Oracle9i for Developers: What You Need to Know

Presented to: AOTC – May 2003

John King

King Training Resources

6341 South Williams Street

Littleton, CO 80121-2627 USA

www.kingtraining.com

800.252.0652 or 303.798.5727

Session Objectives

- Learn new Oracle9i features that are geared to developers
- Be ready to use ISO/ANSI standard SQL to make code more portable
- Know how existing database features have been improved in Oracle9i
- Become aware of some Oracle9i DBA oriented features that will impact developers

Where We Are Going

- New datatypes
- New functions
- New SQL statements
- Multi-table inserts
- New join and case syntax
- External tables
- PL/SQL “in-sync”
- Object improvements

New Datatypes

- Oracle9i provides a series of new and improved datatypes:
  - Date Related:
    - TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE
    - TIMEZONE_HOUR, TIMEZONE_MINUTE, TIMEZONE_REGION
    - INTERVAL YEAR TO MONTH
    - INTERVAL DAY TO SECOND
  - Character Related changes:
    - CHAR, VARCHAR2, NCHAR, NVARCHAR2
  - UROWID: Rowid for IOT rows
  - New Oracle-supplied datatypes
    - SYS.ANYTYPE, SYS.ANYDATA, SYS.ANYDATASET
    - XMLType
    - URIType, DBURIType, HTTPURIType, URIFactoryType
    - MDSYS.SDO_GEOMETRY
    - ORDSYS.ORDAudio, ORDSYS.ORDImage, ORDSYS.ORDVideo

New Date Datatypes

- Timestamps: Year, Month, Day, Hour, Minute, Second
  - TIMESTAMP or TIMESTAMP (n)
    - May specify second fraction used (0-9 decimals), 6 decimals is the default
  - TIMESTAMP WITH TIMEZONE or TIMESTAMP (n) WITH TIMEZONE
    - May specify second fraction used (0-9 decimals), 6 decimals is the default
  - TIMESTAMP WITH LOCAL TIMEZONE or TIMESTAMP (n) WITH LOCAL TIMEZONE
    - May specify second fraction used (0-9 decimals), 6 decimals is the default
- Intervals: Provide period of time
  - INTERVAL YEAR TO MONTH or INTERVAL YEAR (n) TO MONTH
    - Interval in Years and Months
    - May specify number of digits in year (0-9), 2 digits is the default
  - INTERVAL DAY TO SECOND or INTERVAL DAY(d) TO SECOND (s)
    - Interval in Days, House, Minutes, Seconds
    - May specify number of digits for days (0-9), 2 digits is the default
    - May specify second fraction used (0-9 decimals), 6 decimals is the default

Character Modifications

- CHAR, VARCHAR2, NCHAR, and NVARCHAR2 may specify additional size descriptor (BYTE or CHAR)
  - VARCHAR2(n)
  - VARCHAR2(n) BYTE
  - VARCHAR2(n) CHAR
  - CHAR(n)
  - CHAR(n) BYTE
  - CHAR (n) CHAR
- BYTE specifies that the size of the column is specified in bytes
- CHAR specifies that the size of the column is specified in characters
- Maximum size of VARCHAR2 is 4000 bytes (unchanged)
- Maximum size of CHAR is 2000 bytes (unchanged)

Oracle-Supplied “Any” Types

- Oracle supplies a datatype for use in creating tables and stored procedures when the actual type is not know, 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

XMLtype Datatype

- SYS.XMLtype is an Oracle-defined datatype used to store XML data within the database as:
  - Entire document as CLOB/XMLType
  - Document elements as relational table rows and columns
- Member functions include:

– createXML()	Create XMLType instance
– existsNode()	Checks if XPath can find any valid nodes
– extract()	Uses XPath to return fragment as XMLType
– isFragment()	Checks to see if document is really a fragment
– getClobVal()	Gets document as a CLOB
– getStringVal()	Gets value as a string
– getNumberVal()	Gets numeric value as a number

- Lots of XML support is added in Oracle9i, check the reference manual:

  “Application Developer's Guide – XML”

