Oberseminar Datenbanksysteme: Aktuelle Trends

Abstract zu Thema

Objektdatenbanken am Beispiel db4o

Von Daniel Mertens

HTWK Leipzig

Sommersemester 2017

Prof. Thomas Kudraß

Inhaltsverzeichnis

Einleitung

Objektdatenbanken waren in den 1990-er Jahren ein großer Trend und beeinflussten die Weiterentwicklung relationaler Datenbanksysteme hin zu objektrelationalen Systemen. Heutzutage haben Objektdatenbanken als embedded Databases ein neues Anwendungsgebiet mit Wachstumspotential gefunden (Brenner, 2009).

Die db4o-Engine ist für nichtkommerzielle Projekte frei verfügbar. Ein kommerzieller Einsatz der Software erfordert eine Commercial Licence. Jede Datenbankdatei darf bis zu 254 GByte groß sein (Müller, 2009).

Mittlerweile ist die Software nur noch über Drittanbieter erhältlich, da die Firma Actian sich gegen eine duale Lizenz entschieden hat und demnach auch nicht mehr weiterentwickelt wird. (Actian, 2015). Das neue kommerzielle Produkt wird unter dem Namen „Actian NoSQL“ vertrieben (actian.com, 2017).

Motivation

In moderner objektorientierter Softwareentwicklung wird, je nach Anwendung, 10-15% der Zeit und des Geldes darauf verwendet, Objekte in inkompatible, relationale Datenbanktabellen hineinzupressen, was die Abbildung 1 veranschaulicht. (Brenner, 2009)

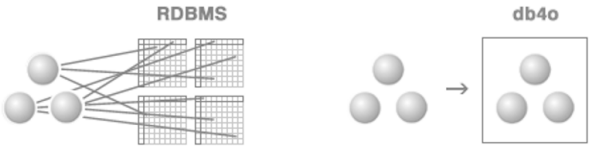

ABBILDUNG 1: VERGLEICH RDBMS UND DB4O

„Tabellen zu verwenden um Objekte abzuspeichern – das ist wie wenn sie Ihr Auto jeden Abend auseinanderbauen um es in die Garage zu bringen. Sie können es am nächsten Morgen wieder zusammenschrauben, aber irgendwann fragt man sich dann doch, ob das die effizienteste Art ist, sein Auto zu parken.“ [Zitat Computer Vordenkerin Esther Dyson].

Basiskonzepte objektorientierter Datenbanken

Die Idee hinter einer Objektdatenbank ist die einfache Speicherung von Objekten aus objektorientierten Programmiersprachen in einen persistenten[1] Speicher. Die Objekte können so in die Datenbank gespeichert werden, wie sie das Programm erzeugt. Die Struktur der Datenablage folgt den Richtlinien der Objektorientierung. Das heißt, die zu einem Objekt gehörenden Daten werden im Objekt selbst abgelegt. Ein Objekt umfasst als Instanz einer Klasse nicht nur die zu manipulierenden Daten, sondern auch die für die Klasse definierten Methoden. Die interne Organisation und Verwaltung der Daten wird komplett vom ODBMS übernommen. So ist die Persistenz für den Programmierer transparent und aufwendige Schnittstellenentwicklung zwischen relationaler Datenbank und objektorientierten Programmcode entfällt. (Abts, 2016, p. 549 ff.)

Objektrelationale Datenbanken

Objektrelationale Datenbankmanagementsysteme sind eine Weiterentwicklung von relationalen Datenbanken. Sie versuchen traditionelle relationale Systeme unter Verwendung objektorientierter Konzepte erweiterbar zu gestalten, um ihnen neue Anwendungsgebiete mit komplexen Daten zu erschließen. Ihr Einsatzgebiet ist dort, wo Mengen von Objekten in Beziehung zu anderen Daten oder Objekten gebracht werden müssen. Häufig wird über eine relationale Datenbank eine objektorientierte Zugriffsschicht gesetzt. Bei manchen Zugriffsschichten werden die Objekte und deren Attribute erst dann geladen, wenn sie in der Anwendung auch benötigt werden. Einsatzgebiete sind unter anderem Systeme zur Erfassung geographischer Daten (GIS), bei denen Koordinaten miteinander verknüpft sind oder andere Daten referenzieren. Beispielsweise referenzieren mehrere Koordinaten-Objekte eine Straße: die Koordinaten stehen in Relation mit einem Straßennamen und sind selbst Objekte, die zueinander eine Beziehung haben. (Lufter, 1999).

