Oracle 12c New Features For Developers & DBAs

Presented by: John Jay King

Download this paper from: http://www.kingtraining.com

Copyright @ 2017, John Jay King

Session Objectives

- Learn new Oracle 12c 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

Copyright @ 2017, John Jay King

Who Am I?

- John King – Partner, King Training Resources
- Oracle Ace Director
- Member Oak Table Network
- Providing training to Oracle and IT community for over 25 years – http://www.kingtraining.com
- "Techie" who knows Oracle, ADF, SQL, Java, and PL/SQL pretty well (along with many other topics)
- Member of AZORA, ODTUG, IOUG, and RMOUG

Copyright @ 2017, John Jay King

Arizona, USA

Copyright @ 2017, John Jay King

“Recent” Releases

• Oracle 11g R1	August 2007
• Oracle 11g R2	September 2009
• Oracle 12c R1	June 2013
• Oracle 12c R1.0.2	June 2014
• Oracle 12c R2	Fall 2016 for Cloud Everyone else 2017?

Copyright @ 2017, John Jay King

Oracle 12c – Exciting DBA Stuff

- Oracle In-Memory Database
- Multi-tenant Architecture:
  (first architecture change to Oracle since V6 in 1988!)
  - Container Database (CDB)
  - Pluggable Database(s) (PDB)
- Performance Improvements:
  - Improved optimization
  - Enhanced Statistics & New Histograms
  - "Heat" maps
  - Adaptive Execution Plans
- More cool stuff (review OOW announcements…)

Copyright @ 2017, John Jay King

What is Multi-Tenant?

- Multi-Tenant architecture is designed to achieve two specific goals:
  - Improved performance
  - Ease of management and consolidation
- Multi-tenant has two types of databases:
  - Container Database (CDB) - "Main" database contains up to 252 PDBs (Oracle EE) or exactly one PDB (Oracle SE)
  - Pluggable Database (PDB) - "Application" databases containing application/function-specific users and data

Copyright @ 2017, John Jay King

CDB & PDBs Share

- Single SGA
- Single set of database processes
- Single database to patch and/or upgrade (CDB)
- Single database to backup (CDB)
- Single configured container as standby database
- Single configuration for High-Availability, Data Guard, or RAC

Copyright @ 2017, John Jay King

What's the Big Deal?

- Less memory required
- Less space required
- For example:
  - Before Oracle 12c: 30 database instances might require approximately 20 background processes (each) to run; or, about 600 processes
  - 12c: 30 PDBs share 20 background processes (that's it)

Copyright @ 2017, John Jay King

Reasons to use Multi-Tenant

- Fast provisioning of new database or copy of existing database
- Fast redeployment to new platform
- Quickly patch and upgrade database version ONCE for all PDBs
- Patch/upgrade unplugging PDB from one CDB and plugging into CDB at later version
- Machine can run more databases as PDBs
- No changes required to user applications

Copyright @ 2017, John Jay King

Oracle In-Memory Database

- Oracle introduced new pay-for “In Memory Option” as part of Oracle Database 12.1.0.2 (so far only for Oracle EE)
- Oracle database normally stores data in tables; one row after another (on disk, pulled into memory for processing)
- In Memory Option ALSO stores table data in columnar format in memory
- Data in columnar format can speed some queries significantly

Copyright @ 2017, John Jay King

In-Memory "Secret Sauce"

- Both row and columnar formats are in memory at the same time; the optimizer decides which data store will work best
- SGA "In-Memory Area" (new pool) stores as much as will fit
- Tables are added to memory with ALTER TABLE xxx IN MEMORY - pivots data and adds to columnar store
- In-Memory is part of database; transparent to applications once tables added

Copyright @ 2017, John Jay King

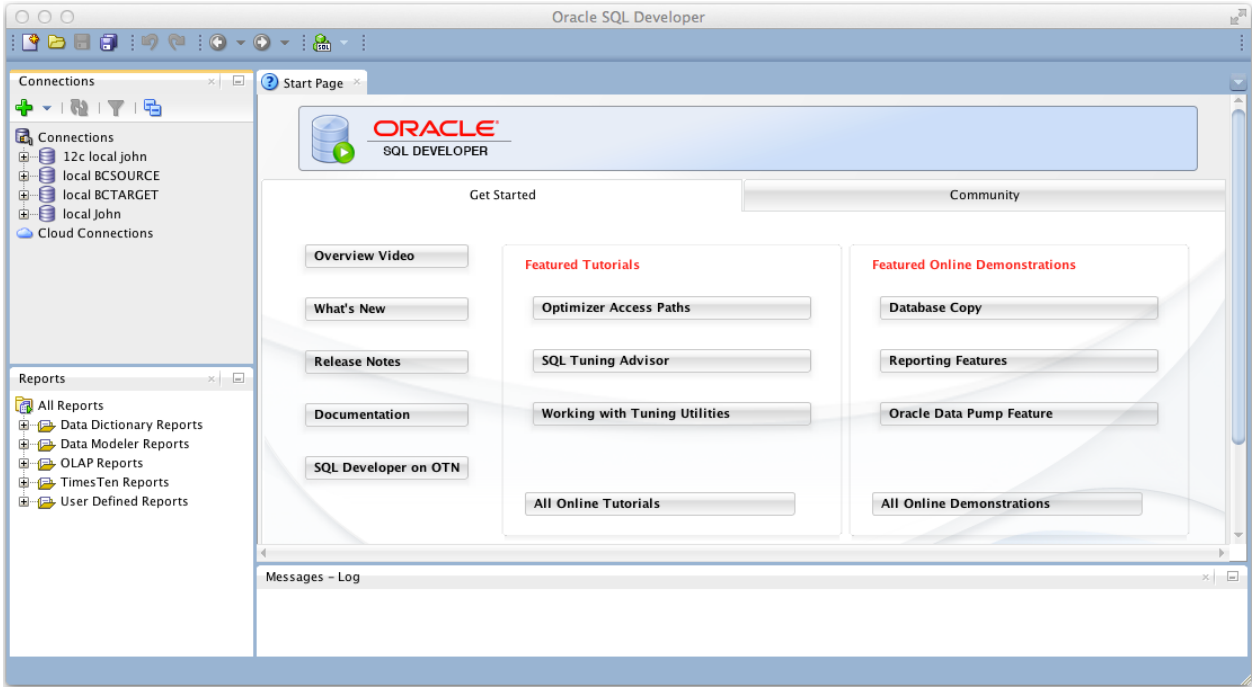
Oracle 12c New Features

- SELECT improvements: Top-n & Pagination, pattern matching, outer join improvements
- Table definition improvements: expanded columns, identity columns, default improvements, invisible columns
- PL/SQL in WITH clause
- Temporal Validity
- Online DML operations
- Truncate CASCADE
- EBR improvements
- JSON in the database (12.1.0.2)

