ORACLE 10G/9I FOR DEVELOPERS: WHAT YOU NEED TO KNOW

John Jay King, King Training Resources

INTRODUCTION

The many new features of Oracle10g and Oracle9i extend the capabilities of the database in many ways. Attendees will be introduced to the new and improved features of Oracle that directly impact application development. Special emphasis is placed on features that can reduce development time, make development simpler, improve performance, or speed deployment. The presentation focuses on features added to Oracle10g and Oracle9i that enable improved application development. Specific topics include: Model, Regular Expressions, CASE, Outer Join, XML, Object improvements, and expanded Java capabilities. Attendees will be better equipped to create new applications or modify existing applications to take full advantage of the Oracle database.

Features added or improved with Oracle10g are marked in these notes to indicate that they are Oracle10g-specific, all other features noted work in both Oracle10g and Oracle9i. Oracle 10g Release 2 opens up new possibilities which will be examined. Examples of some new features are provided during this presentation, however, you should see the Oracle SQL Reference, Oracle PL/SQL Users Guide and Reference, and other relevant Oracle manuals for complete documentation.

Oracle provides rich tools with new features extending the capabilities of the database in many ways. This paper focuses on those improvements and additions to Oracle10g and Oracle9i likely to have the most impact on application developers. This paper assumes a working knowledge of Oracle, SQL, and PL/SQL (from an application developer's perspective). No prior knowledge of Oracle10g or Oracle9i is assumed.

DEVELOPER-ORIENTED FEATURES

Specific developer-oriented features include: Model, Regular Expressions, new data types, statistical functions, multi-table insert, new analytic functions, timezone and timestamp capabilities, direct XML support, ISO/ANSI-standard join syntax, ISO/ANSI-standard CASE syntax, support for object inheritance, synchronization of SQL engine with PL/SQL, external tables allow easy access to non-relational data, and various syntax modifications. SQL and PL/SQL syntax changes are discussed later in this paper.

DBA-ORIENTED FEATURES

Oracle is trying to automate as much of the DBA function as it can. Marketing literature for Oracle9i is full of phrases like "self-tuning" and "automatic management" implying that the DBA role is diminishing. In fact, all of the new "automatic" features add to a DBA's responsibilities in that they must now choose what to or not to let happen automatically, and, they must double-check to make sure that automatic processes are not causing problems.

For years a "black-art" of DBAs has been the management of rollback segments. The new Automatic Undo Management replace standard rollback segments with System Managed Undo (SMU), Oracle9i will create and manage its own rollback information.

Buffer pools may now be resized without cycling the database. Oracle Stand-by Database has been renamed to DataGuard. Flashback query uses the LogMiner to allow a user to refer to (or revert to) the values stored at a specific date and time. Ultra search performs a web-crawler examination to find things in the database. Bitmap join indexes improve the capabilities already provided by bitmaps. The LogMiner has been improved to support Flashback query and provide other meaningful data.

Oracle RAC (Real Application Clusters) provides parallel-processing capabilities to be applied without having to modify applications. Block sizes may vary in a database. Tables and indexes may be modified or reorganized dynamically without significant impact on uptime ("online data evolution").

Oracle Net is the new name for Net8/SQL*Net and the reliance of Oracle Names over tnsnames use is further emphasized.

Primary key and index key indexes may be completely defined (and named) as part of CREATE TABLE or ALTER TABLE.

DBMS_REDEFINITION package provides the ability to dynamically redefine database tables and/or indexes.

PL/SQL native compilation will greatly improve the speed of database code. Cost-based optimization has been improved. A Connection Manager has been added to support greater numbers of concurrent users

New Datatypes

Oracle9i provides a series of new and improved datatypes. New date-related datatypes include: TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE, TIMEZONE_HOUR, TIMEZONE_MINUTE, TIMEZONE_REGION, INTERVAL YEAR TO MONTH, and INTERVAL DAY TO SECOND. New Oracle-supplied datatypes include: SYS.ANYTYPE, SYS.ANYDATA, SYS.ANYDATASET, XMLType, URIType, DBURIType, HTTPURIType, URIFactoryType, MDSYS.SDO_GEOMETRY, ORDSYS.ORDAudio, ORDSYS.ORDImage, and ORDSYS.ORDVideo. Changes have been made to CHAR, VARCHAR2, NCHAR, and NVARCHAR2 datatypes to facilitate multi-byte definitions. UROWID has been added for IOT rows.

New Date Datatypes

Timestamps include Year, Month, Day, Hour, Minute, Second, and fractions of seconds (0-9 decimals, 6 is the default).

