Resolving ORA-00001 ERROR: Unique Constraint Violated (KBA1817)

The page describes how to best mask an Oracle database table and how to avoid and resolve ORA-00001: Unique Constraint Violated.

At a Glance

ORA Error: ORA-00001: Unique Constraint Violated
Affects versions: This issue affects all Masking Versions.
Root Cause: A rule which is defined in the database has been violated when a value was tried to be inserted/updated that is already in the column.

This is Oracle's way to say 'Stop! I can't proceed or I break rules that the database owner has defined'. On Oracle, there is one rule to a Unique Constraint:

The Rule: 1. A **Unique Constraint** ensures that all values in a column are unique - are different.

Note: On Oracle, a **Unique Constraint** accepts multiple **NULLs**.

There are two potential issues when masking:

The Issue:
1. The masked value (lookup or generated) is not **Unique** and hence **duplicated**.
2. The masked value (lookup of generated) is unique but it is already in the table and hence **duplicated**.
3. The Masking Engine (or the user) failed to disable the **Unique Constraint**.

There are two things to think about here and a potential workaround for the 3rd issue:

Solution:
1. Ensure the Algorithm is generating a UNIQUE masked value that is mapped 1:1 (for example Mapping or Segment Mapping Algorithm).
2. Select **Disable Constraints** in the masking job.

The first point above is mandatory. If the values aren't unique and mapped 1:1 Oracle is likely to throw the following error at the end of the job: ORA-02299: cannot validate (xxx) - duplicate keys found.

3. If the disable failed due to owner issues or complexity of the constraint, please disable/re-enable manually.
Masking Job and ORA-00001: Unique Constraint Violated

This article details the root cause and a solution to ORA-00001 error (Unique Constraint Violated) from a masking perspective. The error ORA-00001 is common in masking but is also easy to understand and resolve.

As mentioned, it is Oracle's way to say 'Stop! I can't proceed or I break rules that the database owner has defined'. So the key is to know that this is an error caused by a rule the database designer defined.

So what is a Unique Constraint and why do we get it?

Unique Constraint

This is what defines a Unique Constraint (Oracle):

- A **Unique Constraint** ensures that all values in a column are unique - are different.

Notes:

- On Oracle, a **Unique Constraint** accepts multiple NULLs.
- A **Unique Constraint** is in many ways the same as a **Primary Key Constraint**. There is this one major difference a table can have multiple Unique Constraints while it can only have one Primary Key.

Is this unique to Oracle? No

Unique Constraints is an integrity constraint defined in SQL. Here are other examples from different databases.

- **DB2**
  - 'SQLSTATE: 23505. A violation of the constraint imposed by a unique index or a unique constraint occurred.'

- **MS SQL Server**
  - 'Violation of PRIMARY KEY constraint '[CONSTRAINT]'. Cannot insert duplicate key in object '[TABLE]'. The duplicate key value is ([VALUE]).'

- **MySQL**
  - '-- ERROR 1062 (23000): Duplicate entry '[NEW_VALUE]' for key [VALUE]'

- **Sybase**
  - 'Check constraint violation occurred, dbname = [DATABASE], table name = [TABLE], constraint name = [CONSTRAINT].'

---

Example
**Unique Constraint violation example**

In this section, I will show two different types of Constraint Violations possible when masking a column with a Unique Constraint.

- Example 1: The masked value (lookup or generated) is not **Unique** and hence **duplicated**.
- Example 2: The masked value (lookup or generated) is unique but it is already in the table and hence **duplicated**.

The database error message for these two root causes is the same.

**Background details**

To visualize this, let's create an example. This example will show both types of violations by using an algorithm that is not producing a 1:1 unique mapping (in this example: Secure Lookup with 25 values). The allocation of the masked value is random and the same masked value can appear more than once.

To simplify the illustration the **Commit Size** has been set to 4.

**Explanation and Execution**

In the example, the error happens in the second Commit.

Steps:

- **Update set 1**: Update sets the following 4 masked values (15, 2, 32, 21).

- **Commit 1**: Success - this step is only to highlight errors in example 1.
  - The first 4 masked values are Committed ok.
  - Note that row with ID: ‘1’ is masked to ‘2’. This will later violate a duplicated masked value.

- **Update set 2**: Update sets the following 4 masked values (2, 4, 9, 16).

- **Commit 2**: Error - the second lot of 4 values are masked and the Commit is failing.
  - We are masking ‘6’ to ‘9’ and ‘4’ to ‘2’.
  - Both these violate the constraint for two different reasons.

- **Reason UPDATE Example 1**
  - The masked value ‘2’ is also already in the table.
  - Since the mapping is not 1:1 and unique. ‘1’ is masked to ‘2” and so is ‘4’ as well.
- **Reason UPDATE Example 2**
  - The masked value '9' is already in the table and has not been masked yet.
  - We can never allocate '9' as this will violate the **Unique Constraint**.

---

**Note:** Example 1 **CANNOT** be resolved by Disable Constraints. To resolve example 1 you also need to have an algorithm that maps 1:1, is unique **AND** the data in the column needs to be unique.

More about the resolution below.

<table>
<thead>
<tr>
<th>Original Table # before masking</th>
<th>Masked Table # after masking</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SELECT * FROM Msk_Tbl;</code></td>
<td><code>SELECT * FROM Msk_Tbl;</code></td>
</tr>
<tr>
<td><code>+----+---------+</code></td>
<td><code>+----+---------+</code></td>
</tr>
<tr>
<td>`</td>
<td>ID</td>
</tr>
<tr>
<td><code>+----+---------+</code></td>
<td><code>+----+---------+</code></td>
</tr>
<tr>
<td>`</td>
<td>0</td>
</tr>
<tr>
<td>`</td>
<td>1</td>
</tr>
<tr>
<td>`</td>
<td>2</td>
</tr>
<tr>
<td>`</td>
<td>3</td>
</tr>
<tr>
<td><code>+----+---------+</code></td>
<td><code>+----+---------+</code></td>
</tr>
<tr>
<td>`</td>
<td>4</td>
</tr>
<tr>
<td>`</td>
<td>5</td>
</tr>
<tr>
<td>`</td>
<td>6</td>
</tr>
<tr>
<td>`</td>
<td>7</td>
</tr>
<tr>
<td><code>+----+---------+</code></td>
<td><code>Stop &gt; +----+---------+</code></td>
</tr>
<tr>
<td>`</td>
<td>8</td>
</tr>
<tr>
<td>`</td>
<td>9</td>
</tr>
<tr>
<td>`</td>
<td>..</td>
</tr>
<tr>
<td><code>+----+---------+</code></td>
<td><code>+----+---------+</code></td>
</tr>
</tbody>
</table>

---

**Error message**

The error message we see on a Oracle database is:

- `'ORA-00001: unique constraint ([UNIQUE_CONSTRAINT]) violated'`
Resolution

How to mask columns with Unique Constraints?

1. **Setup**: Define an appropriate **Algorithm** for the masked column(s) and the data.
2. **Inventory**: Assign the **Algorithm** to the **Column**.
3. **Job Configuration**: Select **Disable Constraints** in the Job Configuration.

Fail to Disable Constraints

If the issue is that the Unique Constraint can't be dropped using the **Disable Constraints** due to the constraint is defined using a different **Owner** or the Constraint is too complex then please **Disable** and then **Re-Enable** the **Constraint** manually.

Related Articles

- [Algorithm Summary (KBA4066)](#)
- [Resolving ORA-02299 Error: cannot validate (x) - duplicate keys found (KBA4524)](#)