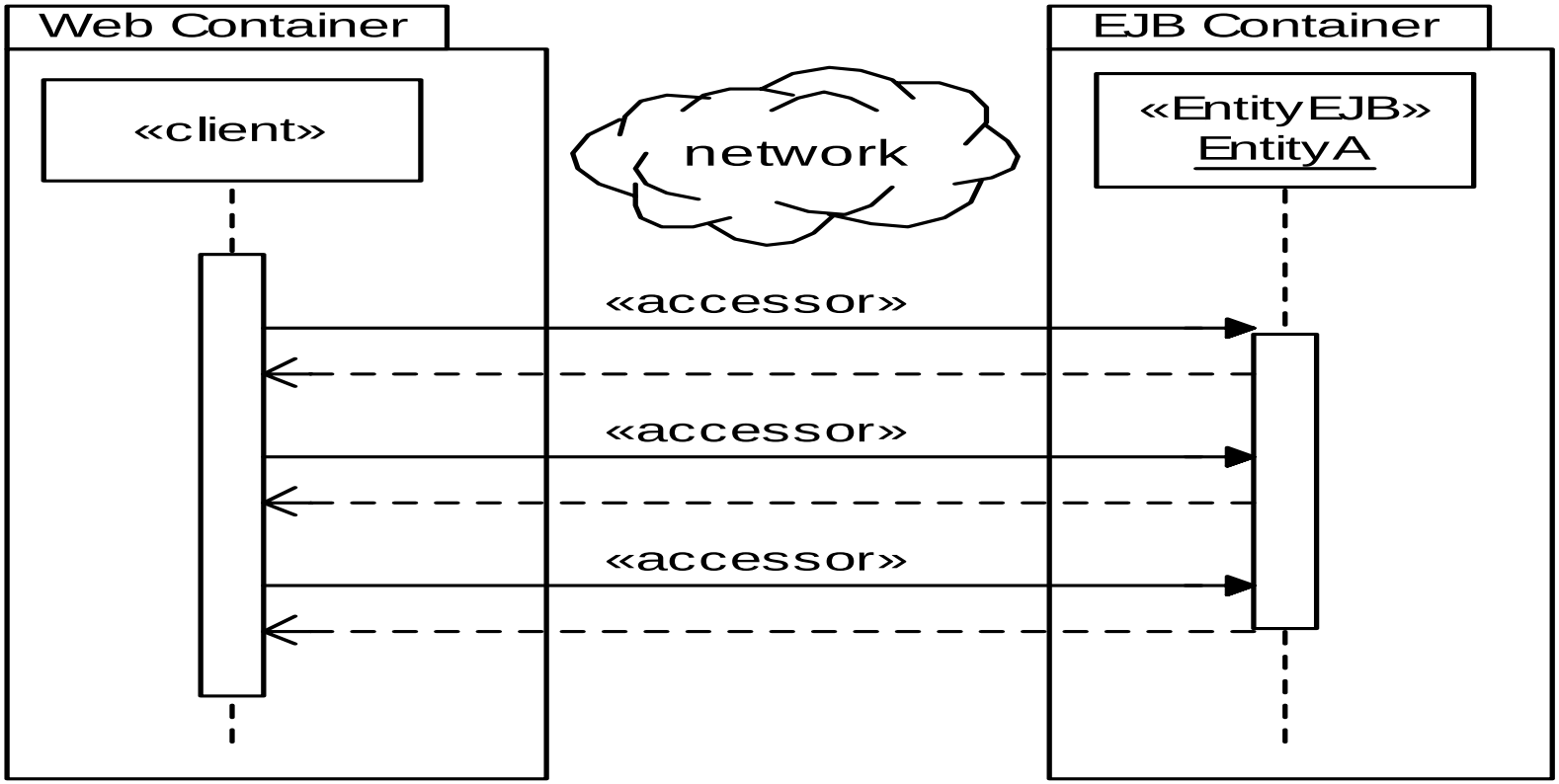
5.5常用模式－Session Façade（会话外观）



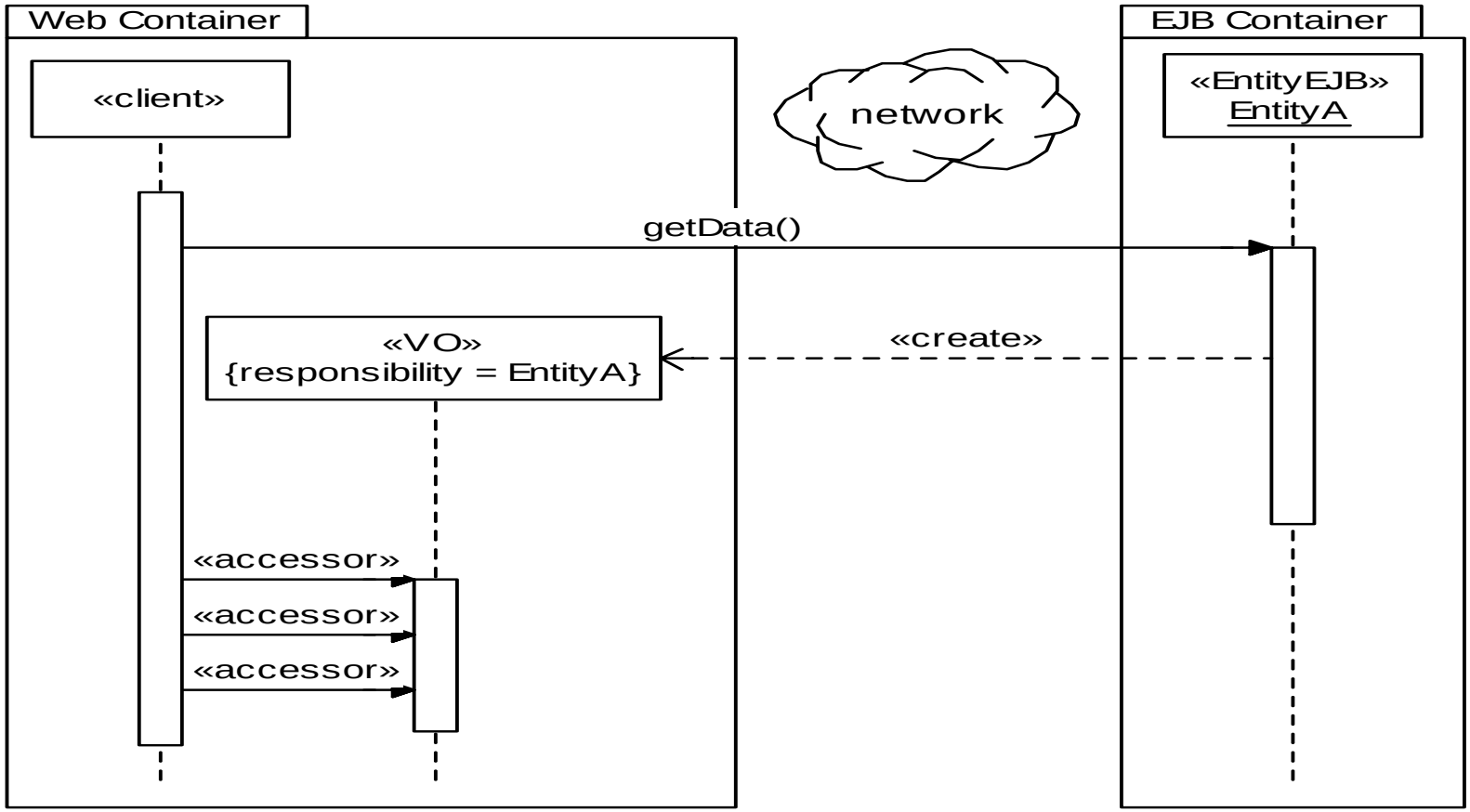
5.5.1常用模式－Session Façade（会话外观）



