How to execute a SQL Query from inside Corticon

Mike Parish

The Use Case

Enterprise Data Connector provides Corticon with an easy way for rules to connect to a database.
It can either automatically retrieve records that are required by the rules (at the point where they are needed) or you can control when Corticon reads records by using the "Extend to Database" option.

In most cases EDC is the simplest approach. It requires no knowledge of SQL or databases to be able to use it.

However sometimes you have rule authors who are also skilled SQL programmers who would like to be able to write their own SQL queries (e. SELECT BALANCE FROM ACCOUNT WHERE ACCTNO=12345)

Can this be done in Corticon?

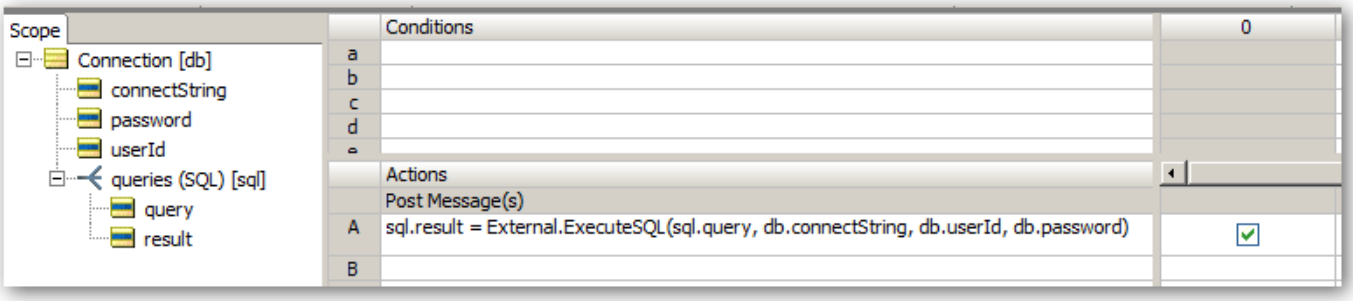
Yes, Of Course!

Restrictions

It does require that you write a generic extended operator that can submit the SQL query to the database on behalf of Corticon. So you will need to know java to do that.
An extended operator can only return a single value (String, Integer, Decimal, Boolean, Date) so you must ensure that your SQL is properly constructed.
If you want to return a set of rows then you will need to do it with a service call out (or just use the standard EDC mechanism that's already built in to Corticon)

Rule Sheet

Here's an example of what it might look like in a rule sheet



Sample Test Results

- **Connection [1]**
  - connectString [jdbc:sqlserver://127.0.0.1:1433;databaseName=Test]
  - password [Password1]
  - userId [sa]
  - **queries (SQL) [5]**
    - query [SELECT sum(price) FROM Test.dbo.THING ;]
    - **result [50.000000]**
  - **queries (SQL) [6]**
    - query [SELECT AVG(price) FROM Test.dbo.THING ;]
    - **result [12.500000]**
  - **queries (SQL) [7]**
    - query [SELECT max(price) FROM Test.dbo.THING ;]
    - **result [20.000000]**
  - **queries (SQL) [8]**
    - query [SELECT stdev(price) FROM Test.dbo.THING ;]
    - **result [5.0]**
  - **queries (SQL) [9]**
    - query [SELECT COUNT(*) FROM Test.dbo.THING WHERE COLOR = 'green' ;]
    - **result [4]**
  - **queries (SQL) [10]**
    - query [SELECT COUNT(*) FROM Test.dbo.THING WHERE price < (select avg(price) from Test.dbo.THING);]
    - **result [3]**

On this table

name	color	price	shape	size
Thing1	green	10.000000	oval	small
Thing2	green	20.000000	triangle	small
Thing3	green	10.000000	oval	large
Thing4	green	10.000000	round	large

Here's the code that does this:

```
package com.corticon.database;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Database {
    public Database() {}
    public ResultSet dbConnect(String SQLQuery, String db_connect_string, String db_userid,
                               String db_password) throws Exception {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection connection = DriverManager.getConnection(db_connect_string, db_userid, db_password);
        Statement statement = connection.createStatement();
        ResultSet rs = statement.executeQuery(SQLQuery);
        return rs;
        }
}

And

import java.io.File;
import java.io.FileWriter;
import java.sql.ResultSet;
import com.corticon.database.Database;
import com.corticon.services.extensions.ICcStandAloneExtension;

public class External implements ICcStandAloneExtension {
   public static String ExecuteSQL(String SQLQuery, String connectString,String userid, String password) {
        Database db = new Database();
        try {
                ResultSet rs = (db.dbConnect(SQLQuery, connectString, userid,password));
                if (rs.next()) {
                        String answer = rs.getString(1);
                        return answer;
                } else {
                        return "No results";
                }
                } catch (Exception e) {
                        e.printStackTrace();
                return "Failed: " + e.getMessage();
            }
        }
}
```

NOTE: When you build the plugin for the extended operators just make sure that sqljdbc4.jar is included in the manifest for the plugin.

You will need to read the Corticon documentation of creating Extended Operators (or our Professional Services Team can do it for you)

DISCLAIMER: I'm not a java programmer so this example is just for illustration purposes.