
ORACLE
ACADEMY

Database Programming with SQL

14-1
Intro to Constraints; NOT NULL and UNIQUE Constraints

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Define the term "constraint" as it relates to data integrity
- State when it is possible to define a constraint at the column level, and when it is possible at the table level
- State why it is important to give meaningful names to constraints
- State which data integrity rules are enforced by NOT NULL and UNIQUE constraints

Objectives

This lesson covers the following objectives:

- Write a CREATE TABLE statement which includes NOT NULL and UNIQUE constraints at the table and column levels
- Explain how constraints are created at the time of table creation

Purpose

- If you think about it, how would society function without rules?
- It is a rule to stop at a red traffic light?
- Would it be safe to drive without this rule?
- For databases, it is a rule that a foreign-key value cannot be entered without first entering a primary-key value.
- What do you think would happen to a database if this rule wasn't enforced?
- A database is only as reliable as the data that is in it.
- Constraints are used to prevent invalid data entry into tables.

Purpose

- Would it make sense to have negative salary values or six students with the same student ID or two tables that no longer reference each other?
- Without rules, how could you trust the integrity of the database?
- In the next three lessons, you will study how to create the constraints that enforce the "rules."
- You will also learn how to manage them and view constraint definitions in the data dictionary.

Constraints in General

- So, what exactly is a constraint?
- Think of constraints as database rules.
- All constraint definitions are stored in the data dictionary.
- Constraints prevent the deletion of a table if there are dependencies from other tables.
- Constraints enforce rules on the data whenever a row is inserted, updated, or deleted from a table.

Constraints in General

- Constraints are important and naming them is also important.
- Although you could name a constraint "constraint_1" or "constraint_2," you'd soon find it difficult to distinguish one constraint from another and would end up redoing a lot of work.

Creating Constraints

- Recall the SQL syntax for creating a table.
- In the CREATE TABLE statement shown on the next slide, each column and its data type is defined.
- You use the CREATE TABLE statement to establish constraints for each column in the table.
- There are two different places in the CREATE TABLE statement that you can specify the constraint details:
  - At the column level next to the name and data type
  - At the table level after all the column names are listed

Creating Constraints

```
CREATE TABLE clients
(client_number NUMBER(4),
 first_name         VARCHAR2(14),
 last_name         VARCHAR2(13));
```

- Note that the column-level simply refers to the area in the CREATE TABLE statement where the columns are defined.
- The table level refers to the last line in the statement below the list of individual column names.

Constraints at the Column Level

- A column-level constraint references a single column.
- To establish a column-level constraint, the constraint must be defined in the CREATE TABLE statement as part of the column definition.
- Examine the following SQL statement that establishes a column-level constraint.

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
 first_name        VARCHAR2(14),
 last_name        VARCHAR2(13));
```

Constraints at the Column Level

- CREATE TABLE clients

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
 first_name        VARCHAR2(14),
 last_name        VARCHAR2(13));
```

- The name of the constraint is clients_client_num_pk.
- It enforces the business rule that the client_number is the primary key of the clients table.

Naming Constraints

- Every constraint in the database has a name. When a constraint is created, it can be given a name, such as clients_client_num_pk, or given no name, in which case the system gives the constraint a name, such as SYS_C00585417.
- A naming convention can be the combination of the tablename abbreviated and a column name abbreviated followed by the constraint abbreviation: table-name_column-name_constraint-type
- If the reserved word CONSTRAINT is used in the CREATE TABLE definition, you must give the constraint a name. Constraint names are limited to 30 characters.

Naming Constraints at the Column Level

- It is best to name constraints yourself because system-generated names are not easy to recognize.
- Look at this table definition:

```
CREATE TABLE clients
(client_number   NUMBER(4),
 last_name       VARCHAR2(13),
 email           VARCHAR2(80));
```

Naming Constraints at the Column Level

- According to our naming convention:
  - A primary key constraint on client_number would be named clients_client_number_pk
  - A not null constraint on last_name would be named clients_last_name_nn
  - A unique constraint on email address would be named clients_email_uk

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_cient_num_pk PRIMARY KEY,
 last_name     VARCHAR2(13) CONSTRAINT clients_last_name_nn NOT NULL,
 email         VARCHAR2(80) CONSTRAINT clients_emil_uk UNIQUE);
```

Constraint Naming Example

- This example shows both a user-named constraint and a system-named constraint:

```
CREATE TABLE clients
 (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
  last_name     VARCHAR2(13) NOT NULL,
  email         VARCHAR2(80));
```

Constraint Naming Example

- System-named constraint:

```
CREATE TABLE clients
 (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
  last_name     VARCHAR2(13) NOT NULL,
  email         VARCHAR2(80));
```

- Two constraints have been created:
  - A user-named constraint named clients_client_num_pk, to enforce the rule that client_number is the primary key
  - A system-named constraint named SYS_Cn (where n is a unique integer) to enforce the rule that last_names cannot be null.

Constraints at the Table Level

- Table-level constraints are listed separately from the column definitions in the CREATE TABLE statement.
- Table-level constraint definitions are listed after all the table columns have been defined.
- In the example shown, the unique constraint is listed last in the CREATE TABLE statement.

