



Oracle 11g/10g for Developers: What You Need to Know

John Jay King

King Training Resources

john@kingtraining.com

Download this paper and code examples from:

<http://www.kingtraining.com>



- Learn new Oracle 11g and Oracle 10g features that are geared to developers
- Know how existing database features have been improved in Oracle
- Become aware of some DBA-oriented features that impact developers



- New and improved data types
- SQL improvements
- SQL*Plus improvements
- iSQL*Plus improvements
- Enhancements to PL/SQL
- Java and XML improvements



- Environment changes
- XML enhancements
- New/improved SQL statements
- New features in PL/SQL
- Java, JDBC, and SQLJ improvements
- Pro* and OCI enhancements



- Oracle provides support for the IEEE754 floating-point specification
- 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
- Both binary floating-point numbers support the special values infinity and NaN (not a number)
- binary_double and binary_float should significantly speed some calculations

	Binary-Float	Binary-Double
Maximum finite value	3.40282E+38F	1.79769313486231E+308
Minimum finite value	1.17549E-38F	2.22507485850720E-308



- Since first introduced the Oracle LOB type has been limited to 4GB (enough for most uses)
- Oracle 10g allows LOB data to be limited only by tablespace page size
- Current limit:
 - 8–128 terabytes
- Supported environments:
 - PL/SQL using DBMS_LOB
 - Java using JDBC
 - C/C++ using OCI



- **CARDINALITY** Returns number of elements in nested tables
- **CV** Current value of dimension in model clause
- **ITERATION_NUMBER** Returns iteration number in model rule
- **LNNVL** Returns TRUE if condition is FALSE or UNKNOWN
- **NANVL** Return alternate value if floating-point is NAN (Not A Number)
- **ORA_HASH** Computes hash value for given expression
- **POWERMULTISET** Return nested table of non-empty sets in nested table
- **POWERMULTISET_BY_CARDINALITY** Returns **POWERMULTISET** for given cardinality
- **PRESENTNNV** Present Value of cell in model clause (nulls converted)
- **PRESENTV** Present Value of cell in model clause
- **PREVIOUS** Returns cell value at beginning of model clause iteration



- REGEXP_INSTR INSTR using regular expression syntax
- REGEXP_REPLACE REPLACE using regular expression
- REGEXP_SUBSTR SUBSTR using regular expression
- REMAINDER Returns remainder after expression is evaluated
- SCN_TO_TIMESTAMP Returns timestamp for given SCN
- SESSIONTIMEZONE Timezone of current session
- SET Converts nested table to set by removing duplicates
- TIMESTAMP_TO_SCN Returns SCN for given timestamp
- TO_BINARY_DOUBLE Return binary_double for given char, number, binary
- TO_BINARY_FLOAT Return binary_float for given char, number, binary



- Oracle 10g adds the following SQL statements
 - ALTER DISKGROUP
 - CREATE DISKGROUP
 - DROP DATABASE
 - DROP DISKGROUP
 - FLASHBACK DATABASE
 - FLASHBACK TABLE
 - PURGE
- Oracle 10g removes the following SQL statement
 - CREATE TEMPORARY TABLESPACE
- Oracle 10g modifies 119 SQL statements (too many to list here)



- Oracle 10g adds two exciting changes to the ALTER SYSTEM statement
- When testing and tuning statements, the values returned by tuning tools are impacted by the actions of Oracle's shared SQL area and buffer cache – take a look!!!

ALTER SYSTEM FLUSH BUFFER_CACHE

ALTER SYSTEM FLUSH SHARED_POOL

(should probably be used in test systems only)



- 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 An exact probability test
- STATS_CROSSTAB Method used to analyze two nominal variables
- STATS_F_TEST Tests whether two variances are significantly different
- STATS_KS_TEST Compares two samples see if they are from the same population or from populations that have the same distribution
- STATS_MODE Returns most frequently occurring value from a set



- `STATS_MW_TEST` 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 variance estimates
- `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` A *t*-test of two independent groups with unequal variance (unpooled variances)
- `STATS_WSR_TEST` Test of paired samples to determine whether the median of the differences between the samples is significantly different from zero



- Given the following data:

