Beginning SQL, Differences Between Oracle and Microsoft

If you're new to SQL or just new to Oracle SQL, perhaps coming from a Microsoft SQL Server environment, it may seem like the two versions should be very similar, and they are, to a certain degree, but they are also very different in some important and basic ways.

You may need to know these differences because of a migration effort from one to the other, or because you need to access both of them in your day-to-day operations. Perhaps you have an Oracle server being downloaded into a SQL Server data warehouse, or perhaps you have distributed SQL Server databases being uploade into a consolidating Oracle database. For these and other circumstances under which these two exist, you may need to be aware of the differences between the two versions of the SQL language.

So what are the differences from SQL Server to Oracle?

Part I. A Quick Intro for the SQL Server User

Don't Use Databases

Well, first of all, we don't use databases, we connect to them:

```
SQL Server

  use mydatabase

Oracle

   connect mydatabase/mypassword
```

Use Dual

And then, our select statements have different options, in this instance requiring a from clause:

```
SQL Server

   select getdate();

Oracle

   select getdate() from dual;
```

so we use that dummy we call DUAL. Did you notice the lack of a from clause in the first version? It's a nice shortcut, but Oracle doesn't allow it, nor does ANSI SQL92.

Select Into

And we don't select rows into a table, but instead, insert the rows by selecting

them:

```
SQL Server

   select getdate() mycolumn
   into mytable;

Oracle

   insert into mytable
   select sysdate() mycolumn from dual;
```

Actually, the SQL Server version creates a table if one doesn't exist, so the Oracle version would require a `CREATE TABLE AS` statement to arrive at the same result.

Inserts

By the way, the into clause of an insert statement is not an option on Oracle; it's required:

```
SQL Server
   insert mytable values('more text');

Oracle
   insert into mytable values('more text');
```

Updates

What about updates? Well, these are different too, and may have to be rewritten entirely to replace the from clause used to get data from one or more tables:

```
SQL Server

   update mytable
   set mycolumn=myothertable.mycolumn
   from mytable,myothertable
   where mytable.mycolumn like 'MY%'
   and myothertable.myothercolumn='some text';

Oracle

   update mytable
   set mycolumn=
   (select a.mycolumn
      from myothertable a
     where myothertable.myothercolumn='some text';
   )
   where mytable.mycolumn like 'MY%';
```

Deletes

And finally, the delete requires a `FROM` clause in Oracle:

```
SQL Server

  delete mytable where mycolumn like 'some%';

Oracle

  delete from mytable where mycolumn like 'some%';
```

which we always try to double-check in either case. Notice that we used a link for thattable on thatdb, which is considered to be a remote database.

Vendor Programs

So where is all this taking place? What programs are we running on Oracle?

Well, in place of iSQL, we're using SQL*Plus to enter our statements, and in place of the Northwind examples, we use Scott's tiger:

```
SQL Server

   command-line-prompt:isql
      or, for queries developed in SQL Analyzer:
   command-line-prompt: osql

   use northwind

Oracle

   command-line-prompt:sqlplus
   scott/tiger
```

Notice that we didn't have to use the connect statement because it's automatic when you first login.

Now that we're logged in, we can query Scott's infamous tables, EMP and DEPT, to execute the examples we find in various reference materials such as the SQL*Plus User's Guide and others.

And now, we're off on our own to pursue our education in Oracle SQL further.

Welcome aboard!

Part II. A Little More Detail

Outer Join

Now, where would we be without the outer join? Missing data, that's where. So, here's an example of that query in dialects:

SQL Server

```
Select d.deptname,e.empname
from dept d, emp e
WHERE d.empno *= e.enum;
```

Oracle

```
Select d.deptname,e.empname
from dept d, emp e
WHERE d.empno = e.enum (+);
```

Notice the slight syntactic difference shown in this example from Scott's beloved EMP and DEPT tables. This may seem like completely opposite forms of expressing this statement, nevertheless all departments are listed even though some have no employees.

Sub-queries in Place of Columns

Another SQL Server extension over both SQL92 and Oracle, is the use of sub-queries wherever a column name is allowed. Here, the quarterly sales columns are returned by a sub-query on the sales table to produce a single row listing all four quarters results for the year:

SQL Server

```
select distinct year,
q1 = (select Amount amt FROM sales
where Quarter=1 AND year = s.year),
q2 = (SELECT Amount amt FROM sales
where Quarter=2 AND year = s.year),
q3 = (SELECT Amount amt FROM sales
where Quarter=3 AND year = s.year),
q4 = (SELECT Amount amt FROM sales
where Quarter=4 AND year = s.year)
from sales s;
```

Oracle

```
SELECT year,
DECODE( quarter, 1, amount, 0 ) q1,
DECODE( quarter, 2, amount, 0 ) q2,
DECODE( quarter, 3, amount, 0 ) q3,
DECODE( quarter, 4, amount, 0 ) q4
FROM sales s;
```

The same one-line result is produced by the decode function, which is reported by Oracle to be faster than the sub-queries and actually looks like a simpler swatch of code than the select statements.

Deletes With Second From Clause

Deleting rows from one table conditionally based on the contents of rows in another table can be expressed with a statement containing two from clauses:

SQL Server

```
   delete
   from products
   from products, product_deletes
   where products.a = product_deletes.a
   and products.b = product_deletes.b
   and product_deletes.c = 'd';

Oracle

   delete
   from products
   where ( a, b ) in
   ( select a, b
     from product_deletes
     where c = 'd' );
```

This can be rewritten to a statement using a single FROM clause, even if there is a multi-column join, with a sub-query to produce the same effect of deleting only those rows marked for the purpose in the other table.

