## FAQs

What if I have a table, view, sequence, or column that produces has a name that produces a reserved Java word?

Tables, view, and sequences do not produce Java keywords since their Java names are by default produced as camel case, and therefore start with an uppercase letter. Columns, on the other hand, could potentially produce a Java keyword. For example, a column named CLASS would produce by default a property class; the resulting Java code would not be valid and the Java class will not even compile. If this is the case, you can specify the Java name you want for a specific column adding a <column> tag a the <table>, <view>, or <select> tag as shown below:

<table name=”VEHICLE”>

<column name=”CLASS” java-name=”vehicleClass” />

</table>

The <column> tag is optional. All the other columns apart from CLASS will use their default names.

Can I use nulls to select by Example?

Yes. The example parameter on the selectByExample(), updateByExample(), and deleteByExample() methods differentiate which properties were set or not. All set properties will participate in the search condition, including properties set to null. For example:

VehicleDAO example = new VehicleDAO();

example.setBrand(”Kia”);

example.setOwnerId(null);

will affect all vehicles of the Kia brand without owner, while:

VehicleDAO example = new VehicleDAO();

example.setBrand(”Kia”);

will affect all vehicles of the Kia brand regardless if they have an owner or not.

Can use parameters for an update or select?

Yes. Arbitrary SQL statements specified with the <select> and <update> tags support Java parameters. Those parameters become part of the Java method signature. The syntax to add those parameters in the SQL statement looks like this:

<select java-class-name="BigAccount">

<![CDATA[

select a.\* , t.\*

from account a

join transaction t on (t.account\_id) = (a.id)

{\*

where a.current\_balance >

#{minBalance,javaType=java.lang.Long,jdbcType=NUMERIC}

\*}

]]>

</select>

For more details see the sections Custom SQL Selects and Custom SQL Updates.

Can I update on a view?

No. The current version of HotRod support retrieving data from a view but does not support updating on views. RDBMS differ on which views can be updates but in general they allow SQL inserts, updates, and deletes on simple views. HotRod may support these operations in the future.

Can I run an exotic SQL that use database-specific syntax?

Yes. All SQL selects and updates can be automated by HotRod. As long as the JDBC driver is able to run them, HotRod can create the corresponding DAOs for it.

For more details see the sections Custom SQL Selects and Custom SQL Updates.

Can I read and update BLOBs and CLOBs?

Yes, BLOBs and CLOBs can be used in SQL select, inserts, and updates as byte[] and String respectively. However, by default they are read and written all at once into the database. It's technically possible to manage them as streams using custom type handlers, but this isn't an out of the box solution.

Can I sort the rows when using the selectByExample() method?

Yes. The selectByExample() method receives a variable length parameter list that accepts zero, one, or more extra parameters to specify the sorting criteria of the rows.

Can I use case-insensitive sorting?

Yes. The enum parameter values that can be used by the selectByExample() method include ascending and descending options for String-like columns that include case insensitive variants.

Can I select rows related by foreign keys?

Yes, foreign keys can be navigated from a DAO object to retrieve related parent row or children rows. All FK relationships are leveraged and the DAO includes Java methods that navigate any direction, including self-referenced tables, and FKs pointing to PKs or unique constraints.

What is "row version control" useful for?

Optimistic Locking is a strategy to manage concurrent access to database data. It reserves a numeric column of the table to be used as the version number of the row.

This concurrency management strategy produces no locks on the database while reading a row—and in that way is very lightweight for the database—considering that most likely the row won't be updated during the time the transaction completes. Later on, when updating the row, it determines if the row was in the mean time updated or not by another process. If not, the row is still “fresh” and the update succeeds; otherwise, the row is considered “stale” and the update fails by throwing a StaleDataException.

See the section Optimistic Locking (Row Version Control) to see how HotRod automates this strategy.

Which RDBMSs are supported?

