How to Enable and Disable Constraints

Using PL/SQL Scripts

Asfaw Gedamu

July 31,2023

Database constraints are rules that are enforced on the data in a table. They can be used to ensure the accuracy, consistency, and integrity of the data.

There are many different types of constraints in Oracle Database, including:

- **Primary key constraints** ensure that each row in a table has a unique value in a particular column.
- **Foreign key constraints** ensure that the values in a column in one table refer to the values in a column in another table.
- **Check constraints** ensure that the values in a column meet certain criteria. For example, a check constraint could be used to ensure that the value in a column is between 1 and 100.
- **Unique constraints** ensure that the values in a column are unique, but not necessarily different from each other. For example, a unique constraint could be used to ensure that the email addresses in a table are unique.

Constraints can be used to improve the performance of queries and to prevent data errors. For example, a primary key constraint can be used to speed up queries that join two tables, and a check constraint can be used to prevent users from entering invalid data into a table.

Here is a visualization of how constraints are enforced in Oracle Database:

```
Database
   |
   |-- Table1 (column1, column2, column3)
   |
   |    |-- Primary key constraint on column1
   |    |-- Foreign key constraint on column2 referencing column1 in Table2
   |    |-- Check constraint on column3
   |
   `-- ...
```

As you can see, constraints are enforced at the table level. This means that the constraints are applied to all rows in the table.

Here are some examples of how constraints can be used in Oracle Database:

- A primary key constraint could be used to ensure that the customer ID in a customer table is unique.
- A foreign key constraint could be used to ensure that the product ID in an order table refers to a product that exists in a product table.
- A check constraint could be used to ensure that the quantity ordered in an order table is greater than 0.

Constraints are a powerful tool that can be used to improve the accuracy, consistency, and integrity of the data in an Oracle Database. They are a valuable tool for database administrators and developers.

There are a few reasons why you might need to disable database constraints. Here are some concrete use cases:

- **Bulk loading data:** When you're bulk loading data into a table, you might want to disable constraints to speed up the process. This is because constraints can slow down bulk loads, especially if the constraints involve complex calculations.
- **Changing the schema:** If you're making changes to the schema of a table, you might need to disable constraints to prevent errors. For example, if you're adding a new column to a table, you'll need to disable the constraints on the other columns so that you can insert rows into the new column without violating the constraints.
- **Troubleshooting:** If you're troubleshooting a problem with a table, you might need to disable constraints to see if they're causing the problem. For example, if you're getting an error message that says "violates unique constraint," you might want to disable the unique constraint to see if that resolves the problem.

It's important to note that disabling constraints can make the data in your table less secure and less consistent. Therefore, you should only disable constraints when absolutely necessary. And once you've finished making the changes to your table, you should re-enable the constraints to ensure the integrity of your data.

Here are some additional considerations when disabling constraints:

- **Make sure you understand the implications of disabling constraints.** Disabling constraints can make the data in your table less secure and less consistent.
- **Only disable constraints when absolutely necessary.** If you can avoid disabling constraints, do so.
- **Re-enable constraints as soon as possible.** Once you've finished making the changes to your table, re-enable the constraints to ensure the integrity of your data.

But, how can you enable or disable table constraints? Well, you can use the following PL/SQL scripts.

1. Enables all check constraints for a specified table, or all tables.

```
DECLARE
  schema_name VARCHAR2(30);
  table_name VARCHAR2(30);
BEGIN
  -- Prompt the user for the schema name and table name.
  dbms_output.put_line('Enter the schema name:');
  schema_name := dbms_input.get_string(undef);
  dbms_output.put_line('Enter the table name:');
  table_name := dbms_input.get_string(undef);

  -- Enable the constraints.
  EXECUTE IMMEDIATE
    'SELECT '
    '"ALTER TABLE "' || schema_name || '.' || table_name || '" ENABLE CONSTRAINT "' ||
a.constraint_name || '";'
    FROM   all_constraints a
    WHERE  a.constraint_type = 'C'
```

```
   AND    a.owner          = Upper('&2')
   AND    a.table_name     = DECODE(Upper('&1'),'ALL',a.table_name,UPPER('&1'));';
END;
```

This PL/SQL script enables all constraints for the specified table or all tables. First, it declares two variables, schema_name and table_name, to store the schema name and table name, respectively. Then prompts the user for the schema name and table name. The user input is stored in the schema_name and table_name variables. Finally, it uses the EXECUTE IMMEDIATE statement to execute the ALTER TABLE command to enable the constraints.