ENAME	SAL	ENAME	SAL
SMITH	800	ALLEN	1600
JAMES	950	CLARK	2450
ADAMS	1100	BLAKE	2850
WARD	1250	JONES	2975
MARTIN	1250	SCOTT	3000
MILLER	1300	FORD	3000
TURNER	1500	KING	5000

- Note the difference between AVG and MEDIAN

```
select avg(sal) average,median(sal) median
      from emp;
```

AVERAGE	MEDIAN
2073.21429	1550



- Oracle 10g Release 1 improved support for XML data and the XMLtype datatype including:
 - Export/Import support
 - SQL*Loader support for both structured and unstructured XMLtype data
 - DBMS_AQ support, globalization support
 - C and C++ APIs allow XML modification
 - DBMS_XMLGEN allows turning off "pretty print"
 - Hierarchical queries (CONNECT) via DBMS_XMLGEN.newContextFromHierarchy
 - Schema evolution via DBMS_XMLSCHEMA (see next page)



- Schema evolution via DBMS_XMLSCHEMA
 - Before Oracle 10g XML registered schemas could not be modified (evolved)
 - Oracle 10g schema "evolution" uses a PL/SQL procedure named CopyEvolve() which is part of the DBMS_XMLSCHEMA package
 - CopyEvolve() copies an XMLType document to temporary tables, drops and re-registers the XML schema, then copies the XMLType data into the new XMLTypes
 - CopyEvolve() has some limits: indexes, triggers, and constraints dependent upon the schemas are lost and must be recreated
 - Changing a top-level element involves additional processing detailed in the Oracle documentation



- Oracle 10g Release 2 further enhances support for XML:
 - XMLtype is fully supported in Java, C, and C++
 - XML options support XSLT 2.0 with XPath functions and operators
 - Built-in XML support includes a JAXB compiler and XQuery support
 - XMLQUERY and XMLTABLE functions support XQuery
 - XMLQUERY builds XML data, queries XML data and relational data using XQuery
 - XMLTABLE creates relational tables and columns from the results of XQuery (COLUMN_VALUE pseudo-column used to retrieve data from XMLTABLE values)
 - XPATH processing has been improved enhancing speed
 - UpdateXML() function is joined by InsertXML(), AppendChildXML(), InsertXMLBefore(), and DeleteXML()



- 10g Release 2 further enhances support for XML (cont):
 - SOAP services are supported in C and C++
 - More SQL/XML (SQX) 2003 functions added including: XMLPI(), XMLComment(), XMLRoot(), XMLSerialize(), XMLCDATA(), and XMLParse()
 - Enterprise Manager Web Console includes the ability to work with XMLType objects
 - Oracle HTTPS server supports XML DB
 - Oracle XML Developers Kit (XDK) PL/SQL packages have been deprecated: XMLDOM(), XMLPARSER(), and XSL_PROCESSOR() have been replaced by DBMS_XMLDOM(), DBMS_XMLPARSER(), and DBMS_XSLPROCESSOR()



- XML Publisher is a powerful report generation tool allowing creation of standardized reports containing Oracle (and other) data quickly and easily (replacement for SQL*Reports?)
- XML Publisher supports:
 - Creating Reports and Report Standardization
 - Report, Data, and Translation Templates
 - Support for regular files, XML files, and database data
 - Online Report Editor, Query Builder, and Analyzer
 - Simple Charts & Formatting Options
 - Row, Column, and Cell formatting
 - Number and Date formatting (e.g internationalization)
 - Advanced reporting via XSL, SQL, and XSL-FO (including PDF)
- Originally intended to support Oracle' various ERP tools XML Publisher is now available as a separate product



- Insert, Update, Delete, and Merge add ERROR logging allowing you to capture DML errors and log them

```
INSERT ... /* or UPDATE, DELETE, MERGE */  
LOG ERRORS  
  [ INTO [schema.] table ]  
  [ (simple_expression) ]  
  [ REJECT LIMIT { integer | UNLIMITED } ]
```

- Default error table defined by DBMS_ERRLOG package: ERR\$_ followed by first 25 characters of DML target table
- Simple expression is value to be used as statement tag (may be result of SQL function call)
- Reject limit default is zero



- Oracle provides a PL/SQL packaged procedure to create the logging table (for each table to be logged)

```
execute DBMS_ERRLOG.CREATE_ERROR_LOG('myemp', 'myemplog');
```

