Database Constraints

Overview

This lesson demonstrates how to define different types of integrity constraints for database tables ~~in order~~ to maintain the accuracy of ~~the~~ stored data. Although most constraints can be created at either the column or table level, all constraints are enforced at the table level. If a constraint is no longer needed in a database, it can be permanently removed using the DROP command.

Objectives

The learning objectives of this lesson are to:

- Distinguish among PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, and NOT NULL constraints and understand the correct use of each constraint.
- Understand how to create constraints when creating a table or modifying an existing table.
- Distinguish between creating constraints at the column level and the table level.
- Create PRIMARY KEY constraints for a single column and a composite primary key.
- Create a FOREIGN KEY constraint.
- Create a UNIQUE constraint.
- Create a CHECK constraint.
- Create a NOT NULL constraint with the ALTER TABLE ... MODIFY command.
- Include constraints during table creation.
- Add multiple constraints on a single column.
- Use the DROP command with constraints.

Additional Files:

- *create_tables_pet_database.sql*
- *insert_data_pet_database.sql*

1. SQL Constraints

Integrity constraints are used to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in referential integrity:

- entity integrity constraints (primary keys)
- referential integrity constraints (foreign keys)
- domain constraints (conditions that must be satisfied by individual fields)
- table constraints (conditions that must be satisfied involving several fields in the same table)

You can add constraints during table creation as part of the CREATE TABLE command, or you can do so after the table is created by using the ALTER TABLE command.

Constraint Types:

- PRIMARY KEY
- FOREIGN KEY
- NOT NULL
- UNIQUE
- CHECK
- DEFAULT

Constraints can be created at the column level by including the constraint as part of the column definition or at the table level by defining the constraint after all columns have been defined. When created at the column level, the column that is the objective of the constraint is identified first, and then the constraint information is provided. If created using the level table approach, the type of constraint is identified first, and then the column is identified.

2. Including Constraints during Table Creation

When you create a constraint at the table level, the constraint definition is separate from any column definitions.

```
CONSTRAINT constraintname] constrainttype
(columnname, ...),
```

If you create the constraint at the same time you are creating a table, list the constraint after all the columns are defined.

In fact, the main difference in the syntax of a column-level constraint and a table-level constraint is that you provide column names for the table-level constraint at the end of the constraint definition inside parentheses, instead of at the beginning of the constraint definition.

A column can be assigned as many constraints as necessary. Each constraint is processed separately.

3. Creating Constraints

The general syntax for creating a constraint at the column level is

columnname [CONSTRAINT constraintname] constrainttype,

3.1. Using the PRIMARY KEY Constraint

The PRIMARY KEY constraint designates the primary key for a table. Each table can have only one PRIMARY KEY constraint. If a primary key consists of more than one column, it is considered a **composite primary key,** and it must be created using the level table approach.

A PRIMARY KEY enforces two rules:

1. NOT NULL
2. UNIQUE.

This method of defining a primary key is accomplished during table creation. The primary key, in this case, is an implied constraint.

PRIMARY KEY Constraints

- A PRIMARY KEY constraint is a rule that the values in one column or a combination of columns must uniquely identify each row in a table
- No primary-key value can appear in more than one row in the table
- To satisfy a PRIMARY KEY constraint, both of the following conditions must be true:
    - No column that is part of the primary key can contain a null
    - A table can have only one primary key
- PRIMARY KEY constraints can be defined at the column or the table level
- However, if a composite PRIMARY KEY is created, it must be defined at the table level

- When defining PRIMARY KEY columns, it is a good practice to use the suffix _pk in the constraint name
- For example, the constraint name for the PRIMARY KEY column named client_number in table named CLIENTS could be clients_client_num_pk

Example 1: Consider the following example, which demonstrates by setting the numeric dnumber field as the primary key for the DEPARTMENT table.

```
CREATE TABLE department (
      dname VARCHAR(25) NOT NULL,
      dnumber INT(4),
      mgrssn CHAR(9) NOT NULL,
      mgrstartdate DATETIME,
      PRIMARY KEY (dnumber),
      UNIQUE (dname),
      FOREIGN KEY (mgrssn) REFERENCES employee(ssn)
)ENGINE = INNODB ;
```

The primary key is defined after the null specification. The null specification may also be specified behind the primary key.

