Hochschule für Angewandte
Wissenschaften Hamburg
Hamburg University of Applied Sciences

Präsentation im Rahmen des Projekts:
“Allgemeine Informatik: Anwendungen”

Persistenz von Objekten

Sommersemester 2005
HAW-Hamburg

Sven Schliesing

Überblick

- Einleitung: Motivation, Dienste, Schnittstellen, Backend
- Datenbanktypen: relational, objekt-relational, objekt
- Mapping: Hibernate
- Objekt-Datenbanken: db4objects, ObjectDB
- Anbindung an das Projekt: “The big picture”

Einleitung: Motivation

- *Dateisystem?*

  nicht geeignet

- *relationale Datenbank?*

  zu aufwendig

- *Objekt-Datenbank?*

  die ideale Spielwiese für das Projekt “Ferienclub”

Einleitung: Angebotene Dienste

- Abspeicherung von Objekten
- Laden von Objekten
- Rechteverwaltung/Speicherung

Einleitung: Schnittstellen

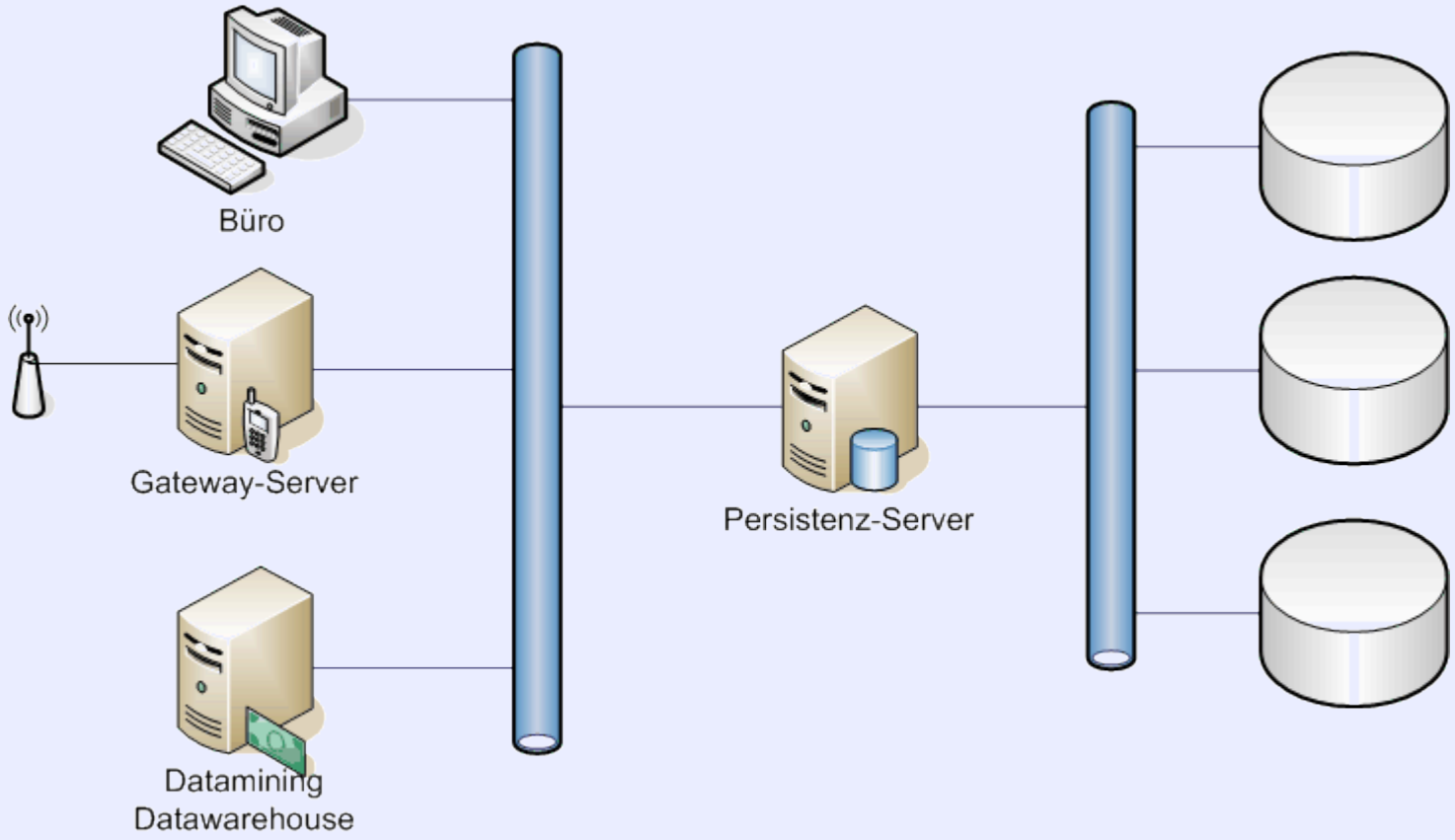
```
module PersistenceModule {
     interface Persistence {
          struct genericObject {
               long owner;
               long time;
               octet[] data;
          };

          typedef octet[160] OpaqueKey;
          exception InvalidKeyException{};

          OpaqueKey saveObject(in genericObject obj);
          genericObject loadObject(in OpaqueKey key, out genericObject obj)
               raises InvalidKeyException;
          genericObject loadAbstract(in OpaqueKey key, out genericObject obj)
               raises InvalidKeyException;
};
```

