ORACLE®

The Sins of SQL Programming that send the DB to Upgrade Purgatory

Abel Macias

Who is Abel Macias ?

- 1994 - Joined Oracle Support
- 2000 - RDBMS Technical Leader
- 2001 - Specialize in Performance (DBPerf/Qtune/WR)
- 2002 - Joined Support Escalations Team
- 2005 - Address Wrong Results issues
- 2006 - Participated in 11g Beta
- 2006 - Became an Open World Speaker
- 2007 - US Performance Support Senior Technical Leader
- 2008 - 2009 Part of Realworld Performance Panel
  & Support Stars Bar in OOW
- 2010 – Joined Exadata Support Team

Agenda

- What do I call a “Sin” ?
- Deep analysis of 4 Real World Cases supplied by customers
- Questions and Answers

What do I call a “Sin” ?

- A “Sin”, is a bad SQL Coding practice that causes great hardship to the business during database upgrades.
- It is a decision made long ago to which the consequences are seen much later.
- The usual argument from customers is that “it used to work”. The fact of being doing something wrong and getting away with it for long does not make it correct.
- The right thing to do is to change things to do them the right way.

Sin 1: Not placing ORDER BY when data is needed Sorted.

Sin 1: Not placing ORDER BY when data is needed Sorted.

```
create table gby ( key number );
begin
 for i in 1 .. 5000 loop
  insert into gby values(mod(DBMS_RANDOM.RANDOM, 3));
 end loop;
 commit;
end;
/
alter session set
optimizer_features_enable='9.2.0.8';
select key,count(*) from gby group by key;

alter session set
optimizer_features_enable='10.2.0.4';
select key,count(*) from gby group by key;
```

9.2.0.8

KEY	COUNT(*)
-2	824
-1	815
0	1714
1	857
2	790

10.2.0.4
11.2.0.3

KEY	COUNT(*)
1	857
2	790
-1	815
-2	824
0	1714

Sin 1: Not placing ORDER BY when data is needed Sorted.

9.2.0.8

```
---------------------------------
| Id  | Operation          | Name |
---------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  SORT GROUP BY     |      |
|   2 |   TABLE ACCESS FULL| GBY  |
---------------------------------
```

10.2.0.4 – 11.2.0.3

```
---------------------------------
| Id  | Operation          | Name |
---------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  HASH GROUP BY     |      |
|   2 |   TABLE ACCESS FULL| GBY  |
---------------------------------
```

order_by_clause

Use the ORDER BY clause to order rows returned by the statement. Without an order_by_clause, no guarantee exists that the same query executed more than once will retrieve rows in the same order.

Sin 1: Not placing ORDER BY when data is needed Sorted.

- Workaround _gby_hash_aggregation_enabled = false
- Has a performance penalty.
- PeopleSoft is one example of an application that has this issue.

Example : Operating System, RDBMS & Additional Component Patches Required for Installation PeopleTools 8.49 (Doc ID 749100.1)

Sin 2: Expect a function to be executed a particular number of times in a SQL

Sin 2: Expect a function to be executed a particular number of times in a SQL

```
CREATE or REPLACE function test_func return number AS
   aNum number:=dbms_random.random;
begin
   dbms_output.put_line('Number:'||aNum);
   return aNum;
end test_func;
/
set serveroutput on

select test_func from dual;

select x.* from
     (select test_func from dual) x;

select x.* from
     (select /*+ no_merge */ test_func from dual) x;
```


Sin 2: Expect a function to be executed a particular number of times in a SQL

```
SQL> set serveroutput on
SQL> select test_func from dual;

 TEST_DATA
----------
1459659030

Number:1459659030

SQL> select x.* from
  2  (select test_func from dual) x;

 TEST_DATA
----------
 660590733

Number:660590733
```

```
SQL> select x.* from
  2  (select /*+ no_merge */
  3     test_func from dual) x;

 TEST_DATA
----------
1883589361

Number:1883589361
Number:1328335145
Number:1224254326
```

Sin 2: Expect a function to be executed a particular number of times in a SQL

- Not related to number of rows
- Not related to column projection
- Not related to view merging only
- Behavior can be different by version and query transformation

```
-----------------------------------------
| Id  | Operation         | Name | Rows  |
-----------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |
|   1 |  VIEW             |      |     1 |
|   2 |   FAST DUAL       |      |     1 |
-----------------------------------------

Column Projection Information
 (identified by operation id):

   1 - "X"."TEST_FUNC"[NUMBER,22]
```