TIMESTAMP or TIMESTAMP (n)

TIMESTAMP WITH TIMEZONE or TIMESTAMP (n) WITH TIMEZONE

TIMESTAMP WITH LOCAL TIMEZONE

TIMESTAMP (n) WITH LOCAL TIMEZONE

Intervals provide a mechanism for storing periods of time. INTERVAL YEAR TO MONTH allows specification of the number of digits in the year (0-9, 2 is the default). INTERVAL DAY TO SECOND allows specification of either (or both) the digits used for days (0-9, 2 is the default) or for fractional seconds (0-9, 6 is the default).

INTERVAL YEAR TO MONTH or INTERVAL YEAR (n) TO MONTH

INTERVAL DAY TO SECOND or INTERVAL DAY(d) TO SECOND (s)

Modifications to CHAR, VARCHAR2, NCHAR, and NVARCHAR2 allow specification of BYTE or CHAR (CHAR is the default). BYTE specifies that the size of the column is specified in bytes and CHAR specifies that the size of the column is specified in characters.

Oracle-Supplied "Any" Types

Oracle supplies a datatype for use in creating tables and stored procedures when the actual type is not known, the so-called "any" types. SYS.ANYTYPE may contain any known SQL datatype or an unnamed datatype. SYS.ANYDATA may contain different types of data in columns of different rows. SYS.ANYDATASET allows sets of data to be passed.

XML Datatype

XMLType is an Oracle-defined datatype used to store XML data within the database (actually a hidden CLOB underneath). The XMLType includes many member functions including: createXML() to create an XMLType instance, existsNode() checks if XPath can find any valid nodes, extract() uses XPath to return a fragment as XMLType data, isFragment() checks to see if a document is really a fragment, getClobVal() gets document contents as

a CLOB, getStringVal() gets values as a string, and getNumberVal() gets numeric value as a number. A great deal of XML support was added in Oracle9i. Check the reference manual "Application Developer's Guide – XML" for more information.

URI DATATYPES AND MISCELLANEOUS ORACLE-SUPPLIED DATATYPES

Oracle9i provides a series of predefined types designed to help programmers work with web applications including: SYS.URITYPE, SYS.DBURITYPE, SYS.HTTPURITYPE, and SYS.URIFACTORYTYPE. Oracle9i also supplies a set of predefined types for working with multimedia: ORDSYS.ORDAUDIO, ORDSYS.ORDIMAGE, and ORDSYS.ORDVIDEO. Finally, Oracle9i provides a spatial datatype called MDSYS.SDO_GEOMETRY.

NEW SYNTAX

Oracle9i includes new syntax options, new functions, and improvements to other features that have been around for years.

SQL: ISO/ANSI JOIN, CASE, ETC…

In the early years of the ISO/ANSI standards Oracle conformed more closely to the ISO/ANSI standards than other vendors did - a fact they were not shy about publicizing. Over the past few years they have hunkered down and left adoption of newer ISO/ANSI standards to other products. Oracle9i adds support for some of the more-common ISO/ANSI constructs making applications less database dependent. Specifically, Oracle9i adopts the ISO/ANSI JOIN syntax, expands on Oracle8i's CASE syntax, and includes support for some standard SQL functions previously not available.

Oracle was the first (and for a long time the only) database vendor to supply an outer-join construct (+). When the ISO/ANSI standard adopted a more complex and useful syntax, Oracle opted out for several years. Oracle9i includes support for INNER, CROSS, NATURAL, and OUTER JOIN syntax. The OUTER JOIN syntax provides RIGHT, LEFT, and FULL JOIN semantics.

Cross Join are intended to be self-documenting Cartesian Products:

```
select ename,dname
           from emp cross join dept
```

It turns out that the Cross Join indicates a join of the same type as when comma-delimited, allowing specification of join conditions in the WHERE clause:

```
select ename,dname
  from emp cross join dept
  where emp.deptno = dept.deptno
```

Natural joins indicate an equi-join where the database automatically chooses join criteria using any column names that happen to match between the two tables. Natural joins may also specify ISO/ANSI join types (INNER, LEFT, RIGHT, FULL; discussed later…). Additional criteria may be specified using the WHERE clause.

```
select ename,dname
  from emp natural join dept
```

Comma-delimited join syntax combines join criteria with non-join criteria in the WHERE clause. The new INNER and OUTER (LEFT, RIGHT, FULL) join syntax separates join criteria from non-join tests.