To run the script, you would need to run the following command from the command line:

```
sqlplus username/password@database < enable_constraints.sql
```

2. Disable all check constraints of a specified table, or all tables.

```
SET SERVEROUTPUT ON
DECLARE
  v_table_name VARCHAR2(30) := UPPER('&1');
  v_owner      VARCHAR2(30) := UPPER('&2');
BEGIN
  FOR c IN (SELECT constraint_name
            FROM   all_constraints
            WHERE  constraint_type = 'C'
            AND    owner          = v_owner
            AND    table_name     = DECODE(v_table_name,'ALL',table_name,v_table_name))
  LOOP
    DBMS_OUTPUT.PUT_LINE('Disabling constraint ' || c.constraint_name || ' on table ' ||
v_owner || '.' || v_table_name);
    EXECUTE IMMEDIATE
```

```
    'ALTER TABLE "' || v_owner || '"."' || v_table_name || '" DISABLE CONSTRAINT "' ||
    c.constraint_name || '"';
  END LOOP;
END;
/
```

This script uses PL/SQL to disable all check constraints of a table in an Oracle database. It uses a cursor to select all check constraints for a given owner and table name. It then generates an ALTER TABLE statement for each constraint that disables the constraint.

To run the script, you would need to run the following command from the command line:

```
sqlplus username/password@database < disable_constraints.sql
```

3. Enables the Primary Key for the specified table, or all tables.

```
-- Enables all primary keys for the specified table or all tables.

DECLARE
  schema_name VARCHAR2(30);
  table_name VARCHAR2(30);
BEGIN
  -- Prompt the user for the schema name and table name.
  dbms_output.put_line('Enter the schema name:');
  schema_name := dbms_input.get_string(undef);
  dbms_output.put_line('Enter the table name:');
  table_name := dbms_input.get_string(undef);

  -- Enable the primary keys.
  EXECUTE IMMEDIATE
    'ALTER TABLE "' || schema_name || '.' || table_name || '" ENABLE PRIMARY KEY;';
END;
```

This PL/SQL script enables all primary keys for the specified table or all tables. First, it declares two variables, schema_name and table_name, to store the schema name and table name, respectively. Then, it prompts the user for the schema name and table name. The user input is stored in the schema_name and table_name variables. Finally, it uses the EXECUTE IMMEDIATE statement to execute the ALTER TABLE command to enable the primary keys.

To run the script, you would need to run the following command from the command line:

```
sqlplus username/password@database < enable_primary_keys.sql
```

4. Disable the Primary Key of a specified table, or all tables.

```
-- Disables all primary keys for the specified table or all tables.

DECLARE
  schema_name VARCHAR2(30);
  table_name VARCHAR2(30);
BEGIN
  -- Prompt the user for the schema name and table name.
  dbms_output.put_line('Enter the schema name:');
  schema_name := dbms_input.get_string(undef);
  dbms_output.put_line('Enter the table name:');
  table_name := dbms_input.get_string(undef);

  -- Disable the primary keys.
  EXECUTE IMMEDIATE
    'ALTER TABLE "' || schema_name || '.' || table_name || '" DISABLE PRIMARY KEY;';
END;
```

Tthis PL/SQL cript disables all primary keys for a specified table or all tables. First, it declares two variables, schema_name and table_name, to store the schema name and table name, respectively.Then prompts the user for the schema name and table name. The user input is stored in the schema_name and table_name variables. Finally, the script uses the EXECUTE IMMEDIATE statement to execute the ALTER TABLE command to disable the primary keys.

To run the script, you would need to run the following command from the command line:

```
sqlplus username/password@database < disable_primary_keys.sql
```

5. Enables all Foreign Keys belonging to the specified table, or all tables.

```
-- Enable all foreign key constraints in a table
DECLARE
  v_table_name VARCHAR2(30) := UPPER('&1');
  v_owner      VARCHAR2(30) := UPPER('&2');
BEGIN
  FOR c IN (SELECT constraint_name
                 , table_name
                 , owner
            FROM   all_constraints
            WHERE  constraint_type = 'R'
            AND    table_name      = DECODE(v_table_name,'ALL',table_name,v_table_name)
            AND    owner           = v_owner)
  LOOP
    EXECUTE IMMEDIATE
      'ALTER TABLE '||c.owner||'.'||c.table_name||' ENABLE CONSTRAINT
      '||c.constraint_name;
  END LOOP;
END;
```

```
/
```

This PL/SQL block enables all foreign key constraints in a table. It declares two variables v_table_name and v_owner to store the table name and owner name respectively. It then uses a FOR loop to select all foreign key constraints from the table and generates an ALTER TABLE statement for each constraint. The ALTER TABLE statement enables the constraint by setting it to the ENABLE VALIDATE state. Finally, it executes the generated ALTER TABLE statements using EXECUTE IMMEDIATE.

