I´m speaking @ OUGN VÅRSEMINAR 2016 10-12. MARS

10th of March - Thursday:

P81 Sometimes it's alright to be lazy (Tools like Vagrant & Packer.io)
5:00 PM – 5:45 PM: Auditorium 1

P5 Oracle 12c New Features for Java-developers
6:00 PM – 6:45 PM: Congress 3

11th of March - Friday:

P49 10 mistakes your (Java) developers will do - if you don't assist 'em!
5:00 PM – 5:45 PM: Parliament 1+2

EVRY

By Lasse Jenssen

Oracle 12c New Features

for Java Developers

EVRY

Work at **EVRY**, Financial Services: Leader of Center of Excellence - Databases

Used to be a **DBA**

Today I consider myself to a …
Oracle Developer or Application DBA

Used to be an Oracle University Instructor

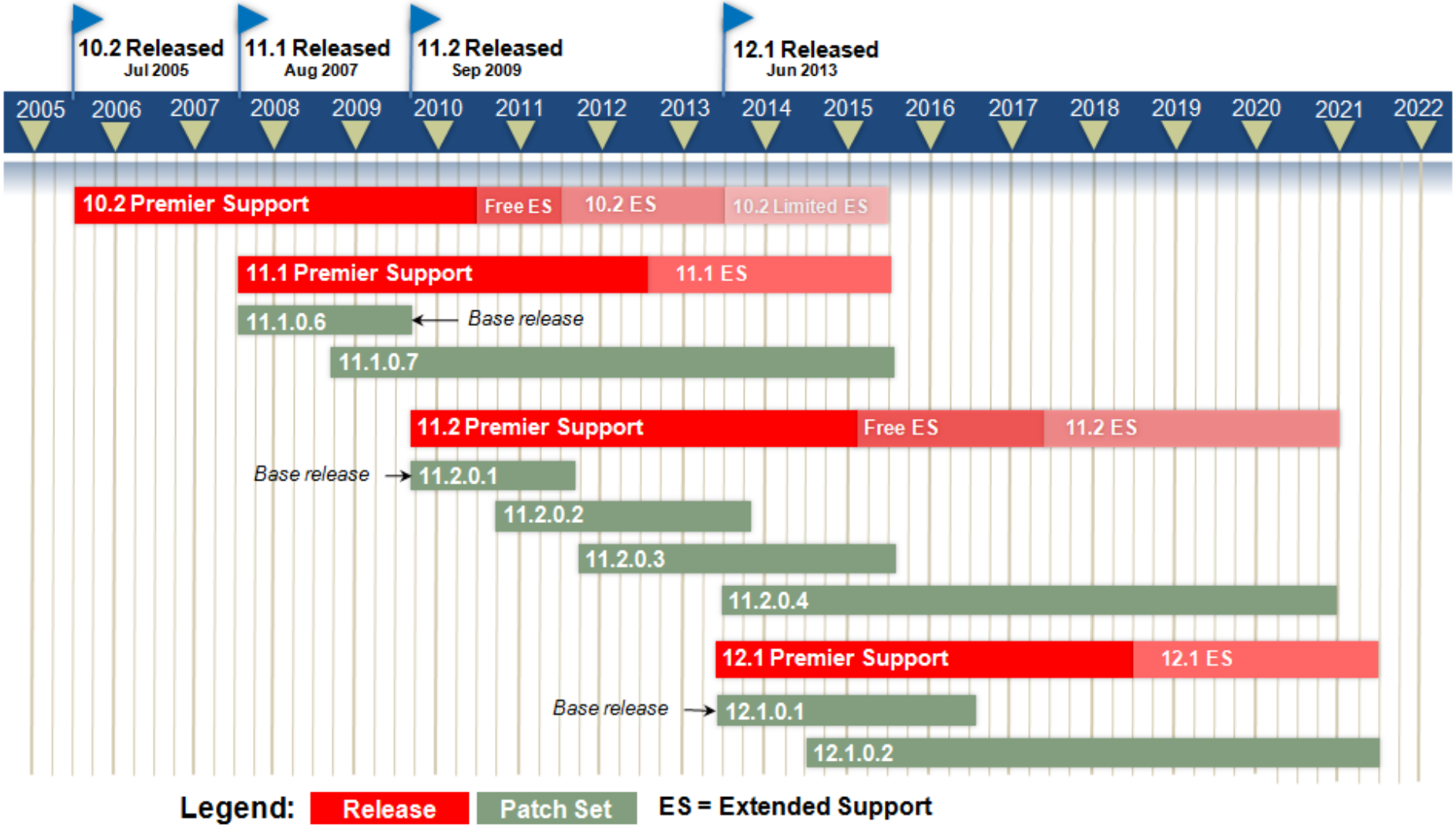
Presented at:
- OUGN Bergen
- OUGN Spring seminar (The Boat)
- JavaZone
- UKOUG Teck Conference
- Oracle OpenWorld
- Internal Courses internal in EVRY