The older syntax names all tables in comma-delimited form and uses the WHERE clause to name join criteria. In the example below join criteria is mixed with row selection criteria.

```
select distinct nvl(dname,'No Dept'),count(empno) nbr_emps
      from many_emps emp,many_depts dept
      where emp.deptno = dept.deptno
       and emp.job in ('MANAGER','SALESMAN','ANALYST')
            group by dname;
```

To use ISO/ANSI Inner Join semantics, use the term INNER JOIN (or simply JOIN) between the table(s) involved and specify one-or-more join criteria with the ON/USING clause. Correlation (alias) table names may be specified in the same manner as with comma-delimited joins. With the new syntax the WHERE clause names only non-join criteria.

```
      select distinct nvl(dname,'No Dept'),count(empno) nbr_emps
            from many_emps emp join many_depts dept
            on emp.deptno = dept.deptno
            where emp.job in ('MANAGER','SALESMAN','ANALYST')
            group by dname;
```

The new join semantics allow the use of parentheses to control the order in which joins occur.

A problem with inner join (old or new syntax) is that only rows that match between tables are returned. Oracle invented the first syntax for solving the outer join issue years ago. This is the "(+)" notation used on side of the join criteria WHERE clause where null rows are to be created to match the other table.

```
select distinct nvl(dname,'No Dept'),count(empno) nbr_emps
            from many_emps emp,many_depts dept
            where emp.deptno(+) = dept.deptno
            group by dname;
```

The new ISO/ANSI join syntax provides three separate capabilities: LEFT, RIGHT, and FULL OUTER JOIN (the word OUTER is redundant and usually omitted). With the new syntax, LEFT and RIGHT indicate which side of the join represents the complete set, the opposite side is where null rows will be created. The example below solves the same problem as the Oracle Outer Join operator example above:

```
select distinct nvl(dname,'No Dept'),count(empno) nbr_emps
            from many_emps emp right join many_depts dept
                  on emp.deptno = dept.deptno
            group by dname;
```

To cause SQL to generate rows on both sides of the join required a UNION using the old Oracle Outer Join operator syntax as illustrated below:

```
select nvl(dname,'No Dept') deptname,count(empno) nbr_emps
            from many_emps emp,many_depts dept
            where emp.deptno(+) = dept.deptno
            group by dname
union
select nvl(dname,'No Dept') deptname,count(empno) nbr_emps
            from many_emps emp,many_depts dept
            where emp.deptno = dept.deptno(+)
            group by dname;
```

The solution using UNION works, unfortunately, it does not conform to the ISO/ANSI standard. Using UNION also means that SQL developers must be sure that the queries involved match properly.

The new ISO/ANSI Outer Join mechanism is simpler to code. To cause rows to be created on either side of a Join as required to align the two tables use the FULL OUTER JOIN (FULL JOIN) syntax.

```
select distinct nvl(dname,'No Dept') deptname,count(empno) nbr_emps
            from many_emps emp full join many_depts dept
                  on emp.deptno = dept.deptno
            group by dname;
```

CASE

For years, one of the best-kept secrets of Oracle was the DECODE function. DECODE is a powerful and useful tool that Oracle first started providing many years ago. Beginning with Oracle8i and expanding in Oracle9i, Oracle has added the CASE expression to allow more complex processing than DECODE (ANSI/ISO standard). CASE allows IF…THEN…ELSE logic to be placed anywhere in SQL that a column or literal can go. The first style of CASE statement is as follows:

```
CASE  WHEN condition1   THEN expression1
      WHEN condition2   THEN expression2
      WHEN conditionn   THEN expressionn
      ELSE expression
END
```

One WHEN THEN pair is required, ELSE is optional (default is NULL), END is required. This form has been available since Oracle 8.1.6.

```
select ename,sal,case when job = 'CLERK' then 'GLUE'
                     when job = 'MANAGER' then 'SUPER'
                     else job
                end job_x
        from emp
        where case when sal < 1000 then sal + 2000
                   when sal < 2000 then sal + 1000
                   else sal
                end    > 2900
        order by case when sal < 1000 then sal + 9000
                      when sal < 2000 then sal + 7000
                      else sal
                end
```

The other style of CASE feels more like DECODE:

```
CASE expression   WHEN value1  THEN expression1
                               WHEN value2  THEN expression2
                               WHEN value3  THEN expression3
                               ELSE  expression4
END
```

In this style of CASE, several values are compared for a single expression or column. Again, ELSE is optional:

```
select ename,
       (case deptno
             when 10 then 'Overhead'
             when 20 then 'Overhead'
             when 30 then 'Operations'
             when 40 then 'Operations'
             else 'Overhead'
       end) EmpType
  from emp;
```

Addition of the CASE statement fills a gap in the PL/SQL language allowing complex conditions without nested if statements.

MISCELLANEOUS

Oracle9i adds new constraints for view definitions: Primary key, Rewrite Integrity (RI), and Unique. You may not specify check constraints for views.

