

OODBMS – Revival oder Renaissance?

Einführung und Werkstattbericht über db40

Vortrag anlässlich der JUGS General Assembly
7. Dezember 2007, Technopark Zürich

Prof. Stefan Keller

Begriffe

- **Objekt-orientierte Datenbanksysteme (OODBMS), kurz: Objekt-Datenbanken**
- ***Revival* = Wiederaufflammen einer Modewelle der Vergangenheit (v.a. Musik)**
- ***Renaissance* = Wiedergeburt**

- **Déja-vu?**

- „Objektorientierte Datenbanken - Back to the future. Eine (zukunfts)kritische Bestandsaufnahme am Beispiel von db4o“. Stefan Edlich, Techn. FH Berlin, JUGS 5.12.06(!)

- **Firma**

- 2002 als Projekt, 2004 als Fa. db4objects, Inc.
- Gründer Carl Roenberger mit Investoren
- Java & .NET
- 30,000 registered developers
- Bosch, Boeing, Ricoh, Seagate, ETH/Uni/FH.

-
- **Schmal**
 - **Schnell (vgl. polepos.sf.net: ~JDBC+HSQL)**
 - **Simpel zu programmieren: vgl. nachfolgend**
 - **Transaktionell: ACID (Read Committed)**
 - **Lazy Loading**
 - **Native Queries**
 - **Zudem: Replikation**

```
// 1: Initialize an object container
ObjectContainer db= null;
try {
    // 2: Open db4o database (embedded mode)
    db= Db4o.openFile("addressbook.yap");
    // 3: Create and store some 'Kellers'
    db.set(new Person("Brian", "Keller", 1960));
    db.set(new Person("Clara", "Keller"));
    db.set(new Person("Test"));
    db.commit();
    // 4: Find and read all Kellers with Query by Example
    ObjectSet result= db.get(new Person(null, "Keller", 0));
    while (result.hasNext())
        System.out.println((Person) result.next());
    // 5: Update Clara's age
    result= db.get(new Person("Clara", "Keller"));
    Person found= (Person) result.next();
    found.setYearOfBirth(1970);
    db.set(found);
    // 6: Delete Test data
    result= db.get(new Person("Test"));
    while (result.hasNext())
        db.delete(result.next());
    // 7: Commit all
    db.commit();
} finally {
    if (db != null)
        db.close();
}
```

-
- **Speichert einfach so jedes Objekt**
 - Kein Pre-Compiler, keine Inheritance, kein Interface (Java Reflection API)
 - Beliebige Klassen mit beliebig vielen und tiefen Subobjekte und Collections

 - **anderes Denken**
 - Brauche ich Joins?
 - Muss ich mich um IDs kümmern? (Ref. Int.)

RDBMS

- Hoher Strukturierungsgr. der Fa. (Datenverantw.)
- Neutrale Form, verschiedene Sichten
- OLAP / Cubes
- Clusterarchitekturen
- mehr DB Services / Tools
- ggf. Administration

Typische Anwendungen:

- Datawarehouses (Rechenzentrum)