@lasjen

http://www.jcon.no/oracle

ORACLE ACE Associate

EVRY


10.2 Released
Jul 2005
11.1 Released
Aug 2007
11.2 Released
Sep 2009
12.1 Released
Jun 2013
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
10.2 Premier Support
Free ES
10.2 ES
10.2 Limited ES
11.1 Premier Support
11.1 ES
11.1.0.6
Base release
11.1.0.7
11.2 Premier Support
Free ES
11.2 ES
Base release
11.2.0.1
11.2.0.2
11.2.0.3
11.2.0.4
12.1 Premier Support
12.1 ES
Base release
12.1.0.1
12.1.0.2
Legend:
Release
Patch Set
ES = Extended Support

“One thing is to upgrade your database, but to actually start using the new features is just another story”

“How many uses COMPRESS on indexes?”

“When was this feature made available?”

EVRY

“How many uses Analytical Functions when writing SQLs?”

“When was this feature made available?”

EVRY

“How many uses MERGE and/or CONDITIONAL INSERTs when writing DML?”

“When was this feature made available?”

EVRY

Adaptive Plans:

Adaptive Join Methods

Parallel Distribution Methods

New Feature: Adaptive Query Optimization

Adaptive Statistics:

Automatic Reoptimization

Dynamic Statistics

SQL Plan directives

EVRY

```
SELECT * FROM TABLE(DBMS_XPLAN.display_cursor(sql_id=> …, format => 'adaptive'));
```

Adaptive Join Methods

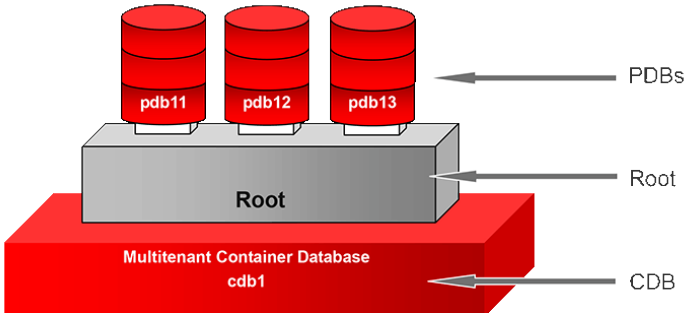
```
SELECT …
FROM p_header p INNER JOIN p_execution e ON (p.hid = e.fk_hid)
WHERE e.debtor_account = 'xxxx0512345'

Plan hash value: 4912255763
```

	Id	Operation	Name	Rows	Bytes	Cost
	0	SELECT STATEMENT				
- *	1	HASH JOIN		25	425	
	2	NESTED LOOPS		25	425	
	3	NESTED LOOPS		25	425	
-	4	STATISTICS COLLECTOR				
	5	TABLE ACCESS BY INDEX ROWID BATCHED	P_HEADER	1	11	
*	6	INDEX RANGE SCAN	P_HEAD_IDX01	1		
*	7	INDEX RANGE SCAN	P_EXE_HEA_FKI	25		
	8	TABLE ACCESS BY INDEX ROWID	P_EXECUTION	25	150	
-	9	TABLE ACCESS FULL	P_EXECUTION	25	150	

```
Note!
   - this is an adaptive plan (rows marked '-' are inactive)
```

EVRY

New Feature:
Multitenant & plugables



Transparant for client if pdb or non-pdb

Might be perfect for test & development systems

ALWAYS use pluggable (Multitenant) database

As long as you only have ONE PDB – you don't utilize the license!

EVRY

Performance improvement for PL/SQL functions called from SQL: Functions & Procedures in the WITH clause

```
WITH PROCEDURE with_procedure(p_id IN NUMBER) IS
     BEGIN
       DBMS_OUTPUT.put_line('p_id=' || p_id);
     END;
     FUNCTION with_function(p_id IN NUMBER) RETURN NUMBER IS
     BEGIN
        with_procedure(p_id);
        RETURN p_id;
     END;
SELECT with_function(id) FROM   t1 WHERE  rownum = 1;

WITH_FUNCTION(ID)
-----------------
                1

p_id=1
```

Example code: Tim Hall, https://oracle-base.com/articles/12c/with-clause-enhancements-12cr1

EVRY

Performance improvement for PL/SQL functions called from SQL: pragma UDF

```
CREATE OR REPLACE FUNCTION normal_function(p_id IN NUMBER) RETURN NUMBER IS
  PRAGMA UDF;
BEGIN
  RETURN p_id;
END;
/
```

Example code: Tim Hall, https://oracle-base.com/articles/12c/with-clause-enhancements-12cr1