SQL - XML Functions

- SQL provides several functions specifically for dealing with XML data including:
  - SYS_DBURIGEN(ts) Generate DBURITYPE URL used to obtain XML data from the database
  - SYS_XMLGEN(exp) Convert specified database row and column into an XML document
  - SYS_XMLAGG(exp) Generate single XML document from aggregate of XML data specified by “exp”
  - XMLELEMENT(name,exp) Generates XML element using name and exp as data
  - XMLATTRIBUTES(exp,list) Generates XML attributes using expression list

SYS_XMLGEN

- SYS_XMLGEN
  - Uses a single input expression representing a particular row/column (scalar value or user-defined type)
    - For scalar value a single XML element representing the value is returned
    - For user-defined type XML elements representing each of the user-defined type's data items is returned
  - Returns an instance of SYS.XMLType data that is an XML document
  - The example on the next page displays using getStringVal since SYS.XMLType data returns as CLOB and is not displayable by SQL*Plus

SYS.XMLGEN Example

```
select sys_xmlgen(ename).getStringVal() Name
     from emp
     where job = 'ANALYST'

NAME
---------------------------------------------------
<?xml version="1.0"?>
 <ENAME>FORD</ENAME>

<?xml version="1.0"?>
 <ENAME>SCOTT</ENAME>
```

SYS_XMLAGG

- SYS_XMLAGG aggregates all XML documents (or fragments of documents) for an expression and produces a single XML document
  - ROWSET is the default tag name
  - Use SYS.XMLGenFormatType to change tag name
- The example on the next page uses the SYS_XMLGEN function to generate an XML document for each dept 20 row of the sample EMP table
- The example on the next page displays using getClobVal since SYS.XMLType data returns as CLOB and is not displayable by SQL*Plus

SYS_XMLAGG Example

```
 select sys_xmlagg(SYS_XMLGEN(Ename)).getClobVal() emps
      from emp
      where deptno = 10

EMPS
------------------------------------------------------
<?xml version="1.0"?>
<ROWSET>
<ENAME>KING</ENAME>
<ENAME>CLARK</ENAME>
<ENAME>MILLER</ENAME>
</ROWSET>
```

SYS_XMLAGG Example 2

```
select sys_xmlagg(SYS_XMLGEN(Ename)
   ,sys.XMLGENFORMATTYPE.createFormat('depts')).getClobVal() emps
from emp group by deptno
   EMPS
   ------------------------
   <?xml version="1.0"?>
   <depts>
   <ENAME>KING</ENAME>
   <ENAME>CLARK</ENAME>
   <ENAME>MILLER</ENAME>
   </depts>
   <?xml version="1.0"?>
   <depts>
   <ENAME>JONES</ENAME>
   <ENAME>ADAMS</ENAME>
   <ENAME>SCOTT</ENAME>
   <ENAME>SMITH</ENAME>
   <ENAME>FORD</ENAME>
   </depts>
```

```
<?xml version="1.0"?>
       <depts>
 <ENAME>BLAKE</ENAME>
  <ENAME>WARD</ENAME>
 <ENAME>JAMES</ENAME>
<ENAME>MARTIN</ENAME>
 <ENAME>ALLEN</ENAME>
<ENAME>TURNER</ENAME>
      </depts>
```

XMLELEMENT

- XMLELEMENT(name,exp) Generates an XML element using name and exp as data

```
select xmlelement("employee",
       xmlelement("empid",empno),
       xmlelement("empname",ename)) myxml
from emp
<employee> <empid>7369</empid>
   <empname>SMITH</empname> </employee>
<employee> <empid>7499</empid>
   <empname>ALLEN</empname> </employee>
<employee> <empid>7521</empid>
   <empname>WARD</empname> </employee>
<employee> <empid>7566</empid>
   <empname>JONES</empname> </employee>
<employee> <empid>7654</empid>

   <empname>MARTIN</empname> </employee>
```

XMLATTRIBUTES

- XMLATTRIBUTES(exp,list) Generates XML attributes using expression list

```
select xmlelement("employee",
       xmlelement("emp",
                xmlattributes(empno as "empno",
                              ename as "ename")),
       xmlelement("job",job),
       xmlelement("hiredate",hiredate),
       xmlelement("pay",
                xmlattributes(nvl(sal,0) as "sal",
                              nvl(comm,0) as "comm"))
       ) as myxml
from emp
```

