



Hibernate and Connector/J performance considerations

Fernando Ipar
June 2011

Agenda

- ORMs: why and when?
- (very) brief Hibernate review
- Performance considerations:
 - Hibernate
 - Connector/J

ORMs: why and when?

What

Object Relational Mapping

Why

Object Relational impedance mismatch

Using RDBMS from an OO lang, you always use *some* form of ORM

When

When your object domain is rich and you implement most business logic in your app

Hibernate review

Key elements

- Persistent Objects

- Configuration

 - Properties

 - Mapping files

- Database

Supports reverse engineering

Can manage transactions, connection pooling, naming, etc.

Hibernate review

Persistent Objects

Your domain

Mapped to database table(s) by use of Mapping XML files

Configuration

Properties for global settings (such as logging, connection pooling, etc)

XML Mapping to configure what and how to persist (including fetch strategies)

Performance considerations: Hibernate

Hibernate

We assume version 3.6

If not, still use anything ≥ 3

Fetch strategies

SELECT

JOIN

Lazyness

Collections

Columns

Hibernate

Fetch strategies

SELECT

Causes the N+1 SELECTs problem

Has it's uses

You only need to access a (very) few items of the associated collection

JOIN

Single SELECT statement

Has it's problems

When SELECT is the best strategy, JOIN will use unnecessary RDBMS, network, and Java memory resources

Hibernate

Clear collections for one shot delete when possible

Alternative is one delete statement for each deleted child

Won't work if your association is mapped with `inverse="true"`

Hibernate

Lazyness

For collections, it's best to configure them lazy and change at runtime when needed (default for hb>= 3)

Too many non lazy collections: entire database fetch into memory in every transaction

For columns, manual says it's just a 'marketing feature'

But not so with covering indexes!

Supports batch fetching

Hibernate

When all mapping fails:

```
session.createQuery("SELECT ...").list();
```

But be sure to use a list of the proper type!

For tuples: `List<Object[]>`

```
List<Object[]> actors_names = session.createQuery  
("SELECT first_name, last_name FROM actor LIMIT  
10").list();
```

Hibernate

Concurrency

Optimistic concurrency control

@Version

on Integer property

on Date / Calendar (for Timestamp versioning)

Note this is for application transactions (or *conversations*) which usually span multiple database transactions.

Performance considerations: Connector/J

Connector/J

Default configuration has no-surprises and jdbc-compatibility goals

Always use latest stable version

Performance improvement with bundled config properties

`src/com/mysql/jdbc/configs/`

`maxPerformance | solarisMaxPerformance`

Good wins and mostly safe:

<http://www.bigdbahead.com/?p=230>

Connector/J

Less 'extra' queries

`useLocalSessionState=true`

`useLocalTransactionState=true`

Less latency

`rewriteBatchedStatements`

`cachePrepStmts=true`

only if `useServerPrepStmts=true`

`dontTrackOpenResources=true`

Though it's better to just always close connections

Connector/J

`maintainTimeStats=true`

won't include elapsed time in error messages

`cacheResultSetMetadata=true`

Only if you know your table definitions don't change

`elideSetAutoCommits=true`



fernando.ipar@percona.com

We're Hiring! www.percona.com/about-us/careers/