EVRY

New Feature:
Improved Defaults

Generating a Default Value from a SEQUENCE.

```
> create sequence emp_seq;
Sequence created.

> create table emp(
     id int  default emp_seq.nextval primary key,
     name varchar2(30));
Table created.

> insert into emp (id, name) values ( default, 'Billy' );
1 row created.
> insert into emp (name)  values ( 'Nancy' );
1 row created.

> select * from emp;
 ID NAME
---------- ------------------------------
         1 Billy
         2 Nancy
```

EVRY

```
> create trigger emp_trg before insert on emp
    for each row
  begin
    if (:new.x is null) then
      :new.x := emp_seq.nextval;
    end if;
  end;
  /
Trigger created.
```

In 11g we had to …

Use a trigger

```
> create table emp(
     id   int  generated as identity primary key,
     name varchar2(30));
Table created.

> insert into emp (id, name) values ( default, 'Billy' );
1 row created.
> insert into emp (name)  values ( 'Nancy' );
1 row created.

create table emp(
     id int  generated as identity
           (start with 10 increment by 1 cache 200) primary key,
     name varchar2(30));

> insert into emp (id, name) values ( 1, 'Cary' );
> insert into emp (id, name) values ( default, 'Tanel' );
> insert into emp (name)  values ( 'Maria' );
```

In 12c we can ...

Use an IDENTITY TYPE

EVRY

```
> create table t (
    x number primary key,
    y varchar2(30),
    z number default 42
  )
/


> insert into t values(1,'Row 1',null);

> select * from t;

         X Y                                                Z
---------- ------------------------------ ----------
         1 Row 1                                          ?
```

Let's have a look at 11g again

Quiz!

EVRY

```
> create table t (
    x number generated as identity primary key,
    y varchar2(30),
    z number default on null 42)
/

> insert into t(y,z) values('Row 1',null);
> insert into t(y,z) values('Row 2',default);
> insert into t(y) values ('Row 3');
> insert into t(y,z) values ('Row 4',100);

> select * from t;

         X Y                                          Z
---------- ------------------------------ ----------
         1 Row 1                                    42
         2 Row 2                                    42
         3 Row 3                                    42
         4 Row 4                                   100
```

In 12c we can …

Set default on null

In 11g –

Adding new column with default value

```
> create table t as select * from all_objects;

> exec show_space('T');
Full Blocks              .............              989
Total Blocks.....................            1,024
Total Bytes......................        8,388,608
Total MBytes.....................                8

> alter table t add (data char(2000) default 'x'); -- timing on
Table altered.
Elapsed: 00:00:14.44

> exec show_space('T');
Full Blocks              .............              989
Total Blocks.....................           23,552
Total Bytes......................      192,937,984
Total MBytes.....................              184
```

EVRY

In 12c –

Adding new column with default value

```
> create table t as select * from all_objects;

> exec show_space('T');
Full Blocks              .............              989
Total Blocks.....................              1,024
Total Bytes......................            8,388,608
Total MBytes.....................                8

> alter table t add (data char(2000) default 'x'); -- timing on
Table altered.
Elapsed: 00:00:00.15
> exec show_space('T');

Full Blocks              .............            1,255
Total Blocks.....................            1,408
Total Bytes......................           11,534,336
Total MBytes.....................               11
```

EVRY

VARCHAR2

Oracle7 => 256

Oracle8 => 4000

Oracle9 => 4000

New Feature:
Bigger Datatypes

Oracle10 => 4000

Oracle11 => 4000

Oracle12 => 32767

EVRY

```
> create table t ( x varchar(32767) );
Error at line 1
ORA-00910: specified length too long for its
datatype
```

WHAT?

Sorry … this is not the default .. Why?

Everything above 4000 characters stored as CLOB

EVRY

```
> Sqlplus “/as sysdba”
SQL> alter system set max_string_size=EXTENDED scope=spfile;
System altered.
SQL> shutdown immediate;

SQL> startup upgrade;
SQL>  @?/rdbms/admin/utl32k.sql

SQL> alter pluggable database all open upgrade;
Pluggable database altered.

SQL> alter session set container=orcl;
SQL> @?/rdbms/admin/utl32k.sql

SQL> alter session set container=cdb$root;
SQL> shutdown immediate;
SQL> startup
SQL> alter pluggable database all open;
```

EVRY

```
> create table t ( x varchar(32767) );
Table created.
```

But now you can not go back!

Warning!
Anything above 4000 characters is stored as a CLOB!

Remember the AXE comersial:
“Don’t use it if you don’t mean it”

I don’t really have any experience on this YET!
Anyone?

EVRY

«least privilege» principle

Any unnecessary privileges represent a possible security loophole.

New Feature:
DBMS_PRIVILEGE_CAPTURE

Harder to implement it on existing systems, where excessive privilege has already been granted.

