Objektorientierte Datenbanken

Vorlesung 12: Zusammenfassung und Ausblick

Sebastian Iwanowski
FH Wedel

Inhalt 12. Vorlesung OODB

1) Zusammenfassung: Zentrale Aussagen der Vorlesung

2) Vergleich Hibernate - JDO

3) Ausblick: Was kommt in der Zukunft?

4) Überblick über die Vorlesung OODB:
   Was ist klausurrelevant ?

Zentrale Aussagen der Vorlesung

Problemstellung

- **Objektorientierte Modellierung an Datenbankmodellierung anbinden**
- **Impedance Mismatch: passt nicht zusammen**

Lösungsstrategien

- **Datenbankmodellierung objektorientiert machen (z.B. Versant)**

  ODMG → JDO 1.1 → JDO 2.0

- **Objektrelationale Übersetzung vornehmen (ORM)**

  JDO 2.0 oder Hibernate 3.x

  Neuer Standard: JPA (wird ab Hibernate 3.2 erfüllt)

Zentrale Aussagen der Vorlesung

Merkmale von modernen ORM-Standards und -Werkzeugen:

- **Persistenzmanagement:**
  Objektmodellierung und DB-Abhängigkeiten (Fetching Strategies, Transitive Persistenz) werden in XML-Metadaten festgelegt
- **Transaktionsmanagement:**
  Optimistische und pessimistische Transaktionen
- **Anfragesprache:**
  SQL-ähnlich aber mit Objekten statt Tabellen oder Filter
- **Datenidentitätskonzepte:**
  Wann gelten zwei Objekte als gleich, wie findet man ein bestimmtes Objekt in der Datenbank?
- **Detach / Attach:**
  Möglichkeit der externen Manipulation von persistenten Objekten ohne Verbindung zur Datenbank
- **Inheritance Mapping Strategies:**
  Abbildung der Vererbungshierarchie in relationalen Tabellen

Unterschiede Hibernate - JDO

Hibernate	JDO
Produkt (open-source)	Spezifikation
nur für SQL-Datenbanken	für beliebige Datenbanken
selten Enhancement (und wenn, dann automatisch)	Enhancement obligatorisch
Criteriafilter unter Kontrolle des Java-Compilers	Anfragen nur als Strings, benötigen Spezialcompiler
Dynamisches Fetching	Fetching nur mit in Metadaten spezifizierten Gruppen
„Big Problem“: Unterscheidung zwischen neu anzulegenden und zu verändernden Objekten	Management über Lebenszykluszustände

Unterschiede Hibernate - JDO

Anmerkungen von Christian Romberg, Mitarbeiter der Versant AG, Projekterfahrung mit Hibernate und JDO:

- **Produkt:**
  Hibernate ist ausschließlich open-source: Es gibt kein kommerzielles Produkt dafür.
  Für JDO gibt es beide Produktarten (open-source: JPOX, kommerziell: z.B. Versant).

- **Metadaten:**
  In JDO sind per Default alle nicht erwähnten Felder persistent,
  in Hibernate transient (➔führt bei Unvorsichtigkeit zu Datenverlust)

- **Persistenzfähige Klassen:**
  In JDO darf fast alles persistenzfähig sein (auch nicht serialisierbare Klassen).
  In Hibernate müssen die Klassen serialisierbar sein, Sonderfall: finale Klassen.

- **Enhancen:**
  hat den Vorteil, dass Feldzugriffe direkt kontrolliert werden können
  (➔ ermöglicht leicht festzustellen, was konkret aktualisiert werden muss)

Unterschiede Hibernate - JDO

Anmerkungen von Christian Romberg, Mitarbeiter der Versant AG, Projekterfahrung mit Hibernate und JDO:

- **Lebenszykluszustände:**
  gibt es auch in Hibernate, aber im Gegensatz zu JDO nicht mit exakt definiertem Verhalten (➔ führt manchmal zu unvorhersagbarem Verhalten)

- **Detach / Attach:**
  bietet in beidem gleiche Möglichkeiten. In JDO werden grundsätzlich Kopien detached, Übertragung der Felder zwischen Original und Kopie erfolgt automatisch.
  reassociate a subgraph: geht auch in JDO 2.0
  implizites detachen in JDO: detach-all-on-commit

- **Transaktionen:**
  Optimistische Transaktionen in JDO wohl definiert (mit Fehlermeldungen), in Hibernate kein garantiertes Verhalten (Werkzeug, kein Standard)

- **Cache:**
  First level ausschließlich für Identitätsgarantien, nur der second level ist richtiger Chache (gilt für Hibernate und JDO).