Copyright @ 2017, John Jay King

New SQL Developer

- Oracle SQL Developer 4.1.5 is now available for download
- Many new features & supports Oracle 12c (still a couple of "wrinkles" …)



Copyright @ 2017, John Jay King

Top-N & Pagination

- Oracle 12c adds “top-n” type queries and paginated queries
  - FETCH FIRST/LAST nn ROWS
    FIRST/LAST n PERCENT ROWS
  - OFFSET nn ROWS
- Optimizer uses analytics under the covers to make this work

Copyright @ 2017, John Jay King

Top-N: Base Query

- Original query; note row sequence

```
select ename,sal from emp order by sal desc;
ENAME              SAL
---------- ----------
KING             5000
FORD             3000
SCOTT            3000
JONES            2975
BLAKE            2850
CLARK            2450
ALLEN            1600
TURNER           1500
MILLER           1300
WARD             1250
*** more ***
SMITH             800
```

Copyright @ 2017, John Jay King

Top-N: Using Rownum

- Original query uses “rownum” – note sequence of data (oops, wrong rows…)

```
select ename,sal from emp
   where rownum < 5 order by sal desc;
--
ENAME              SAL
---------- ----------
JONES             2975
ALLEN             1600
WARD              1250
SMITH              800
```

  – Note use of rownum; RANK, or DENSE_RANK in dynamic view (select from (subquery)) may be used to get correct rows

Copyright @ 2017, John Jay King

Top-N: First nn ROWS

- Here the first five rows (by value) are selected; note no need for analytics

```
select ename,sal from emp
    order by sal desc
    fetch first 5 rows only;

ENAME              SAL
---------- ----------
KING             5000
SCOTT            3000
FORD             3000
JONES            2975
BLAKE            2850
```

Copyright @ 2017, John Jay King

Pagination

- The OFFSET clause may start processing at a given row; when (optionally) paired with FETCH allows pagination in query

```
select ename,sal from emp
   order by sal desc
   offset 2 rows
   fetch first 5 rows only;

ENAME              SAL
---------- ----------
FORD             3000
JONES            2975
BLAKE            2850
CLARK            2450
ALLEN            1600
```

Copyright @ 2017, John Jay King

Top-N: Percentage

- Top-N may use a percentage rather than a number of rows

```
select ename,sal from emp
   order by sal desc
   offset 2 rows
   fetch first 5 percent rows only;

ENAME              SAL
---------- ----------
SCOTT            3000
```

Copyright @ 2017, John Jay King

APPROX_COUNT_DISTINCT

- Oracle 12.1.0.2 documented aggregate first added internally to Oracle 11g
- Provides approximate value without actually processing all of the rows

```
select count(distinct cust_id) from sh.sales;
COUNT(DISTINCTCUST_ID)
----------------------
7059
      Elapsed: 00:00:00.614
```

```
select approx_count_distinct(cust_id) from sh.sales;
APPROX_COUNT_DISTINCT(CUST_ID)
------------------------------
7014
      Elapsed: 00:00:00.074
```

Matching Patterns

- Enhanced ability to use Regular Expressions enabled by Oracle 12c's MATCH_RECOGNIZE
- Using syntax similar to the MODEL clause and Analytics; rows may be compared to other rows using Regular Expressions (beyond capabilities of LAG/LEAD)

Copyright @ 2017, John Jay King

MATCH_RECOGNIZE

- MATCH_RECOGNIZE includes:
  - PARTITION — Segregate data
  - ORDER BY — Order with partitions
  - MEASURES — Define output columns
  - AFTER — Return single/multiple rows
  - PATTERN — Define regular expression
  - DEFINE — Specify expression tags

Copyright @ 2017, John Jay King

Sample MATCH_RECOGNIZE

- The code on the following pages creates a report illustrating sales patterns (highs & lows) for a specific product over time
- Given five periods of data showing sales of:

2229 1191 1333 887 2148

Copyright @ 2017, John Jay King

Sample Code 1

- SELECT uses query in FROM clause to aggregate SH.SALES data by prod_id and day (truncated time_id)

```
select * from
 (select prod_id,trunc(time_id)time_id,
    sum(amount_sold) amount_sold from sh.sales
    where prod_id = 148
       and extract(year from time_id) in (2000,2001)
       group by prod_id, trunc(time_id))
```

Copyright @ 2017, John Jay King

Sample Code 2

```
match_recognize (
   partition by prod_id
   order by time_id
   measures to_char(strt.time_id,'yyyy-mm-dd') as
start_date,
   to_char(last(down.time_id),'yyyy-mm-dd') as bottom_date,
   to_char(last(up.time_id) ,'yyyy-mm-dd') as   end_date,
   last(round(down.amount_sold)) as bottom_amt,
   last(round(up.amount_sold)) as end_amt
   --one row per match
   after match skip to last up
   pattern (strt down+ up+)
   define
     down as down.amount_sold < prev(down.amount_sold),
     up as up.amount_sold > prev(up.amount_sold)
   ) matcher
order by matcher.prod_id, matcher.start_date
```