DBMS_PRIVILEGE_CAPTURE allows you to track the privileges being used

EVRY

Basic use of DMBS_PRIVILEGE_CAPTURE

1. Create a privilege analysis policy. (CREATE_CAPTURE)
2. Enable it. (ENABLE_CAPTURE)
3. Wait for the required analysis period.
4. Disable the privilege analysis policy. (DISABLE_CAPTURE)
5. Analyze the results.
   (GENERATE_RESULT and query dictionary views)
6. Drop the policy if it, and the recorded data, is no longer needed.
   (DROP_CAPTURE)

Source: Tim Hall, https://oracle-base.com/articles/12c/capture-privilege-usage-12cr1

```
-- Connect to a privileged using in a PDB.
CONN / AS SYSDBA   -- MUST have CAPTURE_ADMIN role
ALTER SESSION SET CONTAINER = pdb1;

BEGIN -- Whole database (type = G_DATABASE)
  DBMS_PRIVILEGE_CAPTURE.create_capture(
    name         => 'db_pol',
    type         => DBMS_PRIVILEGE_CAPTURE.g_database);
END;
/
BEGIN -- One or more roles (type = G_ROLE)
  DBMS_PRIVILEGE_CAPTURE.create_capture(
    name         => 'role_pol',
    type         => DBMS_PRIVILEGE_CAPTURE.g_role,
    roles        => role_name_list('DBA', 'RESOURCE')
  );
END;
/
```

EVRY

```
BEGIN -- user defined condition,when user is TEST(type=G_CONTEXT)
  DBMS_PRIVILEGE_CAPTURE.create_capture(
   name      => 'cond_pol',
   type      => DBMS_PRIVILEGE_CAPTURE.g_context,
   condition=> 'SYS_CONTEXT(''USERENV'',''SESSION_USER'')= ''TEST'''
  );
END;
/
BEGIN --Combination of roles and conditions(type = G_ROLE_AND_CONTEXT)
  DBMS_PRIVILEGE_CAPTURE.create_capture(
   name      => 'role_cond_pol',
   type      => DBMS_PRIVILEGE_CAPTURE.g_role_and_context,
   roles     => role_name_list('DBA', 'RESOURCE'),
   condition=> 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') IN
                                          (''TEST'',''EMP'')');
END;
/
```

```
BEGIN
  DBMS_PRIVILEGE_CAPTURE.enable_capture('db_pol');
  DBMS_PRIVILEGE_CAPTURE.enable_capture('cond_pol');
END;
/
BEGIN
  DBMS_PRIVILEGE_CAPTURE.disable_capture('db_pol');
  DBMS_PRIVILEGE_CAPTURE.disable_capture('cond_pol');
END;
/
BEGIN
  DBMS_PRIVILEGE_CAPTURE.generate_result('db_pol');
END;
/
BEGIN
  DBMS_PRIVILEGE_CAPTURE.drop_capture('cond_pol');
  DBMS_PRIVILEGE_CAPTURE.drop_capture('db_pol');
  DBMS_PRIVILEGE_CAPTURE.drop_capture('role_cond_pol');
  DBMS_PRIVILEGE_CAPTURE.drop_capture('role_pol');
END;
/
```

Now we wait …

EVRY

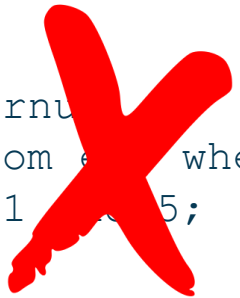
DBA_PRIV_CAPTURES
DBA_USED_OBJPRIVS
DBA_USED_OBJPRIVS_PATH
DBA_USED_PRIVS
DBA_USED_PUBPRIVS
DBA_USED_SYSPRIVS
DBA_USED_SYSPRIVS_PATH
DBA_USED_USERPRIVS
DBA_USED_USERPRIVS_PATH
DBA_UNUSED_OBJPRIVS
DBA_UNUSED_OBJPRIVS_PATH
DBA_UNUSED_PRIVS
DBA_UNUSED_SYSPRIVS
DBA_UNUSED_SYSPRIVS_PATH
DBA_UNUSED_USERPRIVS
DBA_UNUSED_USERPRIVS_PATH

EVRY

New Feature:
Top-N Queries and Pagination

The 11g Way of doing PAGINATION

```
SELECT i.*, rownum rnu
FROM   (select * from e    where deptno=10 order by empid) i
WHERE rnum between 1 a    5;
```



```
SELECT *
FROM (SELECT i.*, rownum rn
      FROM   (select * from emp where deptno=10 order by empid) i
      WHERE rownum<=5)
WHERE rn >=1;
```

EVRY

The 12c Way of doing PAGINATION

```
SELECT *
FROM emp
ORDER BY empno
FETCH FIRST 5 ROWS ONLY;
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |     5 |   795 |     1   (0)| 00:00:01 |
|*  1 |  VIEW                         |               |     5 |   795 |     1   (0)| 00:00:01 |
|*  2 |   WINDOW NOSORT STOPKEY       |               |   107 | 14231 |     1   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |   107 | 14231 |     1   (0)| 00:00:01 |
|   4 |     INDEX FULL SCAN           | EMP_EMP_ID_PK |   107 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

The 12c Way of doing PAGINATION

