HotRod MyBatis Cheat Sheet

Select by PK

```
UserDAO u = UserDAO.select(17);
```

Available on tables with a PK.

Insert

```
UserDAO u = new UserDAO();
u.setUsername("jsmith");
u.setFirstName("John");
u.setLastName("Smith");
u.setGroupId(101);
u.setState(1);
u.insert();
```

Available on all tables.

Update by PK

```
UserDAO u = UserDAO.select(17);
u.setFirstName("Jamie");
u.update();
```

Available on tables with a PK.

Delete by PK

```
UserDAO u = new UserDAO();
u.setId(17);
u.delete();
```

Available on tables with a PK.

Select by Unique Constraint

```
UserDAO u = UserDAO.
  selectByUIUsername("jsmith");
```

Available on tables with unique constraints.

Auto-generated PK

```
// Sequence (e.g. Oracle)
GroupDAO g = new GroupDAO();
g.setName("admin");
g.insert();
System.out.println("id=" + g.getId());
// shows the new id value
```

```
// Identity column (e.g. MySQL)
GroupDAO g = new GroupDAO();
g.setName("admin");
g.insert();
System.out.println("id=" + g.getId());
// shows the new id value
```

```
// Optional identity column (DB2)
GroupDAO g = new GroupDAO();
g.setId(123); // value 123 is forced!
g.setName("admin");
g.insert();
System.out.println("id=" + g.getId());
// shows the value 123
```

Same method in Java. Available on tables with PK auto-generation specified in the config file.

Select by Example

```
UserDAO example = new UserDAO();
example.setState(1); // Get active users
List<UserDAO> users = UserDAO.
  selectByExample(example);
```

Available on tables and views.

Select by Example, with Order

```
UserDAO example = new UserDAO();
example.setState(1); // Get active users
List<UserDAO> users = UserDAO.
  selectByExample(example,
    UserOrderBy.LAST_NAME,
    UserOrderBy.FIRST_NAME);
```

Available on tables and views.

Update by Example

```
// Deactivate all active users of group 78
UserDAO example = new UserDAO();
example.setState(1); // active
example.setGroup(78);
UserDAO newValues = new UserDAO();
newValues.setState(0); // inactive
UserDAO.updateByExample(example,
  newValues);
```

Available on all tables.

Delete by Example

```
// Delete all inactive users
UserDAO example = new UserDAO();
example.setState(0); // inactive
UserDAO.deleteByExample(example);
```

Available on all tables.

Transactions

```
TxManager tx = null;
try {
  tx = UserDAO.getTxManager();
  tx.begin();

  UserDAO u1 = UserDAO.select(12);
  u1.setGroup(204);
  u1.update(); // updates group to 204

  UserDAO u2 = UserDAO.select(17);
  u2.setGroup(252);
  u2.update(); // updates group to 252

  tx.commit();
} finally { // don't forget to free resources!
  if (tx != null) {
    tx.close();
  }
}
```

Supports simple transactions (depicted above), custom transactions, custom mappers, and interlaced transactions (if supported by your RDBMS/driver). Also supports isolation levels.

Custom & Native SQL Select

```
// Get all privileges set by a specific user
// from all active users on a specific group
```

- In HotRod's configuration file:

```
<select java-class-name="ActivePrivilege">
 select
   u.username, p.*
  from privilege p
  join user u on (u.id) = (p.user_id)
  {*
  where u.state = 1
   and u.group =
     #{group,javaType=java.lang.Long}
   and p.created_by =
     #{createdBy,javaType=java.lang.Long}
  *}
</select>
```

- In the java application:

```
List<ActivePrivilege> privs =
  ActivePrivilege.select(123, 5);
```

Creates a new DAO java class. Complex and database-specific SQL selects can be used. Dynamic SQL selects can also be used. Specified parameters make up the list of method parameters. Available on all selects specified in the config file.

Custom & Native SQL Update

```
// Move all users without privileges to a
// specific user group
```

- In HotRod's configuration file:

```
<table name="user">
  ...
  <update java-method-name=
    "moveUnprivilegedUsers">
    update user u set group =
      #{group,javaType=java.lang.Long}
     where u.id not in (
       select id from privilege p
         where p.user_id = id)
  </update>
</select>
```

- In the java application:

```
UserDAO.moveUnprivilegedUsers(1041);
```

Adds a new method to a DAO class. Can be added to <table> and <dao> tags. Complex and database-specific SQL updates can be used. Dynamic SQL updates can also be used.

Select Sequence Value

```
// Get the value of sequence user_seq
```

- In HotRod's configuration file:

```
<table name="user">
  ...
  <sequence name="user_seq" />
</table>
```

- In the java application:

```
long value = UserDAO.
  selectSequenceUserSeq();
```

Can be added to <table> and <dao> tags.

Select Parent Row by FK

```
UserDAO u = UserDAO.select(17);
GroupDAO g = u.
  selectParentGroup().byGroupId();
```

Available on tables with imported FKs.

Select Children Rows by FK

```
UserDAO u = UserDAO.select(17);
List<PrivilegeDAO> privs = u.
  selectChildrenPrivilege().
  byUserId();
```

Available on tables with exported FKs.

Reflexive Select Parent by FK

```
UserDAO u = UserDAO.select(17);
UserDAO creator = u.
  selectParentUser().byCreatedBy();
```

Available on tables with imported reflexive FKs.

Reflexive Select Children by FK

```
UserDAO u = UserDAO.select(17);
List<UserDAO> created = u.
  selectChildrenUser().byCreatedBy();
```

Available on tables with exported reflexive FKs.

Update with Optimistic Lock

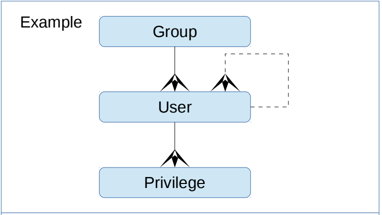
```
UserDAO u = UserDAO.select(17); // #1
u.setGroupId(102);
try {
  u.update(); // #2
  // Successfully updated
} catch (StaleDataException e) {
  // Row had been updated/deleted by
  // other process between steps #1 and #2
}
```

Available on all tables with version control specified in the config file.

Delete with Optimistic Lock

```
UserDAO u = UserDAO.select(17); // #1
u.setGroupId(102);
try {
  u.delete(); // #2
  // Successfully deleted
} catch (StaleDataException e) {
  // Row had been updated/deleted by
  // other process between steps #1 and #2
}
```

Available on all tables with version control specified in the config file.



- A group may have many users.
- A user always belongs to a group.
- A user may be created by another user.
- A user may create many users.
- A user may have many privileges.
- The username of a user is unique.
- A privilege always belong to a user.