- myemp Table DML is being applied to
- myemplog Logging table for rejected rows
- Creates a database table containing:
 - ORA_ERR_NUMBER\$ Error number
 - ORA_ERR_MESG\$ Error message
 - ORA_ERR_ROWID\$ Rowid of impacted rows
 - ORA_ERR_OPTYP\$ Operation type (I,U,D,M)
 - ORA_ERR_TAG\$ Text from LOG_ERRORS
 - All column values (good & bad) as varchar2(4000)



```
insert into emp
  select * from myempbig
  log errors into myemplog ('Log test3')
  reject limit unlimited;
0 rows created.
```

```
ORA_ERR_NUMBER$      12899
ORA_ERR_MESG$        ORA-12899: value too large for column
                      "JOHN"."EMP"."JOB" (actual: 13, maximum: 9)
ORA_ERR_ROWID$
ORA_ERR_OPTYP$       I
ORA_ERR_TAG$         Log test3
EMPNO                6543
ENAME                STEPHENSON
JOB                  WEB DEVELOPER
MGR                  7369
HIREDATE             03-SEP-06
SAL                  3000
COMM
DEPTNO               40
```



- 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 dual and implements an operation called “fast dual” greatly speeding access
- Here is the “Explain” output from a simple query:

```
select sysdate from dual;
```

```
-----
```

Id	Operation	Name	Rows	Cost (%CPU)	Time
0	SELECT STATEMENT		1	2 (0)	00:00:01
1	FAST DUAL		1	2 (0)	00:00:01

```
-----
```



- The login.sql and glogin.sql files are automatically executed upon entering SQL*Plus,
- 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...



- SET SERVEROUTPUT ON now works immediately within PL/SQL block where executed
- DBMS_OUTPUT.PUT_LINE text line maximum increased from 255 to 32767 bytes
- Recycle Bin keeps deleted database objects until Purged
- DESCRIBE automatically attempts to validate invalid objects before display
- White space now allowed in file names
- Substitution variables allowed in SET PROMPT
- Three pre-defined SQL*Plus variables added:
 - `_DATE` Current date or a user defined fixed string.
 - `_PRIVILEGE` Privilege level of connection (AS SYSDBA, AS SYSOPER or blank)
 - `_USER` Currently connected userid
- APPEND, CREATE, REPLACE extensions to SPOOL



- It is common for PL/SQL developers to use the `DBMS_OUTPUT.PUT_LINE` procedure to write to the console during testing and debugging
- To enable output from `DBMS_OUTPUT.PUT_LINE` you must enable `SERVEROUTPUT`
- In Oracle 10g this command has been enhanced to include a default of `UNLIMITED` buffer size eliminating the need to specify a buffer size
- You may also specify “`WORD_WRAPPED`” to cause `DBMS_OUTPUT.PUT_LINE` output to be wrapped at clear word breaks

```
set serveroutput on size 1000000 -- size limited
set serveroutput on size unlimited -- size unlimited
set serveroutput on -- size unlimited (default)
set serveroutput on size unlimited format word_wrapped
```



- SQL*Plus now provides an “oops” capability for object drops
- 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;
```



- Careful! Dropping tables no longer really drops them... This might be a problem for applications with lots of “temp”-type tables (consider drop ... purge)

```
drop table myTable;
```

```
show recyclebin
```

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

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

```
drop table myTable purge;  
purge recyclebin;
```



- Dropping a table:
 - Removes indexes on the dropped table
 - Invalidates views, procedures, functions, and other objects dependent upon the table
- After using “FLASHBACK TABLE”
 - Indexes are restored with different names
 - Primary key constraints are restored using a different name
 - Foreign key constraints are not restored
 - Views, procedures, functions, and other objects dependent upon the table will work once validated (may be done manually or automatically)



- Oracle now has three 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



```
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$');
```



- Here are two statements that generate exactly the same output and nearly the same execution plan

```
select prod_id
       , substr(prod_name,1,20) prod_name
       , substr(prod_desc,1,30) prod_desc
from sh.products
where prod_name like ('E%')
      or prod_name like ('P%')
order by prod_id;
```