```
SELECT *
FROM emp
ORDER BY empno
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

New Feature:
In Database Archiving

Examples: Tim Hall, https://oracle-base.com/articles/12c/in-database-archiving-12cr1

EVRY

```
-- Create the table with in-database archiving enabled.
CREATE TABLE tab1 (
  id          NUMBER,
  description VARCHAR2(50),
  CONSTRAINT tab1_pk PRIMARY KEY (id)) ROW ARCHIVAL;

-- Disable, then re-enable in-database archiving.
ALTER TABLE tab1 NO ROW ARCHIVAL;
ALTER TABLE tab1 ROW ARCHIVAL;

-- Populate the table with 1000 rows.
INSERT /*+ APPEND */ INTO tab1
SELECT level, 'Description of ' || level
FROM   dual
CONNECT BY level <= 1000;
COMMIT;

SELECT COUNT(*) FROM tab1;

  COUNT(*)
----------
      1000
```

```
SELECT column_id, column_name, data_type, data_length, hidden_column
FROM   user_tab_cols
WHERE  table_name = 'TAB1'
ORDER BY column_id;

 COLUMN_ID COLUMN_NAME          DATA_TYPE            DATA_LENGTH HID
---------- -------------------- -------------------- ----------- ---
         1 ID                   NUMBER               22          NO
         2 DESCRIPTION          VARCHAR2             50          NO
           ORA_ARCHIVE_STATE    VARCHAR2             4000        YES

SELECT ora_archive_state, COUNT(*)
FROM   tab1
GROUP BY ora_archive_state
ORDER BY ora_archive_state;

ORA_ARCHIVE_STATE      COUNT(*)
------------------- ----------
0                         1000
```

```
UPDATE tab1
SET    ora_archive_state = '1'      -- OR use DBMS_ILM.ARCHIVESTATENAME(1)
WHERE  id BETWEEN 751 and 1000;
COMMIT;

SELECT COUNT(*) FROM tab1;

  COUNT(*)
----------
       750

-- Make archived rows visible.
ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ALL;
-- ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ACTIVE;

SELECT COUNT(*) FROM tab1;

  COUNT(*)
----------
      1000
```

EVRY

```
COLUMN ora_archive_state FORMAT A20

SELECT ora_archive_state, COUNT(*)
FROM   tab1
GROUP BY ora_archive_state
ORDER BY ora_archive_state;

ORA_ARCHIVE_STATE      COUNT(*)
---------------------- ----------
0                             750
1                             250

-- Deleting
ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ALL;
DELETE FROM tab1 where ora_archive_state=DBMS_ILM.ARCHIVESTATENAME(1);
```

New Feature:
Support for JDBC 4.1 Standard

EVRY

NOT **New Feature:**

Oracle End-to-End Metrics

EVRY

Introduce an non-functional requirement:
The Oracle End-to-end Metrics (10g & 11g)

```
metrics = new String[OracleConnection.END_TO_END_STATE_INDEX_MAX];
metrics[OracleConnection.END_TO_END_MODULE_INDEX] = "PAY";
metrics[OracleConnection.END_TO_END_ACTION_INDEX] = "PaymentHistoricRead";
metrics[OracleConnection.END_TO_END_CLIENTID_INDEX] = "067e6162-3b6f-…";
((OracleConnection) connection).setEndToEndMetrics(metrics,(short)0);

// The PaymentHistoricRead code

metrics[OracleConnection.END_TO_END_MODULE_INDEX] = "";
metrics[OracleConnection.END_TO_END_ACTION_INDEX] = "";
metrics[OracleConnection.END_TO_END_CLIENTID_INDEX] = "";
((OracleConnection) connection).setEndToEndMetrics(metrics,(short)0);
```

EVRY

Introduce an non-functional requirement:
The Oracle End-to-end Metrics

```
Properties p = new Properties();
p.put("OCSID.MODULE",      "PAY");
p.put("OCSID.ACTION",      "PaymentHistoricRead");
p.put("OCSID.CLIENTID",    "067e6162-3b6f-4ae2-a171-2470b63…");

connection.setClientInfo(p);

// The PaymentHistoricRead code

p.put("OCSID.MODULE",      "");
p.put("OCSID.ACTION",      "");
p.put("OCSID.CLIENTID",    "");

connection.setClientInfo(p);
```

Concept:

User Experience ID

Java:

```
UUID.randomUUID().toString();
```

EVRY

Working together in development (or test)

Hello …
… my dear Oracle resource.
I've rewritten the "PaymentHistoricRead" service in our "PAY" application.
Could you have a look?

Developer

DBA or
other Oracle resource

Sure …
… dear developer colleague
I'll start **trace** on it right away. Let me know when you have ran some tests, and send some of the clientids from your logs.