Einleitung: Schnittstellen

Weitere denkbare Methoden:

- loadObjectByAlias(ein/aus objectAlias : string(idl))
- loadAbstractByAlias(ein/aus objectAlias : string(idl))
- getOwnerByObject(ein/aus objectId : long(idl))
- etc.

Einleitung: Backend

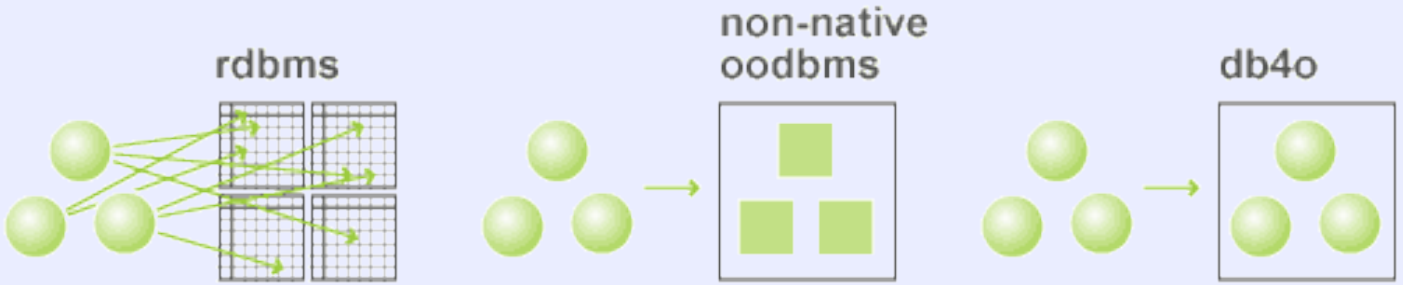


Sven Schliesing, HAW-Hamburg, Sommersemester 2005

Einleitung: Backend

- Datenbank
- relational oder Objekt-Datenbank?
- Filesystem für Grafik-/Binärdaten?

Datenbanktypen



Quelle: http://www.db4objects.com/about/productinformation/

Datenbanktypen: Überblick

- relationale Datenbanken
- objekt-relationale Datenbanken
- Objekt-Datenbanken

Datenbanktypen: relational

- Produkte:
  - Oracle 7.x
  - IBM DB/2
  - MSSQL
  - mySQL
  - etc.

Datenbanktypen: relational

- Impedance mismatch: Problem beim Übergang von Objekten in eine relationale Datenbank
  - Objekt-Ansatz besitzt Software-Architektur als Basis
  - relationaler Gedanke basiert auf Mathematik

  -> Problem beim Übergang (Mapping notwendig)

Datenbanktypen: objekt-relational

- Produkte:
  - Oracle 8.x
  - Universal Database (DB/2 Extenders)
  - PostgreSQL
- Bindeglied zwischen relational und objektbasiert
- realisieren die Anforderungen an Objektdatenbanken durch Erweiterung des relationalen Modells --> Hybrid

Datenbanktypen: Objektdatenbank

- Anforderungen (nach Atkinson):
  - Objekte / komplexe Objekte
  - Klassenhierarchien
  - Sprachvollständigkeit
  - Erweiterbarkeit
  - Fehlersicherheit

"[...] we are taking a Darwinian approach: we hope that, out of the set of experimental prototypes being built, a fit model will emerge."