XMLATTRIBUTES Example

```
<employee>
  <empno>7782</empno>
  <job>MANAGER</job>
  <job>MANAGER</job>
  <hiredate>09-JUN-81</hiredate>
  <pay sal="2450" comm="0"/>
</employee>

<employee>
  <empno>7839</empno>
  <job>PRESIDENT</job>
  <job>PRESIDENT</job>
  <hiredate>17-NOV-81</hiredate>
  <pay sal="5000" comm="0"/>
</employee>
```

Other XML Functions

• XMLColattval	Creates series of XML fragments using an element name of “column” and column names and values as attributes
• XMLConcat	Concatenates a series of XMLType objects (opposite of XMLElement)
• XMLForest	Creates XML fragments from a list of arguments/parameters
• XMLSequence	Creates Varray of XMLType instances
• XMLTransform	Uses input XMLType and XSL style sheet (also XMLType) to create a new XMLType
• UpdateXML	Uses an XMLType and an XPATH reference and returns an updated XMLType

URI Datatypes

- Oracle9i provides a series of predefined types designed to help programmers work with web applications including:
  - SYS.URITYPE
  - SYS.DBURITYPE
  - SYS.HTTPURITYPE
  - SYS.URIFACTORYTYPE

UROWID

- UROWID allows manipulation of values used to identify the base-64 value representing the address of a row in an index-organized table (IOT)

```
mycol1 UROWID
mycol2 UROWID(nn)
```

  - nn Size of UROWID is optional:
    default size = 4000
    max = 4000

Misc. Oracle-Supplied Datatypes

- Oracle9i supplies a set of predefined types for working with multimedia:
  - ORDSYS.ORDAUDIO
  - ORDSYS.ORDIMAGE
  - ORDSYS.ORDVIDEO
- Finally, Oracle9i provides a spatial datatype:
  - MDSYS.SDO_GEOMETRY

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
  - CONNECT BY

Silly Examples

- Here are four oddball statements that would not be possible in earlier versions

```
select ename,job,sal,(select avg(sal) from emp where job = main.job) jobavgsal
                    from emp main;

select ename,sal from emp
  where sal between (select avg(sal) from emp where job = 'SALESMAN')
                  and (select avg(sal) from emp where job = 'ANALYST');

select deptno from dept
       where (select avg(sal) from emp) > (select avg(sal) from emp
                                                    where emp.deptno = dept.deptno);
select ename,sal from emp
  order by (select dname from dept where dept.deptno = emp.deptno),ename;
```

Misc. SQL Improvements

- Most variable-character functions now allow CLOB arguments (SUBSTR, etc…)
- New operators:
  - LIKEC, LIKE2, LIKE4 similar to LIKE with Unicode, UCS2, and UCS4 data respectively
  - IS OF xxx determines the type of an object instance
- WAIT option: SELECT … FOR UPDATE WAIT waits specified number of seconds for locked row
- DEFAULT keyword for INSERT and UPDATE
- When using CONNECT BY may use ORDER BY SIBLINGS to sort within hierarchy
- New sample tables!
- Scrollable cursor support (3GLs)
- ***Group by Grouping Sets***
- New hints

New Functions

- Many (over 50!) new functions have been added to Oracle9i including:
  - ANSI-standard functions
  - Date and Time functions
  - Analytical functions (added to those from Oracle8i)
  - Unicode functions
  - Character conversion functions
  - XML functions
  - Object functions

ANSI-Standard Functions

- COALESCE is similar to NVL, but, returns first non-null value:

```
COALESCE(qtr4,qtr3,qtr2,qtr1)
```

- NULLIF returns NULL if the specified value is matched

```
NULLIF(PREFCODE, 'N/A')
```

Date and Time Functions

- Oracle9i includes many date and time functions, among the more useful are:
  - CURRENT_DATE
  - CURRENT_TIMESTAMP
  - DBTIMEZONE
  - EXTRACT(timestamp)
  - SYSTIMESTAMP
  - TO_CHAR(timestamp)
  - TO_DSINTERVAL(instring)
  - TO_YMINTERVAL(instring)
  - TO_TIMESTAMP(instring)
- Several other functions that work with time zones have been added

Timezone Functions

- Several timezone-specific functions have also been added:

– DBTIMEZONE	Get UTC offset from database
– SESSIONTIMEZONE	Get UTC offset from current session (uses ORA_SDTZ env. variable)
– FROM_TZ	Convert timestamp at timezone to timestamp with timezone value
– TZ_OFFSET	Returns UTC offset for specified time zone
– SYS_EXTRACT_UTC	Get UTC time from specified timestamp with time zone
– DBTIMEZONE	Get UTC offset from database
– TO_TIMESTAMP_TZ	Convert string to timestamp with timezone using normal formatting

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** - Compares values of rows to other rows in same table: LAG, LEAD
  - **Ranking** - Supports "top n" queries: CUME_DIST, DENSE_RANK, NTILE, PERCENT_RANK, RANK, ROW_NUMBER
  - **Reporting Aggregate -** Compares aggregates to non-aggregates (pct of total): RATIO_TO_REPORT
  - **Window Aggregate -** Moving average type queries: FIRST_VALUE, LAST_VALUE
- Analytic functions allow users to divide query result sets into ordered groups of rows called partitions (not the same as database partitions)

Example Analytical Function

- Rank may be used with GROUP aggregation:

```
select dname,
       nvl(avg(sal),0) avg_sal,
       count(empno) nbr_emps,
       rank() over (order by nvl(avg(sal),0)) rank
   from emp right join dept using (deptno)
group by dname

DNAME             AVG_SAL   NBR_EMPS       RANK
-------------- ---------- ---------- ----------
OPERATIONS              0          0          1
TESTER                  0          0          1
SALES          1566.66667          6          3
RESEARCH             2175          5          4
ACCOUNTING     2916.66667          3          5
```

New Analytical Functions

- Oracle9i adds additional Analytical functions:

– FIRST	Gets first sorted group row
– LAST	Gets last sorted group row
– GROUP_ID	Group Identifier for GROUP BY
– GROUPING_ID	Number matching GROUPING
– PERCENTILE_CONT	Pct. when continuous distribution
– PERCENTILE_DISC	Pct. When discrete distribution
– WIDTH_BUCKET	Use to create same-size intervals in histogram

Example FIRST/LAST

```
select dname,
       min(salary) keep (dense_rank first order by hire_date)
             "First Hired",
       max(salary) keep (dense_rank last order by hire_date)
             "Last Hired"
   from hr.employees right join dept
          on department_id = deptno
   group by dname

DNAME          First Hired Last Hired
-------------- ----------- ----------
ACCOUNTING            4400       4400
OPERATIONS            6500       6500
RESEARCH             13000       6000
SALES                11000       2500
```

GROUPING SETS

- Oracle 9i Release 2 adds the ability to created cube-type statistics on command

```
   select deptno
          ,job
          ,count(empno) nbremps
          ,sum(sal)     totpay
        from emp
        group by grouping sets (deptno,job)

    DEPTNO JOB          NBREMPS     TOTPAY
---------- --------- ---------- ----------
           CLERK              4       4150
           ANALYST            2       6000
           MANAGER            3       8275
           SALESMAN           4       5600
           PRESIDENT          1       5000
        10                    3       8750
        20                    5      10875
        30                    6       9400
```

ORDER SIBLINGS BY

- Sorting data returned by CONNECT BY has been an issue for years, Oracle 9i Release 2 changes things:

```
   select lpad(' ',level*2)||empno empid,ename,mgr,deptno,level
         from emp
         connect by prior empno = mgr
         start with mgr is null
         order siblings by (ename)
EMPID        ENAME             MGR     DEPTNO      LEVEL
------------ ---------- ---------- ---------- ----------
  7839       KING                          10          1
    7698     BLAKE            7839         30          2
      7499   ALLEN            7698         30          3
      7900   JAMES            7698         30          3
      7654   MARTIN           7698         30          3
      7844   TURNER           7698         30          3
      7521   WARD             7698         30          3
    7782     CLARK            7839         10          2
      7934   MILLER           7782         10          3
```

TO_CHAR & New Dates

- TO_CHAR has been modified with attributes to describe desired TIMESTAMP components:

– FF1-FF9	Fractional Seconds
– TZD	Abbrev. Timezone with Daylight time notation
– TZH	Timezone UTC offset hours
– TZM	Timezone UTC offset minutes
– TZR	Timezone Region
– X	Local radix operator

