

ORACLE DATABASE PL/SQL

Select Language ▼

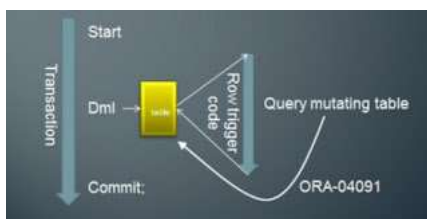
Powered by Google Translate



MUTATING Table Error and How to Resolve it?

Mutating table Error Occurs when Trigger is Querying or Modifying a “Mutating Table”

Mutating error normally occurs when we are performing some DML operations and we are trying to select the affected record from the same trigger. So basically we are trying to select records in the trigger from the table that owns the trigger. This creates inconsistency and Oracle throws a mutating error.



Let's take an example in which we have to know total number Employees with Active status after any status is updated to 'Active'. We will see it with an example. First let us create a table and then trigger.

Create a table and insert records into that table.

Here is the table definition I have created and inserted some records.

```
SQL> DESC TEST;
Name                Null?  Type
-----
ID                   NUMBER
NAME                 VARCHAR2(10)
SALARY               NUMBER(20)
STATUS               VARCHAR2(10)

SQL> SELECT * FROM TEST;
  ID  NAME    SALARY STATUS
-----
 102  Ankit    8000   Active
 104  Nikhil   69000  Active
 105  Rajan    18000  InActive
 107  Karan   101000 Active
 110  Sajal    88000  InActive
 101  Ravi     89000  Active
 109  Manu     777000 InActive

7 rows selected.
```

Now, Let's create a Mutating Trigger (Row Level) on Update of Status column of above table.

```
SQL> CREATE OR REPLACE TRIGGER MutatingTrigger /*Row Level TRIGGER*/
2 AFTER UPDATE OF STATUS ON TEST
3 FOR EACH ROW
4 DECLARE
5 V_count NUMBER;
6 BEGIN
7 Select count(*) Into V_count From TEST
8 Where STATUS='Active';
9 DBMS_OUTPUT.PUT_LINE('Total Number of Active Records: '|| V_count);
10 END;
```

Follow For More UPDATES

Ravikant Baluni

G+ Follow

94 followers



ORACLE Topics

- COLLECTIONS and TYPES of COLLECTI
- Which Collection type should be used
- Examples of COLLECTIONS and COLLECTION Methods
- DATABASE Normalization Techniques
- PACKAGE Overloading
- Creating PACKAGES and Call it's Methc
- ORACLE 11g Features
- CONTINUE and CONTINUE WHEN Statement
- Passing parameters in Functions/Procedures
- Stored Procedure Vs. Functions

```
11 /
Trigger created.
```

Now if we try to change status of any record to "Active", Oracle will throw a **Mutating Table Error** as we are trying to update the records and trigger is trying to select affected records in the same trigger.

```
SQL> UPDATE TEST
2 SET STATUS='Active' Where ID IN(109,110);
UPDATE TEST
*
ERROR at line 1:
ORA-04091: table HR.TEST is mutating, trigger/function may not see it
ORA-06512: at "HR.MUTATINGTRIGGER", line 4
ORA-04088: error during execution of trigger 'HR.MUTATINGTRIGGER'
```

So, When you encounter an ORA-04091 error, the following error message will appear:

ORA-04091: table name is mutating, trigger/function may not see it

Cause - A statement executed a **Trigger** or **custom PL/SQL Functions**. That trigger/function tried to modify or query a table that is currently being modified by the statement that fired the trigger/function.

Now Let's Resolve this error.

Solution 1:

Create Statement Level Trigger instead of Row Level Trigger. Row Level Triggers can't read Mutating tables- Change Row Level Trigger to Statement Level Trigger:

If we omit the 'For Each Row' clause from above Trigger, it will become statement level Trigger. Let's modify above trigger to statement level trigger.

```
SQL> CREATE OR REPLACE TRIGGER MutatingTrigger /*Statement Level TRIGGER*/
2 AFTER UPDATE OF STATUS ON TEST
3 DECLARE
4 V_count NUMBER;
5 BEGIN
6 Select count(*) Into V_count From TEST
7 Where STATUS='Active';
8 DBMS_OUTPUT.PUT_LINE('Total Number of Active Records: ' || V_count);
9 END;
10 /
Trigger created.
```

Let's try to fire the same Update statement now and see if you still get Mutating Error.

```
SQL> UPDATE TEST
2 SET STATUS='Active' Where ID IN(109,110);
Total Number of Active Records: 6
2 rows updated.
SQL> SELECT * From TEST;
   ID NAME      SALARY STATUS
-----
  102 Ankit      8000   Active
  104 Nikhil     69000  Active
  105 Rajan     18000  InActive
  107 Karan     101000 Active
  110 Sajal     88000  Active
  101 Ravi       89000  Active
  109 Manu      777000 Active
7 rows selected.
```

As you can see it works fine with Statement Level Trigger. With 'For Each Row', session cannot query the same table. This restriction applies to all row level triggers and hence we run into mutating table error.