```
select prod_id
       , substr(prod_name,1,20) prod_name
       , substr(prod_desc,1,30) prod_desc
from sh.products
where regexp_like (prod_name, '^E|^P')
order by prod_id;
```



- The SQL MODEL clause is a powerful extension of the SELECT statement
- MODEL provides the ability to present the output of a SELECT 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



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> ... )
```



```

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['Zoopert',2002] = sales['Zoopert',2001]
        + sales['Zoopert',2000],
        sales['HulaWhirl',2002] = sales['HulaWhirl',2001],
        sales['HulaZoop Pkg',2002] = sales['Zoopert',2002]
        + sales['HulaWhirl',2002])
  ORDER BY country, prod, year;

```

COUNTRY	PROD	YEAR	SALES
-----	-----	-----	-----
Canada	HulaZoop Pkg	2002	92613.16
Canada	Zoopert	2002	9299.08
Canada	HulaWhirl	2002	83314.08
Germany	HulaZoop Pkg	2002	103816.6
Germany	Zoopert	2002	11631.13
Germany	HulaWhirl	2002	92185.47



- The statement on the preceding page calculates sales values for two products and defines sales for a new product based upon the other two products
 - 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
 - First rule defines sales of “ZooperT” game in 2002 as the sum of its sales in 2000 and 2001
 - The second rule defines sales for “HulaWhirl” in 2002 to be the same value they were for 2001
 - Third rule defines "HulaZoop Pkg" that is the sum of the ZooperT and HulaWhirl values for 2002 -- the rules for ZooperT and HulaWhirl must be executed before the HulaZoop Pkg rule



- MERGE now allows:
 - Specification of either update, or insert, or both
 - Deletion of rows during update

```
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
        delete
        where bonus.sal > 3999 -- values after
merge
  when not matched
  then insert (ename,job,sal,comm)
  values
    (emp.ename,emp.job,emp.sal,emp.comm);
```



- For years Oracle SQL has allowed “sampling” to occur using randomized value
- Beginning with Oracle 10g you may now use a SEED to get the same (pretty close anyway) randomized value each time

```
select count(*) nbr_rows
  from sh.sales;                                NBR_ROWS = 918843
select count(*) * 10 nbr_rows
  from sh.sales sample(10);                    NBR_ROWS = 921400
select count(*) * 10 nbr_rows
  from sh.sales sample(10);                    NBR_ROWS = 917400
select count(*) * 10 nbr_rows
  from sh.sales sample(10) seed(1);           NBR_ROWS = 913280
select count(*) * 10 nbr_rows
  from sh.sales sample(10) seed(1);           NBR_ROWS = 913280
```



- Oracle 10g R2 added the ability to conditionally include PL/SQL code at compile time
- PL/SQL conditional compilation occurs before the rest of PL/SQL compilation
- Using the “\$” delimiter PL/SQL may include:
 - **Conditional directives** tell the compiler to choose between different code fragments using: \$IF, \$THEN, \$ELSE, \$END, and \$ELSIF
 - **Inquiry directives** use environment values to determine whether something should be included such as: PL/SQL compile parameters and flags (two predefined: PLSQL_Unit & PLSQL_Line)
 - **Error Directives** allow a developer to force a compile error using the specified VARCHAR2 string (may be concatenation)



- The `plsql_ccflags` session variable may be set to include name/value pairs

```
alter session set plsql_ccflags = 'hoohah:true';
```

- Conditional code may test `plsql_ccflags` as shown below

```
begin
    $if $$hoohah $then
        dbms_output.put_line('hoo hah!');
    $end
    dbms_output.put_line('testing');
end;
/
```



- Oracle Application Express (APEX; formerly HTML DB) is a complete web development and deployment environment built into Oracle 10g
- HTML DB is based upon the home-grown software that helped make Tom Kyte's "Ask Tom" website so powerful
- HTML DB 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
- 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



- Some of the built-in features in Oracle Application Express include:
 - Page Rendering and Processing Engine Rather than generating code, Oracle HTML DB stores user interface properties and data access and logic behaviors in an application definition; when an HTML DB 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



- Oracle has released the free SQL Developer tool (formerly known as Project Raptor) and it is super!
- If you've used the Free Toad tool this tool will feel familiar plus it has more features and supports Oracle 10g too!
- SQL Developer is a Java-based GUI tool (works in Windows, Unix, and Linux)
 - Power of SQL*Plus without a local Oracle client installation
 - Intuitive software
 - Provides features often found only in expensive third-party software including:
 - GUI browsing of database objects
 - Debugging complete with breakpoints
 - Query execution
 - Database reporting
 - Multiple connections