Sin 2: Expect a function to be executed a particular number of times in a SQL

```
SQL> select count(*)  from dual
  2  connect by level <=3;

  COUNT(*)
----------
         3

SQL>
SQL> select count(*)  from dual
  2  where  1 <> test_func
  3  connect by level <=3;

  COUNT(*)
----------
         3

Number:1080442802
Number:-1062755320
Number:698313632
```

```
SQL>
SQL> select count(*)  from dual
  2  where  1 <> (select test_func from dual)
  3  connect by level <=3;

  COUNT(*)
----------
         3

Number:-1620232937
```


Sin 2: Expect a function to be executed a particular number of times in a SQL

Can It Happen with Deterministic Functions too ?

```
CREATE or REPLACE function test_func(aNum number)  return number deterministic AS
begin
   dbms_output.put_line('Number:'||aNum);
   return aNum;
end test_func;
/
select test_func(1) from dual;

select x.* from (select test_func(1) from dual) x;

select x.* from (select /*+ no_merge */ test_func(1) from dual) x;

select test_func(rownum) from dual;

select x.* from (select test_func(rownum) from dual) x;

select x.* from (select /*+ no_merge */ test_func(rownum) from dual) x;
```

Sin 2: Expect a function to be executed a particular number of times in a SQL

```
SQL> select test_func(1)
 2 from dual;

TEST_FUNC(1)
------------
           1

Number:1
SQL>
SQL> select x.* from
 2 (select test_func(1)
 3 from dual) x;

TEST_FUNC(1)
------------
           1

Number:1
```

```
SQL>
SQL> select x.* from
 2 (select /*+ no_merge */ test_func(1)
 3 from dual) x;

TEST_FUNC(1)
------------
           1

Number:1
```


Sin 2: Expect a function to be executed a particular number of times in a SQL

```
SQL> select test_func(rownum)
 2 from dual;

TEST_FUNC(ROWNUM)
-----------------
                1

Number:1
SQL>
SQL> select x.* from
 2 (select test_func(rownum)
 3 from dual) x;

TEST_FUNC(ROWNUM)
-----------------
                1

Number:1
Number:1
```

```
SQL>
SQL> select x.* from
 2 (select /*+ no_merge */ test_func(rownum)
 3 from dual) x;

TEST_FUNC(ROWNUM)
-----------------
                1

Number:1
Number:1
```


Sin 2: Expect a function to be executed a particular number of times in a SQL

Documentation Bug 7239930 :RELATIONSHIP BETWEEN NO. OF SELECT LIST FUNCTION CALLS AND NO. OF ROWS RETURNED

Invoking Stored PL/SQL Functions from SQL Statements

Caution: Because SQL is a declarative language, rather than an imperative (or procedural) one, you cannot know how many times a function invoked from a SQL statement will execute—even if the function is written in PL/SQL, an imperative language.

If your application requires that a function be executed a certain number of times, do not invoke that function from a SQL statement. Use a cursor instead.

For example, if your application requires that a function be called once for each selected row, then open a cursor, select rows from the cursor, and call the function for each row. This guarantees that the number of calls to the function is the same as the number of rows fetched from the cursor.

Sin 2: Expect a function to be executed a particular number of times in a SQL

- No Workaround
- Performance impact is unpredictable

Sin 3: Expect predicates to be evaluated in a particular order

Sin 3: Expect predicates to be evaluated in a particular order

```
CREATE TABLE TEST_VAL( COL_01  NUMBER(15) NOT NULL, COL_02  NUMBER(15) NOT NULL,
  COL_03  NUMBER(15) NOT NULL, VAL_03  VARCHAR2(600),  VAL_04  VARCHAR2(600));
Insert into TEST_VAL (COL_01, COL_02, COL_03, VAL_03, VAL_04)
Values (111, 2222, 333, 'HSM', '        ');
COMMIT;

SQL> alter session set optimizer_features_enable='9.2.0';

Session altered.

SQL> SELECT COL_01 FROM TEST_VAL
  2  WHERE COL_02 = 2222 AND COL_03 = 333
  3  AND ( VAL_03 NOT IN ('Z1', 'Z2', 'ZD') OR VAL_04 != 0 );

    COL_01
----------
       111
```