Copyright @ 2017, John Jay King

Results

- Here are the results and a sample of the data to see what happened
- Two result rows:

```
148 2000-01-18 2000-01-23 2000-01-27     1191      1333
148 2000-01-27 2000-02-02 2000-02-14      887      2148
```

- Matching base data rows:

```
148 18-JAN-00          2229
148 23-JAN-00          1191
148 27-JAN-00          1333
148 02-FEB-00           887
148 14-FEB-00          2148
```

Copyright @ 2017, John Jay King

Outer Join Improvements

- Oracle 12c expands the use of the "traditional" Oracle Outer Join syntax (+) to make it more useful
- The (+) notation to create null rows may now be used for multiple tables & columns

Copyright @ 2017, John Jay King

Outer Join Example

```
select region_name, country_name, department_name, city,
count(employee_id) nbr_emps
  from hr.regions r, hr.countries c, hr.locations l,
       hr.departments d, hr.employees e
  where r.region_id = c.region_id(+)
    and c.country_id = l.country_id(+)
    and l.location_id = d.location_id(+)
    and d.department_id = e.department_id(+)
    group by region_name,country_name,department_name, city
    order by region_name,country_name,department_name, city
```

Copyright @ 2017, John Jay King

CROSS & OUTER APPLY

- Oracle 12c adds the ability to JOIN values in a generated table collection to regular tables using correlated column values:
  - CROSS APPLY — Join table to generated collection when values match
  - OUTER APPLY — Join table to generated collection when values match and create matches for non-match rows too

Copyright @ 2017, John Jay King

Example APPLY - Setup

```
create or replace type name_table_type
       as table of varchar2(100);
create or replace function department_employees
(in_department_id varchar2)
   return name_table_type
is
  mynames name_table_type;
begin
  select cast(collect(last_name || ', ' || first_name)
                    as name_table_type)
     into mynames
     from hr.employees
     where department_id = in_department_id;
  return mynames;
end;
/
```

Copyright @ 2017, John Jay King

Example APPLY

```
select *
  from hr.departments d
       cross apply
       department_employees(d.department_id) dept_emps;

select *
  from hr.departments d
       outer apply
       department_employees(d.department_id) dept_emps;

select department_name
      ,department_employees(department_id) deptemps
  from hr.departments;
```

Copyright @ 2017, John Jay King

LATERAL Inline Views

- Lateral inline views introduce a new keyword allowing correlated references to other tables in a join
  - Correlated tables appear to the left of the inline view in the query's FROM list
  - Correlation names may be used anywhere within the inline view a correlation name usually occurs
    (e.g. SELECT, FROM, WHERE, …)

Copyright @ 2017, John Jay King

Example Lateral Inline View

- Here is an example using a lateral inline view; this syntax would fail without the "LATERAL" keyword

```
select last_name,first_name,department_name
   from hr.employees e, LATERAL(select *
                                from hr.departments d
                                where e.department_id
                                      = d.department_id);
```

Copyright @ 2017, John Jay King

New Column Sizes

- 12c increases max size of VARCHAR2, NVARCHAR2, and RAW to 32,767
- Stored out of line as SECUREFILE CLOB when > 4k
- Now matches PL/SQL variables
- Not default required DBA action:
  - MAX_SQL_STRING_SIZE set to EXTENDED
  - COMPATIBLE must be 12.0.0.0.0+
  - Probably requires system restart to change
  - Once set cannot be undone

Copyright @ 2017, John Jay King

Identity Columns

- Oracle has had SEQUENCES for years; the IDENTITY column allows use of a SEQUENCE as part of a column definition (much like some competitor databases)
  - Use “GENERATED AS IDENTITY” clause
  - Default starts with 1 increments by 1
  - May set values using START WITH and INCREMENT BY
  - IDENTITY column resets if table is dropped and recreated

Copyright @ 2017, John Jay King

Identity Example 1

```
create table id_test1
(id number generated as identity,
 col1 varchar2(10));
--
insert into id_test1 (col1) values ('A');
insert into id_test1 (col1) values ('B');
insert into id_test1 (col1) values ('C');
--
select * from id_test1;
 ID COL1
---------- ----------
         1 A
         2 B
         3 C
```

Copyright @ 2017, John Jay King

Identity Example 2

```
create table id_test1
(id number generated as identity (
       start with 10 increment by 11),
 col1 varchar2(10));
--
insert into id_test1 (col1) values ('A');
insert into id_test1 (col1) values ('B');
insert into id_test1 (col1) values ('C');
--
select * from id_test1;
 ID COL1
---------- ----------
        10 A
        21 B
        32 C
```

Copyright @ 2017, John Jay King

Enhanced Column DEFAULT

- Oracle 12c enhances the capabilities of column default settings
  - Columns may be set to a default when NULL values are INSERTed
  - Column default values may be based upon a SEQUENCE (.nextval or .currval)

Copyright @ 2017, John Jay King

Example Defaults

```
drop sequence default_test_seq;
drop table default_test;
create sequence default_test_seq start with 1 increment by 1;
create table default_test
(id number default default_test_seq.nextval not null,
 col1 varchar2(10) ,
 col2 varchar2(10)default on null 'N/A' not null);
insert into default_test (col1,col2) values ('A',null);
insert into default_test (col1) values ('B');
insert into default_test (col1,col2) values ('C','test');
select * from default_test;
 ID        COL1       COL2
---------- ---------- ----------
         1 A          N/A
         2 B          N/A
         3 C          test
```

Copyright @ 2017, John Jay King

Session-Specific Sequence

- CREATE SEQUENCE now offers a SESSION parameter allowing a sequence to be reset each time the Global Temporary Table is reinitialized (default is GLOBAL)

```
create sequence session_sample_seq
  start with 1 increment by 1
  session;
```

  - Rows in Global Temporary Tables exist either for the life of the session or transaction
  - While particularly useful for GTTs; session-specific sequences are NOT limited to GTTs