The SELECT statement FOR UPDATE clause adds WAIT as an option. The specified wait delays when a lock is acquired to support the update for a number of seconds. If WAIT is used without a number of seconds, the wait will be indefinite until the resource is unlocked. The default is NOWAIT.

```
select ename,hiredate,job from emp for update wait 4;
```

MERGE

MERGE uses a SELECT (table/view/subquery) to UPDATE or INSERT rows in another table/view.

```
merge
    into bonus
    using emp
    on ( bonus.ename = emp.ename )
    when matched
            then  update  -- only one update match allowed!
           set bonus.sal = emp.sal,
           bonus.comm = emp.comm
    when not matched
        then insert
           (ename,job,sal,comm)
         values
           (emp.ename,emp.job,emp.sal,emp.comm)
```

EXTERNAL TABLES

Oracle9i allows access to an external sequential file as a read-only table. Before Oracle9i external file access was via SQL*Loader, UTL_FILE PL/SQL package, Pro* or OCI programs written in 3GLs, and BFILE in Oracle8 and later for large objects. The CREATE TABLE statement uses a combination of standard syntax and field definition syntax

from SQL*Loader. CREATE TABLE has two parts: an Internal table description using normal column definitions and the External table description using SQL*Loader-like syntax. When an External Table is referenced in SQL, the file data is loaded and made available.

For the following External File Data:

```
7402,LINCOLN,SALESMAN,7839,20-JAN-1980,2372.50,500.00,10
7418,MORRIS,CLERK,7782,01-APR-1982,1100.00,0,10
7422,LITTLE,CLERK,7782,12-NOV-1982,980.00,0,10
7437,BILLINGS,MANAGER,7839,23-FEB-1983,2923.75,0,20
7443,ALLEN,SALESMAN,7698,30-MAR-1982,1500.00,600.00,30
7456,GARCIA,ANALYST,7698,22-APR-1980,2312.50,0,30
7464,SOUK,ANALYST,7566,14-JUL-1981,3450.00,0,20
7473,CHANG,SALESMAN,7839,18-DEC-1982,2372.50,500.00,10
7484,SMITH,CLERK,7782,09-SEP-1982,925.50,0,10
7489,LIBUTTI,CLERK,7782,04-JUN-1980,1005.00,0,10
7495,HIPSON,MANAGER,7839,15-OCT-1982,3876.00,0,20
7498,MICHELL,SALESMAN,7698,16-NOV-1983,1600.00,750.00,30
7504,JORDAN,ANALYST,7698,21-APR-1982,2370.50,0,30
7518,SANCHEZ,ANALYST,7566,02-JAN-1981,3005.00,0,20
```

The Create Table would look something like this:

```
create table ioug_newemp
  ( empno      number(4)
   ,ename      char(10)
   ,job        char(9)
   ,mgr        number(4)
   ,hiredate   date
   ,sal        number(7,2)
   ,comm       number(7,2)
   ,deptno     number(2)
  )
  organization external
      (type oracle_loader default directory ioug_src
          access parameters
          ( records delimited by newline
           badfile ioug_bad:'ioug_newemp.bad'
           discardfile ioug_dis:'ioug_newemp.dis'
           logfile ioug_log:'ioug_newemp.log'
           fields terminated by ','
           missing field values are null
             (  empno, ename, job, mgr,
                hiredate char date_format date mask "mm-dd-yyyy",
                sal, comm, deptno
              )
          )
          location ('personc.dat')
      )
  reject limit unlimited;
```

An SQL statement using the External Table might look something like this:

```
SQL> select empno,ename,hiredate,sal from ioug_newemp
     EMPNO ENAME      HIREDATE         SAL
---------- ---------- --------- ----------
      7402 LINCOLN    20-JAN-80     2372.5
      7418 MORRIS     01-APR-82       1100
      7422 LITTLE     12-NOV-82        980
      7437 BILLINGS   23-FEB-83    2923.75
      7443 ALLEN      30-MAR-82       1500
      7456 GARCIA     22-APR-80     2312.5
      7464 SOUK       14-JUL-81       3450
      7473 CHANG      18-DEC-82     2372.5
      7484 SMITH      09-SEP-82      925.5
      7489 LIBUTTI    04-JUN-80       1005
      7495 HIPSON     15-OCT-82       3876
```

```
      7498 MICHELL     16-NOV-83       1600
      7504 JORDAN      21-APR-82     2370.5
      7518 SANCHEZ     02-JAN-81       3005
```

MULTI-TABLE INSERT

The multi-table insert allows a single INSERT statement to insert rows into several tables unconditionally (ALL) or conditionally (WHEN). Inserts may only name local tables (no views), may not include the RETURNING clause, and may not use sequences.

