A Quick Database Comparison of Db4o and SQL Databases through Cayenne

Peter Karich

August 11, 2007, Bayreuth

Contents

This short evaluation is based on only some little knowledge I gained through my project at gstpl.sourceforge.net. It will give you a short summary, which persistent mechanism could be good for you. I compare the object oriented database 'db4o' (database for object) with the object-relational mapper 'Cayenne' and the SQL database 'Derby'. I used db4o 6.1, cayenne 2.0.2 and derby 10.2.2.0.
Of course you have the choice of the database if you use Cayenne, e.g. you could choose any SQL database e.g. 'hsqldb'. And of course there are several other object-relational mappers like Cayenne e.g. 'Hibernate'.

The first entry is related to db4o; the other is for cayenne.

1 Design

- In **Db4o** you can persist any class, even unknown.
- In **Cayenne** you have to extend your classes, which you want to persist. And you need some configuration files.

1.1 Pros

- It is very easy to use existing classes and you can easily separate persistence mechanism from application code.
- Cayenne will do the job for you. So you don't need to know how deep your references go or will go in future. And you don't need to update your objects explictly after their properties may changed.

1.2 Cons

- The activation of the classes is static. So if you access deep references you have to configure it. And if you perform a rollback operation on the database you have to explicitly (!!) refresh all live objects which may have participated in a rollback transaction.

- With Cayenne you get in some little troubles, if you want to preserve the separation of application and persistence code, but of course you can separate this if you want. Another problem is that you need to map the classes. That means Cayenne needs to know which attributes, relationships etc. you want to store and what names they have. I think it is very easy to be done, because Cayenne comes with the so called 'modeler', where you can do this job graphically and you can generate the necessary classes from this modeler.
  But it is a little bit complicated to add a property (a column) while your application is running.
  Another problem is that you cannot create an object (of these classes), which you

don't want to persist. Because cayenne automatically register it if you add it as a ToManyTarget (or ToOneTarget). E.g. you don't need to save this object, because you only use it for calculation purposes or there are too many and persisting will take a long time. A simple workaround is to rollback the registerd objects while calculation. There is a more complicated workaround look here:

`http://objectstyle.org/cayenne/lists/cayenne-user/2007/03/0181.html`
`http://objectstyle.org/cayenne/lists/cayenne-user/2007/04/0054.html`

2 License

- Db4o uses multiple licenses: commercial, GPL and a license like the GPL, but a little bit lesser: It has the exception of some licenses e.g. projects which are under the LGPL can use this and this will NOT force you to make your project available under the GPL, which would be the case if you choose the GPL.

- Cayenne stands under the apache license version 2.

2.1 Pros

- Db4o is a company AND has a community, so they can push the project by marketing, events etc.
- You can use Cayenne even in commercial projects.

2.2 Cons

- You can't use Db4o in projects with commercial licenses without paying something.
- ?

3 Community

3.1 Pros

- Intact community of db4o.
- Cayenne: The best community ever, Thanks to Andrus!

3.2 Cons

Both project are open source project so a responding time of 2 or 3 days is normal. But mostly faster.

4 Querying raw SQL-statement

4.1 Pros

- Db4o has also a project, which supports sql-like statements (db4o-sql). You can even access db4o by a JDBC driver. (But I did not try it!)
- You can call raw SQL statements through Cayenne API.

4.2 Cons

- The db4o-sql statements are very basic.
- If you use raw SQL statement to UPDATE objects in cayenne. The object graph used internally by cayenne will not record those changes, so this is not a recommended approach.

5 Performance

I can't find a very big difference here. But keep in mind:

- If you think your application is slow because of the database, there are a lot of configuration mistakes you could have made.

For further performance stuff look at 'db4o written' sites: polepos.org or
http://www.jpox.org/docs/performance.html

6 Specials

6.1 Db4o

- Native queries in db4o are very easy to use and typesafe, but they not as performant as the fastest S.O.D.A. API (simply object direct access??).
- Another querying mechanism is: Querying By Example. Very easy to use. Even for database newbies.
- JPOX allows querying using either JDOQL, SQL, or JPQL. Please look at http://www.jpox.org/ for more information. It should be equivalent to JPA.

6.2 Cayenne

- Java Persistent Architechture (JPA) will be implemented in cayenne 3.0.
- Remote persistent framework. TODO

6.3 Direct comparison db4o - ORM

Most of this section was taken from Stefan Edlich:
`http://best-practice-software-engineering.blogspot.com/2006/11/event-java-persistenz-23-nov-2006_27.html`
It is really important to select the right tool (db4o or ORM) for a specific problem. Stefan suggests RDBM systems when:

- Data needs to be stored in a neutral way, different views are required. Where 'neutral' means: different systems with even different computer languages need access. So that for instance a BitSet.java object shouldn't be stored directly into the db. In ORM you would use BLOB's or even Strings.
- OLAP is applied. Data storage does not happend in rows, but in multidimensional spaces. See www.palo.net as an implementation.
- Clustering is necessary
- additional DB features required

But use db4o:

- No mapping is needed (no neutreal representation is required)
- Higher performance for deeper object graphs is necessary
- If you want to support Scheme changes
- Less administration of the DB required