Copyright @ 2017, John Jay King

Invisible Columns

- Columns may be marked “INVISIBLE” in CREATE/ALTER table
- Invisible columns do not normally appear in SQL*Plus DESCRIBE or SQL Developer column display (does show in SQL Developer table column list, SQL*Plus COLINVISIBLE ON)
- Invisible columns may be inserted into or omitted from INSERT statements
- When made visible columns appear at end of table (why?? See next page)

Copyright @ 2017, John Jay King

COL$ View

- What happens when a column is marked invisible?
- The database marks column number to 0

```
SELECT c.name,c.type#,c.col#,c.intcol#,c.segcol#,
       TO_CHAR (c.property,'XXXXXXXXXXXX') AS property
FROM sys.col$ c, sys.obj$ o, sys.user$ u
WHERE c.obj# = o.obj#
AND o.owner# = u.user#
AND u.name = 'MYUSER'
AND o.name = 'MYTABLE';
```

  - Col# is set to 0
  - Property is set to x'40000020'

Copyright @ 2017, John Jay King

Invisible Column Example 1

```
drop table invisible_test;
create table invisible_test (
 id  number,
 col1 varchar2(10),
 col2 varchar2(10) invisible,
 col3 varchar2(10));
desc invisible_test;
Name Null Type
---- ---- ------------
ID        NUMBER
COL1      VARCHAR2(10)
COL3      VARCHAR2(10)
```

Copyright @ 2017, John Jay King

Invisible Column Example 2

```
insert into invisible_test
(col1,col2,col3) values (1,'a','a');
insert into invisible_test
(col1,col3) values (2,'b');
insert into invisible_test values (3,'c');
select * from invisible_test;
alter table invisible_test modify col2 visible;
desc invisible_test;
Name Null Type
---- ---- ------------
ID        NUMBER
COL1      VARCHAR2(10)
COL3      VARCHAR2(10)
COL2      VARCHAR2(10)
```

Copyright @ 2017, John Jay King

PL/SQL in WITH

- Oracle 12c allows definition of PL/SQL Functions and Procedures using SQL's Common Table Expression (WITH)
  - Defining PL/SQL locally reduces SQL-PL/SQL context-switching costs
  - Local PL/SQL overrides stored PL/SQL with the same name
  - Local PL/SQL is not stored in the database
  - Local PL/SQL is part of the same source code as the SQL that uses it
  - PL/SQL Result Cache no use in Local PL/SQL

Copyright @ 2017, John Jay King

Example PL/SQL in WITH

```
with function times_42(inval number)
 return number
as
begin
  return inval * 42;
end;
select channel_id,count(*) nbr_rows,
       sum(quantity_sold) qtysold,
       sum(times_42(cust_id)) cust42
  from sh.sales
  group by channel_id
  order by channel_id
/
```

Copyright @ 2017, John Jay King

PL/SQL UDF

- Oracle 12c allows functions to be defined using “PRAGMA UDF” to specify that a function will be used in SELECTS (behaving similar to function in WITH)
- This optimizes code for use within a SELECT or other SQL

Probably not a good option for functions also used from PL/SQL !

Copyright @ 2017, John Jay King

Example PL/SQL UDF

```
create or replace function times_42(inval number)
 return number
as
  pragma udf;
begin
  return inval * 42;
end;
/
```

Copyright @ 2017, John Jay King

How Did They Rate?

- Here's how the three options stacked up:

	1st Run	2nd Run	3rd Run
Function in WITH	0.854	0.825	0.929
Compiled Function in database	2.018	1.945	1.928
Compiled UDF Function in database	0.667	0.602	0.664

Copyright @ 2017, John Jay King

Temporal Validity

- Oracle 12c adds options to CREATE TABLE, ALTER TABLE, and SELECT allowing use of time dimensions in conjunction with FLASHBACK QUERY
  - Periods are defined using TIMESTAMP columns
  - CREATE/ALTER TABLE's PERIOD clause specifies period starting and ending times
  - SELECT statements AS OF PERIOD FOR clause allows selection of rows falling within periods

Copyright @ 2017, John Jay King

Temporal Validity Example

```
CREATE TABLE temporal_emp_test(
  employee_id NUMBER,
  last_name   VARCHAR2(50),
  start_time  TIMESTAMP,
  end_time    TIMESTAMP,
  PERIOD FOR my_time_period (start_time, end_time));
INSERT INTO temporal_emp_test
      VALUES (1000, 'King', '01-Jan-10', '30-Jun-11');
INSERT INTO temporal_emp_test
      VALUES (1001, 'Manzo', '01-Jan-11', '30-Jun-11');
INSERT INTO temporal_emp_test
      VALUES (1002, 'Li', '01-Jan-12', null);
SELECT * from temporal_emp_test AS OF PERIOD
      FOR my_time_period TO_TIMESTAMP('01-Jun-10');
SELECT * from temporal_emp_test VERSIONS PERIOD FOR
      my_time_period BETWEEN TO_TIMESTAMP('01-Jun-10')
                          AND TO_TIMESTAMP('02-Jun-10');
```

Copyright @ 2017, John Jay King

Online DDL

- Some DDL statements may be performed ONLINE in Oracle 12c, eliminating the DML lock from earlier releases
  - DROP INDEX … ONLINE
  - ALTER INDEX … UNUSABLE ONLINE
  - ALTER TABLE … SET UNUSED … ONLINE …
  - ALTER TABLE … DROP … ONLINE
  - ALTER TABLE …
    MOVE PARTITION … ONLINE
  - ALTER TABLE …
    MOVE SUBPARTITION …. ONLINE
  - ALTER DATABASE MOVE DATAFILE (….) TO (….)