Using TO_CHAR

- Use the new TO_CHAR attributes in the same manner as other attributes:

```
select to_char(t4,'yyyy-mm-dd hh24:mi:ssxff tzh:tzm')
            t4_value
   from timetest1;

T4_VALUE
-----------------------------------------------------
2002-10-28 16:23:50.9530000 -07:00
```

Extract Syntax

- EXTRACT is used to get specific parts of a TIMESTAMP of INTERVAL

```
EXTRACT ( YEAR  FROM  datetime   )
          MONTH       timestamp
          DAY         interval
          HOUR
          MINUTE
          SECOND
          TIMEZONE_HOUR
          TIMEZONE_MINUTE
          TIMEZONE_REGION
          TIMEZONE_ABBR
```

SYS_EXTRACT_UTC

- Query below extracts current UTC timestamp from provided timestamp with time timezone value

```
select sys_extract_utc(to_timestamp_tz(current_timestamp))
   from dual
SYS_EXTRACT_UTC(TO_TIMESTAMP_TZ(CURRENT_TIMESTAMP))
------------------------------------------------------
01-APR-02 08.56.15.013000 PM
```

Note: UTC (Coordinated UniversalTime)
was formerly known as
GMT (Greenwich Mean Time)

Unicode Functions

- Oracle9i Unicode-specific functions:

– COMPOSE	Return string from Unicode
– DECOMPOSE	Return Unicode for string
– INSTRC	Search string for Unicode characters
– LENGTHC	Length of Unicode string
– SUBSTRC	Return partial Unicode string
– UNISTR	Convert string to Unicode

- INSTRC, LENGTHC, AND SUBSTRC functions are replicated as INSTR2, INSTR4, LENGTH2, LENGTH4, SUBSTR2, and SUBSTR4 for data using UCS2 and UCS4 codepoints

Character Conversion Functions

- Several functions have been added or improved:
  - ASCIISTR
  - BIN_TO_NUM
  - CAST
  - DECODE
  - NCHR
  - RAW_TO_HEX
  - ROWIDTONCHAR
  - TO_CHAR
  - TO_CLOB
  - TO_NCHAR
  - TO_NCLOB

New SQL Statements

- Oracle9i adds some new SQL statements:
  - CREATE PFILE — Export database parameters as text file
  - CREATE SPFILE — Create server parameters from external text file
  - MERGE — Combination INSERT and UPDATE, if row exists change it, otherwise build new row

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:
  - – SQL*Loader
  - – UTL_FILE PL/SQL package
  - – Pro* or OCI programs written in 3GLs
  - – BFILE in Oracle8 and later for large objects

How Do External Tables Work?

- 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
    - Uses normal column definitions
    - Constraints are not valid
    - No indexes may be defined
  - External table description
    - Uses DIRECTORY objects to find files
    - Uses SQL*Loader-like syntax
- When an External Table is referenced in SQL, the file data is loaded and made available

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
```

Create Table (Internal desc)

```
create table 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
```

Rest of CREATE TABLE on next page!

Create Table (External desc.)

```
(type oracle_loader default directory iouga_src
    access parameters
    ( records delimited by newline
     badfile iouga_bad:'newemp.bad'
     discardfile iouga_dis:'newemp.dis'
     logfile iouga_log:'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
;
```

Using External Table

```
SQL> select empno,ename,hiredate,sal from 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

- Multi-table insert allows a single INSERT statement to insert rows into several tables:

– ALL	Unconditionally INSERT
– WHEN	Conditionally INSERT

- Rules
  - – May only insert into local tables (no views)
  - – RETURNING clause invalid
  - – Insert subqueries 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 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 newemp;
```

New ISO/ANSI Join Syntax

- ISO/ANSI Join syntax has been used for several years in some non-Oracle SQL environments
- Oracle invented the original Outer-join syntax and was slow to accept the new style
- ISO/ANSI Join syntax is supported by many third party SQL tools
- The new semantics separate join criteria from other row selection criteria

Cross Join

- Cross Join is designed to provide a “Cartesion Product” type join. It works the same as a comma-delimited join, requiring specification of join conditions in the WHERE clause to avoid the Cartesian Product:

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

Natural Join

- Natural joins indicate an equi-join automatically using any column names match to join
- 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
```