UNCONDITIONAL MULTI-TABLE INSERT

```
insert all
  into emp
     (empno,ename,job,mgr,hiredate,sal,comm,deptno)
   values
     (empno,ename,job,mgr,hiredate,sal,comm,deptno)
  into bonus
     (ename,job,sal,comm)
   values
     (ename,job,sal,comm)
  select empno,ename,job,mgr,hiredate,sal,comm,deptno
         from ioug_newemp;
```

CONDITIONAL MULTI-TABLE INSERT

```
insert first
   when job = 'SALESMAN' then
     into emp
       (empno,ename,job,mgr,hiredate,sal,comm,deptno)
        values
       (empno,ename,job,mgr,hiredate,sal,comm,deptno)
     into bonus (ename,job,sal,comm)
          values (ename,job,sal,comm)
   else
     into emp
       (empno,ename,job,mgr,hiredate,sal,comm,deptno)
     values
       (empno,ename,job,mgr,hiredate,sal,comm,deptno)
  select empno,ename,job,mgr,hiredate,sal,comm,deptno
         from ioug_newemp;
```

NESTED CURSOR EXPRESSIONS

Cursor expressions are new in Oracle9i. If a cursor expression is used in a SELECT statement (below), the cursor will be opened for each value fetched by the query. Cursor expressions may also be used to provide a REFCURSOR value being passed to a procedure or function.

```
SELECT dname,
       CURSOR(SELECT sal, comm FROM emp
              WHERE emp.deptno=dept.deptno) curval
    FROM dept;
```

SCROLLABLE CURSORS

Oracle9i adds support for scrollable cursors to provide compatibility with other database products. Scrollable cursors are read-only and allow fetch of specific rows, or, previous rows. So far, OCI programs and Java programs are the only place where these may be used.

PL/SQL "IN-SYNC"

Oracle9i provides a PL/SQL engine that uses the same SQL as the database! SQL inside PL/SQL may use the full SQL provided by the database. This means that developers no longer need to be concerned that some SQL capabilities supported by the database will not be available to PL/SQL code. All new SQL features are supported by PL/SQL. Plus, bulk bind may now apply to EXECUTE IMMEDIATE statements.

PL/SQL Native Compilation

Stored PL/SQL may now be compiled into native binary files to improve performance. This requires DBA involvement; system parameters must be set in the configuration file using ALTER SYSTEM, or using ALTER SESSION to modify the PLSQL_COMPILER_FLAGS setting. "Native" is used to compile to native binary, "Interpreted" (the default) is the way it has always worked. To the user of the PL/SQL procedure or function there is no difference other than speed of execution.

Subqueries Almost Anywhere!

Oracle8i allowed subqueries just about anywhere in the SQL statement, Oracle9i allows subqueries that return a single value anywhere **except** for the following: default value for columns, check constraints, RETURNING clause, function-based indexes, when condition in CASE, GROUP BY, HAVING, START WITH, or CONNECT BY.

New Functions

Oracle has added over 50 new functions to Oracle9i including: ANSI-standard functions, Date and Time functions, Analytical functions (added to those from Oracle8i), Unicode functions, Character conversion functions, XML functions, and Object functions. ANSI-Standard functions include: COALESCE, similar to NVL, but, returns first non-null value, NULLIF returns NULL if the specified value is matched.

ISO/ANSI Functions

Among the new functions added to Oracle9i are two added to conform to ISO/ANSI standards: NULLIF and COALESCE. Both of these are simple and are similar to functionality that has been part of Oracle for years.

NULLIF returns a null value based upon a simple comparison:

```
select ename,NULLIF(job,'PRESIDENT') job from emp;
```

COALESCE is similar to the NVL function Oracle has had for years, except it has the option of choosing the non-null from a list. The first non-null value in the list of parameters passed to COALESCE is output:

```
select deptno,coalesce(qtr4,qtr3,qtr3,qtr1) last_qtr from dept_summary;
```

Date and Time Functions

Oracle9i includes many date and time functions to complement the new datatypes, among the more useful are: CURRENT_DATE, CURRENT_TIMESTAMP, DBTIMEZONE, EXTRACT, SYSTIMESTAMP, TO_CHAR, TO_DSINTERVAL, TO_YMINTERVAL, and TO_TIMESTAMP. Several other functions that work with time zones have been added.

Oracle 8.1.6 Analytic Functions

Oracle 8.1.6 included a set of functions providing expanded support for data mining operations (topic is too rich to cover in the context of this paper). The analytic functions are divided into four "families": Lag/Lead compare values of rows to other rows in same table, Ranking supports "top n" queries, Reporting Aggregate compares aggregates to non-aggregates (pct of total), and Window Aggregate is used for moving average type queries. Analytic functions allow users to divide query result sets into ordered groups of rows called partitions based upon WHERE-clause style logic (not the same as database partitions).