OODBMS

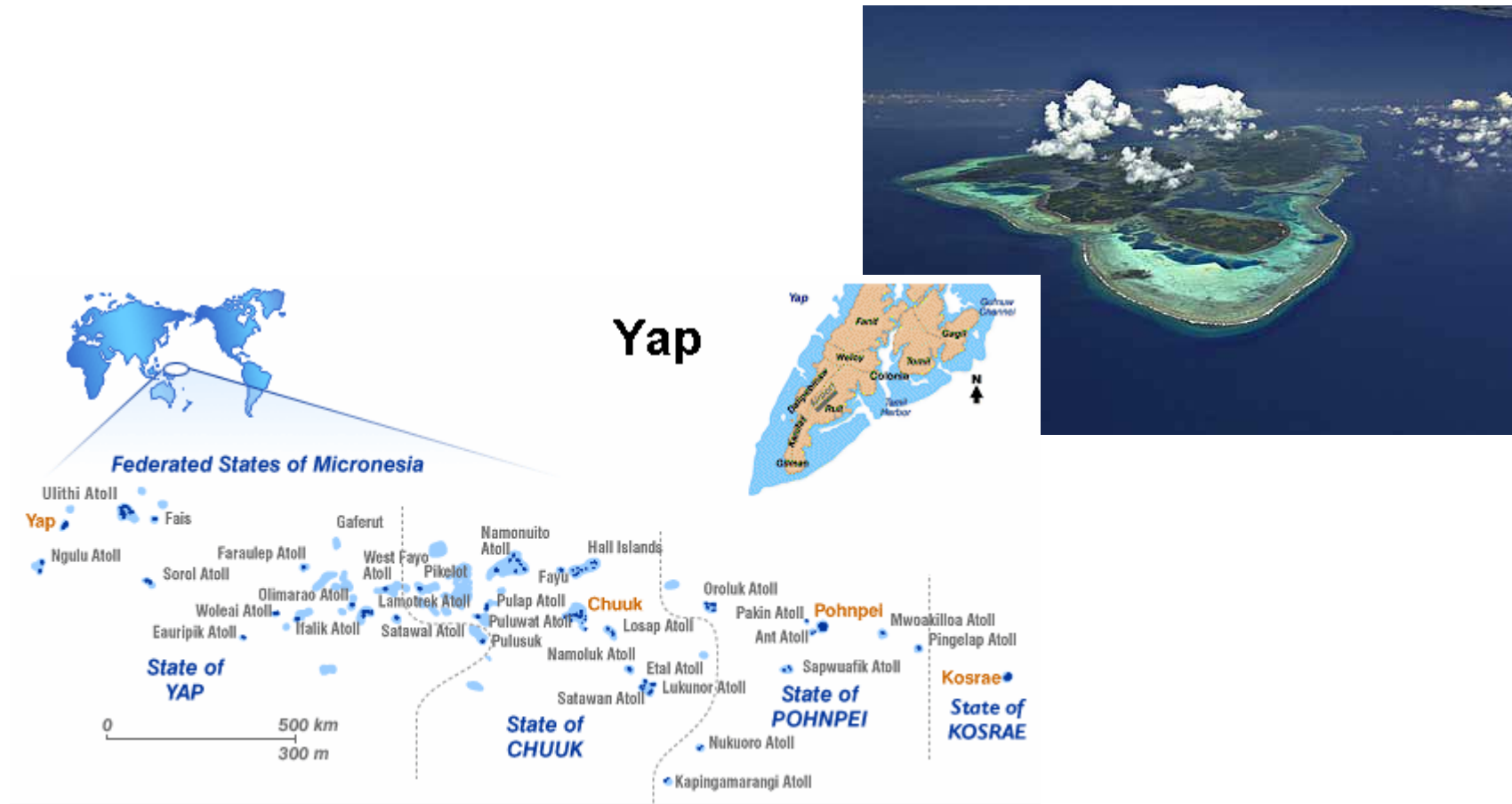
- Kein Mappingaufwand
- Mehr Performance (v.a. ORM)
- tiefe Objektgraphen
- Schema Changes
- keine Administration

Typische Anwendungen:

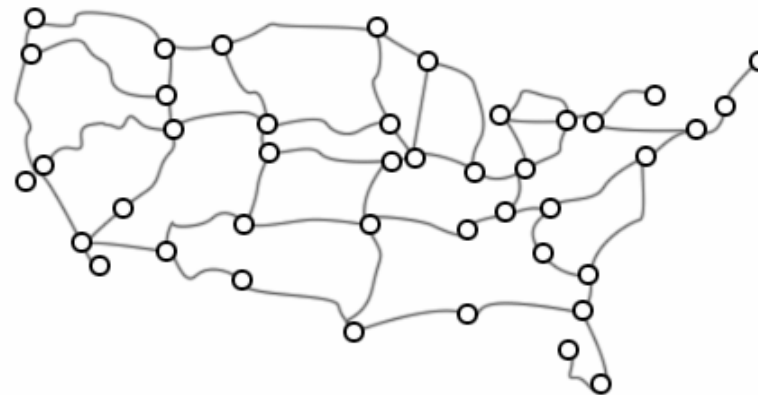
- Embedded Systems

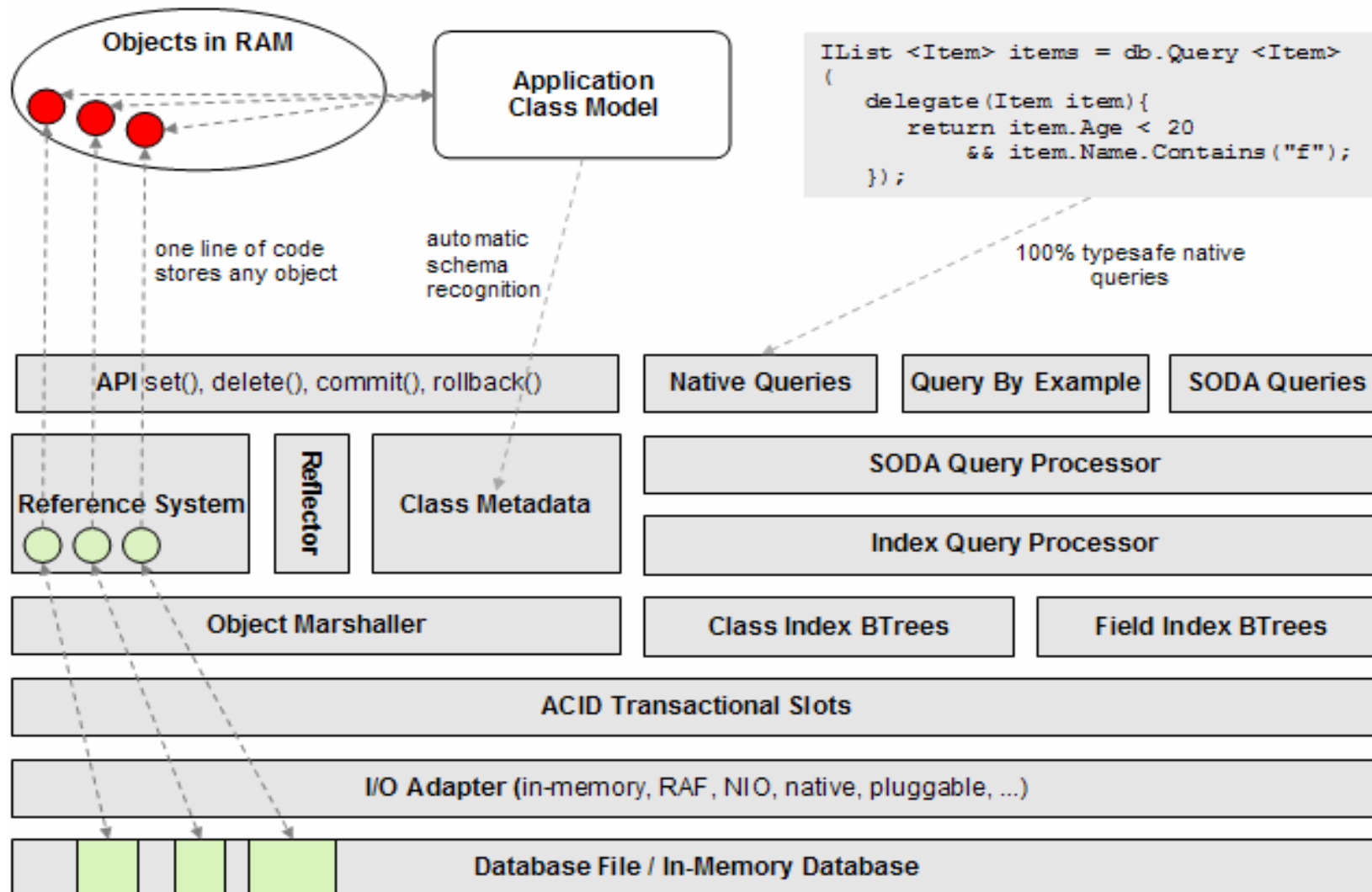
-
- **Erfahrungen**
 - **Erweiterung mit Geometrie-Basistypen**
 - **Mapping einer SQL-orientierten Anfragesprache auf db4o**
 - HSR-Projekt JPOX4db4o: Mapping von JPOX' JDOQL queries auf db4o's SODA.
 - Mapt JDOQL => SODA