Copyright @ 2017, John Jay King

TRUNCATE … CASCADE

- Oracle 12c's TRUNCATE statement allows the use of CASCADE to eliminate values in tables that are referentially connected

```
TRUNCATE TABLE ID_TEST1 CASCADE;
```

  - Child table referential security must specify "ON DELETE CASCADE" or statement will fail

Copyright @ 2017, John Jay King

UTL_CALL_STACK

- Oracle has provided PL/SQL debug aids for a long time; perhaps your shop uses one: dbms_utility.format_call_stack, dbms_utility.format_error_backtrace, or dbms_utility.format_error_stack
- Oracle 12c adds UTL_CALL_STACK providing greater insight into the stack

Copyright @ 2017, John Jay King

UTL_CALL_STACK Functions

- See documentation for a complete list of subprograms – here are a few:
  - CONCATENATE_SUBPROGRAM
    Concatenated unit name
  - DYNAMIC_DEPTH
    Number of subprograms on call stack
  - LEXICAL_DEPTH
    Lexical nesting level of subprogram
  - UNIT_LINE
    Line number in backtrace unit

Copyright @ 2017, John Jay King

Using UTL_CALL_STACK

```
create or replace procedure Print_Call_Stack
As
   DEPTH pls_integer := UTL_CALL_STACK.dynamic_depth();
   procedure printheaders is
    /* more code */
   procedure print is
     begin
      printheaders;
      for stunit in reverse 1..DEPTH loop
       dbms_output.put_line(
        rpad( UTL_CALL_STACK.lexical_depth(stunit), 10 )
        || rpad( stunit, 7)
        || rpad(to_char(UTL_CALL_STACK.unit_line(stunit),
            '99'), 9 )
        || UTL_CALL_STACK.concatenate_subprogram
      end loop;
    /* more code */
```

Copyright @ 2017, John Jay King

Anatomy of Test Package

- The example package illustrates code nested within code:

```
package body TestPkg is
  procedure proc_a is
    procedure proc_b is
      procedure proc_c is
        procedure proc_d is
          Print_Call_Stack();
```

Copyright @ 2017, John Jay King

UTL_CALL_STACK Results

```
begin TestPkg.proc_a; end;
Error report -
ORA-06501: PL/SQL: program error
ORA-06512: at "JOHN.TESTPKG", line 11
ORA-06512: at "JOHN.TESTPKG", line 14
ORA-06512: at "JOHN.TESTPKG", line 17
ORA-06512: at "JOHN.TESTPKG", line 20
ORA-06512: at line 1
06501. 00000 - "PL/SQL: program error"
*Cause: This is an internal error message. An error has
been detected in a PL/SQL program.
*Action: Contact Oracle Support Services
TESTPKG.PROC_A
TESTPKG.PROC_A.PROC_B
TESTPKG.PROC_A.PROC_B.PROC_C
TESTPKG.PROC_A.PROC_B.PROC_C.PROC_D
PRINT_CALL_STACK
PRINT_CALL_STACK.PRINT
```

Copyright @ 2017, John Jay King

12c (12.1.0.2) and JSON

- 12c patch-set 2 (12.1.0.2) adds JSON data
- JSON documents are stored as VARCHAR2, CLOB, or BLOB data type
- JSON data works with all existing Oracle features including SQL and Analytics
- 12c supports path-based queries of JSON data stored in the database, JSON Path Language, and JSON Path Expressions
- JSON is used in SQL via SQL/JSON views
- JSON documents may be indexed

Copyright @ 2017, John Jay King

JSON-XML Similarities

- JSON is text only, just like XML and thus is an excellent vehicle for data interchange– JSON and XML are both plain text
- JSON and XML are “human readable” and "self-describing”
- JSON and XML are hierarchical (data sets nested within data sets)
- JSON and XML offer validation capability; XML’s is more mature and capable today

Copyright @ 2017, John Jay King

JSON-XML Dissimilarities

- XML is verbose, JSON is shorter
- JSON does not end tags, required in XML
- JSON is quicker to read and write
- Reading XML documents requires “walking the DOM” – JSON does not
- JSON works more easily and is faster than XML when working with AJAX
- XML documents must be tested for “well-formed”-ness before processing

Copyright @ 2017, John Jay King

XML File

```
<?xml version="1.0"?>
<myBooks>
  <book>
    <name>Learning XML</name>
    <author>Eric T. Ray</author>
    <publisher>O'Reilly</publisher>
  </book>
  <book>
    <name>XML Bible</name>
    <author>Elliotte Rusty Harold</author>
    <publisher>IDG Books</publisher>
  </book>
  <book>
    <name>XML by Example</name>
    <author>Sean McGrath</author>
  </book>
</myBooks>
```

Copyright @ 2017, John Jay King

JSON File

```
{"myBooks":
  [ {"book":
     "name":"Learning XML",
     "author":"Eric T. Ray",
     "publisher":"O'Reilly"
     },
     {"book":
     "name":"XML Bible",
     "author":"Elliotte Rusty Harold",
     "publisher":"IDG Books"
     },
     {"book":
     "name":"XML by Example",
     "author":"Sean McGrath",
     "publisher":"Prentice-Hall"
     }
  ]}
```

Copyright @ 2017, John Jay King

Oracle as JSON Data Store

- JSON documents are stored in the database using existing data types
  - VARCHAR2, CLOB and BLOB for character mode JSON
  - External JSON data sources accessible through external tables
  - JSON in file system (also HDFS) can be accessed via external tables

Copyright @ 2017, John Jay King

JSON SQL

- JSON content is accessible from SQL via new operators
  - JSON_VALUE Used to query a scalar value from a JSON document
  - JSON_TABLE Used to query JSON document and create relational-style columns
  - JSON_EXISTS Used in query to see if JSON path exists in document IS JSON Used to validate JSON, usually in CHECK constraint
- JSON operators use JSON Path language to navigate JSON objects

Copyright @ 2017, John Jay King

JSON Check Constraint

```
create table deptj
(id raw(16) not null,
 dept_info clob constraint deptjson
                check (dept_info is json)
);
```

Copyright @ 2017, John Jay King

JSON and DML

```
insert into deptj values
(sys_guid(),
 '{"departments":{
   "DEPTNO": 10, "DNAME": "ACCOUNTING", "LOC": "NEW YORK",
    "deptemps": [
                 { "EMPNO": 7782,
                   "ENAME": "CLARK",
                   "JOB": "MANAGER",
                   "MGR": 7839,
                   "HIREDATE": "09-JUN-81",
                   "pay":{
                          "SAL": 2450,
                          "COMM": null},
                   "DEPTNO": "10"
                  },
                  /* more */
```

