ORACLE®

Academy

Database Programming with SQL

14-3
Managing Constraints

ORACLE
Academy

Objectives

This lesson covers the following objectives:

- List four different functions that the ALTER statement can perform on constraints
- Write ALTER TABLE statements to add, drop, disable, and enable constraints
- Name a business function that would require a DBA to drop, enable, and/or disable a constraint or use the CASCADE syntax
- Query the data dictionary for USER_CONSTRAINTS and interpret the information returned

Purpose

- Would it make any difference if a new student ID number was entered into the school's database when no actual student enrolled?
- Is it likely that a credit-card company would issue the same credit-card number to more than one account or that a business would hire an employee for a department that didn't exist?
- What do you predict would happen if a business could not trust the reliability of the information in its database?

Purpose

- A database system needs to be able to enforce business rules and, at the same time, prevent adding, modifying, or deleting data that might result in a violation of the referential integrity of the database.
- In this section, you will learn how to make changes to table constraints so that referential integrity and, in turn, database reliability are maintained when data needs to be changed.

Managing Constraints

- The ALTER TABLE statement is used to make changes to constraints in existing tables.
- These changes can include adding or dropping constraints, enabling or disabling constraints, and adding a NOT NULL constraint to a column.

Managing Constraints

- The guidelines for making changes to constraints are:
  - You can add, drop, enable, or disable a constraint, but you cannot modify its structure.
  - You can add a NOT NULL constraint to an existing column by using the MODIFY clause of the ALTER TABLE statement.
  - MODIFY is used because NOT NULL is a column-level change.
  - You can define a NOT NULL constraint only if the table is empty or if the column contains a value for every row.

The ALTER Statement

- The ALTER statement requires:
  - name of the table
  - name of the constraint
  - type of constraint
  - name of the column affected by the constraint
- In the code example shown below, using the employees table, the primary-key constraint could have been added after the table was originally created.

```
ALTER TABLE employees
ADD CONSTRAINT  emp_id_pk PRIMARY KEY (employee_id);
```

Adding Constraints

- To add a constraint to an existing table, use the following SQL syntax:

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name] type of constraint (column_name);
```

- If the constraint is a FOREIGN KEY constraint, the REFERENCES keyword must be included in the statement.
- Syntax:

```
ALTER TABLE tablename
ADD CONSTRAINT constraint_name FOREIGN KEY(column_name) REFERENCES
tablename(column_name);
```

Adding Constraints Example

- Consider the employees database.
- The primary key from the DEPARTMENTS table is entered in the EMPLOYEES table as a foreign key.

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

Adding Constraints Example

- The following example demonstrates the syntax to add this foreign key to the EMPLOYEES table:

```
ALTER TABLE employees
ADD CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
   REFERENCES departments (department_id)
ON DELETE CASCADE;
```

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

Adding Constraints Conditions

- If the constraint is a NOT NULL constraint, the ALTER TABLE statement uses MODIFY in place of ADD.
- NOT NULL constraints can be added only if the table is empty or if the column contains a value for every row:

```
ALTER TABLE table_name
MODIFY (column_name CONSTRAINT constraint_name NOT NULL);
```

```
ALTER TABLE employees
MODIFY (email CONSTRAINT emp_email_nn NOT NULL);
```

Why Enable and Disable Constraints?

- To enforce the rules defined by integrity constraints, the constraints should always be enabled.
- In certain situations, however, it is desirable to temporarily disable the integrity constraints of a table for performance reasons, such as:
  - When loading large amounts of data into a table
  - When performing batch operations that make massive changes to a table (such as changing everyone's employee number by adding 1,000 to the existing number)

Dropping Constraints

- To drop a constraint, you need to know the name of the constraint.
- If you do not know it, you can find the constraint name from the USER_CONSTRAINTS and USER_CONS_COLUMNS in the data dictionary.
- The CASCADE option of the DROP clause causes any dependent constraints also to be dropped.
- Note that when you drop an integrity constraint, that constraint is no longer enforced by the Oracle Server and is no longer available in the data dictionary.

Dropping Constraints

- No rows or any data in any of the affected tables are deleted when you drop a constraint.

```
ALTER TABLE table_name
DROP CONSTRAINT  name [CASCADE}
```

```
ALTER TABLE copy_departments
DROP CONSTRAINT  c_dept_dept_id_pk CASCADE;
```

Disabling Constraints

- By default, whenever an integrity constraint is defined in a CREATE or ALTER TABLE statement, the constraint is automatically enabled (enforced) by Oracle unless it is specifically created in a disabled state using the DISABLE clause.

Disabling Constraints

- You can disable a constraint without dropping it or re-creating it by using the ALTER TABLE option DISABLE.
- DISABLE allows incoming data, whether or not it conforms to the constraint.
- This function allows data to be added to a child table without having corresponding values in the parent table.
- DISABLE simply switches off the constraint.

Using the DISABLE Clause

- You can use the DISABLE clause in both the ALTER TABLE statement and the CREATE TABLE statement.