New Analytical Functions

Oracle9i adds additional Analytical functions including: FIRST to get the first sorted group row, LAST to get the last sorted group row, GROUP_ID to provide a group identifier for GROUP BY, GROUPING_ID provides a number matching GROUPING, PERCENTILE_CONT shows a percentage when given continuous distribution, PERCENTILE_DISC creates a percentage when given a discrete distribution, WIDTH_BUCKET creates same-size intervals in a histogram.

Analytic functions lend statistical muscle to SQL that has in the past called for joins, unions, and complex programming. Performance is improved (sometimes significantly) because the functions are performing work that previously required self-joins and unions. Using Analytic functions requires far less SQL coding than previously required to accomplish the same task because one SQL statement takes the place of many.

Unicode Functions

Oracle9i adds a list of Unicode-specific functions: COMPOSE to return a string from Unicode data, DECOMPOSE to return the Unicode for a string, INSTRC to search a string for Unicode characters, LENGTHC returns the length of a Unicode string, SUBSTRC returns a partial Unicode string, and UNISTR converts a string to Unicode.

XML Functions

Several XML functions have been added to support those built into the XMLtype: EXISTSNODE, EXTRACT, SYSDBURIGEN, SYS_XMLAGG, and SYS_XMLGEN.

Oracle10g New Features

Oracle10g ("g" is for "grid") allows another evolutionary leap forward in developer productivity. Developer-related features of the new release include: new and improved data types, SQL improvements, SQL*Plus/iSQL*Plus improvements, enhancements to PL/SQL, and enhancements to Java and XML interfaces.

10g New Datatypes

10g Binary_double and Binary_float

Oracle10g provides support for the IEEE754 floating-point specification via two new datatypes. BINARY_FLOAT is a 32-bit, single-precision floating-point number stored as 5 bytes, including a length byte. BINARY_DOUBLE is a 64-bit, double-precision floating-point number stored as 9 bytes, including a length byte. When processing a NUMBER column, floating point numbers have decimal precision. In a BINARY_FLOAT or BINARY_DOUBLE column, floating-point numbers have binary precision and process more efficiently. Both binary floating-point numbers support special values of infinity and NaN (not a number). The maximum values of the two types are:

```
                          Binary-Float       Binary-Double
Maximum finite value          1.79e308       3.4e38
Minimum finite value           -1.79e308    -3.4e38
Smallest positive value         2.3e-308     1.2e-38
Smallest negative value -2.3e-308    -1.2e-38
```

10g Sgo_georaster

Oracle10g adds a new SDO_GEORASTER object type defined as follows:

```
CREATE TYPE SDO_GEORASTER AS OBJECT (
            rasterType NUMBER,
            spatialExtent SDO_GEOMETRY,
            rasterDataTable VARCHAR2(32),
            rasterID NUMBER,
            metadata XMLType);
```

SI_StillImage

SI_StillImage is an object type that represents digital images and metadata including height, width, and format.

10g "Limitless" LOB

Since first introduced the Oracle LOB type (BLOB, CLOB, BFILE) has been limited to 4GB (enough for most of us). Oracle 10g allows LOB data to be limited only by tablespace page size. This means that with 10g the current limit is between 8 and 128 terabytes depending upon your platform. These new, larger LOBs are supported in PL/SQL (using DBMS_LOB), Java via JDBC, and C/C++ (OCI interface).

10g New Statistical Functions

Oracle10g adds a new set of statistical functions to augment those already in the database. The new functions include:

- CORR returns the coefficient of correlation of a set of number pairs
- CORR_S calculates the Spearman's rho correlation coefficient
- CORR_K calculates the Kendall's tau-b correlation coefficient

- MEDIAN calculates the statistical median
- STATS_BINOMIAL_TEST is an exact probability test
- STATS_CROSSTAB method is used to analyze two nominal variables
- STATS_F_TEST tests whether two variances are significantly different
- STATS_KS_TEST compares two samples to see if they are from the same population or from populations that have the same distribution
- STATS_MODE returns the most frequently occurring value from a set
- STATS_MW_TEST - This Mann Whitney test compares two independent samples to test the null hypothesis that two populations have the same distribution function against the alternative hypothesis that the two distribution functions are different
- STATS_ONE_WAY_ANOVA tests differences in means (for groups or variables) for statistical significance by comparing two different estimates of variance
- STATS_T_TEST_ONE is a one-sample t-test
- STATS_T_TEST_PAIRED is a two-sample, paired t-test (also known as a crossed t-test)
- STATS_T_TEST_INDEP is a t-test of two independent groups with the same variance (pooled variances)
- STATS_T_TEST_INDEPU is a t-test of two independent groups with unequal variance (unpooled variances)
- STATS_WSR_TEST tests paired samples to determine whether the median of the differences between the samples is significantly different from zero

Example CORR Function

```
select country,
       corr(sale,cnt)
      from sales_view
      group by country
```

10g Fast Dual

For years, developers have used the Dual table for "quick and dirty" queries only to find during performance tuning that scans involving dual could be expensive. In Oracle 10g the optimizer knows about the Dual table and implements an operation called "fast dual" greatly speeding access.