Unterschiede Hibernate - JDO

Anmerkungen von Christian Romberg, Mitarbeiter der Versant AG, Projekterfahrung mit Hibernate und JDO:

Fazit (Romberg):

- **Hibernate wirkt einfach auf Gelegenheitsbenutzer, zeigt aber bei tieferer Beschäftigung seine Unzulänglichkeiten**
- **JDO ist professioneller und systematischer**

Ausblick

EJB 3: Enterprise Java Beans mit POJOs (JPA)

- wird in Zukunft von allen ORM-Anbietern unterstützt
- nach Aussage Chris Richardson schlechter als Hibernate / JDO, z. B.:
  - fetch joins und detachments umständlicher
  - keine Collections primitiver Datentypen
- Vorhersage Richardson: Nutzer finden über kommerzielle EJB3-Anbieter ihren Weg auch zu JDO

Weitere Entwicklungshilfen für Java-Programmierer:

- Spring: Aspektorientierte Programmierung
- iBATIS: Direkte Einbindung von Java-Klassen in SQL

OODB: Überblick und Klausurrelevanz

Vorlesung 1: Grundlegende Prinzipien relationaler und objektorientierter Datenmodellierung

Gemeinsamkeiten und Unterschiede, Konversionen von Beispielen zwischen den Modellierungswelten, Impedance Mismatch

Vorlesung 2: ODMG

Objektmodell, verschiedene Persistenzkonzepte (Abhängigkeit von der jeweiligen Programmiersprache), Transaktionskonzept allgemein,
Integritätsbedingungen: Charakterisierungen, Realisierung durch OOP im Allgemeinen und ODMG im Besonderen

Vorlesung 3: JDO: Grundlagen

Zielsetzung, genereller Aufbau eines JDO-Programms

Vorlesung 4: JDO: Persistenz- und Transaktionskonzepte

Persistenzkonzept: Attributmanagement via XML-Metadaten, transiente / transaktionale und persistente Attribute, First-Class- und Second-Class-Objekte, Transaktionseigenschaften, Isolationsniveaux, JDO-Transaktionsstrategien (pessimistisch und optimistisch)

OODB: Überblick und Klausurrelevanz

Vorlesung 5 / 6: JDOQL

Vorteile einer Anfragesprache allgemein gegenüber anderen Extraktionsmöglichkeiten eines Objekts (benennen!),
Filter-Queries und Single-String-Queries: Gemeinsamkeiten und Unterschiede, Parameter- und Variablendeklarationen, Anfragen mit collection-wertigen Attributen, Selektionen und Projektionen, Aggregatfunktionen, ~~Vergleich mit SQL-ähnlichen Fragen~~,
Lösen von Anwendungsaufgaben,
~~Unterschiede zwischen JDO 1.1 und JDO 2.0, Funktionalität von Versant~~

Vorlesung 7: JDO: Datenidentitätskonzepte

ID-Klassen, Datastore- und Application-Identity,
verschiedene Zugriffsmöglichkeiten auf ein Datenbankobjekt

Vorlesung 8: JDO: Lebenszykluszustände

Obligatorische Lebenszykluszustände: Wirkung und Zweck (besonders **`hollow`**), Grund für detach / attach, Funktionsweise in JDO 2.0, Zustandsabfrage

Optionale Lebenszykluszustände: transaktionale Objekte, nichttransaktionaler Datenzugriff

OODB: Überblick und Klausurrelevanz

Vorlesung 9: JDBC und EJB

ODBC und JDBC: Prinzipien und Unterschiede, verschiedene Datenbankanbindungsmöglichkeiten, minimal notwendige Aktionen zur Anbindung, ~~Java API für JDBC~~
~~Überblick über Enterprise Java Beans und seine Einbindung in J2EE~~

Vorlesung 9-12: Hibernate

Hibernate als Beispiel für ein O/R-Mapping-Tool:
Transparente Persistenz, Transitive Persistenz, Grundprinzip des Detached Object Support, Inheritence mapping strategies, Fetching strategies, Grundprinzip des Caching, ~~Automatic dirty object checking~~.
In allem: Gemeinsamkeit und Unterschiede zu JDO.
HQL, Criteria und JDOQL: Prinzipien und Unterschiede, kleine Anwendungsbeispiele.

Generelle Richtlinie für die Klausur:

Gefragt sind eher Konzepte als Code. Dennoch wird auch nach Codebeispielen gefragt, da Konzepte anders nicht zu verdeutlichen sind. Hier geht es vor allem darum, in welchen Situationen welche Konstruktionen gebildet werden müssen. Typische Anwendungsbeispiele hierfür sind Definitionen in den Metadaten sowie Objektzugriffe, insbesondere die unterschiedlichen Zugriffsmöglichkeiten bei Queries. Wichtig ist die Bildung der richtigen Konstruktionen. Kleinere syntaktische Ungenauigkeiten in der Antwort spielen dagegen keine Rolle für die Bewertung.

Das war die Vorlesung OODB