```
CREATE TABLE copy_employees
( employee_id NUMBER(6,0) PRIMARY KEY DISABLE,
   ...
   ...);

ALTER TABLE copy_employees
DISABLE CONSTRAINT c_emp_dept_id_fk;
```

- Disabling a unique or primary-key constraint removes the unique index.

Using the CASCADE Clause

- The CASCADE clause disables dependent integrity constraints. If the constraint is later enabled, the dependent constraints are not automatically enabled.
- Syntax and example:

```
ALTER TABLE table_name
DISABLE CONSTRAINT constraint_name [CASCADE];
```

```
ALTER TABLE copy_departments
DISABLE CONSTRAINT c_dept_dept_id_pk CASCADE;;
```

Enabling Constraints

- To activate an integrity constraint currently disabled, use the ENABLE clause in the ALTER TABLE statement.
- ENABLE ensures that all incoming data conforms to the constraint.
- Syntax and example:

```
ALTER TABLE table_name
ENABLE CONSTRAINT constraint_name;
```

```
ALTER TABLE copy_departments
ENABLE CONSTRAINT c_dept_dept_id_pk;
```

- You can use the ENABLE clause in both the CREATE TABLE statement and the ALTER TABLE statement.

Enabling Constraint Considerations

- If you enable a constraint, that constraint applies to all the data in the table.
- All the data in the table must fit the constraint.
- If you enable a UNIQUE KEY or PRIMARY KEY constraint, a UNIQUE or PRIMARY KEY index is created automatically.
- Enabling a PRIMARY KEY constraint that was disabled with the CASCADE option does not enable any foreign keys that are dependent on the primary key.
- ENABLE switches the constraint back on after you switched it off.

Cascading Constraints

- Cascading referential-integrity constraints allow you to define the actions the database server takes when a user attempts to delete or update a key to which existing foreign keys point.
- The CASCADE CONSTRAINTS clause is used along with the DROP COLUMN clause.
- It drops all referential-integrity constraints that refer to the primary and unique keys defined on the dropped columns.
- It also drops all multicolumn constraints defined on the dropped columns.

Cascading Constraints

- If an ALTER TABLE statement does not include the CASCADE CONSTRAINTS option, any attempt to drop a primary key or multicolumn constraint will fail.
- Remember, you can't delete a parent value if child values exist in other tables.

```
ALTER TABLE table_name
DROP(column name(s)) CASCADE CONSTRAINTS;
```

When CASCADE is Not Required

- If all columns referenced by the constraints defined on the dropped columns are also dropped, then CASCADE CONSTRAINTS is not required.
- For example, assuming that no other referential constraints from other tables refer to column PK, it is valid to submit the following statement without the CASCADE CONSTRAINTS clause:

```
ALTER TABLE tablename DROP
(pk_column_name(s));
```

- However, if any constraint is referenced by columns from other tables or remaining columns in the target table, you must specify CASCADE CONSTRAINTS to avoid an error.

Viewing Constraints

- After creating a table, you can confirm its existence by issuing a DESCRIBE command.
- The only constraint that you can verify using DESCRIBE is the NOT NULL constraint.
- The NOT NULL constraint will also appear in the data dictionary as a CHECK constraint.

Viewing Constraints

- To view all constraints on your table, query the USER_CONSTRAINTS table.

```
SELECT constraint_name, table_name, constraint_type, status
FROM USER_CONSTRAINTS
WHERE table_name ='COPY_EMPLOYEES';
```

CONSTRAINT_NAME	TABLE_NAME	CONSTRAINT_TYPE	STATUS
COPY_EMP_PK	COPY_EMPLOYEES	P	ENABLED
CDEPT_DEPT_ID_FK	COPY_EMPLOYEES	R	ENABLED

Query USER_CONSTRAINTS

- The constraint types listed in the Data Dictionary are:
  - P – PRIMARY KEY; R – REFERENCES (foreign key);
  - C – CHECK constraint (including NOT NULL);
  - U – UNIQUE.

CONSTRAINT_NAME	TABLE_NAME	CONSTRAINT_TYPE	STATUS
COPY_EMP_PK	COPY_EMPLOYEES	P	ENABLED
CDEPT_DEPT_ID_FK	COPY_EMPLOYEES	R	ENABLED

Terminology

Key terms used in this lesson included:

- ALTER TABLE
- CASCADE clause
- CASCADE CONSTRAINT clause
- DISABLE CONSTRAINT
- DROP COLUMN
- DROP CONSTRAINT
- ENABLE CONSTRAINT

Summary

In this lesson, you should have learned how to:

- List four different functions that the ALTER statement can perform on constraints
- Write ALTER TABLE statements to add, drop, disable, and enable constraints
- Name a business function that would require a DBA to drop, enable, and/or disable a constraint or use the CASCADE syntax
- Query the data dictionary for USER_CONSTRAINTS and interpret the information returned

ORACLE®
Academy