10g Connect login.sql/glogin.sql

As before, in Oracle 10g the login.sql and glogin.sql files are automatically executed upon entering SQL*Plus. In addition, in Oracle 10g login.sql and glogin.sql are also executed upon execution of CONNECT. This is either a blessing or a curse, just be aware that it is happening…

10g SQL*Plus Miscellaneous new Features.

- SET SERVEROUTPUT ON now works immediately within the block where it is executed
- Recycle Bin keeps deleted database objects until PURGEd
- DESCRIBE now validates before display
- White space now allowed in file names
- APPEND, CREATE, REPLACE extensions to SPOOL

10g Recycle Bin

SQL*Plus now provides an "oops" capability for object drops if the database is running in "Flashback mode" allowing a DROP TABLE (or other drop) to be undone. After issuing "DROP object xyz" the user may issue the following SQL*Plus command to see "dropped objects:

```
SHOW RECYCLEBIN
```

Then, one of these SQL statements might be executed:

```
PURGE table xx|index yy |recyclebin|tablespace zz;


FLASHBACK TABLE xxx TO BEFORE DROP;
```

Now that 10g uses the recycle bin be careful! Dropping tables no longer really drops them… This might be a problem for applications with lots of "temp"-type tables where successive CREATE TABLE - DROP TABLE statements occur (Temporary Tables function as before).

Recycle Bin sequence of events

Drop a table as before:

```
drop table myTable;
```

To view the recyclebin's contents:

```
show recyclebin
```

```
            ORIGINAL     RECYCLEBIN NAME      TYPE  DROP TIME
            myTable      RB$$41506$TABLE$0    TABLE 2004-04-01:22:11:13
```

To restore the table:

```
flashback table myTable to before drop;
```

To drop a table and avoid the recyclebin:

```
drop table myTable purge;
```

To "empty" the recyclebin:

```
purge recyclebin;
```

iSQL*Plus Changes

With Oracle 10g, iSQL*Plus has been improved to include multi-page output and now supports prompting for input values.

10g Regular Expressions

Oracle now has functions that allow the use of POSIX-compliant regular expressions in SQL:

- REGEXP_LIKE Allows pattern matching
- REGEXP_INSTR Search for string matching pattern and return position
- REGEXP_REPLACE Find string matching pattern and replace it
- REGEXP_SUBSTR Search for string matching pattern and return substring

Regular Expression Examples

```
select employee_id,phone_number
      from hr.employees
      where REGEXP_LIKE(phone_number,
      '[[:digit:]]{3}[[:punct:]][[:digit:]]{2}[[:punct:]]');

select first_name, last_name
      from hr.employees
      where REGEXP_LIKE (first_name, '^ste(v|ph)en$');
```

10g Model Clause

The SQL MODEL clause is a powerful extension of the SELECT statement providing the ability to perform spreadsheet-like processing in the form of multi-dimensional arrays and apply formulas to the array values. The Model

clause defines a multidimensional array by mapping the columns of a query into three groups: partitioning, dimension, and measure columns. Partitions define logical blocks of the result set in a way similar to the partitions of the analytical functions; each partition is viewed by the formulas as an independent array. Dimensions identify each measure cell within a partition; each column identifies characteristics such as date, region and product name. Measures are similar to the measures of a fact table in a star schema, they normally contain numeric values such as sales units or cost; each cell is accessed within its partition by specifying its full combination of dimensions.