- **Wirklich einfach zu programmieren**
 - Objekte in Datei gespeichert; beliebig viele Dateien gleichzeitig
`db= Db4o.openFile("myobjects.yap");`
 - Zwei Modi: Embedded und Server-Mode
- **Nützliche Funktionen**
 - Constraints als external Callbacks
- **Hindernisse**
 - Häufige Releases, wandelndes API
 - Release-Management? Non-Test-Driven?



- **HSR-Projekt db4o2D**
- **Geometrie in 2D (2.5D) plus 'Metadaten'**
- **Zusätzliche Basistypen**
 - Point, LineString Polygon
 - sowie MultiPoint, MultiLineString and MultiPolygon





- **Aktuell drei Arten:**
 1. Query by Example (QbE)
 2. Criteria Queries (SODA)
 3. Native Queries (NQ)
- **Beispiel JDOQL (wird in realisiert)**

```
Query query =
    persistenceManager.newQuery(Student.class,
        "age < 20");
Collection students =
    (Collection)query.execute();
```
- **Beispiel db4o SODA**

```
Query query = database.query();
query.constrain(Student.class);
query.descend("age").constrain(20).smaller();
List students = query.execute();
```

Generische anonyme Klasse (Java):

```
// ...  
List <Student> students =  
    database.query <Student> (  
        new Predicate <Student> () {  
            public boolean match(Student student) {  
                return student.getAge() < 20  
                    && student.getName().contains("f");  
            }  
        }  
    );  
// ...
```

OODBMS:

- **Welcher Anfragesprachen-Standard**
- **Lasttests?**

db4o:

- **Datenstrukturen:**
 - Scalability: Datei kann max. 2 GB adressieren
 - Memory-based Index, noch nicht disk-based
- **Funktionen:**
 - Server-Management, z.B. für Defragmentieren?
 - Scalability: Thread-Management?
- **Anfragesprache:**

- **db4o – ein OODBMS mit erstaunlichen Eigenschaften und mit Grenzen**
- **Liegt die Zukunft der OODBMS in den Anfragesprachen?**
 - Einfachheit? Mächtigkeit? Typsicherheit? Erlernbarkeit
 - Auch Akzeptanz?
- **Theoretische Foundation für ODBMS?**
 - MS LINQ: Language INtegrated Query.
 - Eine von MS entwickelte Methode, um SQL-, XLink- und XQuery-Anfragen in eine Programmiersprache als Code statt als String einzubinden
 - Vorteil: Code wird durch Compiler geprüft. Die Syntax an SQL angelehnt
 - Whitepaper on Next-Generation Object Database Standardization, OGM, Prof. Subieta, Polish-Japanese Institute of IT, Warschau: siehe ...

- **OMG's Object Database Technology Working Group (ODBTWG)**
 - Dezember 2005
 - Übernimmt Rechte des ODMG 3.0-Standards
- **ODBTWG diskutierte 3 Standardisierungs-Ansätze:**
 - 1. Develop a theoretical underpinning for object databases and base all standards on it (object database equivalent of the relational calculus)
 - 2. Quickly produce a standard on a vendor-neutral technology...
 - 3. Quickly develop a standard for C#/.NET programming language bindings...
- **Es wurde Ansatz 1 gewählt**

- **Revival oder Renaissance?**

- Beides

- OODBMS

- im Wandel der IT

- Wandel der Vernunft

Weblinks:

- <http://developer.db4o.com/>
- <http://wiki.hsr.ch/StefanKeller/wiki.cgi?Db4o>
- odbms.org

Kontakt:

Prof. Stefan Keller
IFS Institut für Software
HSR Hochschule für Technik Rapperswil
Oberseestrasse 10
8640 Rapperswil (Schweiz)

www.ifs.hsr.ch
[sfkeller \(at\) hsr.ch](mailto:sfkeller@hsr.ch)
T 055 222 47 46