- SQL Developer is freely downloadable from Oracle using the following URL:

http://www.oracle.com/technology/products/database/sql_developer/index.html



Oracle SQL Developer

File Edit View Navigate Run Debug Source Tools Help

Connections DBConnection1 GOOFY GOOFY Snippets

```

1 CREATE OR REPLACE
2 function goofy
3 (inval in number)
4 return number
5 is
6     myvar1 number := 1;
7     myvar2 number := 2;
8     myvar3 number := 0;
9 begin
10    if inval = myvar1
11    then
12        return 1;

```

Optimizer Hints

- ALL_ROWS
- APPEND
- CACHE(table-name)
- CLUSTER(table-name)
- CURSOR_SHARING_EXA
- DRIVING_SITE(table-name)
- DYNAMIC_SAMPLING(tab
- FACT(table-name)
- FIRST_ROWS(n)
- FULL(table-name)
- HASH(table-name)
- INDEX(table-name index-)
- INDEX_COMBINE(table-na
- INDEX_FFS(table-name in
- INDEX_JOIN(table-name in

Messages - Log Breakpoints X

Saved nodes(1) Editing



- ORDER BY has always been the only way to guarantee output sequence
- Some developers have been "sloppy" and **assumed** that since some statements seemed to be solved via sort that an additional sort was not needed...
- Oracle 10g DOES NOT always return rows in the same way earlier releases did
- The two simple statements below would have shown sorted output in earlier releases; without an ORDER BY you might get surprised!

```
select distinct ename from emp;  
select deptno,count(*) from emp  
group by deptno;
```



- iSqlPlus and SQLPLUSW gone
(SQL*Plus & SQL Developer still there)
- Virtual Columns
- XML DB Binary XMLTYPE
- SQL Pivot/Unpivot
- REGEXP_COUNT
- PL/SQL compiler enhancement
- Assign sequence numbers in PL/SQL
- PL/SQL CONTINUE
- Trigger improvements
- New JDBC driver support Java 5 (1.5) & 6



- Oracle11g does not include iSQL*Plus
- Oracle 11g does not include the windows version of SQL*Plus (sqlplusw.exe)
- Oracle 11g does still include SQL*Plus (command line)
- Oracle 11g fully supports Oracle SQL Developer (introduced in Oracle 10g)
- Oracle SQL Developer is Oracle's suggested mechanism for SQL and PL/SQL development
- SQL*Plus has been enhanced to deal with BLOB, CLOB, and BFILE data more effectively



- Oracle continues its XML leadership in Oracle 11g
- Biggest change is the addition of a new “binary” XMLType
 - “binary xml” is a third method for storing XML data in the database
 - “structured” and “unstructured” XMLType still supported
 - Oracle 11g’s XML processors includes a binary XML encoder, decoder, and token manager
 - XML 1.0 text may be parsed via SAX events with or without a corresponding schema into “binary” XML form
 - “binary” XMLType allows optimization of some XML applications by reducing memory and CPU expense



- Oracle 11g provides a new, more-secure, faster mechanism for storing Large Objects (e.g. XMLType data)
- LOB column specifications in CREATE TABLE or ALTER TABLE include STORE AS SECUREFILE
- SECUREFILE provides compression and encryption for Large Objects (LOBs)
 - Oracle 11g will detect duplicate LOB data and conserve space by only storing one copy ("de-duplication" if SECUREFILE is specified).
 - PL/SQL packages and OCI functions have been added to take advantage of SECUREFILE LOBs
 - SECUREFILE lobs provide higher performance through reduced size and resource use.