Embedded Databases

Ein eingebettetes Datenbanksystem ist von außen als solches nicht zu erkennen und kann auch nicht von Fremd-Systemen zur Datenspeicherung genutzt werden. Die Vorteile von eingebetteten Datenbanksystemen ergeben sich daraus, dass der Hersteller eine für die spezielle Anwendung ausgerichtete Anpassung vornehmen kann, die über die Möglichkeiten der normalen Administration und Beschleunigung hinausgeht.

---

[1] Persistenz ist in der Informatik der Begriff, der die Fähigkeit bezeichnet, Daten (oder Objekte) oder logische Verbindungen über lange Zeit (insbesondere über einen Programmabbruch hinaus) bereitzuhalten.

Ein weiterer Vorteil ist eine einfachere Installation und Lizenzierung eines Produktes, das ein eingebettetes Datenbanksystem verwendet. Der Produkt-Hersteller kann sein Produkt als Ganzes an seine Kunden ausliefern. Lizenzen für sein Produkt können ohne Beteiligung des Datenbank-Herstellers zwischen dem Produkt-Hersteller und seinem Kunden verhandelt werden. Ein Nachteil ist zum Beispiel, dass man die so gespeicherten Daten schwieriger auf ein System eines anderen Herstellers übertragen kann (wikipedia.org, 2017).

Beziehungen in Objektdatenbanken

Die Objektdatenbanken verwalten die Beziehungen eigenständig. Da die Objekte über die Beziehungen abgefragt werden können, die in der Datenbank gespeichert sind, ist es nicht notwendig, mehrere Tabellen der Datenbank durch Verbindungen zu komplexen Datenobjekten zusammenzusetzen. Klassenbeziehung entspricht Objektbeziehung auf Instanzenebene. 1:1 Beziehungen werden durch Attribute der jeweils anderen Klasse definiert. Bei 1:n Beziehungen bekommt die 1-Seite eine Array-List und die n-Seite ein Attribut der anderen Klasse.

(eu-datenbank, 2017)

CRUD-Operationen

CREATE → db.store(object) wird ein beliebiges Objekt gespeichert.

READ → Eine Personen-Instanz wird als Muster erstellt und beliebige Felder werden gesetzt.

UPDATE → z.B. mit *setAge(25)* kann gespeichert werden. Im JPA Standard entspricht dieses einem em.merge(object), wobei das Objekt immer von der Session referenziert werden muss.

DELETE → db.delete(object) löscht ein Objekt, das dem Objektmanager bekannt sein muss.

Anfrageschnittstellen

QBE (Query By Example)

Query by Example steht für eine Suche anhand von Beispielen. Bei db4o kann ein beliebiges Objekt erstellt werden und danach:

- Kein Feld gesetzt, so dass alle Felder null sind, oder es wird direkt eine Referenz auf die Klasse übergeben (z.B. Person.class): In diesem Fall werden alle in der Datenbank gespeicherten Instanzen der Klasse zurückgeliefert.
- Es werden beliebig viele Felder gesetzt: Dann werden alle gespeicherten Objekte zurückgeliefert, bei denen die im Muster gefüllten Felder mit dem Wert im Objekt übereinstimmen.

- Es wird null übergeben: Dieses liefert alle Objekte aus der Datenbank zurück.

Komplexe Abfragekriterien lassen sich bei Verwendung dieser Art der Abfrage nicht ausdrücken (Brenner, 2009, p. 87).

S.O.D.A. / Criteria Queries

Intern werden alle Abfragen bei db4o auf SODA abgebildet. SODA ist die Programmierschnittstelle des auf SourceForge gehosteten Projekts S.O.D.A. – Simple Object Database Acces. Die SODA-Abfragen ähneln denen von Hibernate-Criteria-Abfragen. Ein Beispiel:

```
Query query=db.query();
query.constrain(Person.class);
query.descend("alter").constrain(50).greater();
ObjectSet result=query.execute();
```

Hier wird die Abfrage zunächst auf die Klasse Person eingeschränkt, danach weiter auf alle Personen, die älter als 50 sind.

Sie ist daher mächtiger als die anderen Abfragemechanismen, allerdings auch schwerer zu erstellen, da der Code schwerer zu lesen ist. Durch die Verwendung von Zeichenketten als Abfrageparameter wird die Typsicherheit zum Kompilationszeitpunkt nicht gewährleistet (Brenner, 2009, p. 87).

Native Abfragen

Native Abfragen sind Abfragen, die in der Programmiersprache der Client-Anwendung formuliert sind, also bei db4o beispielsweise in Java oder C#.

Der Code von nativen Abfragen wird bei db4o normalerweise nicht ausgeführt, sondern zur Laufzeit analysiert. Basierend auf dem Ergebnis der Analyse wird eine SODA-Criteria-Query erstellt. (Brenner, 2009, p. 87).

