Objektorientierte Datenbanken

Vorlesung 6 (JDOQL, Übung + Details)

Sebastian Iwanowski
FH Wedel

Bsp.: Sortieren der Antwort

Single-String-Query:

- Finde alle Studenten, die von Iwanowski geprüft wurden:

select s **from** Student

where s.wurdeGeprüft.contains (p) && p.Prüfer.Name == „Iwanowski“

order by s.Semesterzahl **DESC**, s.Name **ASC**

Filter-Query ?

Achtung:
Diese Lösung setzt voraus,
dass wurdeGeprüft ein mehrwertiges Attribut ist,
d.h. jeder Student mehrere Prüfungen macht.

Vergleiche mit der unterschiedlichen
Voraussetzung in Folie OODB 5-10.

Funktionalität von JDOQL-Queries

Sortieren der Antwort für Queries:

Interface Query

public void setOrdering (String orderingInfo);

- **orderingInfo enthält Sortieranweisungen, durch Komma getrennt, die von links nach rechts lexikographisch befolgt werden.**
- **Jede Sortieranweisung hat die Form: `<Ausdruck> descending` oder `<Ausdruck> ascending`. Hierbei ist `<Ausdruck>` ein Wert von einem primitiven oder Wrapper-Typ oder vom Typ `BigDecimal`, `BigInteger`, `String`, `Date`.**
  Eigen definierte Objekte sind nicht zugelassen.
- **In `<Ausdruck>` sind nur navigierende Zugriffe auf Attribute erlaubt (also keine beliebigen Rechenoperationen).**

Existenzquantoren

SQL-ähnliche Frage:

- Finde die Vorlesungen, in denen weibliche Studenten sitzen und die von Iwanowski gehalten werden:

select v

from v **in** Vorlesung

where (v.gelesenVon.Name = „Iwanowski“) and (**exists** s in v.Hörer: s.female)

JDOQL-Lösung ?

2. Variante:

- 1. Frage: Finde die Vorlesungen, in denen weibliche Studenten sitzen.
- 2. Frage: Finde unter den Vorlesungen aus der 1. Frage die, die von Iwanowski gelesen werden.

Allquantoren

SQL-ähnliche Frage:

- Finde die Vorlesungen, in denen nur weibliche Studenten sitzen und die von Iwanowski gehalten werden:

select v

from v **in** Vorlesung

where (v.gelesenVon.Name = „Iwanowski“) and (**for all** s in v.Hörer: s.female)

JDOQL-Lösung ?

Verwendung von Aggregatfunktionen: max, min, sum, avg, count im Ergebnis

SQL-ähnliche Frage:

- Finde Anzahl und Durchschnittsalter aller jungen Professoren

```
select count(p), avg(p.alter)
from p in Professor
where p.alter <= 40
```

JDOQL-Lösung ?

ist in JDO möglich

Nicht bei Versant möglich !

Zulässige Ausdrücke im Ergebnis

Auszug aus Spec. 2.0, Kap. 14.6.9, S. 164 (final release):

The result expressions include:

- "`this`": indicates that the candidate instance is returned
- <field>: this indicates that a field is returned as a value; the field might be in the candidate class or in a class referenced by a variable
- <variable>: this indicates that a variable's value is returned as a persistent instance
- <aggregate>: this indicates that an aggregate of multiple values is returned
  - `count`(<expression>): the count of the number of instances of this expression is returned; the expression can be "`this`" or a variable name
  - `sum`(<numeric field expression>): the sum of field expressions is returned
  - `min`(<field expression>): the minimum value of the field expressions is returned
  - `max`(<field expression>): the maximum value of the field expressions is returned
  - `avg`(<numeric field expression>): the average value of all field expressions is returned
- <field expression>: the value of a numeric expression using any of the numeric operators allowed in queries applied to fields is returned
- <navigational expression>: this indicates a navigational path through single-valued fields or variables as specified by the Java language syntax; the navigational path starts with the keyword "this", a variable, a parameter, or a field name followed by field names separated by dots.
- <parameter>: one of the parameters provided to the query.

Allgemeine Verwendung von Aggregatfunktionen: max, min, sum, avg, count im Filter

SQL-ähnliche Frage:

- Finde alle Professoren, die umfangreiche Vorlesungen halten:

```
select p
from p in Professor
where max (select v.SWS from v in p.liest) >= 4
```

JDOQL-Lösung ?

ist in JDO nicht möglich

hier durch Umformulierung möglich

Allgemeine Verwendung von Aggregatfunktionen: max, min, sum, avg, count im Filter

SQL-ähnliche Frage:

- Finde alle Professoren, die viel zu tun haben:

```
select p
from p in Professor
where sum (select v.SWS from v in p.liest) >= 14
```

JDOQL-Lösung ?

ist in JDO nicht möglich

hier auch nicht durch Umformulierung möglich ?

Funktionalität von JDOQL-Queries

Weitere Funktionen:

Interface Query

public void close (Object result);

public void compile ();

public void setIgnoreCache (boolean transactionChangesAreNotConsidered);

public long deletePersistentAll ();

public long deletePersistentAll (Map parameters);

public long deletePersistentAll (Object[] parameters);

public long setUnique (boolean unique);

Zusammenspiel mit anderen JDO-Funktionen:

- **Einbettung einer JDOQLQuery in eine Transaktion nicht zwingend erforderlich, aber empfohlen**

Funktionalität von JDOQL-Queries: Zusammenfassung

- JDOQL verlangt Filter für Boolesche Auswertungen
- Die Auswertungen beziehen sich auf beliebige Eigenschaften von Attributen der dem Filter übergebenen Elemente
- Andere als Booleschen Operationen können im Filter nicht durchgeführt werden (mit wenigen Ausnahmen)
- Die Eingabe in den Filter ist eine Menge von Elementen derselben Klasse (Collection oder Extent)
- Die Elemente können durch Parameter verallgemeinert werden
- Existenzeigenschaften von Collection-wertigen Attributen können über **`contains`** mit Variablen nachgeprüft werden
- Die Ausgabe vom Filter ist eine Collection von Elementen der Eingabeklasse
- Die Ausgabe kann bereits durch den Filter sortiert werden
- Single-String-Queries (nur JDO 2.0) bieten nur eine SQL-ähnlichere Syntax, aber keine andere Funktionalität als Filter-Queries

Analogien bzgl. der Funktionalität

SINGLE STRING	*Filter-Query*
SELECT [UNIQUE] [<result>]	setUnique() setResult()
[INTO <result-class>]	setResultClass()
[FROM <candidate-class> [EXCLUDE SUBCLASSES]]	setCandidates(Extent), setClass()
[WHERE <filter>] subqueries in <filter>	setFilter() setCandidates(Collection)
[VARIABLES <variable declarations>]	declareVariables()
[PARAMETERS <parameter declarations>]	declareParameters()
[IMPORTS <import declarations>]	declareImports()
[GROUP BY <grouping>]	setGrouping()
[ORDER BY <ordering>]	setOrdering()
[RANGE <start>, <end>]	setRange() ----------
	getFetchPlan()... setExtensions() setIgnoreCache() setUnmodifiable()

Aufstellung von Maximilian Herold