- Beginning with Oracle 11g tables may now include virtual columns
- Virtual columns obtain their value by evaluating an expression that might use:
 - Columns from the same table
 - Constants
 - Function calls (user-defined functions or SQL functions)
- Virtual columns might be used to:
 - Eliminate some views
 - Control table partitioning (DBA stuff)
 - Manage the new "binary" XMLType data
- Virtual columns may be indexed!



```
CREATE TABLE NEWEMP
  (EMPNO NUMBER(4) NOT NULL,
   ENAME VARCHAR2(10),
   JOB VARCHAR2(9),
   MGR NUMBER(4),
   HIREDATE DATE,
   SAL NUMBER(7, 2),
   COMM NUMBER(7, 2),
   INCOME NUMBER(9,2)
     GENERATED ALWAYS
     AS (NVL("SAL",0)+NVL("COMM",0))
     VIRTUAL,
   DEPTNO NUMBER(2));
```

- Datatype defaults if not specified (based upon expression)
- Expression result appears as data in table but is “generated always” (whether or not specified in table definition)
- “VIRTUAL” is not required, but adds clarity



- Oracle 11g also allows specification of Virtual Columns via ALTER TABLE

```
alter table myemp  
  add (totpay as  
        (nvl(sal,0)+nvl(comm,0)));
```



- Oracle joins other vendors by adding the PIVOT clause to the SELECT statement
- Adding a PIVOT clause to a SELECT allows rotation of rows into columns while performing aggregation to create cross-tabulation queries
- The PIVOT clause:
 - Computes aggregations (implicit GROUP BY of all columns not in PIVOT clause)
 - Output of all implicit grouping columns followed by new columns generated by PIVOT
- UNPIVOT performs the same activity but converts columns into ROWS (does not “undo” PIVOT)
- Clever developers have used PL/SQL and/or CASE to achieve PIVOT results before, but now it is part of Oracle's standard SQL



```
select * from
  (select job,deptno,income from newemp) query1
  pivot (avg(income)
        for deptno in (10 AS ACCOUNTING,
                      20 AS RESEARCH,
                      30 AS SALES))
  order by job;
```

Job	ACCOUNTING	RESEARCH	SALES
ANALYST	30000		
CLERK	13000	9500	9500
MANAGER	24500	29750	28500
PRESIDENT	50000		
SALESMAN		19500	



```
select * from pivot_emp_table
  unpivot include nulls
    (avgpay for dept in (ACCOUNTING,RESEARCH,SALES))
  order by job;
```

JOB	DEPT	AVGPAY
ANALYST	ACCOUNTING	
ANALYST	RESEARCH	30000
ANALYST	SALES	
/** more rows **/		
SALESMAN	ACCOUNTING	
SALESMAN	RESEARCH	
SALESMAN	SALES	19500



- New functions have also been added to Oracle 11g including:
 - CUBE_TABLE Extracts two-dimensional table from a cube or dimension
 - XMLCAST Cast XML data to SQL datatype
 - XMLEXISTS Determine if XQuery returns values
 - XMLDIFF Used to compare two XMLType documents
 - XMLPATCH Used to patch an XMLType document



- Oracle 11g's changes to PL/SQL are very interesting to the developer
 - PL/SQL has been improved to include all of the XMLType, BLOB, Regular Expression, and other functionality added to SQL
 - Improvements have been made to the compiler
 - New new PL/SQL data types
 - Sequence number use is easier
 - “continue” added for loop control
 - CALL syntax has improved



- In previous releases, the PL/SQL compiler required a standalone “C” compiler
- Oracle 11g now provides a native compiler for PL/SQL eliminating the need for a separate compiler



- Oracle 11g adds three new PL/SQL datatypes: `Simple_integer`, `Simple_float`, `Simple_double`
 - The three new datatypes take advantage of native compilation features providing faster arithmetic via direct hardware implementation
 - `SIMPLE_INTEGER` provides a binary integer that is neither checked for nulls nor overflows
 - `SIMPLE_INTEGER` values may range from -2147483648 to 2147483647 and is always NOT NULL
 - Likewise, `SIMPLE_FLOAT` and `SIMPLE_DOUBLE` provide floating point without null or overflow checks



```
declare
-- mytestvar pls_integer := 2147483645;
  mytestvar simple_integer := 2147483645;
begin
  loop
    mytestvar := mytestvar + 1;
    dbms_output.put_line('Value of mytestvar is now '
                        || mytestvar);
    exit when mytestvar < 10;
  end loop;
end;
```

Results in:

```
Value of mytestvar is now 2147483646
Value of mytestvar is now 2147483647
Value of mytestvar is now -2147483648
```