Copyright @ 2017, John Jay King

Simple JSON Query

```
select dept_info
from deptj;

DEPT_INFO
-----------------------------------------------
{"departments":{
 "DEPTNO": 10,
 "DNAME": "ACCOUNTING",
 "LOC": "NEW YORK",
 "deptemps": [
               {
                "EMPNO": 7782,
                "ENAME": "CLARK",
  **** more ****
```

Copyright @ 2017, John Jay King

Query with JSON_VALUE

```
select json_value(dept_info, '$.departments.DNAME')
from deptj;
DNAME
-----------------
ACCOUNTING
RESEARCH
SALES
OPERATIONS
```

Copyright @ 2017, John Jay King

Query with JSON_TABLE

```
select dname,ename,job,sal
from deptj, json_table(dept_info,'$.departments'
            columns (dname varchar2(15) path '$.DNAME'
            ,nested path '$.deptemps[*]'
              columns (ename varchar2(20) path '$.ENAME'
              ,job varchar2(20) path '$.JOB'
                ,nested path '$.pay'
                  columns (sal number path '$.SAL')
              )
));
DNAME          ENAME      JOB                SAL
------------ -------    --------    ----------
ACCOUNTING    CLARK     MANAGER            2450
ACCOUNTING    KING      PRESIDENT          5000
**** more ****
```

Copyright @ 2017, John Jay King

EBR Improvements

- Time does not permit detailed EBR coverage
- Edition-Based Redefinition made its debut in Oracle 11g and provides an ability to significantly reduce downtime due to changes in PL/SQL and/or SQL
- Oracle 12c removes some limitations present in 11gR2 implementation of EBR:
  - Public Synonyms may point to editioned objects
  - Materialized Views and Types may be used with editioning
  - Virtual Columns may be used with EBR

Copyright @ 2017, John Jay King

System READ Privilege

- Oracle 12.1.0.2 added two new privileges READ and READ ANY TABLE
- SELECT privilege (been there forever) also allows locking to occur
  - LOCK TABLE ...
  - SELECT ... FOR UPDATE
- READ privilege allows SELECT statements but does not allow statements to lock rows

```
GRANT READ ON xxx.yyy TO user,role;
GRAND READ ANY TABLE  TO user,role;
```

Copyright @ 2017, John Jay King

READ Privilege at Work

```
SQL> grant read on dept to t1;
```

```
SQL> select * from john.dept;
    DEPTNO DNAME    LOC
---------- -------------- -------------
        10 ACCOUNTING       NEW YORK
        20 RESEARCH    DALLAS
        30 SALES       CHICAGO
        40 OPERATIONS       BOSTON
```

```
SQL> select * from john.dept for update;
select * from john.dept for update
                      *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Copyright @ 2017, John Jay King

Merge Partitions

- Partitions may be merged easily

```
alter table ORDERS
  merge partitions P2014Q1,P2014Q2,P2014Q3,P2014Q4
      into P2014;
```

```
alter table ORDERS
   merge partitions P2014Q1 to P2014Q4 into P2014;
```

Copyright @ 2017, John Jay King

Partition Split

- Partitions may be split

```
alter table ORDERS split partition P2016 into
   (partition P2016Q1 values
      less than to_date('01.04.2016','DD.MM.YYYY')),
   (partion P2016Q2 values
      less than to_date('01.07.2016','DD.MM.YYYY')),
   (partition P2016Q3 values
      less than to_date('01.10.2016','DD.MM.YYYY')),
   (partition P2016Q4 values
      less than to_date('01.01.2017','DD.MM.YYYY');
```

Copyright @ 2017, John Jay King

Partition Add/Drop/Truncate

```
alter table ORDERS add
 partition P2017Q1 value
     less than to_date('01.04.2017','DD.MM.YYYY')),
 partition P2017Q1 value
     less than to_date('01.07.2017','DD.MM.YYYY')),
 partition P2017Q1 value
     less than to_date('01.10.2017','DD.MM.YYYY')),
 partition P2017Q1 value
     less than to_date('01.01.2018','DD.MM.YYYY')),
```