Transaktionen

Unter Transaktionen versteht man eine oder mehrere Operationen, die logisch zusammengehören. Startpunkt einer Transaktion ist die Methode *openDB()*. Innerhalb einer Transaktion gibt es zwei wichtige Methoden. Die Methode *rollback()*, mit der alle Operationen rückgängig gemacht werden können und die Methode *commit()*, die sicherstellt, dass alle Speichervorgänge in die Datenbank geschrieben werden und für andere Nutzer sichtbar werden. Eine Transaktion wird durch die Methode *close()* beendet und schließt die Datenbank (Brenner, 2009, p. 151 ff.).

Client-/Server-Modes in db4o

Neben dem Embedded Mode bietet db4o auch einen Client/Server-Mode an. Dafür wird Db4o als Server gestartet und kann von verbundenen Clients Anfragen über TCP/IP entgegennehmen. Um db4o als Server zu starten, steht die Methode openServer() bereit. Übergeben werden können zwei Parameter, der Pfad zur Datenbankdatei und optional den TCP/IP Port. Damit ein Client Zugriff auf den Server erhält, muss mit der Methode server.grantAccess() ein Nutzername und Passwort festgelegt werden.

Ein Client wird mit der Methode openClient() und den Parametern Hostname, Port, Nutzername und Passwort erstellt. Nachdem der Server gestartet ist, folgt die Methode wait(), die eine InterruptedException wirft, sobald ein Client mit dem Server in Interaktion tritt. Deshalb muss die Methode in einem Try-Catch-Block stehen. (Brenner, 2009, p. 139 ff.).

weitere Eigenschaften

Replikation

Mittels weniger Zeilen können beliebig viele Server repliziert (geclustert) werden. Dies geschieht mit dem db4o-*dRS*-Modul.

Die Objekte bekommen dazu eine eindeutige UUID. Dadurch können einzelne Objekte oder die gesamte Datenbank uni- oder bidirektional verteilt werden, d.h., jedes neue Objekt wird an alle anderen Datenbanken gesendet. Dies erlaubt es große und redundante Architekturen aufzusetzen.

Außerdem bietet db4o die Option, die Objektdatenbank in eine relationale Datenbank zu replizieren. db4o verwendet dazu Hibernate. Mit Hilfe der dRS-Bridge können ebenfalls einzelne Objekte oder ganze Datenbanken per Trigger wahlweise in eine oder beide Richtungen transferiert werden. In vielen Fällen können so die Vorteile beider Welten gut ausgenutzt werden. (Paterson Edlich, 2006, p. 275 ff.)

Ladeverhalten

Bei db4o gibt es drei Modi für das Laden von Objekten:

IMMEDIATE

Hier werden bei einer Abfrage alle gesuchten Objekte unmittelbar ermittelt. Bei großen Ergebnismengen kann dies viel Zeit und Arbeitsspeicher in Anspruch nehmen.

LAZY

Hier wird sofort ein ObjectSet zurückgeliefert, obwohl noch kein Objekt ermittelt worden ist. Ein Thread sucht alle weiteren Ergebnisse. Dieses Verhalten ist ideal, wenn die ersten Ergebnisse schnell angezeigt werden sollen. Bei diesem Lademodus muss beachtet werden, dass die Abfrage nicht zu einem definierten Zeitpunkt ausgewertet wird. Hierdurch kann es zu Nebeneffekten kommen, wenn zwischen dem Absetzen und dem vollständigen Auswerten der Anfrage Änderungen am Datenbestand vorgenommen werden.

SNAPSHOT

In diesem Zwischenmodus wird die Auswertung aller indizierten Bestandteile der Abfrage zu einem definierten Zeitpunkt vorgenommen. Anschließend wird der Zustand der möglichen Ergebnismenge in einem Snapshot aufgezeichnet. Die weitere Verarbeitung der nichtindizierten Abfragebestandteile findet in einem nebenläufigen Thread statt. Hierdurch werden die Vorteile der verzögerten Auswertung unter Vermeidung ihrer Nebeneffekte bereitgestellt. Für den erzeugten Snapshot wird jedoch Arbeitsspeicher verwendet, der erst mit Freigabe des ObjectSet freigegeben werden kann.

Objekte bilden normalerweise tiefe Referenzstrukturen. In db4o kann beim Laden von Objekten angegeben werden, bis zu welcher Tiefe referenzierte Objekte implizit mit dem Objekt geladen werden sollen. Zusätzlich besteht die Möglichkeit, diese referenzierten Objekte explizit zu einem späteren Zeitpunkt zu aktivieren.