Join Using

- When join column names are the same, the new syntax now allows the USING clause

```
select dname,ename
 from dept join newemp
 using (deptno)
```

Inner Join

- Traditional Inner Joins match rows tables
- 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;
```

ISO/ANSI Inner Join

- Use 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
- 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;
```

Joining More Then Two

```
select distinct nvl(dname,'No Dept') dept
      ,count(empno) nbr_emps
      ,round(avg(grade),1) avg_paygrade
    from      many_emps emp
         join many_depts dept
              on emp.deptno = dept.deptno
         join salgrade
              on emp.sal between losal and hisal
    where emp.job in ('MANAGER','SALESMAN','ANALYST')
    group by dname
```

Oracle Outer Join Operator

- Oracle invented the first syntax for solving the outer Join issue years ago
- This is the “(+)” notation used on the 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;
```

ISO/ANSI Outer Join

- 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

Left/Right Join

- The example below solves the same problem as the Oracle Outer Join operator example earlier:

```
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;
```

Full Outer Join (Union)

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

```
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;
```

ISO/ANSI Full Outer Join

- 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;
```

Searched CASE Syntax

- Oracle8i added the CASE expression to allow more complex (ANSI/ISO standard) processing than DECODE
- CASE allows IF…THEN…ELSE logic to be placed anywhere in SQL that a column or literal can go
- CASE syntax 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
- The example on the next page shows CASE being used in three parts of the statement

Searched CASE Example

```
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
```

Output:

```
ENAME       SAL     JOB_X
JONES      2975     SUPER
FORD       3000     ANALYST
SCOTT      3000     ANALYST
KING       5000     PRESIDENT
JAMES       950     GLUE
```

Simple CASE Syntax

- New with Oracle9i, simple CASE syntax compares values to an expression

```
select ename,sal,
       case job when 'CLERK' then 'Producer'
                when 'ANALYST' then 'Producer'
                when 'PRESIDENT' then 'Overhead'
                when 'SALESMAN' then 'Producer'
                else 'Overhead'
       end emptype
   from emp;
```

WITH Subquery Reuse

- WITH allows a subquery to be named and reused

```
with SUMMARY as
   (SELECT dname
              , sum(sal) saltot
              , round(avg(sal),2) avgsal
              , count(distinct empno) nbremps
           from emp join dept
                   on emp.deptno = dept.deptno
                group by dname
   )
 select dname, nbremps, avgsal
   from SUMMARY
   where saltot > ( select sum(saltot) * .25
                          from SUMMARY)
   ORDER BY saltot DESC;
```

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 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
- 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	Compile to native binary
• Interpreted	The default, the way it has always worked

- To the user of the PL/SQL procedure/function there is no difference other than speed of execution

Oracle C++ Call Interface

- The Oracle C++ Call Interface (OCCI) allows C++ programmers to create fast database applications
  - Speed of OCI
  - Object-oriented flavor of C++

Object Improvements

- SQL type inheritance and synonyms
- Object view hierarchies
- Type evolution
- User-defined aggregate functions
- User-defined constructors
- Generic and transient data types
- Function-based index support
- Multi-level collections
- C++ interface to Oracle
- Java object storage

DBA-Oriented Features

- As developers, you should be aware of some of the new features at the DBA-level
  - Flashback Query, point-in-time queries
  - Cost-Based Optimizer improvements, new hints
  - Multiple block sizes in a single tablespace
  - Constraints on views

Oracle Documentation

- Oracle9i SQL Reference
- Oracle9i PL/SQL User’s Guide and Reference
- Oracle9i Application Developer's Guide - Object-Relational Features
- Oracle9i Concepts
- Oracle9i Application Developer’s Guide - XML
- Lots of papers and examples: http://technet.oracle.com

Wrapping it all Up

- Oracle9i adds significant new functionality to the already robust database environment
- While an emphasis is sometimes placed on the features of Oracle9i that support the Data Base Administrator, this paper shows many Developer-oriented features of great usefulness
- The importance of ISO/ANSI constructs cannot be underestimated, with the frequent use of third-party tools and occasional movement of applications; following an international standard makes sense

To contact the author:

John King

King Training Resources

6341 South Williams Street

Littleton, CO 80121-2627 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com

Paper & Sample Code: www.kingtraining.com

Thanks for your attention!