The Object-Oriented Database System Manifesto, 1995

Datenbanktypen: Resumee

- relational

  pro:

  - umfangreiche Replikation
  - ausgereifte Mapping-Lösungen (z.B. Hibernate)
  - volle SQL-Unterstützung

Datenbanktypen: Resumee

- relational

  contra:

  - kein nativer Übergang in die Persistenz möglich
  - aufwendiges Mapping notwendig

Datenbanktypen: Resumee

- Objektdatenbanken:

  pro:

  - einfache Integration
  - kürzester Weg *Objekt* <-> *Persistenz*
  - erweiterbar durch den Anwender

"ODBMSs provide the more flexible, extensible alternative..."

Object Database vs. Object-Relational Databases, August 1997

Datenbanktypen: Resumee

- Objektdatenbanken:

  contra:

  - weniger verbreitetes Know-How
  - Konzentration auf Bekanntes und Bewährtes

"The other advantage that RDBMSs and the SQL-based ORDBMSs have is the availability of experienced developers and the plethora of SQL-based developer tools, books, and consultants. ODBMSs won't be in a similar position for another five years."

"Despite this clear advantage, some user organizations, and even ISVs, feel safer using a relational database..."

Object Database vs. Object-Relational Databases, August 1997

plethora = die Fülle
ISV = Independent Software Vendor

Mapping

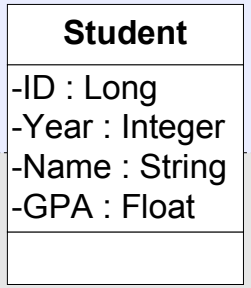
- Abbildung von Objekten/Objektstrukturen auf relationale Datenmodelle
- Performance-Einbußen:
  - Komplette Objekte werden geladen
- Performance-Gewinn:
  - effizientes Caching

"The term object/relational mapping (ORM) refers to the technique of mapping a data representation from an object model to a relational data model with a SQL-based schema."

hibernate.org - Introduction

Mapping: Konfiguration

- Beispielkonfiguration




```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
 "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping package="studentControl">
 <class name="Student" table="student">
  <id name="id" column="ID" type="long"><generator class="increment"/></id>
  <property name="Year" column="Year" type="int" not-null="true"/>
  <property name="Name" column="Name" type="string" length="50" not-null="true"/>
  <property name="GPA" column="GPA" type="float" not-null="true"/>
 </class>
</hibernate-mapping>
```

Mapping: lazy-load / proxy

- lazy-load

```
class A {
     private Set items;
     public Set getItems() {
          return items;
     }
     ...
}
```

- proxy

```
class A {
     private B classB;
     public Set getClassB() {
          return classB;
     }
     ...
}
```

Objektdatenbanken

- db4objects:
  - db4objects Inc., Kalifornien (http://www.db4objects.com)
- JDO:
  - Implementation: ObjectDB (http://www.objectdb.com)

Objektdatenbanken: db4objects

> ***"What is needed?" in a single, short sentence: "A database library that is small, fast, powerful, and easy to use."***
>
> Rick Grehan, The Database Behind the Brains

Klartext:

- Resourcensparend
- hohe Performance
- zuverlässig
- einfache Implementation
- portierbar
- Lizenz: GPL

Objektdatenbanken: db4objects

Genauer:

- Resourcensparend:
  - Bibliothek lediglich 250kB gross
- hohe Performance
  - vergleichbare (oft sogar höhere) Performance mit anderen (auch nicht-Objekt-)Datenbanken

Objektdatenbanken: db4objects

barcelona benchmarks	read	write	query	dele
db4o	1.0	1.0	1.0	1.0
Hibernate / MySQL	19.0	[illegible]	6.0	9.0
Hibernate / HSQLDB	14.4	15.[illegible]	388.6	6.3
JDBC / MySQL	9.[illegible]	18.7	1.0	4.0
JDBC / HSQLDB	0.4	3.5	453.6	0.6
[illegible]DBC / Derby	6,367.4	25.6	2,403.2	4.6
[illegible]BC / "redmondsql"	35.9	21.8	8.5	2.8

fastest

slowest

Fragwürdig!