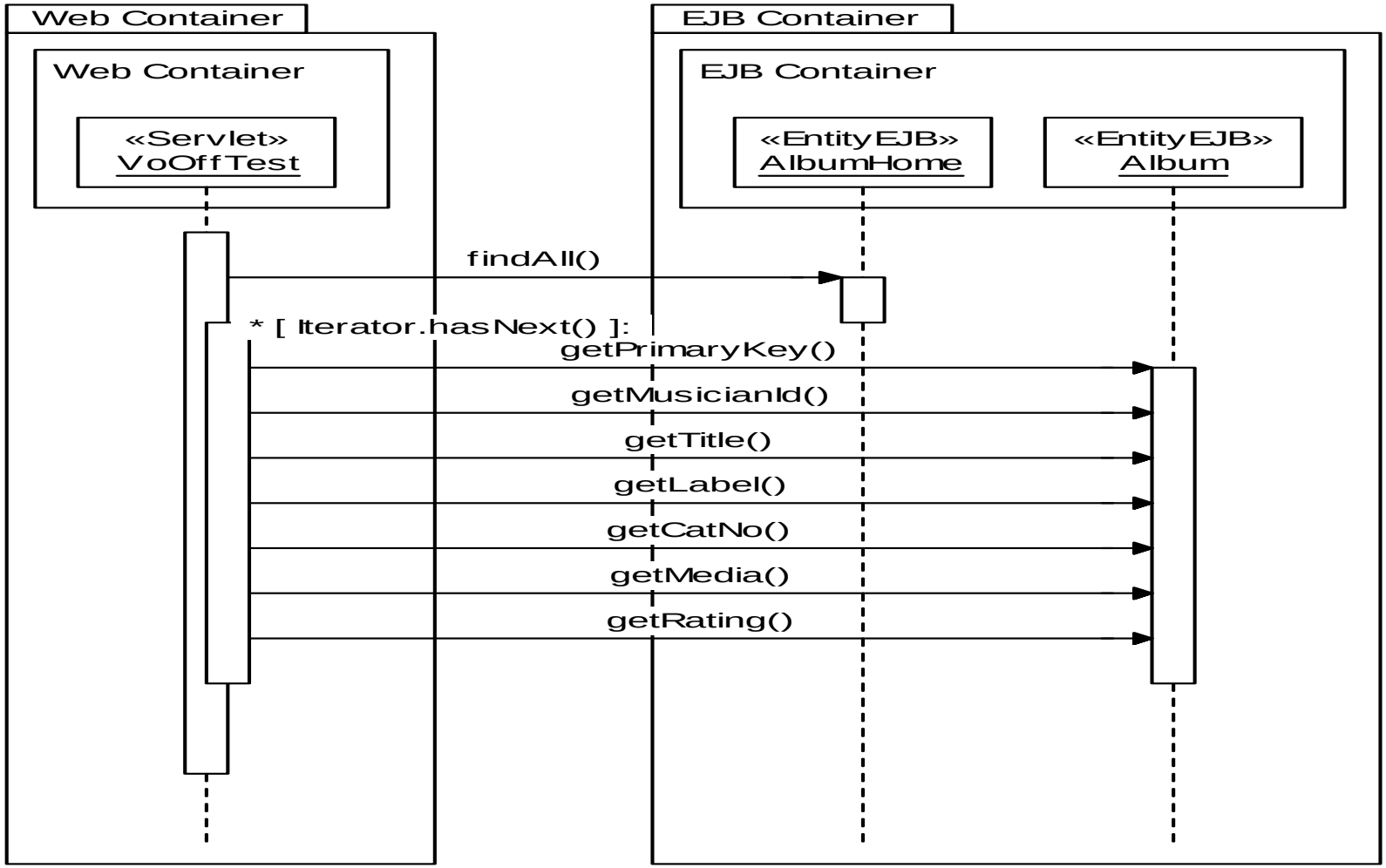
5.6常用模式—Value Object（值对象）



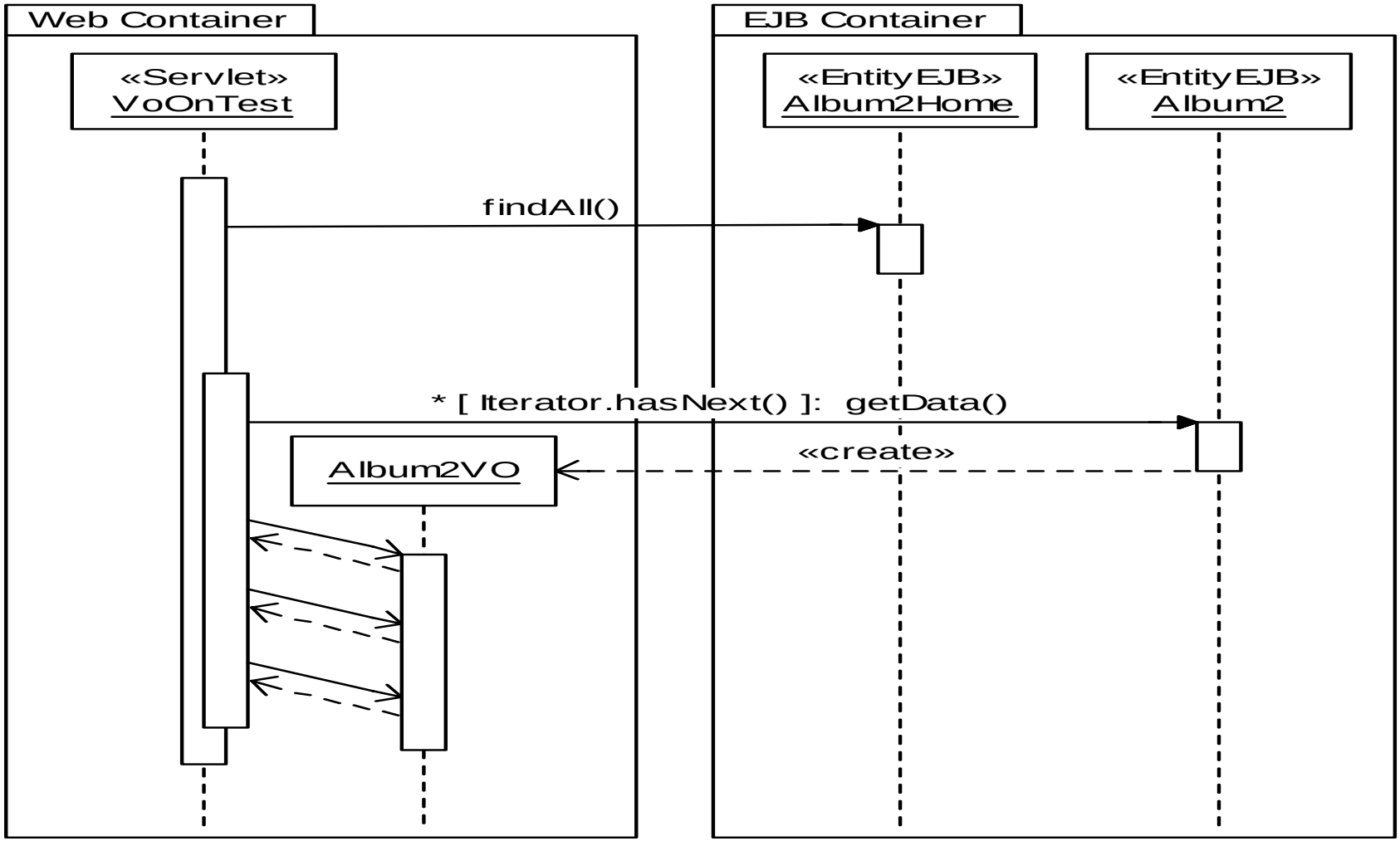
5.6.1常用模式—Value Object（值对象）



