

Review of SwisSQL SQLOne Console, SQL Query Converter

Narayana Vyas Kondreddi

Microsoft Most Valuable Professional (SQL Server)

Feb 9, 2005

Available Online at http://vyaskn.tripod.com/review_swissql_sqlone_console.htm

Recently, SwisSQL approached me for a review of their product that can convert SQL statements from one dialect to another. They told me that "SwisSQL SQLOne Console" is a GUI application that can convert SQL statements from one RDBMS implementation to another and the dialects supported include:

Microsoft SQL Server

Sybase

Oracle

IBM DB2

Informix

MySQL

PostgreSQL

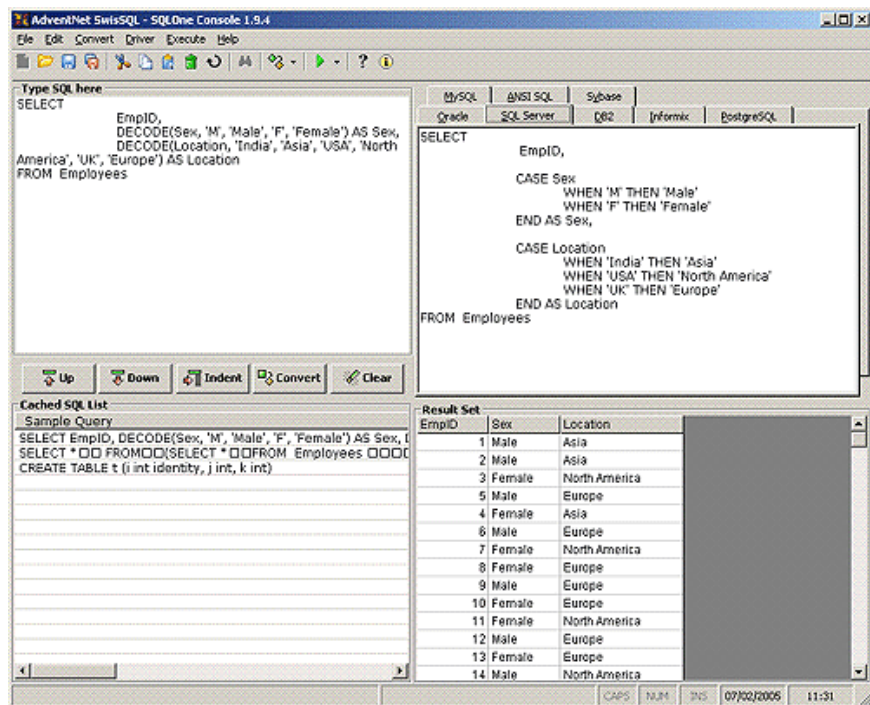
MySQL

and last but not the least, ANSI SQL

This sounded too good to be true, so I decided to review the product.

I went ahead and downloaded the fully functional version (with the limitation of 50 SQL query conversions in the 30 day evaluation period) of SwisSQL - SQLOne Console (Visual Basic) 1.9.4 from their website. Before downloading the product, you are required to fill-in a form with some basic details like your name, email, phone, company etc. The product itself is downloadable as a .Zip file. Installation can be started by extracting the contents of the .Zip file and running the setup.exe file. Installation is quite simple, but you do get some dialog boxes about replacing some of your existing files that are more recent than the ones in the installation package. Always choose to keep your existing and recent files. One of the requirements of this application is that, you should have .Net framework 1.1 installed on your system. My installation went fairly smooth. But when I started the application from the Start menu, I got an error message about the environmental variable 'path' not containing the path to the .Net framework. I fixed it by appending the .Net framework path to the path environmental variable, and the application started fine.

It turned out to be a very simple, single screen, standard application. You can see this for yourself in the below image:



The application window consists of four main areas:

- ❖ The top left hand side frame consists of a free text area where you can type in your SQL queries that you want to convert to other dialects of SQL.
- ❖ The top right hand side tabbed dialog box lets you switch from one dialect of SQL to another. That is, if you type in a T-SQL query into the top left text box, and click convert, the equivalent MySQL, ANSI SQL, Sybase, PL/SQL, DB2, Informix and PostgreSQL queries can be seen in this tabbed dialog box. Good thing is, by clicking the 'Convert' button only once, your SQL query is converted into all supported dialects in one go.
- ❖ The bottom left hand side list box can be used to cache your frequently converted SQL statements. This is a handy feature, as you can store your frequently used SQL commands in this list box, and move them in to the top left text box for conversion, on demand, by pressing the 'up' and 'down' buttons.
- ❖ The bottom right hand side grid shows you the results of query. This application can connect to the supported data sources, and you can run your converted queries against the data source, and check your results conveniently.