Quelle: http://www.db4objects.com/about/productinformation/benchmarks/

Objektdatenbanken: db4objects

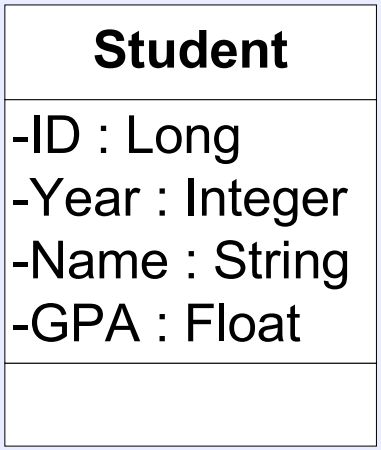
Genauer:

- zuverlässig:
  - volle ACID-Unterstützung
- einfache Implementierung
- portierbar
  - lauffähig auf allen Umgebungen auf denen JAVA/.NET läuft

Objektdatenbanken: db4objects

Implementation:

Klasse “Student”:



Objektdatenbanken: db4objects

Implementation:

- relational:

```
ResultSet results = statement.executeQuery("SELECT ID, Year, Name, GPA FROM
Student WHERE ID = 1");
if (results.next()) {
     student = new Student();
     student.ID = results.getInt("ID");
     student.Year = results.getInt("Year");
     student.name = results.getString("Name");
     student.GPA = results.getInt("GPA");
     // Do something with the student}
```

- relational mit Mapping:

```
Configuration cfg = new Configuration();
cfg.addClass(Student.class);
SessionFactory factory = cfg.buildSessionFactory();
Session session = factory.openSession();
Student student = (Student) session.load(Student.class, 1);
```

Objektdatenbanken: db4objects

Implementation:

Objektdatenbank:

```
studentTemplate = new Student(1, 0, null, 0.0);

ObjectSet result = studentDB.get(studentTemplate);

if (result.hasNext()) {
     Student student = (Student)result.next();
     // Do something with the student
}
```

```
Query query = studentDB.query();
query.constrain(Student.class);
query.descend("Name")
        .constrain("Maik Hansen");
ObjectSet result = query.execute();
...
```

Objektdatenbanken: db4objects

- Indices:

```
Configuration config = Db4o.configure();
ObjectClass clazz = config.objectClass(Student.class);
ObjectField field = clazz.objectField("Name");
field.indexed(true);
```

- Update-Tiefe

```
Configuration config = Db4o.configure();
ObjectClass clazz = config.objectClass(Student.class);
clazz.updateDepth(3);
config.updateDepth(3);
```

Objektdatenbanken: db4objects

- Lazy-load (activationDepth, std.: 5)

```
Configuration config = Db4o.configure();
config.activationDepth(3);
```

- Nachträgliches Laden

```
ObjectContainer container=Db4o.openFile(dbFilename);
container.activate(member5, 1);
```