Solution2:

Make the transaction independent using PRAGMA AUTONOMOUS TRANSACTION.

Declare a Row level trigger as an Autonomous Transaction so that it is not in the same scope of the session issuing DML statements.

The **AUTONOMOUS_TRANSACTION** PRAGMA changes the way a subprogram works within a transaction. A subprogram marked with this PRAGMA can do SQL operations and commit or roll back those operations, without committing or rolling back the data in the main transaction. Autonomous transactions allow you to leave the context of the calling transaction, perform an independent transaction, and return to the calling transaction without affecting its state.

- SQL Query Order Execution
- DWH(OLAP) Vs. Operational DB(OLTP)
- Data Migration Steps and SCD Change:
- ROLLBACK behaviour when FOR ALL is used
- Handling BULK Exception using SAVE EXCEPTION
- BULK Collect with NATIVE Dynamic SQ
- BULK Collect and Collection of Recor
- Using BULK Collect and BULK Binds
- ORACLE Table Locking
- How to kill ORACLE Session?
- Handling PL/SQL Errors(Exception Handling)
- RAISE_APPLICATION_ERROR Built-IN Procedure
- Exception Trapping Functions
- WHERE and HAVING clause Alternative
- TRIGGER and Types of TRIGGERS
- Identify Columns having all NULLS
- TABLE Vs. MATERIALIZED View
- VIEWS in ORACLE
- SYNONYMS in ORACLE
- How INDEXES stored in DB
- Local and Global Indexes
- CLUSTERED and NON-CLUSTERED Inde
- INDEXES in ORACLE
- Opening Parameterized Cursor in Different ways
- Sub-Queries-And-Types-of-Sub-Querie
- COMMIT inside Trigger
- Difference between Primary and Uniq Key
- Difference between %TYPE Vs. %ROWTYPE
- WITH Clause in ORACLE
- DECODE Vs. CASE
- ROWNUM Vs. ROW_NUMBER()
- ROWNUM Vs. ROWID
- INSERT and DELETE Execution Plan
- Different types of JOINS in ORACLE
- NOT IN Vs. NOT EXISTS Operator
- IN Vs. EXISTS Operator
- How Count Function behaves with different operators
- DELETE Vs. TRUNCATE Vs. DROP
- Find Highest/Minimum Salary and Employee Information
- Identify and Remove DUPLICATE Reco
- MUTATING Table Error and How to Avc It.
- GLOBAL TEMPORARY Tables in ORACLE
- CHAR-NCHAR-VARCHAR-VARCHAR2-NVARCHAR
- UNION Vs. UNION ALL(SET OPERATORS)
- How CURSOR works Internally?
- ORACLE Cursors and its Types
- Separate NUMERIC/NON-NUMERIC/DA values From a Column
- ANALYTICAL Vs. AGGREGATE Function
- DBMS_PROFILER Installation Steps
- How DBMS_PROFILER helps in identify long running SQL's
- ORACLE SQL Tuning Tips
- Dynamic Where Clause
- ORACLE SQL EXECUTION PLAN
- COLLECTION having NULL Values
- ORACLE SQL* Loader
- ORACLE External Tables
- RULE BASED and COST BASED OPTIMIZ
- OPTIMIZER Modes in ORACLE
- ORACLE Driving Tables

Following is the ROW Level Trigger defined as **PRAGMA AUTONOMOUS TRANSACTION**.

```
SQL> CREATE OR REPLACE TRIGGER MutatingTrigger
2 AFTER UPDATE OF STATUS ON TEST
3 FOR EACH ROW
4 DECLARE
5 V_count NUMBER;
6 PRAGMA AUTONOMOUS_TRANSACTION;
7 BEGIN
8 Select count(*) Into V_count From TEST
9 Where STATUS='Active';
10 DBMS_OUTPUT.PUT_LINE('Total Number of Active Records: '|| V_count);
11 END;
12 /
Trigger created.
```

Now let's issue the Update statement again and see if now it works with ROW Level Trigger as now we have Created a Trigger with PRAGMA.

```
SQL> Select * From TEST;
  ID NAME      SALARY STATUS
-----
 102 Ankit      8000 Active
 104 Nikhil     69000 Active
 105 Rajan     18000 InActive
 107 Karan     101000 Active
 110 Sajal      88000 InActive
 101 Ravi       89000 Active
 109 Manu       777000 InActive
```

7 rows selected.

```
SQL> UPDATE TEST
2 SET STATUS='Active' Where ID IN(109,110);
```

Total Number of Active Records: 4

Total Number of Active Records: 4

2 rows updated.

```
SQL> SELECT * From TEST;
  ID NAME      SALARY STATUS
-----
 102 Ankit      8000 Active
```

By defining Row Level trigger as an AUTONOMOUS TRANSACTION, we got rid of Mutating table error but result is not correct. Updated records are not getting reflected. So one has to be very careful while using this approach.

Solution 3:

Avoid Mutating Error Using Compound Trigger. In Oracle 11g, Oracle has made it much easier by introducing Compound Trigger.