As of version 3.0.0 HotRod supports Oracle database, IBM DB2, SAP ASE (ex-Sybase), Microsoft SQL Server, PostgreSQL, MySQL, MariaDB, HyperSQL, and H2.

I want database XXX to be supported. Will you add it?

We can theoretically add any database with a decent JDBC driver, as long as there it interest on it... and, of course, as long as we have time, and database licenses, and hardware resources to devote on it.

What if I don't know the catalog and schema for my dev database?

Leave it blank and run HotRod. HotRod will complain that is blank, and will show the full list of catalog and/or schemas available on your database. Then you can pick the one you want. Make sure you use the correct case. Databases are very picky.

What if I have two/three/multiple database schemas?

Run HotRod once per schema and use separate JDBC data sources for each one. As of version 3.0.0 HotRod supports a single schema per data source. This is suitable for most cases, but if you happen to have multiple database schemas, using multiple JDBC data sources is the simplest solution. If you happen to need database transactions across multiple schemas make sure you enable the XA option on the JDBC data sources so the JTA can control the transaction effectively.

Can I divide the HotRod configuration file in multiple sections?

Sure. Instead of having a single gigantic configuration file you can divide it into fragments. Each fragment can include fragments as well, in multiple levels, as long as there are no circular references. To include a fragment, add the <fragment> tag in the configuration file as shown below:

<fragment file="accounting/reporting.xml" />

This tag will include the definitions from the accounting/reporting.xml file.

This can be quite useful to reduce the conflicts on you source control repository. If the configuration file is divided per modules or separate sections, developers will work on specific sections at a time without stepping into each other's toes.

Is there a plug in for Eclipse?

No. So far Ant seems to work pretty well, but a plug-in for Eclipse could be simpler to use.

Does HotRod generate stored procedures for SQL in order to improve performance?

No. It seems that in the early 2000s some databases (notably Microsoft SQL Server) suffer significant performance loss when using straight SQL statements compared to the same SQL statements precompiled in stored procedures. This lead to the expensive strategy of adding all SQL statements as stored procedures in the database.

Nowadays, this is not needed anymore since all world-class databases deal with this cases well by enabling clever caching default on all SQL statements.

It may still be a good practice to use stored procedures to run SQL, but performance improvement against straight SQL is not one of them anymore.

What database column types are supported out of the box?

Common column types such as character char, varchar, numeric types, dates, times, timestamp, blobs, and CLOBs work out of the box for the supported databases. Other column types can also be used if specified in the configuration file using the <column> tag.

To see the full list of out of the box column types see the corresponding subsection of the Supported Databases & Column Types, for the specific database you're using.

I have a column with an exotic type not supported by HotRod. What do I do?

HotRod does not provide default types for all possible data types each database offers. However, if you think JDBC can deal with this column type, you can specify its java and JDBC type using the <column> tag, as shown below:

<table name=”property”>

<column name="main\_boundary"

java-type="byte[]"

jdbc-type=”VARBINARY” />

</table>

In this case we assume the MAIN\_BOUNDARY column is of the type POLYGON (not shown in the example) and the JDBC driver can read/write it as a byte[] using the JDBC type VARBINARY.

In any case, these cases are heavily dependent on the database and JDBC driver version you are using. Consult the database documentation to try to find out about the best Java and JDBC types to use.

Can I use auto-increment/identity PK columns?

Yes. You'll need to specify the PK column name using the <auto-generated> on the table tag. The newly generated PK value is populated back into the DAO right after the insert() method is executed.

Can I use database sequences to generate PK values?

Yes. You'll need to specify how the PK value is generated using the <auto-generated> on the table indicating the corresponding sequence name. The newly generated PK value is populated back into the DAO right after the insert() method is executed.

Also, sequence values can also be retrieved separately for any other purpose by adding a <sequence> tag into any <table>, <view>, or <dao> tag. Java methods are generated that return a new sequence value each time they are called. These methods do not produce any SQL insert but only retrieve the sequence value for any purpose deemed necessary.