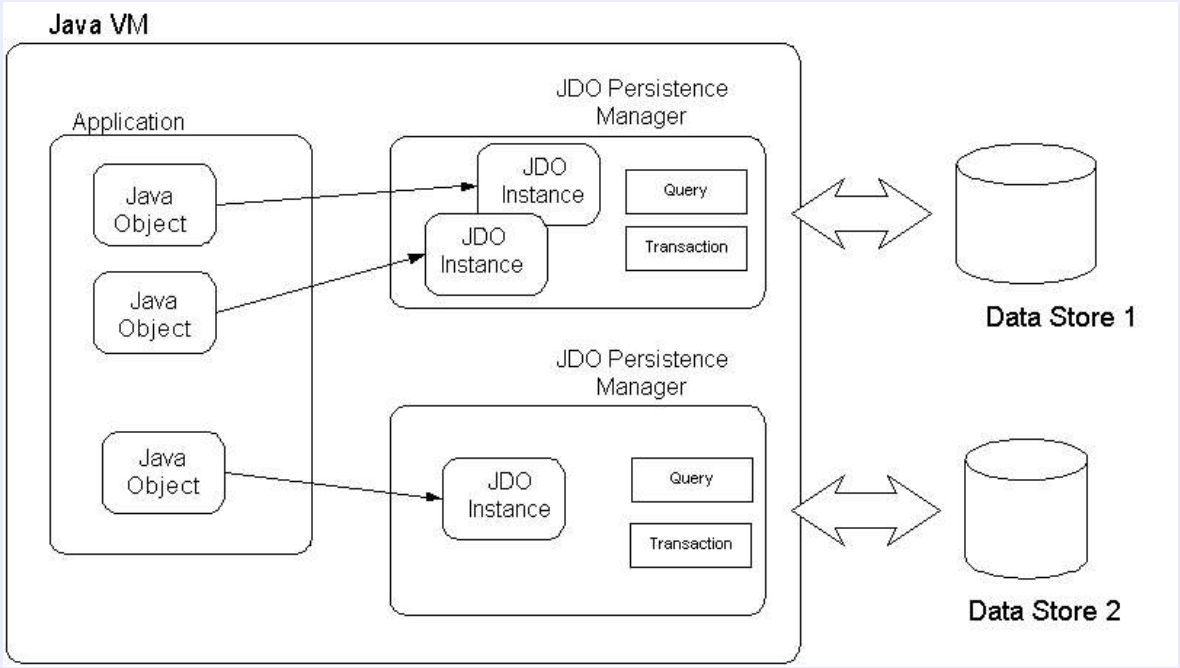
Objektdatenbanken: db4objects

No Impedance Mismatch:

direkter Übergang von (OO-)Programmiersprache in Persistenz

Sven Schliesing, HAW-Hamburg, Sommersemester 2005

Objektdatenbanken: JDO



Quelle: http://www.javaworld.com/javaworld/jw-04-2002/images/jw-0412-jdo1.gif

"In addition, the same object based persistence API will be usable both with J2SE and J2EE, allowing many more developers to take advantage of a standard object persistence API."