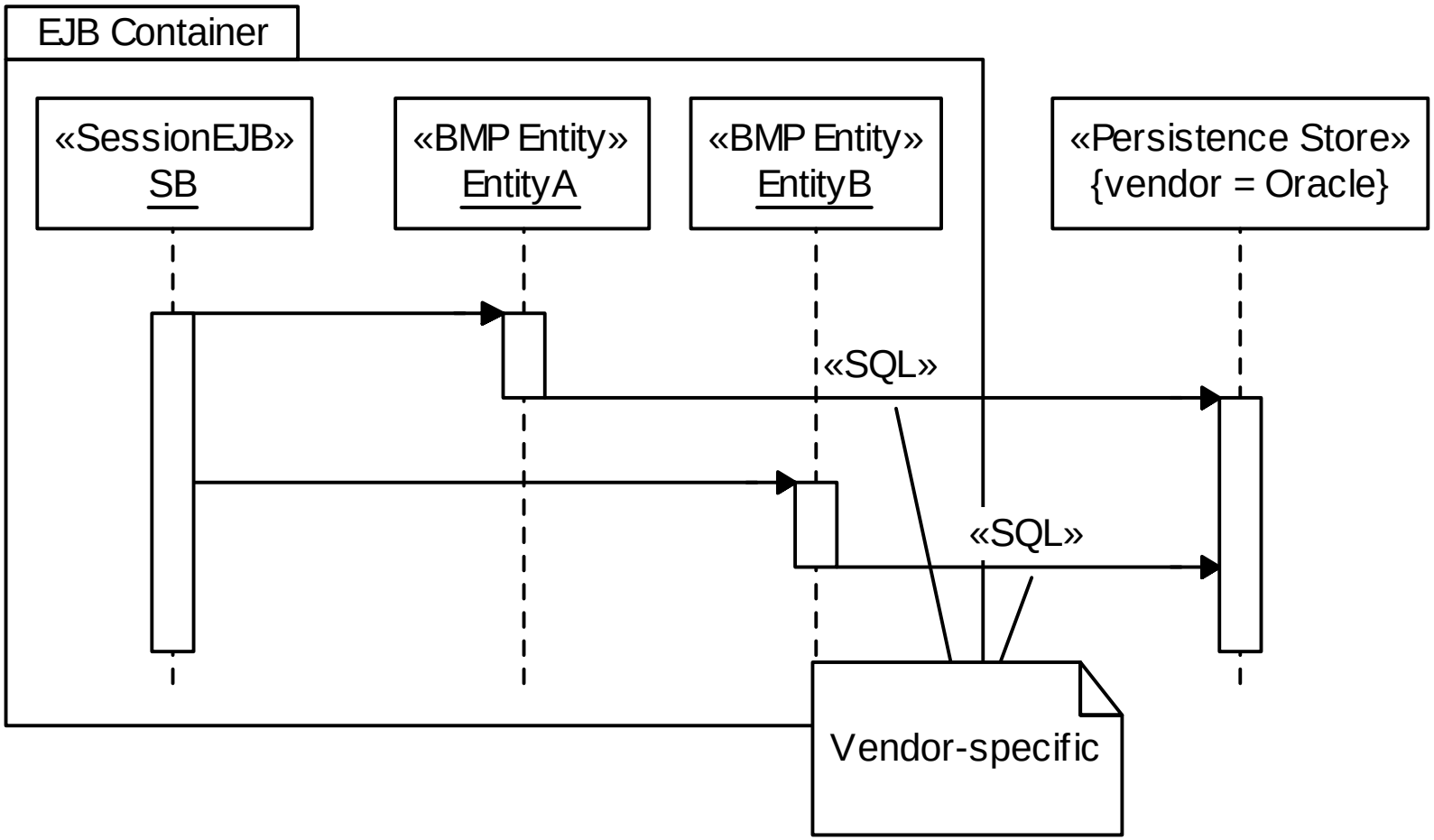
5.6.2常用模式—Value Object（值对象）



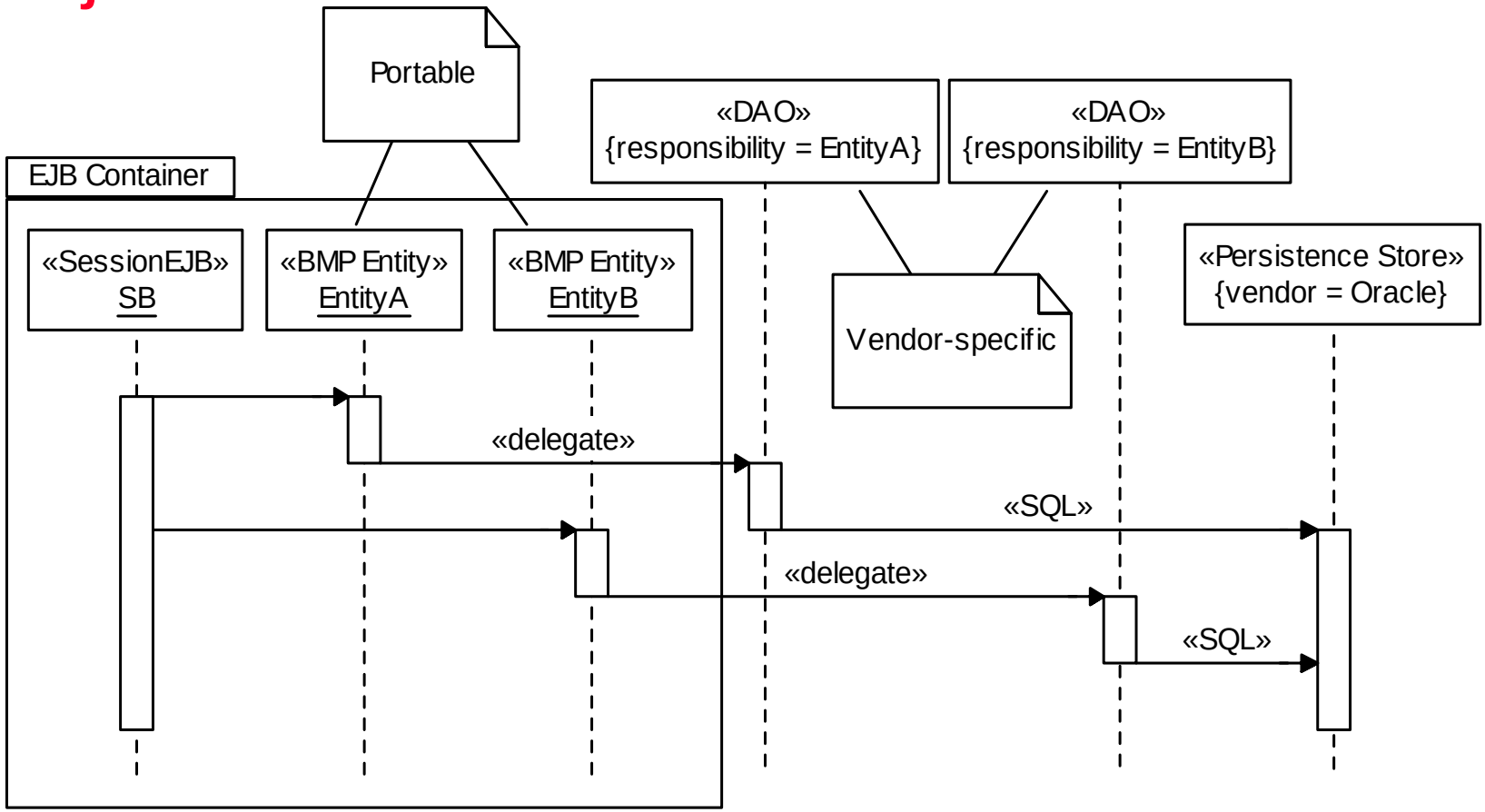
5.6.3常用模式—Value Object（值对象）



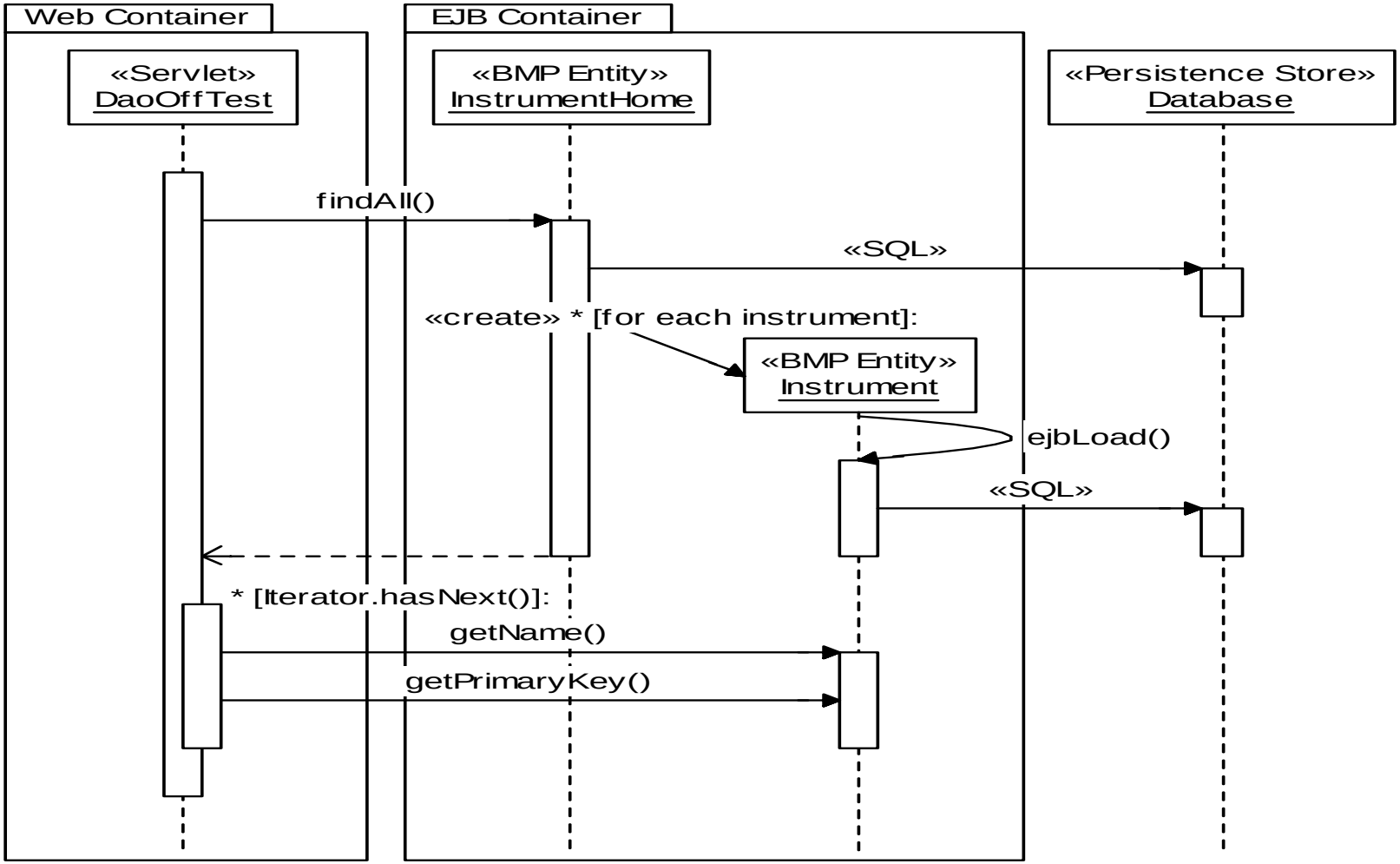