Sin 3: Expect predicates to be evaluated in a particular order

```
SQL> alter session set optimizer_features_enable='10.2.0.4';

Session altered.

SQL> SELECT COL_01 FROM TEST_VAL
  2  WHERE COL_02 = 2222 AND COL_03 = 333
  3  AND ( VAL_03 NOT IN ('Z1', 'Z2', 'ZD') OR VAL_04 != 0 );
   OR VAL_04 != 0 )
      *
ERROR at line 4:
ORA-01722: invalid number
```

```
COL_01 | 111
COL_02 | 2222
COL_03 | 333
VAL_03 | 'HSM'
VAL_04 | '       '
```

```
SQL> alter session set events '10158 trace name context forever, level 1';

Session altered.

SQL> SELECT /*+ hardparse me */ COL_01 FROM TEST_VAL
  2  WHERE COL_02 = 2222 AND COL_03 = 333 AND ( VAL_03 NOT IN ('Z1', 'Z2', 'ZD') OR VAL_04 != 0);

    COL_01
----------
       111
```

Sin 3: Expect predicates to be evaluated in a particular order

9.2.0

```
--------------------------------------
| Id  | Operation         | Name     |
--------------------------------------
|   0 | SELECT STATEMENT  |          |
|*  1 |  TABLE ACCESS FULL| TEST_VAL |
--------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(("VAL_03"<>'Z1' AND "VAL_03"<>'Z2' AND "VAL_03"<>'ZD' OR
              TO_NUMBER("VAL_04")<>0) AND "COL_03"=333 AND "COL_02"=2222)
```

```
COL_01 | 111
COL_02 | 2222
COL_03 | 333
VAL_03 | 'HSM'
VAL_04 | '        '
```

10.2.0.4

```
-------------------------------------------------------------------------------
| Id  | Operation         | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |     1 |   643 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| TEST_VAL |     1 |   643 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("COL_02"=2222 AND "COL_03"=333 AND(TO_NUMBER("VAL_04")<>0 OR
              "VAL_03"<>'Z1' AND "VAL_03"<>'Z2' AND "VAL_03"<>'ZD'))
```

Sin 3: Expect predicates to be evaluated in a particular order

Documentation Bug.8554306 "CONDITION PRECEDENCE" IS MISSING INFO THAT THE CBO MAY REARRANGE CONDITIONS

Condition Precedence

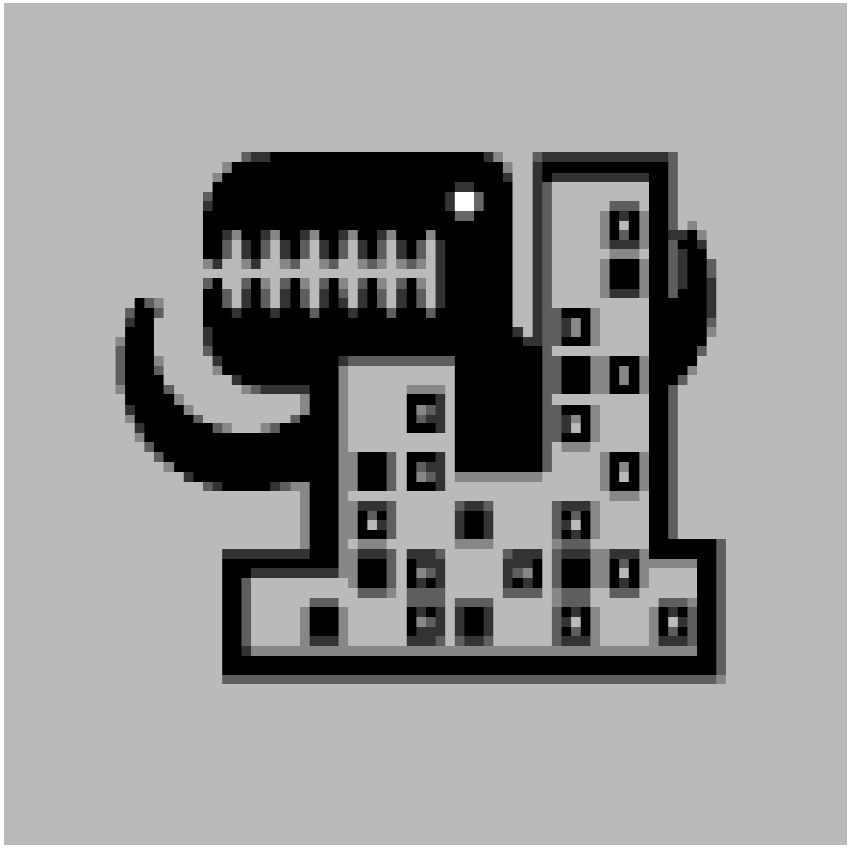
… Oracle evaluates conditions with equal precedence from left to right within an expression, with the following exceptions:

- Left to right evaluation is not guaranteed for multiple conditions connected using AND
- Left to right evaluation is not guaranteed for multiple conditions connected using OR

Sin 3: Expect predicates to be evaluated in a particular order

- Workaround (though not very effective)
  - ORDERED_PREDICATES hint
  - Event 10158
  - Changing the query

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.



Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

```
SELECT tablespace_name, 'literal value' str, count(*)
FROM all_tables
GROUP BY tablespace_name;
```

Customer Says “In the above query, the literal is not required to be included in the GROUP BY clause, as the server understands that the value is unchanging over the data set and thus does not need to be included.

This query will run in both our 10g and 11g environments.

However, if I nest the above query using the WITH clause but leave the GROUP BY in the outer query, things become a bit murkier:”

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

```
WITH qry AS
(SELECT tablespace_name, 'literal value' str
  FROM all_tables
)
SELECT tablespace_name, str, count(*)
FROM qry
GROUP BY tablespace_name;
```

“In 10g, this query executes without error, while in 11g, I get the following error:

ORA-00979: not a GROUP BY expression

So, the 10g server still identifies that the 'str' column is a literal and does not require inclusion in the GROUP BY clause, while the 11g server does not. I can run the query in 11g only if I include 'str' in the GROUP BY clause.

In our query that is failing in our 11g test environment, I happen to be binding in the literal value at runtime, but, as I demonstrated above, this is not a requirement. The failure is caused by the nesting of a query containing a literal value, with the outer query performing a GROUP BY.” – Customer.

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

```
Alter session set optimizer_features_enable='10.2.0.1';

WITH qry AS
(SELECT tablespace_name, 'literal value' str
  FROM all_tables
)
SELECT tablespace_name, str, count(*)
FROM qry
GROUP BY tablespace_name;

TABLESPACE_NAME                STR             COUNT(*)
------------------------------ ------------- ----------
                               literal value         247
SYSAUX                         literal value         497
USERS                          literal value           3
SYSTEM                         literal value         740
```

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

```
Alter session set optimizer_features_enable='11.2.0.1';

WITH qry AS
(SELECT tablespace_name, 'literal value' str
  FROM all_tables
)
SELECT tablespace_name, str, count(*)
FROM qry
GROUP BY tablespace_name;

SELECT tablespace_name, str, count(*)
                        *
ERROR at line 5:
ORA-00979: not a GROUP BY expression
```

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

```
alter session set "_fix_control"= '5520732:off';

WITH qry AS
(SELECT tablespace_name, 'literal value' str
  FROM all_tables
)
SELECT tablespace_name, str, count(*)
FROM qry
GROUP BY tablespace_name;

TABLESPACE_NAME                STR             COUNT(*)
------------------------------ ------------- ----------
                               literal value        247
SYSAUX                         literal value        497
USERS                          literal value          3
SYSTEM                         literal value        740
```

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

Oracle Dev Answer in bug 5520732

The checks for whether the group by list contains the columns/expressions in the select list are being done after the view is already merged, so the error is not raised.

However, the query is in fact illegal and should not sometimes work depending on what transformations are chosen.

So, The error is the expected behavior.

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

```
Alter session set optimizer_features_enable='10.2.0.1';

WITH qry AS
(SELECT /*+ NO_MERGE */ tablespace_name, 'literal value' str
  FROM all_tables
)
SELECT tablespace_name, str, count(*)
FROM qry
GROUP BY tablespace_name;

SELECT tablespace_name, str, count(*)
                        *
ERROR at line 5:
ORA-00979: not a GROUP BY expression
```

Sin 4:To depend on a transformation to be applied to the query for the successful execution of the query.

- Workarounds
  - `_fix_control` or Parameters
  - Look for MOS notes with the error
  - Review release notes with the phrase
    “Notable change of behaviour introduced in {version}”.

Q&A

Abel.macias@oracle.com

Hardware and Software

ORACLE

Engineered to Work Together

Backup Slides