EVRY

Turning on trace based on metrics

```
exec dbms_monitor.serv_mod_act_trace_enable(
                        service_name  => 'PAY_SRV',
                        module_name   => 'PAY',
                        action_name   => 'PaymentHistoricRead',
                        waits         => true,
                        binds         => true,
                        plan_stat     => 'ALL_EXECUTIONS');

SELECT * FROM dba_enabled_traces;
```

Warning!

BUT this works only if you code sets the metric tags in your application!!

EVRY

Working together in production

Hello …
… my dear Oracle resource. Eva in Sparebank1 is experiencing slowness in the “PaymentHistoricRead” service in our “PAY” application. Could you have a look?

Sure …
… dear developer colleague I’ll start **trace** on it right away. Let me know when Eva have ran the feature again, and send some of the **clientids** from your logs.

EVRY

Profile Report: **pay_1_ora_16974264.html**

METHOD R™ Profiler

License id 1000079 registered to lasse.jenssen@evry.com / EVRY AS.

Profiler version 5.2.8.11 built on Wed Jan 30 10:54:18 CST 2013.

Profiler XML created 2014-09-19T14:25:11.449832; kernel duration 0.084 seconds (1.31 MB/sec).

Total measured task response time:	$R = t_1 - t_0$ = 0.277924 seconds ≈ **0.278 seconds**
Task begin time:	t_0 = 1:34:47.015945 p.m. 19 September 2014 Friday
Task end time:	t_1 = 1:34:47.293869 p.m. 19 September 2014 Friday
Oracle release:	11.2.0.3.0
Oracle session id:	288.58427
Transactions:	1 commit, 0 rollback
Workload:	777 database buffer cache accesses
Errors and warnings:	0 parse errors, 0 other errors, 0 Profiler kernel warnings
Service/module/action:	pay_srv / PAY / PaymentHistoricRead
Client identifier:	067e6162-3b6f-4ae2-a171-2470b63dff00
Viewing characteristics:	screen | projection | print

Profile by Subroutine

e following table shows the decomposition of total task response time by internal subroutine.

	Subroutine	Duration		Cumulative duration		Call	Duration per call (seconds)			
		seconds	% *R*	seconds	% *R*	count	mean	min	skew	max
1.	**db file sequential read**	**0.129**	**46.3%**	0.129	46.3%	3	**0.042887**	0.021816		0.053708
2.	**unaccounted-for within dbcalls**	**0.065**	**23.5%**	0.194	69.8%	**330**	0.000198	-0.000109		0.032567
3.	**CPU service, EXEC calls**	**0.059**	**21.3%**	0.253	91.1%	**89**	0.000665	0.000000		0.034788
4.	CPU service, PARSE calls	0.008	2.9%	0.261	94.1%	16	0.000506	0.000226		0.000961
5.	SQL*Net message from client	0.007	2.7%	0.269	96.7%	6	0.001229	0.000000		0.001979
6.	CPU service, FETCH calls	0.004	1.3%	0.272	98.0%	182	0.000020	0.000002		0.000360
7.	unaccounted-for between dbcalls	0.003	1.1%	0.276	99.1%	15	0.000207	0.000018		0.001522
8.	row cache lock	0.002	0.6%	0.277	99.7%	6	0.000279	0.000005		0.000973
9.	rdbms ipc reply	0.000	0.2%	0.278	99.9%	4	0.000112	0.000049		0.000274
0.	CPU service, unreported call(s)	0.000	0.0%	0.278	100.0%	1	0.000125	0.000125		0.000125
1.	CPU service, CLOSE calls	0.000	0.0%	0.278	100.0%	86	0.000001	0.000000		0.000036
2.	SQL*Net message to client	0.000	0.0%	0.278	100.0%	6	0.000002	0.000001		0.000003
3.	Total	0.278	100.0%							

New Feature:
Improved JDBC batch execution

```
PreparedStatement pstmt =
        conn.prepareStatement("INSERT INTO employees VALUES(?, ?)");

pstmt.setInt(1, 2000);
pstmt.setString(2, "Milo Mumford");
pstmt.addBatch();

pstmt.setInt(1, 3000);
pstmt.setString(2, "Sulu Simpson");
pstmt.addBatch();

int[] updateCounts = pstmt.executeBatch();
```

- Return ARRAY with size like number of records in batch Example: 2
- Each value reflects the number of rows affected by the record execution

New Feature:
JDBC support for Implicit Results

```
create procedure foo as
  c1 sys_refcursor;
  c2 sys_refcursor;
begin
  open c1 for select * from hr.employees;
  dbms_sql.return_result(c1); --return to client

-- open 1 more cursor
  open c2 for select * from hr.departments;
  dbms_sql.return_result (c2); --return to client
end;
/
```

```
String sql = "begin foo; end;";
...
Connection conn = DriverManager.getConnection(jdbcURL, user, passw);
try {
        Statement stmt = conn.createStatement ();
        stmt.executeQuery (sql);

        while (stmt.getMoreResults())
        {
              ResultSet rs = stmt.getResultSet();
              System.out.println("ResultSet");
              while (rs.next())
             {
                   /* get results */
             }
        }
    }
```