5.7常用模式—DAO（ Data Access Object ）



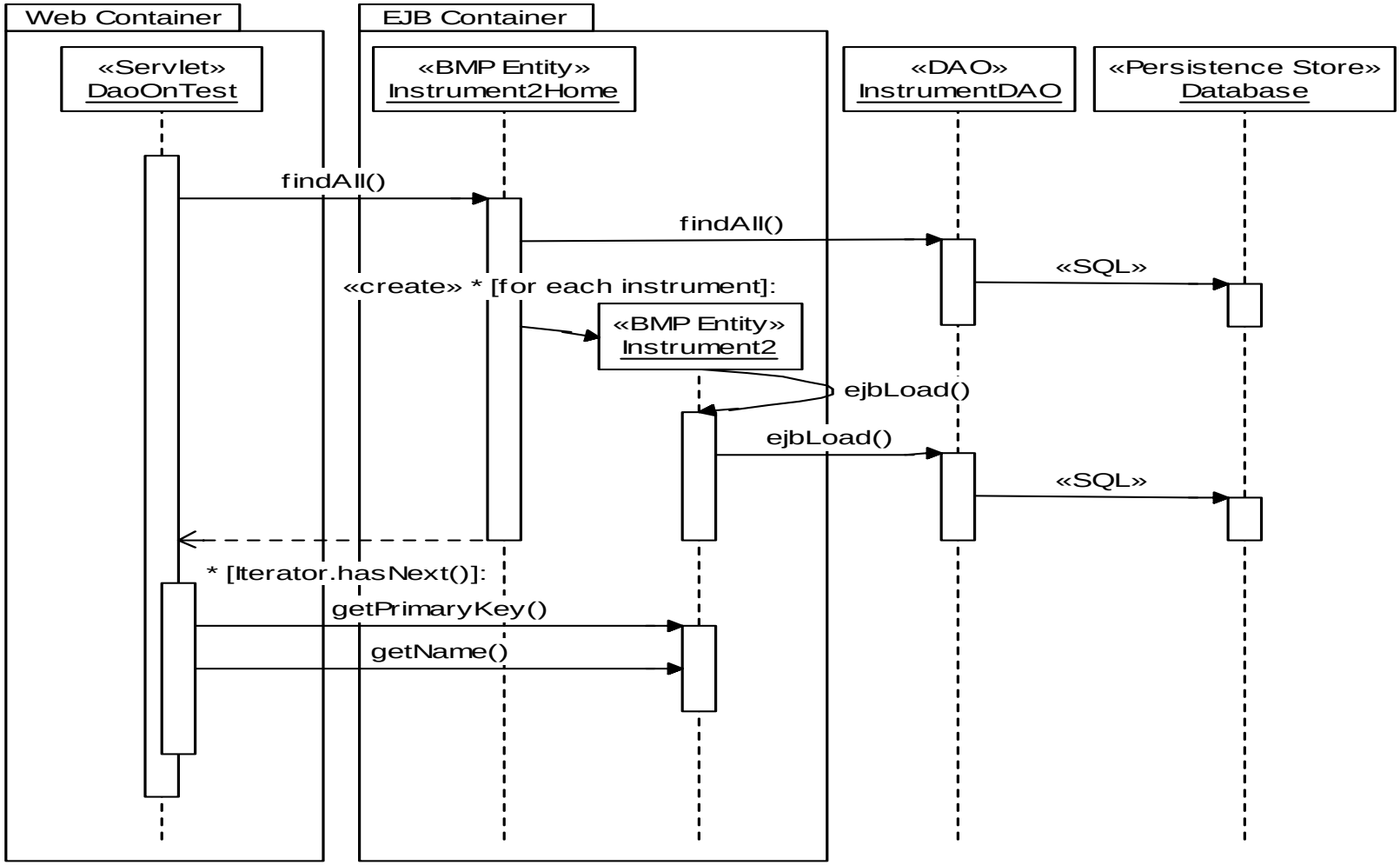
5.7.1常用模式—DAO（Data Access Object）



5.7.2常用模式—DAO（Data Access Object）



5.7.3常用模式—DAO（Data Access Object）



5 .8.3时间消耗对比

单位：ms　执行次数：１００次　每次取一条记录

	CMP	JDBC	JDBC(不带JNDI查找)
第一次执行时间	90	687	390
第二次执行时间	78	188	141
第三次执行时间	60	125	62
最小执行时间	30	78	47
平均执行时间	42	123	71

5 .8.4打开WLS中CMP的SQL监视功能

- 在startWebLogic.cmd或者startWebLogic.sh中加入：
- SET JAVA_OPTIONS=-Dweblogic.ejb20.cmp.rdms.codegen.debug=true –Dweblogic.ejb20.cmp.rdms.codegen.verbose=true

5.9常见J2EE性能问题

- 数据库连接池太小
- JNDI查询调用未被缓存
- SQL语句运行时间太长
- 数据库调用太多
- CMP事务过于精细

5.10实体Bean在WLS上的最佳实践

- 使用只读实体Bean
- 使用以读为主的实体Bean
- 合理使用Session Façade和Value Object模式
- 使用Local JNDI或者是声明的ejb-ref，不要用Remote JNDI来访问实体Bean
- 合理使用并发策略，缓存策略，调整策略

谢谢大家！