Oracle9i for Developers: What You Need to Know

John Jay King, King Training Resources

Introduction

The many new features of Oracle9i extend the capabilities of the database in many ways. Attendees will be introduced to the new and improved features of Oracle9i 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 Oracle9i that enables improved application development. Specific topics include: CASE, Outer Join, Object improvements and expanded Java capabilities. Attendees will be better equipped to create new applications or modify existing applications to take full advantage of Oracle9i.

Oracle8 and Oracle8i provide support for Objects with all versions, but, with Oracle9i some of the issues raised by earlier releases have been addressed.

Features added or improved with Oracle9i are marked in these notes to indicate that they are Oracle9i-specific, also, some of the more recent Oracle8i changes are noted. Examples of some new features are provided during this presentation, however, you should see the Oracle9i SQL Reference, Oracle9i PL/SQL Users Guide and Reference, and other relevant Oracle manuals for complete documentation. All sample code was tested using Oracle 9.2.

Oracle9i provides rich tools with many new features extending the capabilities of the database in many ways. This paper focuses on those improvements and additions to 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 Oracle9i is assumed.

Introduction to Oracle9i

Oracle9i provides rich tools with many new features extending the capabilities of the database in many ways. This paper focuses on those improvements and additions to 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 Oracle9i is assumed.

Developer-oriented features

Specific developer-oriented features include: 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 log miner 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 Log Miner has been improved to support Flashback query and provide other meaningful data.

Log-miner improvements.

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 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 is 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 provide 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, note that 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 the 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 expresssion2
        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;
```

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: 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 iouga_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 iouga_src
       access parameters
       ( records delimited by newline
        badfile iouga_bad:'iouga_newemp.bad'
        discardfile iouga_dis:'iouga_newemp.dis'
        logfile iouga_log:'iouga_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 iouga_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 iouga_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 iouga_newemp;
```

Nested Cursor Expressions

Cursor expressions are new in Oracle9i. If a cursor expression is used in a select (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 function 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.

Oracle Documentation

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

- Oracle9i SQL Reference
- Oralce9i PL/SQL Users Guide and Reference
- Oracle9i Java Developer's Guide
- Oracle9i Java Tools Reference
- Oracle9i JDBC Developer's Guide and Reference
- Oracle9i Supplied Java Packages Reference
- Lots of papers and examples: at http://technet.oracle.com

Wrapping it all Up

Oracle9i continues 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, provided 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 Unix, Linux, IBM mainframe, personal computers, and the web. 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 Oracle, DB2, UDB, Java, XML, C++, and various programming languages. He has presented papers at various industry events including: IOUG-A Live!, UKOUG Conference, EOUG Conference, RMOUG Training Days, 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
```