Callbacks

Bei db4o gibt es interne und externe Rückruffunktionen (callbacks). Diese ähneln in ihrer Bedeutung den Datenbanktriggern anderer Datenbanken. Rückrufmechanismen gibt es in vielen Produkten zur Datenbankanbindung und sind ein mächtiges Werkzeug, um beispielsweise Überprüfungen durchzuführen, Standardwerte zu setzen oder Sicherheitsaspekte zu berücksichtigen.

Interne Callbacks

Interne Callbacks werden den Klassen der zu speichernden Objekte als Methoden hinzugefügt. Diese werden von db4o dynamisch unter Verwendung der Reflexion erkannt und aufgerufen. Hierzu ist es nicht notwendig, die Klassen von speziellen Schnittstellen abzuleiten. Aufrufe interner Callbacks sind vor und nach datenbankbezogenen Ereignissen möglich. Der Aufrufzeitpunkt wird hierbei durch das Präfix des Methodennamens festgelegt: objectCan...–Methoden werden vor und objectOn...–Methoden werden nach dem Ereignis aufgerufen. Durch den Rückgabewert der objectCan...–Methoden kann gesteuert werden, ob die mit dem Ereignis verbundene Aktion stattfinden soll. Für die internen Callbacks stehen folgende Ereignisse zur Verfügung: New, Update, Delete, Activate und Deactivate.

Externe Callbacks

Externe Callbacks sind keinen Klassen zugeordnet. Stattdessen werden sie beim ObjectContainer registriert. Zusätzlich zu den bei den internen Callbacks möglichen Ereignissen stehen für diese Callbacks QueryStarted und QueryFinished zur Verfügung. Die Verwendung externer Callback ermöglicht eine Ereignisbehandlung ohne Abänderung der zu speichernden Klassen. Dieses kann den Vorteil haben, dass Aspekte der Persistenz nicht mit der Geschäftslogik vermischt werden müssen. Wenn der Quelltext der zu speichernden Klassen nicht zur Verfügung steht, muss mit externen Callbacks gearbeitet werden.

Alternative Apache Derby

2006 wurde Derby als Java DB im Java Development Kit ab Java 6 integriert. Auch Derby beherrscht einen eingebetteten Modus und einen Netzwerkmodus. Als dritten Modus kann Derby die Datenbank statt auf der Festplatte auch im Hauptspeicher halten. Dieser In-Memory-Datenbank-Modus ist insbesondere bei Tests vorteilhaft oder wenn die Daten nicht sofort persistiert werden müssen, weil bei dieser Betriebsart der einzelne Datenbankzugriff weitaus schneller abläuft als in den anderen Modi.

Demonstration

Im Anschluss an den Vortrag folgt eine Demonstration zum Speichern, Auslesen und Löschen von Objekten. Außerdem wird in einem Vergleich gezeigt, wie sich die S.O.D.A Queries von den Nativen Abfragen unterscheiden.

Literaturverzeichnis

Abts, P. D. D., 2016. *Grundkurs JAVA.* 9 Hrsg. Wiesbaden: Springer Vieweg.

actian.com, 2017. *www.actian.com.* [Online]
Available at: https://www.actian.com/data-management/versant-nosql-object-database/
[Zugriff am 06 14 2017].

Actian, 2015. *Supportservices Actian.* [Online]
Available at: http://supportservices.actian.com/versant/default.html
[Zugriff am 14 06 2017].

Brenner, I., 2009. *Datenbankentwicklung mit db4o.* 1 Hrsg. Norderstedt: Books on Demand GmbH.

eu-datenbank, 2017. *www.eu-datenbank.de.* [Online]
Available at: http://www.eu-datenbank.de/fachartikel/vor-und-nachteile-von-objektdatenbanken/
[Zugriff am 15 06 2017].

Lufter, J., 1999. *Gesellschaft für Informatik.* [Online]
Available at: https://www.gi.de/service/informatiklexikon/detailansicht/article/objektrelationale-datenbanksysteme.html
[Zugriff am 16 06 2017].

Müller, F., 2009. *heise.de.* [Online]
Available at: https://www.heise.de/ix/artikel/Mittendrin-statt-nur-dabei-794690.html
[Zugriff am 02 06 2017].

Paterson Edlich, S. a. J., 2006. *The Definitive Guide to db4o.* 1 Hrsg. New York: Springer.

wikipedia.org, 2017. *Wikipedia.org.* [Online]
Available at: https://de.wikipedia.org/wiki/Eingebettetes_Datenbanksystem
[Zugriff am 02 06 2017].