6. Disables all Foreign Keys belonging to the specified table, or all tables.

```
-- Disable all foreign key constraints in a table
DECLARE
  v_table_name VARCHAR2(30) := UPPER('&1');
  v_owner      VARCHAR2(30) := UPPER('&2');
BEGIN
  FOR c IN (SELECT constraint_name
                 , table_name
                 , owner
            FROM   all_constraints
            WHERE  constraint_type = 'R'
            AND    table_name      = DECODE(v_table_name,'ALL',table_name,v_table_name)
            AND    owner           = v_owner)
  LOOP
    EXECUTE IMMEDIATE 'ALTER TABLE '||c.owner||'.'||c.table_name||' DISABLE
CONSTRAINT '||c.constraint_name;
  END LOOP;
```

```
END;
/
```

This PL/SQL block disables all foreign key constraints in a table. It declares two variables v_table_name and v_owner to store the table name and owner name respectively. It then uses a FOR loop to select all foreign key constraints from the table and generates an ALTER TABLE statement for each constraint. The ALTER TABLE statement disables the constraint by setting it to the DISABLE state. Finally, it executes the generated ALTER TABLE statements using EXECUTE IMMEDIATE.

7. Enable all foreign keys that reference the specified table or all tables.

```
-- Enables all foreign keys that reference the specified table or all tables.
DECLARE
  schema_name VARCHAR2(30);
  table_name VARCHAR2(30);
BEGIN
  -- Prompt the user for the schema name and table name.
  dbms_output.put_line('Enter the schema name:');
  schema_name := dbms_input.get_string(undef);
  dbms_output.put_line('Enter the table name:');
  table_name := dbms_input.get_string(undef);
  -- Enable the foreign keys.
  EXECUTE IMMEDIATE
    'SELECT '
    '"ALTER TABLE "' || schema_name || '.' || table_name || '" ENABLE CONSTRAINT "' ||
a.constraint_name || '";'
```

```
   FROM   all_constraints a
   WHERE  a.owner          = Upper('&2')
   AND    a.constraint_type = 'R'
   AND    a.r_constraint_name IN (SELECT a1.constraint_name
                          FROM   all_constraints a1
                          WHERE  a1.table_name =
DECODE(Upper('&1'),'ALL',a.table_name,Upper('&1'))
                          AND    a1.owner      = Upper('&2'));';
END;
```

This PL/SQL script allows the user to specify the schema name and table name to enable foreign keys. It does not require the user to know the name of the all_constraints view. Additionally, the script uses the EXECUTE IMMEDIATE statement to run the ALTER TABLE commands directly.

8. Disable all foreign keys that reference the specified table or all tables.

```
-- Disables all foreign keys that reference the specified table or all tables.
DECLARE
  schema_name VARCHAR2(30);
  table_name VARCHAR2(30);
BEGIN
  -- Prompt the user for the schema name and table name.
  dbms_output.put_line('Enter the schema name:');
  schema_name := dbms_input.get_string(undef);
  dbms_output.put_line('Enter the table name:');
  table_name := dbms_input.get_string(undef);
```

```
-- Disable the foreign keys.
EXECUTE IMMEDIATE
  'SELECT '
  '''ALTER TABLE ''' || schema_name || '.' || table_name || ''' DISABLE CONSTRAINT ''' ||
a.constraint_name || ''';'
  FROM   all_constraints a
  WHERE  a.owner          = Upper('&2')
  AND    a.constraint_type = 'R'
  AND    a.r_constraint_name IN (SELECT a1.constraint_name
                  FROM   all_constraints a1
                  WHERE  a1.table_name =
DECODE(Upper('&1'),'ALL',a.table_name,Upper('&1'))
                  AND    a1.owner      = Upper('&2'));';
END;
```

This script allows the user to specify the schema name and table name. It doesn't require the user to know the name of the all_constraints view. Additionally, the script uses the EXECUTE IMMEDIATE statement to run the ALTER TABLE commands directly.

Generally, disabling constraints can be a useful tool, but it's important to use it carefully. By understanding the implications of disabling constraints and using it only when necessary, you can help to ensure the integrity of your data.