Part III. More Depth

The Connect Concept

SQL Server provides connection to a server which allows access to multiple databases, while Oracle's Server provides access to one database with multiple users and roles, so a database is roughly equivalent to a tablespace, user, schema and role. One can change roles or connect as a different user, but the one server, one database concept remains.

For all the similarities between the two SQL versions, there are a few key conceptual differences:

A SQL Server:	is an Oracle:
Database owner, DBO	Schema
Group/Role	Role
Non-unique Index	Index
Transact SQL stored procedure	PL/SQL procedure
T-SQL stored procedure	PL/SQL function
Trigger	BEFORE trigger
Complex rule	AFTER trigger
Column identity property	Sequence

And a few that are only available in Oracle:

Clusters
Packages
Triggers for each row
Synonyms
Snapshots

Data Type Differences

Here's a summary of the datatype differences between the two versions:

SQL Server	Oracle
INTEGER	NUMBER(10)
SMALLINT	NUMBER(6)
TINYINT	NUMBER(3)
REAL	FLOAT
FLOAT	FLOAT
BIT	NUMBER(1)
VARCHAR(n)	VARCHAR2(n)
TEXT	CLOB
IMAGE	BLOB
BINARY(n)	RAW(n) or BLOB
VARBINARY	RAW(n) or BLOB
DATETIME	DATE
SMALL-DATETIME	DATE
MONEY	NUMBER(19,4)
NCHAR(n)	CHAR(n*2)
NVARCHAR(n)	VARCHAR(n*2)
SMALLMONEY	NUMBER(10,4)
TIMESTAMP	NUMBER
SYSNAME	VARCHAR2(30), VARCHAR2(128)

As you may imagine, there are also differences in the concepts of data storage, such as page versus data block, but our purposes are limited to SQL.

Time

Oracle's default time storage in the date datatype resolves down to the second, while SQL Server's DATETIME datatype will store to the 1/300th second, but the new Oracle TIMESTAMP datatype will store 1/100 millionth of a second in accuracy, if one remembers to use it instead of the default type. See the Migration Guide for an extended example on this subject.

Alias

A column alias is useful sometimes in cutting down the clutter in an SQL statement:

```
SQL Server

  select a=deptid,b=deptname,c=empno from dept;

Oracle
  select deptid a, deptname b, empno c from dept;
```

One can think of these as being the reverse of each other, as a memory aid, with the SQL Server version coming before the the column name and the Oracle version coming after it.

Sub-queries

SQL Server

```
SELECT ename, deptname
FROM emp, dept
WHERE emp.enum = 10
AND(SELECT security_code
FROM employee_security
WHERE empno = emp.enum) =
    (SELECT security_code
    FROM security_master
    WHERE sec_level = dept.sec_level);
```

Oracle

```
SELECT empname, deptname
FROM emp, dept
WHERE emp.empno = 10
AND EXISTS (SELECT security_code
FROM employee_security es
WHERE es.empno = emp.empno
AND es.security_code =
     (SELECT security_code
     FROM security_master
     WHERE sec_level =
     dept.sec_level));
```

Both versions of SQL support multiple subqueries, but with differing syntax. The select in place of a column name in SQL Server can produce the same effect as the query within the subquery in Oracle, which is the version supported by SQL92.

Part III: Something New

With the recent update of Oracle SQL to support the use of regular expressions, the expressive power of simple queries is greatly expanded. New features have been introduced for this purpose, such as the operator `REGEXP_LIKE` and the functions `REGEXP_INSTR`, `REGEXP_SUBSTR`, and `REGEXP_REPLACE`.

We can now write queries for non-digit zipcodes in one, short statement:

```
select zip
  from zipcode
 where regexp_like (zip, '[^[:digit:]]')
```

This one shows the starting column of both the 5 and 9 digit

zip code:

```
SELECT REGEXP_INSTR('Joe Smith, 10045 Berry Lane, San Joseph, CA
91234-1234',
       ' [[:digit:]]{5}(-[[:digit:]]{4})?$')
    AS starts_at
  FROM dual
```

Further examples can be found in the excellent article, on OTN, written by Alice Rischert.

Summary

This discussion has been an attempt at a light and lively introduction to the Oracle database world for those familiar with the Microsoft SQL Server database products. Much more in-depth examples are available in the references shown that follow, from which many of the examples were drawn and for which we can thank the authors involved.

References

(1) Oracle Migration Workbench Reference Guide for SQL Server and Sybase Adaptive Server Migrations, Release 9.2.0 for Microsoft Windows 98/2000/NT and Microsoft Windows XP, Part Number B10254-01

(2) Microsoft Transact-SQL Reference:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_tsqlcon_6lyk.asp

(4) Oracle Technology Network, OTN:

http://otn.oracle.com/software/index.html

As Oracle puts it:

"All software downloads are free, and each comes with a development license that allows you to use full versions of the products only while developing and prototyping your applications. You can buy Oracle products

with full-use licenses at any time from the online Oracle Store or from your Oracle sales representative."

(5) Writing Better SQL Using Regular Expressions, By Alice Rischert

http://otn.oracle.com/oramag/webcolumns/2003/techarticles/rischert_regexp_pt1.html