New Feature:
Enhanced Partitioning

11g New Feature:
Partition By Reference

```
CREATE TABLE payment(
  payment_id          NUMBER,
  customer_id         NUMBER,
  debtor_account      VARCHAR2(11),
  created_date        DATE,
  CONSTRAINT payment_pk PRIMARY KEY (payment_id)
)
PARTITION BY RANGE (created_date)
(PARTITION part_2015 VALUES LESS THAN (TO_DATE('01/01/2016', 'DD/MM/YYYY')),
 PARTITION part_2016 VALUES LESS THAN (TO_DATE('01/01/2017', 'DD/MM/YYYY'))
);

CREATE TABLE payment_detail (
  payment_id            NUMBER NOT NULL,
  creditor_account      VARCHAR2(11) NOT NULL,
  amount                NUMBER,
  amountccy             VARCHAR2(100),
  CONSTRAINT paym_detail_pk PRIMARY KEY (payment_id, creditor_account),
  CONSTRAINT paym_detail_paym_fk FOREIGN KEY (payment_id)
                              REFERENCES payment (payment_id) ON DELETE CASCADE
)
PARTITION BY REFERENCE (paym_detail_paym_fk );
```

```
INSERT INTO payment VALUES (1, 1, '12340512345', TO_DATE('11/03/2015', 'DD/MM/YYYY'));
INSERT INTO payment VALUES (2, 1, '12340512346', TO_DATE('23/07/2016', 'DD/MM/YYYY'));

INSERT INTO payment_detail VALUES (1, '43210512345', 100, 'NOK');
INSERT INTO payment_detail VALUES (2, '43210512345', 1000, 'NOK');
INSERT INTO payment_detail VALUES (2, '43210512346', 10, 'SEK');
COMMIT;

EXEC DBMS_STATS.gather_table_stats(USER, 'payment');
EXEC DBMS_STATS.gather_table_stats(USER, 'payment_detail');

SELECT table_name, partition_name, num_rows
FROM   user_tab_partitions
ORDER BY 1,2;

TABLE_NAME           PARTITION_NAME         NUM_ROWS
-------------------- -------------------- ----------
PAYMENT              PART_2015                     1
PAYMENT              PART_2016                     1
PAYMENT_DETAIL       PART_2015                     1
PAYMENT_DETAIL       PART_2016                     2
```

EVRY

12g New Feature:
Partition Cascade Truncate

```
ALTER TABLE payment TRUNCATE PARTITION part_2015 CASCADE UPDATE INDEXES;

EXEC DBMS_STATS.gather_table_stats(USER, 'payment');
EXEC DBMS_STATS.gather_table_stats(USER, 'payment_detail');

SELECT table_name, partition_name, num_rows
FROM   user_tab_partitions
ORDER BY 1,2;

TABLE_NAME           PARTITION_NAME         NUM_ROWS
-------------------- -------------------- ----------
PAYMENT              PART_2015                     0
PAYMENT              PART_2016                     1
PAYMENT_DETAIL       PART_2015                     0
PAYMENT_DETAIL       PART_2016                     2
```

11g –
Had to truncate from child and upwards

Some SQLs:
Preparing to Load Data

```
CREATE TABLE payment_tmp(
  payment_id          NUMBER,
  customer_id         NUMBER,
  debtor_account      VARCHAR2(11),
  created_date        DATE,
  CONSTRAINT payment_tmp_pk PRIMARY KEY (payment_id)
);

CREATE TABLE payment_detail_tmp (
  payment_id            NUMBER NOT NULL,
  creditor_account      VARCHAR2(11) NOT NULL,
  amount                NUMBER,
  amountccy             VARCHAR2(100),
  CONSTRAINT paym_detail_tmp_pk PRIMARY KEY (payment_id, creditor_account),
  CONSTRAINT paym_detail_tmp_paym_fk FOREIGN KEY (payment_id)
                              REFERENCES payment_tmp (payment_id) ON DELETE CASCADE
);

INSERT INTO payment_tmp VALUES (2, 1, '12340512345',
                                                TO_DATE('15/04/2016', 'DD/MM/YYYY'));
INSERT INTO payment_detail_tmp VALUES (2, '43210512345', 100, 'NOK');
COMMIT;
```

EVRY

12g New Feature:
Partition Cascade On Exchange

```
ALTER TABLE orders
  EXCHANGE PARTITION part_2016
  WITH TABLE payment_tmp
  CASCADE UPDATE INDEXES;
```

Fastest way to load data into related tables

```
-- Check the data in the partitioned data.
COLUMN amountccy FORMAT A20

SELECT p.debtor_account, d.creditor_account, d.amount, d.amountccy
FROM   payment p JOIN payment_detail d ON p.payment_id = d.payment_id;

DEBTOR_ACCOUNT CREDITOR_ACCOUNT     AMOUNT AMOUNTCCY
-------------- ---------------- ---------- --------------------
70010512345    90010512345             100 NOK
```

Have you ever had trouble with global indexes on a partitioned table?

