# Example 14 - Dynamic SQL

This example shows how to run dynamic SQL statements.

Dynamic SQL is a powerful extension provided by the MyBatis persistence layer. It can dramatically reduce the effort of developing and debugging complex SQL statements that vary depending on the runtime parameter values, all without writing any Java code.

Dynamic SQL shares the syntax with Regular SQL and Native SQL. They can be combined all at once when creating SQL statements.

## How to Run this example

The **Example 14** is included in the download package. To run this example please refer to the section How to Run the Examples above.

## Case #1: Searching by a dynamic criteria

The SQL select is extended using XML tags that define sections that are dynamically included or excluded depending on the runtime parameter values. The configuration file includes the following section:

<select java-class-name=*"SearchedClient"*>

<![CDATA[

select c.\* from client c

{\*

<where>

<if test="#{minPurchases,javaType=java.lang.Integer,jdbcType=NUMERIC} != null">

and (select count(\*) from purchase p where p.client\_id = c.id ) >= #{minPurchases}

</if>

<if test="#{state,javaType=java.lang.String,jdbcType=VARCHAR} != null">

and c.state = #{state}

</if>

<if test="#{createdSince,javaType=java.sql.Date,jdbcType=DATE} != null">

and created\_at > #{createdSince}

</if>

</where>

\*}

]]>

</select>

The example runs the SQL twice with different parameters to show different sections being activated for each search.

The SQL select can be run using the select(Integer minPurchases, String state, java.sql.Date createdSince) Java method on the DAO SearchedClient.

## Case #2: Update using a dynamic criteria

The SQL update is extended using XML tags that define sections that are dynamically included or excluded depending on the runtime parameter values. The configuration file includes the following section:

<query java-method-name=*"applyDiscountToVehicles"*>

<![CDATA[

update vehicle set

list\_price = list\_price - #{discount,javaType=java.lang.Integer,jdbcType=NUMERIC}

<where>

<if test="#{unsold,javaType=java.lang.Boolean,jdbcType=BOOLEAN} != null">

and not sold

</if>

<if test="#{minMileage,javaType=java.lang.Integer,jdbcType=NUMERIC} != null">

and mileage > #{minMileage}

</if>

<if test="#{type,javaType=java.lang.String,jdbcType=VARCHAR} != null">

and type = #{type}

</if>

</where>

]]>

</query>

The example runs the SQL twice with different parameters to show different sections being activated for the update.

The SQL update can be run using the applyDiscountToVehicle(Integer discount, Boolean unsold, Integer mileage, String type) Java method on the DAO GeneralOperations.