- If the “mytestvar” variable is switched to PLS_INTEGER, an ORA-1426 NUMERIC OVERFLOW exception occurs

Error report:

```
ORA-01426: numeric overflow
```

```
ORA-06512: at line 7
```

```
ORA-01426. 00000 - "numeric overflow"
```

```
*Cause:      Evaluation of an value expression causes an  
overflow/underflow.
```

```
*Action:     Reduce the operands.
```

```
Value of mytestvar is now 2147483646
```

```
Value of mytestvar is now 2147483647
```



- Sequence values NEXTVAL and CURRVAL may be use in PL/SQL assignment statement

```
myvar := myseq.nextval;
```



- CONTINUE “iterates” a loop; branching over the rest of the code in the loop and returning to the loop control statement

```
begin
  dbms_output.put_line('Counting down to blastoff!');
  for loopctr in reverse 1 .. ctr loop
    if loopctr in (4,2) then
      continue;
    end if;
    dbms_output.put_line(to_char(loopctr));
  end loop;
  dbms_output.put_line('Blast Off!');
end;
Counting down to blastoff!
6
5
3
1
Blast Off!
```

– Values “4” and “2” do not appear in the output



- REGEXP_COUNT counts the number of times a pattern occurs in a source string

```
select  ename,regexp_count(ename,'l',1,'i') from emp;
SMITH      0
ALLEN      2
WARD       0
JONES      0
MARTIN     0
BLAKE      1
/** more rows **/
MILLER     2
```

- string expression and/or column to match pattern
- Regular Expression pattern
- Beginning position in the source string (default=1)
- Match parameters (i = case insensitive, c = case sensitive, m = multiple line source delimited by '^' or '\$', n = matches '.' newline characters (default no), and x = ignore whitespace characters (default is to match))



- PL/SQL allows function and procedure parameters to be specified in two ways; by position and by name
- With Oracle 11g SQL, parameter types may now be mixed
- Given the following function:

```
CREATE OR REPLACE
FUNCTION TEST_CALL (inval1 IN NUMBER, inval2 IN NUMBER,
    inval3 IN NUMBER) RETURN NUMBER AS
BEGIN
    RETURN inval1 + inval2 + inval3;
END TEST_CALL;
/
```

- The following calls all now work:

```
test_call(vara, varb, varc)
test_call(inval3=>varc, inval1=>vara, inval2=>varb)
test_call(vara, inval3=>varc, inval2=>varb)
```



- Both Oracle 11g and Oracle 10g add significant new functionality to the already robust database environment
- While an emphasis is sometimes placed on the features of Oracle that support the Data Base Administrator, this paper shows many Developer-oriented features of great usefulness



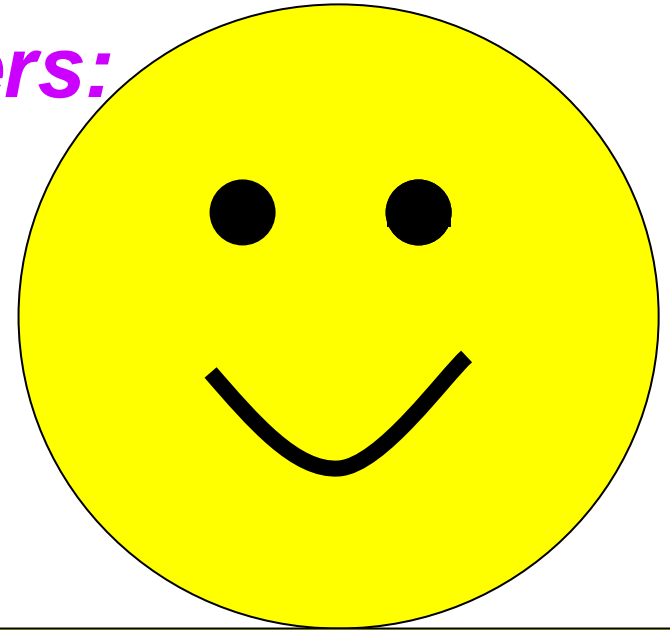
Training Days 2008

Mark your calendar for:

February 13-14 2008!



Oracle 11g/10g for Developers: What You Need to Know



Thanks for your attention!

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

Today's slides and examples are on the web:
<http://www.kingtraining.com>