Before we actually start using the product, let's see where this product can be useful. This automated conversion tool can save huge amounts of man hours, when migrating a database application from one RDBMS platform to another. For example, think about migrating an Oracle database application to Microsoft SQL Server. It is going to be a humongous task migrating the SQL queries, as the PL/SQL and T-SQL implementations are vastly different. Another scenario where I think such a utility can add great value is, if you are an Independent Software Vendor (ISV), that supports different database backends. In this case, an automated SQL conversion utility can cut down the development time vastly.

I started off my review with a simple and very common test, that retrieves the current time from the SQL Server, and the SQL Query is: `SELECT GETDATE()`. When I clicked on the "Convert" button, the tool came up with the following queries for each of the supported database dialects. You'll be amazed to see how a simple query like this one can be so different, in different database management systems. Here are the results:

MySQL: SELECT (CURRENT_TIMESTAMP)

ANSI SQL: SELECT (CURRENT_TIMESTAMP)

Sybase: SELECT (CURRENT_TIMESTAMP)

Oracle: SELECT (SYSDATE) FROM SYS.DUAL

DB2: SELECT (CURRENT_TIMESTAMP) FROM SYSIBM.SYSDUMMY1 FETCH FIRST 1 ROW ONLY

Informix: SELECT FIRST 1 (CURRENT) FROM SYSTABLES

PostgreSQL: SELECT (CURRENT_TIMESTAMP)

Next I decided to find out how this tool translates the Oracle's rownum functionality (that let's Oracle users access individual rows from a resultset using a row number) to other databases. I converted the following query:

```
SELECT *
FROM
(
  SELECT *
  FROM Employees
  ORDER BY EmployeeID DESC
)
WHERE ROWNUM < 2
```

I got the following results for various database platforms:

SQL Server:

```
SELECT TOP 1 *
FROM
(
  SELECT *
  FROM Employees
  ORDER BY EmployeeID DESC
) AdventNet_ALIAS1
```

DB2:

```
SELECT *
FROM
(
  SELECT *
  FROM Employees
  ORDER BY EmployeeID DESC
) AdventNet_ALIAS1
FETCH FIRST 1 ROWS ONLY
```

Informix:

```
SELECT FIRST 1 *
FROM
(
  SELECT *
  FROM Employees
  ORDER BY EmployeeID DESC
) AdventNet_ALIAS1
```

PostgreSQL:

```
SELECT *
FROM (SELECT *
FROM Employees
ORDER BY EmployeeID DESC
) AdventNet_ALIAS1
LIMIT 1
```

Next, I decided to test how the SQL Server's proprietary implementation of IDENTITY values can be translated to other database platforms. I converted the following SQL Server CREATE TABLE statement:

```
CREATE TABLE Orders
(
  OrderID int IDENTITY(1, 1) PRIMARY KEY NOT NULL,
  CustomerID int NOT NULL,
  OrderDate datetime NOT NULL,
  OrderAmount money NOT NULL
)
```

..and got the following results:

Oracle:

```
CREATE SEQUENCE Orders_OrderID_SEQ
START WITH 1
INCREMENT BY 1
```

```
CREATE TABLE Orders
(
  OrderID int PRIMARY KEY ,
  CustomerID int NOT NULL ,
  OrderDate DATE NOT NULL ,
  OrderAmount DECIMAL (19, 4) NOT NULL
)
```

DB2:

```
CREATE TABLE Orders
(
  OrderID int GENERATED BY DEFAULT AS IDENTITY(START WITH 1 INCREMENT BY
1) PRIMARY KEY NOT NULL,
  CustomerID int NOT NULL ,
  OrderDate TIMESTAMP NOT NULL ,
  OrderAmount DECIMAL (19, 4) NOT NULL
)
```

MySQL:

```
CREATE TABLE Orders
(
  OrderID int AUTO_INCREMENT PRIMARY KEY NOT NULL,
  CustomerID int NOT NULL ,
  OrderDate datetime NOT NULL ,
  OrderAmount DECIMAL (19, 4) NOT NULL
)
```

Then I checked how Oracle's DUAL table functionality can be implemented in various other RDBMSes, and here are the results, when I converted the following Oracle's PL/SQL: `SELECT 'Operation Succeeded' FROM Dual`

SQL Server:

```
SELECT 'Operation Succeeded'
```

DB2:

```
SELECT 'Operation Succeeded'
FROM SYSIBM.SYSDUMMY1
FETCH FIRST 1 ROWS ONLY
```

Informix:

```
SELECT FIRST 1 'Operation Succeeded'  
FROM SYSTABLES
```

So far so good. When I tried to convert Oracle's proprietary START WITH...CONNECT BY (for processing hierarchical data) syntax to SQL Server's T-SQL, the tool couldn't do it. It is understandable, as there's no equivalent syntax in T-SQL, but the same can be achieved by using a multi-step procedural code and SQLOne console didn't go that length. But it did convert the query successfully into Informix and DB2.