EJB/JDO Persistence FAQ

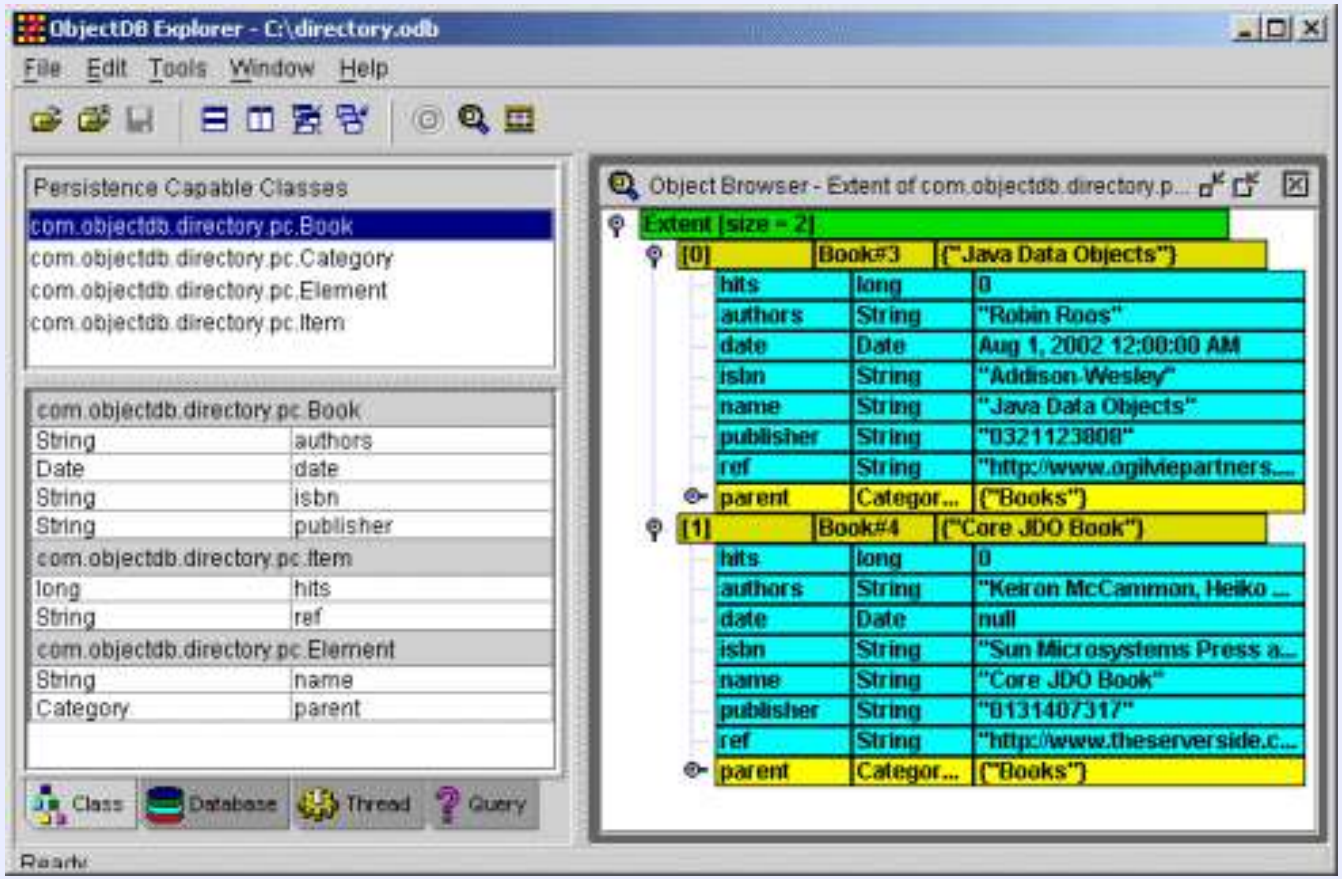
Objektdatenbanken: JDO

- ObjectDB als JDO-konforme Implementation
- Embedded / Client-Server – Mode (ebenso db4objects)

```
Student student = new Student(1, 2005, "Maik Hansen", 2.0);

pm.currentTransaction().begin();
pm.makePersistent(student);
pm.currentTransaction().commit();
```

```
Query query = pm.newQuery(Student.class, "this.ID == 1");
Collection result = (Collection)query.execute();
try {
  Iterator itr = result.iterator();
  while (itr.hasNext())
    System.out.println(itr.next());
} finally {
  query.close(result);
}
```