10G MODEL SYNTAX

```
SELECT
  -- rest of SELECT goes here -
      MODEL [main]
      [reference models]
      [PARTITION BY (<cols>)]
      DIMENSION BY (<cols>)
      MEASURES (<cols>)
      [IGNORE NAV] | [KEEP NAV]
      [RULES
      [UPSERT | UPDATE]
      [AUTOMATIC ORDER | SEQUENTIAL ORDER]
      [ITERATE (n) [UNTIL <condition>] ]
      ( <cell_assignment> = <expression> ... )
```

10G MODEL EXAMPLE

```
SELECT SUBSTR(country,1,20) country, SUBSTR(prod,1,15) prod, year, sales
      FROM sales_view
      WHERE country IN ('Canada','Germany')
      MODEL RETURN UPDATED ROWS
            PARTITION BY (country)
            DIMENSION BY (prod, year)
            MEASURES (sale sales)
            RULES (
                  sales['Zoop Tube', 2002] = sales['Zoop Tube', 2001] +
                                             sales['Zoop Tube', 2000],
                  sales['Hula Twirl', 2002] = sales['Hula Twirl', 2001],
                  sales['HulaZoop Pkg', 2002] = sales['Zoop Tube', 2002] +
                                                sales['Hula Twirl', 2002])
      ORDER BY country, prod, year;

COUNTRY PROD YEAR SALES
-------------------- --------------- ---------- ----------
Canada HulaZoop Pkg 2002 92613.16
Canada Zoop Tube 2002 9299.08
Canada Hula Twirl 2002 83314.08
Germany HulaZoop Pkg 2002 103816.6
Germany Zoop Tube 2002 11631.13Germany Hula Twirl 2002 92185.47
```

10G MODEL EXAMPLE EXPLAINED

The preceding SQL statement calculates sales values for two products and defines sales for a new product based upon the other two products. The statement partitions data by country, so formulas are applied to one country at a time. Sales fact data ends with 2001, any rules defining values for 2002 or later will insert new cells. The first rule defines sales of the "Zoop Tube" game in 2002 as the sum of its sales in 2000 and 2001. The second rule defines sales for "Hula Twirl" in 2002 to be the same value they were for 2001. The third rule defines "HulaZoop Pkg" that is the sum of the Zoop Tube and Hula Twirl values for 2002. The rules for Zoop Tube and Hula Twirl must be executed before the HulaZoop Pkg rule.

10G MERGE ENHANCED

MERGE now allows specification of either update, or insert, or both. Deletion is now allowed during update.

10G APPLICATION EXPRESS (FORMERLY HTML DB)

Oracle Application Express (formerly HTML DB) is a complete web development and deployment environment built into Oracle 10g. Application Express is based upon the home-grown software that helped make Tom Kyte's "Ask Tom" website so powerful. Application Express is designed to make building web applications easy without compromising flexibility when building web applications. Pre-built components are assembled using wizards and declarative programming eliminating most need to write code. Some of the built-in features include Page Rendering and a Processing Engine. Rather than generating code, Oracle Application Express stores user interface properties and data access and logic behaviors in an application definition; when an Application Express application is run pages are rendered in real time based upon an application definition stored in the database. Logic to determine how a user flows from page to page, data validation and form handlers are all built-in to the processing engine. Deployment is automatic, immediately after an application is built or changed users can start using it. Pre-Built components are used with wizards, to assemble applications with forms, reports, and charts without writing code; the pre-built components include: navigational controls, authentication schemes and user interface themes.

ORACLE DOCUMENTATION

Thorough documentation may be found for Oracle's features in the following on-line references (go to http://tahiti.oracle.com or http://technet.oracle.com if you do not have the disks).

1. Oracle SQL Reference
2. Oracle PL/SQL Users Guide and Reference
3. Oracle Java Developer's Guide
4. Oracle Java Tools Reference
5. Oracle JDBC Developer's Guide and Reference
6. Oracle Supplied Java Packages Reference
7. Lots of papers and examples: at http://technet.oracle.com

WRAPPING IT ALL UP

Oracle10g and Oracle9i continue the tradition of fine products and significant enhancements we have come to expect from Oracle. The many new features that are geared to the developer will save time, provide enhanced performance, and allow us to create richer applications. The DBA-oriented features of the new release will make performance and reliability easier to guarantee for our users.

ABOUT THE AUTHOR

John King is a Partner in King Training Resources, a firm providing instructor-led training since 1988 across the United States and Internationally. John specializes in application development software on a variety of platforms including Web, Unix, Linux, IBM mainframe, and personal computers. John has worked with Oracle products and the database since Version 4 and has been providing training to Oracle application developers since Oracle Version 5. John develops and presents customized courses in a variety of topics including SOA, Web Services, Oracle, DB2, UDB, Java, XML, C#, and various programming languages. He has presented papers at many industry events including: IOUG-A Live!, UKOUG Conferences, EOUG Conferences, AUSOUG Conferences, RMOUG Training Days, OOUG, TOUG, MAOP-AOTC, NYOUG, and the ODTUG conference.


```
John Jay King
King Training Resources
6341 South Williams Street
Littleton, CO 80121-2627
U.S.A.

Phone:      1.303.798.5727    1.800.252.0652 (within the U.S.)
Fax:        1.303.730.8542
Email:      john@kingtraining.com
Website:    www.kingtraining.com
```