A compound trigger allows code for one or more timing points for a specific object to be combined into a single trigger. The individual timing points can share a single global declaration section, whose state is maintained for the lifetime of the statement. Once a statement ends, due to successful completion or an error, the trigger state is cleaned up.

Let's create a Compound Trigger to resolve Mutating Table Error.

```
SQL> CREATE OR REPLACE TRIGGER MutatingTrigger
2 FOR UPDATE
3 ON TEST
4 COMPOUND TRIGGER
5 /*Declaration Section*/
6 V_count NUMBER;
7
8 AFTER EACH ROW IS
9 BEGIN
10 DBMS_OUTPUT.PUT_LINE('Record Updated');
11 END AFTER EACH ROW;
12
13 AFTER STATEMENT IS
14 BEGIN
15 Select count(*) Into V_count
16 From TEST Where STATUS = 'Active';
17 DBMS_OUTPUT.PUT_LINE('Total Number of Active Records: '|| V_count);
18 END AFTER STATEMENT;
19
20 END MutatingTrigger;
```

- EXECUTION PLAN and Its Components
- CURSOR_SHARING in ORACLE
- INDEX Usage with LIKE Operator and DOMAIN Index
- DYNAMIC_SAMPLING and its Impact on OPTIMIZER
- NOT NULL and Indexed Column
- ORACLE AUTOTRACE Utility
- Pass COMMA Separated Value to IN Operator
- ANALYTICAL & AGGREGATE Functions Examples
- ORACLE UTL_FILE Package
- ORACLE UTL_FILE Exceptions
- UTL_FILE Operations and Functions
- Comma Separated Values
- RETURNING Table From a Function
- REGULAR Expressions in ORACLE
- RESTRICT DROP/TRUNCATE on TABLE
- Export Table Data to CSV
- IMPORT Data from Flat Files to ORACLE Tables
- ORACLE PIVOT/UNPIVOT
- PARTITIONING IN ORACLE
- Equality Test of Two COLLECTION Types
- Compare and Merge COLLECTION Objects
- NESTED Table Functions
- DETERMINISTIC FUNCTIONS
- HANDLING CURSOR Exceptions
- When CROSS JOIN Will Be Useful?
- DML Error Logging
- Handle CONCURRENT Updates
- Pessimistic and Optimistic Oracle Locking
- Returning REF CURSOR From a Procedure
- Prevent VALUE_ERROR Exception
- SYS_REFCURSOR Vs. REF CURSOR



TIMEX FASHION ANALOG
SILVER DIAL WOMEN'S ...

Rs. 1,741.00
(details + delivery)



IMAGIMAKE
INDIA WIT

Rs. 374.00
(details -)



FOSSIL J/
ANALOG B

Rs. 8,
(details -)

Popular Posts

- Analytic Functions Vs. Aggregate Functions

21 /
Trigger created.

Now let's issue the Update statement and see how it works with **Compound Trigger**.

```
SQL> SELECT * From TEST;
  ID NAME      SALARY STATUS
-----
 102 Ankit      8000 Active
 104 Nikhil     69000 Active
 105 Rajan      18000 InActive
 107 Karan     101000 Active
 110 Sajal      88000 InActive
 101 Ravi       89000 Active
 109 Manu       777000 InActive

7 rows selected.

SQL> UPDATE TEST
 2 SET STATUS='Active' Where ID IN(109,110);
Record Updated
Record Updated
Total Number of Active Records: 6
2 rows updated.

SQL> SELECT * From TEST;
  ID NAME      SALARY STATUS
-----
 102 Ankit      8000 Active
 104 Nikhil     69000 Active
 105 Rajan      18000 InActive
 107 Karan     101000 Active
 110 Sajal      88000 Active
 101 Ravi       89000 Active
 109 Manu       777000 Active

7 rows selected.
```

Here we get correct result without getting any Mutating Table error.

There are also other ways to resolve this problem like using Temporary Table. I will update this or post a new blog for the same.

Get involved and leave your Comments in the Box Below. The more people get involved, the more we all benefit. So, leave your thoughts before you leave the page.



2 comments:

Anonymous October 17, 2016 at 10:15 AM

nicely explained...

Reply

Replies



Ravikant Baluni

January 29, 2017 at 4:18 AM

Thank You.

Reply

Enter your comment...



Comment as:

Unknown (Go ▼)

Sign out

Publish

Preview

☐ Notify me

- [ORACLE COLLECTIONS](#)
- [ORACLE SQL* Loader](#)
- [EXPORT TABLE Data to Flat Files](#)
- [SYS_REFCURSOR Vs. REF CURSOR](#)
- [ORACLE TABLE PARTITIONING](#)
- [ORACLE PL/SQL Tuning](#)
- [UTL_FILE Import Data into ORACLE TA](#)
- [ORACLE UTL_FILE](#)
- [Using BULK Collect](#)



[FOSSIL](#)

4/18/2018

ORACLE SQL, PL/SQL: MUTATING Table Error and How to Resolve it?

[Newer Post](#)

[Home](#)

[Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)

BaluniGroups. Travel theme. Powered by [Blogger](#).