```
CREATE TABLE clients (
 client_number NUMBER(6) NOT NULL,
 first_name    VARCHAR2(20),
 last_name     VARCHAR2(20),
 phone         VARCHAR2(20),
 email         VARCHAR2(10) NOT NULL,
CONSTRAINT clients_phone_email_uk UNIQUE (email,phone));
```

Basic Rules for Constraints

- Constraints that refer to more than one column (a composite key) must be defined at the table level
- The NOT NULL constraint can be specified only at the column level, not the table level
- UNIQUE, PRIMARY KEY, FOREIGN KEY, and CHECK constraints can be defined at either the column or table level
- If the word CONSTRAINT is used in a CREATE TABLE statement, you must give the constraint a name

Examine the Violations

```
CREATE TABLE clients(
        client_number  NUMBER(6),
        first_name     VARCHAR2(20),
        last_name      VARCHAR2(20),
        phone          VARCHAR2(20) CONSTRAINT phone_email_uk
                              UNIQUE(email,phone),
         email         VARCHAR2(10) CONSTRAINT NOT NULL,
     CONSTRAINT emailclients_email NOT NULL,
     CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));
```

Examine the Violations

COMPOSITE UNIQUE KEY VIOLATION
Composite keys must be defined at the table level.

```
CREATE TABLE clients(
        client_number   NUMBER(6),
        first_name      VARCHAR2(20),
        last_name       VARCHAR2(20),
        phone           VARCHAR2(20) CONSTRAINT phone_email_uk
                                UNIQUE(email,phone),
         email          VARCHAR2(10) CONSTRAINT NOT NULL,
     CONSTRAINT emailclients_email NOT NULL,
     CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));
```

NOT NULL VIOLATION
NOT NULL constraints can only be defined at the column level.

NAME VIOLATION
When using the term CONSTRAINT, it must be followed by a constraint name.

Five Types of Constraints

- Five types of constraints exist within an Oracle database.
- Each type enforces a different rule.
- The types are:
  - NOT NULL constraints
  - UNIQUE constraints
  - PRIMARY KEY constraints
  - FOREIGN KEY constraints
  - CHECK constraints

Five Types of Constraints

- In the remainder of this lesson, you will learn about NOT NULL and UNIQUE constraints.
- The next lesson will teach you about the other three types of constraints.

NOT NULL Constraint

- A column defined with a NOT NULL constraint requires that for every row entered into the table, a value must exist for that column.
- For example, if the email column in an employees table was defined as NOT NULL, every employee entered into the table MUST have a value in the email column.
- When defining NOT NULL columns, it is customary to use the suffix _nn in the constraint name.
- For example, the constraint name for the NOT NULL email column in the employees table could be emp_email_nn.

UNIQUE Constraint

- A UNIQUE constraint requires that every value in a column or set of columns (a composite key) be unique; that is, no two rows of a table can have duplicate values.
- For example, it may be important for a business to ensure that no two people have the same email address.
- The email column could be defined using a UNIQUE constraint.
- The column or set of columns that is defined as UNIQUE is called a unique key.

UNIQUE Constraint

- If the combination of two or more columns must be unique for every entry, the constraint is said to be a composite unique key.
- Stating that all combinations of email and last name must be UNIQUE is an example of a composite unique key.
- The word "key" refers to the columns, not the constraint names.

Unique Constraint Example

- If the email column in the table is defined with a UNIQUE constraint, no other client entry can have an identical email.
- What if two clients live in the same household and share an email address?

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net

```
INSERT INTO clients (client_number, first_name, Last_name, phone,
email)
   VALUES (7234, 'Lonny', 'Vigil', 4072220091, 'lbv@lbv.net');
```

```
ORA-00001: unique constraint
(USWA_SKHS_SQL01_T01.CLIENT_EMAIL_UK) violated
```

Defining UNIQUE Constraints

- When defining UNIQUE constraints, it is customary to use the suffix _uk in the constraint name.
- For example, the constraint name for the UNIQUE email column in the employees table could be emp_email_uk.
- To define a composite unique key, you must define the constraint at the table level rather than the column level.
- An example of a composite unique-key constraint name is:

```
CONSTRAINT clients_phone_email_uk UNIQUE(email,phone)
```

Composite Unique Key

- UNIQUE constraints allow the input of nulls unless the column also has a NOT NULL constraint defined.
- A null in a column (or in all columns of a composite unique key) always satisfies a UNIQUE constraint because nulls are not considered equal to anything.

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net
7234	Lonny	Vigil	4072220091	lbv@lbv.net

This combination of columns must be **UNIQUE**

Composite Unique Key

- To satisfy a constraint that designates a composite unique key, no two rows in the table can have the same combination of values in the key columns.
- Also, any row that contains nulls in all key columns automatically satisfies the constraint.

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net
7234	Lonny	Vigil	4072220091	lbv@lbv.net

This combination of columns must be **UNIQUE**

Constraints Created at Table Creation

- When you add a NOT NULL constraint as part of a table creation statement, the Oracle database will create a Check Constraint in the database to enforce a value in the NOT NULL column.
- This constraint creation can be almost invisible to you when you create your table—Oracle just does it.
- At the end of your table creation statement, the message "Table created" displays, but no details are provided about the number or types of constraints that were also created.

Terminology

Key terms used in this lesson included:

- CHECK constraint
- Constraint
- Column level constraint
- FOREIGN KEY
- NOT NULL constraints
- PRIMARY KEY

Terminology

Key terms used in this lesson included:

- Table level constraint
- UNIQUE constraints
- UNIQUE KEY

Summary

In this lesson, you should have learned how to:

- Define the term "constraint" as it relates to data integrity
- State when it is possible to define a constraint at the column level, and when it is possible at the table level
- State why it is important to give meaningful names to constraints
- State which data integrity rules are enforced by NOT NULL and UNIQUE constraints

Summary

In this lesson, you should have learned how to:

- Write a CREATE TABLE statement which includes NOT NULL and UNIQUE constraints at the table and column levels
- Explain how constraints are created at the time of table creation


ORACLE
ACADEMY