```
alter table ORDERS drop partitions
  P2010Q1, P2010Q2, P2010Q3, P2010Q4;
```

```
alter table ORDERS truncate partitions
  P2010Q1, P2010Q2, P2010Q3, P2010Q4;
```

Copyright @ 2017, John Jay King

EBR & Materialized Views

- CREATE/ALTER MATERIALIZED VIEW now add the ability to specify use with editioning by specifying the Edition(s) to be used:
  - UNUSABLE BEFORE
    - CURRENT EDITION
    - EDITION XXX
  - UNUSABLE BEGINNING
    - CURRENT EDITION
    - EDITION XXX
    - NULL EDITION

Copyright @ 2017, John Jay King

EBR & Types

- CREATE/ALTER TYPE now add the ability to specify use with editioning by specifying the Edition(s) to be used:
  - UNUSABLE BEFORE
    - CURRENT EDITION
    - EDITION XXX
  - UNUSABLE BEGINNING
    - CURRENT EDITION
    - EDITION XXX
    - NULL EDITION

Copyright @ 2017, John Jay King

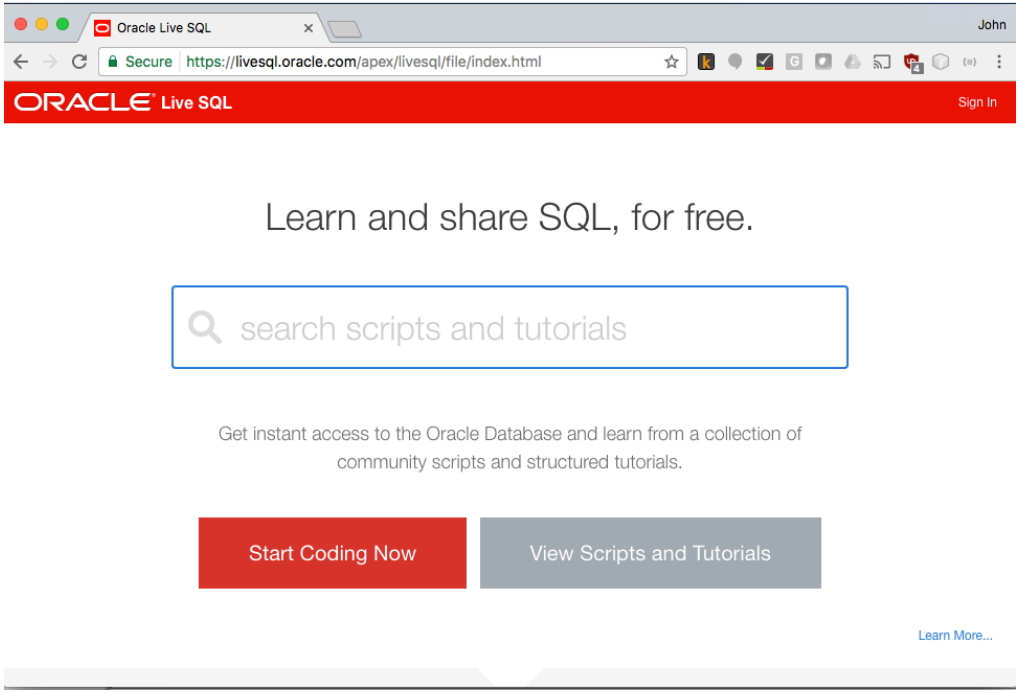
EBR & Virtual Columns

- Non-editioned Virtual Columns may depend upon editioned objects
  - May specify expression is to be resolved by searching the specified edition:
    - CURRENT EDITION
    - EDITION XXX
    - NULL EDITION
  - May use UNUSABLE EDITION or UNUSABLE BEGINNING clause (see previous page) to limit Virtual Columns “visibility” into editions

Copyright @ 2017, John Jay King

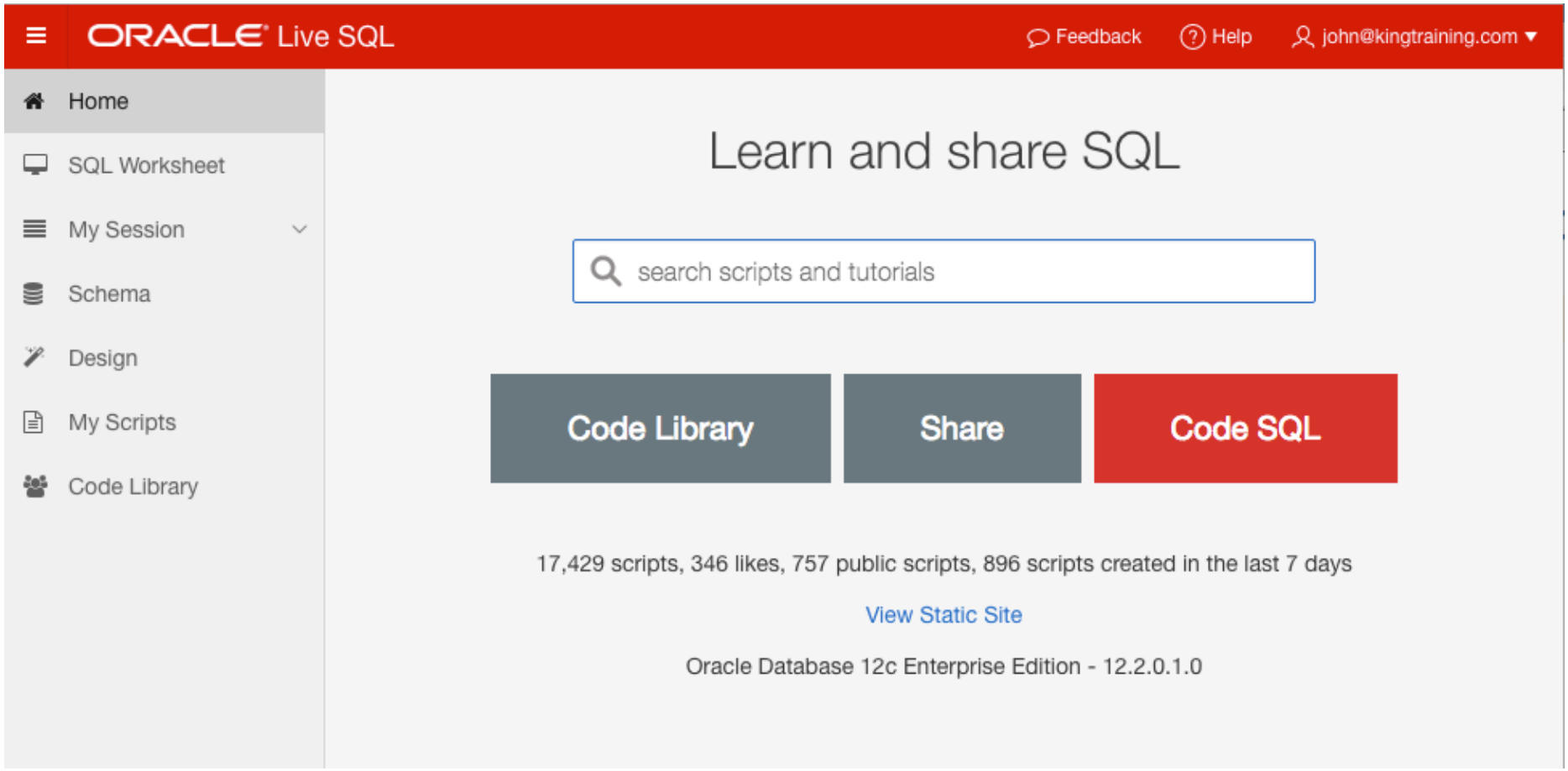
livesql.oracle.com

- Oracle provides free "live" 12.2 SQL tool
  - Includes available code library (cut & paste capab
  - Ability to save scripts and share
  - Online database design
  - Available sample schemas or build your own



Copyright @ 2017, John Jay King

livesql.oracle.com - Options



Copyright @ 2017, John Jay King

livesql.oracle.com - SQL

Oracle Live SQL - SQL Worksh

Secure https://livesql.oracle.com/apex/f?p=590:1:102547413411826::::

ORACLE Live SQL

Feedback Help john@kingtraining.com

SQL Worksheet

Clear View Session Reset Save Run