Next I tried to convert another commonly used Oracle function DECODE() into other platforms. I converted the following query:

```
SELECT DBMS_Code,  
decode  
(  
DBMS_Code, 'Ora', 'Oracle',  
'Syb', 'Sybase',  
'mssql', 'Microsoft SQL Server'  
)  
FROM DBMS_Table
```

...and I got the correct result for SQL Server, as shown below:

```
SELECT  
DBMS_Code,  
CASE DBMS_Code  
WHEN 'Ora' THEN 'Oracle'  
WHEN 'Syb' THEN 'Sybase'  
WHEN 'mssql' THEN 'Microsoft SQL Server'  
END  
FROM DBMS_Table
```

Here's how the application looks like in action:

The screenshot shows the AdventNet SwissSQL - SQLOne Console 1.9.4 interface. The main window is divided into several sections:

- Type SQL here:** Contains the original Oracle query:

```
SELECT EmpID,  
DECODE(Sex, 'M', 'Male', 'F', 'Female') AS Sex,  
DECODE(Location, 'India', 'Asia', 'USA', 'North  
America', 'UK', 'Europe') AS Location  
FROM Employees
```
- Convert:** A button that has been used to convert the query.
- Result Set:** Displays the converted SQL Server query:

```
SELECT  
EmpID,  
CASE Sex  
WHEN 'M' THEN 'Male'  
WHEN 'F' THEN 'Female'  
END AS Sex,  
CASE Location  
WHEN 'India' THEN 'Asia'  
WHEN 'USA' THEN 'North America'  
WHEN 'UK' THEN 'Europe'  
END AS Location  
FROM Employees
```
- Result Set Table:** A table showing the output of the converted query. The table has three columns: EmpID, Sex, and Location. The data is as follows:

EmpID	Sex	Location
1	Male	Asia
2	Male	Asia
3	Female	North America
5	Male	Europe
4	Female	Asia
6	Male	Europe
7	Female	North America
8	Female	Europe
9	Male	Europe
10	Female	Europe
11	Female	North America
12	Male	Europe
13	Female	Europe
14	Male	North America
- Cache SQL List:** A list of saved queries, including a sample query and a table creation statement:

```
CREATE TABLE t (i int identity, j int, k int)
```

The interface also includes a menu bar (File, Edit, Convert, Driver, Execute, Help) and a toolbar with various icons for file operations and execution. The status bar at the bottom shows the date and time: 07/02/2005 11:31.

So! What can I say? AdventNet's SwisSQL SQLOne Console is doing what it says on the tin. Very impressive. I'd definitely recommend this product to anyone involved in migrating databases between different platforms, or to those developing database independent applications. It is also useful for DBAs and developers trying to learn a new RDBMS. And it comes in very handy for a DBA/developer who needs to support multiple database platforms.

Before I conclude this review, here are the other features that are worth mentioning: The ability to load files containing SQL commands and convert them from one SQL dialect to another. You can also execute the converted SQL against a database of your choice, using an ODBC DSN or using a connection string, for validation purposes. The 'Indent' button can be used for formatting and proper casing the input SQL commands, and the output always comes out properly formatted. If you input an SQL command that cannot be converted into some of the supported dialects, you will see an error for those database dialects, but the tool will successfully convert the SQL to the rest of the SQL dialects, where a proper conversion is possible.

I am not saying that this product will convert SQL 100% accurately. It won't. I did encounter some cases, where the output SQL needed slight changes and corrections. So, it is not eliminating a DBA/programmer intervention completely, but it is reducing it by about 90%, I'd say. You'll need someone to check the output SQL and make sure it is produced correctly. But the effort and cost involved in this is significantly smaller than a completely manual database migration.

If you are interested in any of the migration products offered by AdventNet, checkout their site at: <http://www.swissql.com>. You can also try out an online conversion utility on their site at: <http://www.adventnet.info/license/commonSql/jsp/index.jsp>. Also check out [SwisSQL - Oracle to SQL Server Edition 2.6](#) if you are specifically interested in Oracle to SQL Server migration.

The SQLOne console review is also available online at:
http://vyaskn.tripod.com/review_swissql_sqlone_console.htm

Copyright © 1997 - 2005 Narayana Vyas Kondreddi. All rights reserved.