Preparing:
Creating a Global Index

```
-- Creating a partitioned table with some global indexes
CREATE TABLE payment(
  payment_id          NUMBER,
  customer_id         NUMBER,
  debtor_account      VARCHAR2(11),
  created_date        DATE
)
PARTITION BY RANGE (created_date)(
  PARTITION part_2015 VALUES LESS THAN (TO_DATE('01/01/2016', 'DD/MM/YYYY')),
  PARTITION part_2016 VALUES LESS THAN (TO_DATE('01/01/2017', 'DD/MM/YYYY'))
);

ALTER TABLE payment ADD CONSTRAINT payment_pk PRIMARY KEY (payment_id);

CREATE INDEX payment_idx01 ON payment(created_date);
```

EVRY

Preparing:
Inserting Data

```
-- Populate it so segments are created.
INSERT /*+ APPEND */ INTO payment
SELECT level,                       -- payment_id
       mod(level,1000),             -- customer_id
       '90010712345',               -- debtor_account
       DECODE(MOD(level,2),0,
              TO_DATE('01/02/2015', 'DD/MM/YYYY'),
              TO_DATE('01/02/2016', 'DD/MM/YYYY'))
FROM   dual
CONNECT BY level <= 50000;
COMMIT;

EXEC DBMS_STATS.gather_table_stats(USER, 'payment');
```

EVRY

12c New Feature:
Keeping Global Indexes Valid

```
-- Check the indexes.
SELECT table_name,  index_name, status
FROM   user_indexes WHERE table_name = 'PAYMENT'
ORDER BY 1,2;

TABLE_NAME           INDEX_NAME           STATUS
-------------------- -------------------- --------
ORDERS               ORDERS_IDX01         VALID
ORDERS               ORDERS_PK            VALID

-- Truncate a partition (or drop partition)
ALTER TABLE orders TRUNCATE PARTITION part_2015 DROP STORAGE UPDATE INDEXES;
-- ALTER TABLE t1 DROP PARTITION part_2014 UPDATE INDEXES;

SELECT table_name,  index_name, status
FROM   user_indexes WHERE table_name = 'PAYMENT'
ORDER BY 1,2;

TABLE_NAME           INDEX_NAME           STATUS
-------------------- -------------------- --------
ORDERS               ORDERS_IDX01         VALID
ORDERS               ORDERS_PK            VALID
```

12c New Feature:
Cleaning Up Global Indexes

```
-- Check if index maintenance is needed.
SELECT index_name, orphaned_entries
FROM   user_indexes WHERE table_name = 'PAYMENT'
ORDER BY 1;

INDEX_NAME           ORP
-------------------- ---
PAYMENT_IDX           YES
PAYMENT_PK            YES

EXEC DBMS_PART.cleanup_gidx(USER, 'payment');

SELECT index_name, orphaned_entries
FROM   user_indexes WHERE table_name = 'PAYMENT'
ORDER BY 1;

INDEX_NAME           ORP
-------------------- ---
PAYMENT_IDX           NO
PAYMENT_PK            NO
```

New Feature:
JSON support in the database

P56 Read, Store and Create XML and JSON

Thursday 6:00 PM – 6:45 PM: Bundestag 1+2+3

```
CREATE TABLE j_purchaseorder(
  id           RAW (16) NOT NULL,
  date_loaded TIMESTAMP WITH TIME ZONE,
  po_document CLOB CONSTRAINT ensure_json CHECK (po_document IS JSON)
);

INSERT INTO j_purchaseorder
  VALUES (SYS_GUID(),
          SYSTIMESTAMP,
          '{"PONumber"              : 1600,
            "Reference"             : "ABULL-20140421",
            "Requestor"             : "Alexis Bull",
            "User"                  : "ABULL",
            "CostCenter"            : "A50",
            "ShippingInstructions"  : {...},
            "Special Instructions"  : null,
            "AllowPartialShipment"  : true,
            "LineItems"             : [...]}');
```

```
SELECT po.po_document.PONumber FROM j_purchaseorder po;


SELECT po.po_document.ShippingInstructions.Phone
FROM j_purchaseorder po;
```

https://docs.oracle.com/database/121/ADXDB/json.htm#ADXDB6256

EVRY

“One thing is to upgrade your database, but to actually start using the new features is just another story”

Any Questions

www.jcon.no/oracle

#lasjen

EVRY

We bring information to life