```
select dname,ename
  from scott.dept d join scott.emp e
    on d.deptno = e.deptno
  order by 1,2
```

DNAME	ENAME
ACCOUNTING	CLARK
ACCOUNTING	KING

Oracle 12.2

- Oracle 12.2 was released for Oracle Exadata Express Cloud Service users in October 2016 (OOW)
- Oracle DBaaS uses got access to 12.2 in November 2016
- On-premise versions of 12.2 will be available for download sometime in the first quarter of this year
(current rumor is March??)

Copyright @ 2017, John Jay King

Sampling of 12.2 Features

- Many-many new features including:
  - JSON generating functions
  - Analytic Views
  - Max number of PDBs from 252 to 4096
  - PDB memory and resource management
  - Local UNDO for PDBs and "hot clone"
  - SQL*Plus history and csv output
  - Partition tables online
  - READ-only partitions/subpartitions
  - Oracle sharding
  - Partitioned External Tables

Copyright @ 2017, John Jay King

Wrapping it all Up

- Oracle 12c has added significant new functionality to the already robust Oracle database environment; release 12.1.0.2 adds even more (12.2 for all soon!)
- Oracle 12c represents the first major architectural change to Oracle since Version 6
- With the release of Oracle 12c it's probably time for your shop to finally move to 11g R2
- 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
- I am still actively testing the new features presented here (and some others); your mileage may vary; watch for future editions of this talk or blog posts for more

Copyright @ 2017, John Jay King

Watch this space !!!

Dates & Venue for RMOUG 2018 coming soon

PHOTO CREDIT: Mike Landrum, SQL Developer and the "Data Tsunami" from i-Behavior


COLLABORATE17
TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY
Save the Date for
COLLABORATE 17^
We'll see you again in Vegas
WELCOME
TO The Fabulous
LAS VEGAS
NEVADA
APRIL 2 - 6, 2017 | MANDALAY BAY RESORT & CASINO

King Training Resources

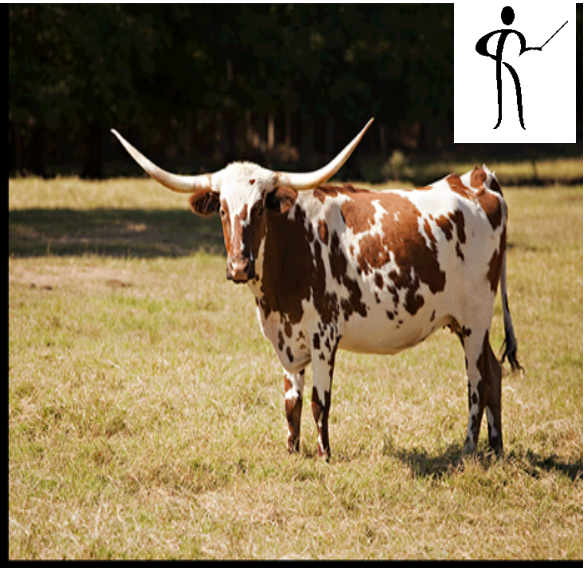
SAN ANTONIO
JUNE 25-29

ODTUG
Kscope17

"Great event, great content #Kscope16 many thanks #orclapex #letswreckthistogether"
- Simon Greenwood @APEXORADEV

"#kscope16 was a blast. On the way to the airport with a heavy heart. Thanks @odtug for making this event what it is: the best!"
- Christian Berg @Nephentur

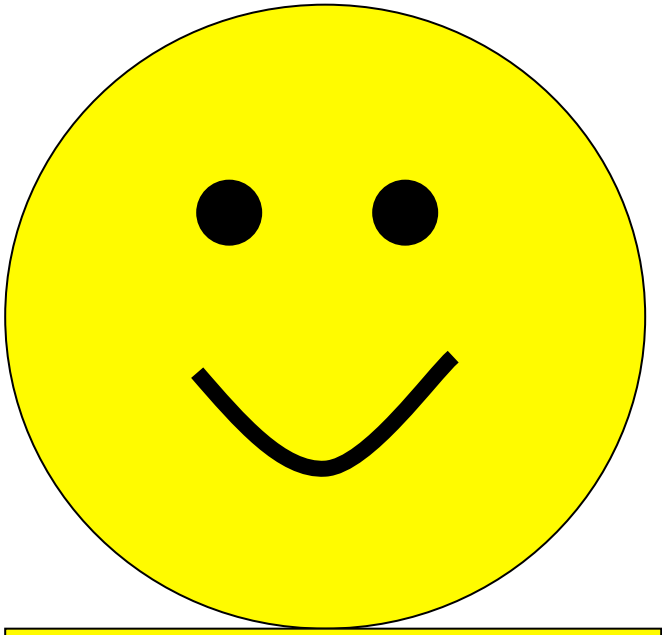
www.kscope17.com



Copyright @ 2017 John Jay King

Please Complete Session Evaluations

Oracle 12c
New Features For Developers
(& DBAs)



Thanks for your attention!

To contact the author:

John King

King Training Resources

P. O. Box 1780

Scottsdale, AZ 85252 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com

Twitter: royaltwit

Today's slides and examples are on the web:

http://www.kingtraining.com

Copyright @ 2017, John Jay King

- End

Copyright @ 2017, John Jay King