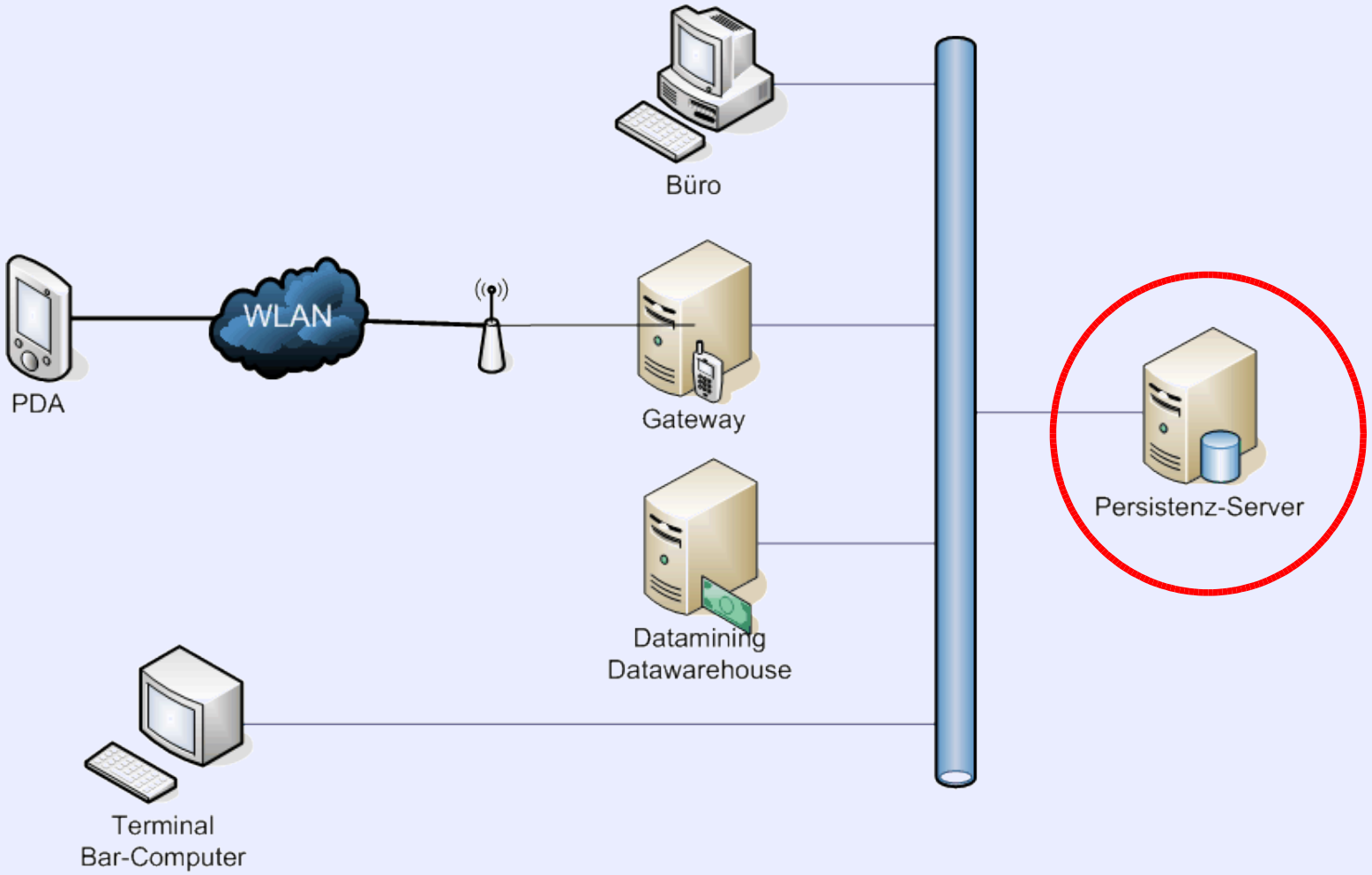
Objektdatenbanken: JDO



Quelle: http://www.objectdb.com/database/jdo/java-explorer/

Einleitung – DBTypen – Mapping – ObjektDBs – The big picture

The big picture



The big picture

- Persistenz-Server stellt sichere Datenhaltung zur Verfügung
- Anwendungen (Webservices, Datamining/Datawarehouse, SOA) greifen über ausreichend definierte Schnittstellen auf die Daten zu
- Authentifizierung wird von den Applikationen übernommen
- Verbindungsabrisse werden von der mobilen Persistenzschicht gehandlet

Offene Fragen

- Welche Schnittstellen-Methoden werden noch benötigt?
- Performance bei grossen Datenmengen (Binärdaten)?
- Objektdatenbanken ausgereift genug?

Quellen

•Object Database vs. Object-Relational Databases, August 1997: http://www.ca.com/products/jasmine/analyst/idc/14821E.htm

•The Object-Oriented Database System Manifesto, 1995:
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/clamen/OODBMS/Manifesto/htManifesto/Manifesto.html

•The Database Behind the Brains, unbekanntes Datum
http://www.db4objects.com/about/productinformation/whitepapers/db4o%20Whitepaper%20-%20The%20Database%20Behind%20tl

•Relationale Datenbank: http://de.wikipedia.org/wiki/Relationale_Datenbank

•The Object-Relational Impedance Mismatch: http://www.agiledata.org/essays/impedanceMismatch.html

•EJB/JDO Persistence FAQ: http://java.sun.com/j2ee/persistence/faq.html

Vielen Dank
für eure
Aufmerksamkeit!