Example 2: In this example, we can also define the primary key as a table integrity constraint.

```
CREATE TABLE department(
      dname VARCHAR(25) NOT NULL,
      dnumber INT(4) NOT NULL,
      mgrssn CHAR(9) NOT NULL,
      mgrstartdate DATETIME,
      CONSTRAINT  DEPARTMENT_PK PRIMARY KEY (dnumber),
      UNIQUE (dname),
      FOREIGN KEY (mgrssn) REFERENCES EMPLOYEE (ssn));
```

PRIMARY KEY constraints can be specified for either a single field or for a **composite of multiple fields**.

Example 3: Consider the following example, which demonstrates by constructing a table containing a composite primary key.

```
CREATE TABLE dependent (
      essn CHAR(9),
      dependent_name VARCHAR(15),
      sex CHAR,
      bdate DATE,
      relationship VARCHAR(8),
      PRIMARY KEY (essn , dependent_name),
      FOREIGN KEY (essn)
      REFERENCES employee (ssn)
)  ENGINE=INNODB;
```

The PRIMARY KEY Constraint on ALTER TABLE

The ALTER TABLE command with the ADD clause can be used to add a PRIMARY KEY constraint to an existing table. When a PRIMARY KEY constraint has been added to a column, the DESCRIBE command will display that the column cannot contain NULL values. This is because a PRIMARY KEY constraint will not allow duplicate values or NULL values in that column.

```
ALTER TABLE tablename
ADD [CONSTRAINT constraintname] PRIMARY KEY (columnname);
```

DROP a PRIMARY KEY Constraint

To drop a PRIMARY KEY constraint, use the following SQL:

```
ALTER TABLE dependent
DROP PRIMARY KEY;
```

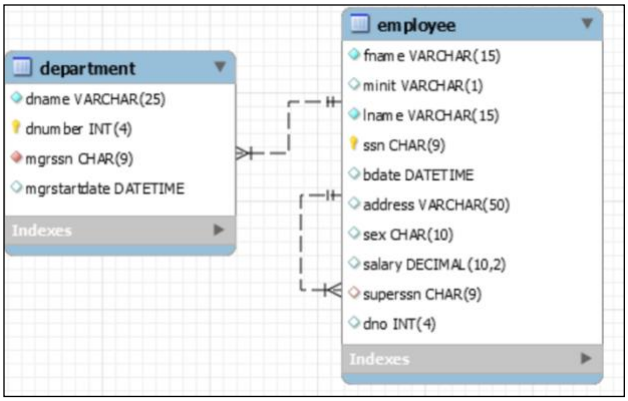
Follow these rules when you define primary keys:

- Only one primary key can be defined for each table.
- The relational model requires that one primary key be defined for each table. However, MySQL does not enforce this; you can create tables without a primary key.
- Two different rows in a table may never have the same value for the primary key.
- A column name may occur only once in the column list of a primary key.
- The populations of the columns belonging to a primary key may not contain null values. This rule is known as either the first integrity constraint or the entity integrity constraint.

3.2. Using the FOREIGN KEY Constraint

A FOREIGN KEY in one table points to a PRIMARY KEY in another table. A FOREIGN KEY constraint is used to ensure referential integrity. In addition, the FOREIGN KEY constraint can only reference a column that has already been designated as the primary key for the parent table.

- FOREIGN KEY constraints are also called "referential integrity" constraints
- Foreign Key constraints designate a column or combination of columns as a foreign key
- A foreign key links back to the primary key (or a unique key) in another table, and this link is the basis of the relationship between tables
- To satisfy a referential-integrity constraint, a foreign-key value must match an existing value in the parent table or be NULL
- A primary-key value can exist without a corresponding foreign-key value; however, a foreign-key must have a corresponding primary key
- The rule is: before you define a referential-integrity constraint in the child table, the referenced UNIQUE or PRIMARY KEY constraint on the parent table must already be defined
- In other words, you must first have a parent primary key defined before you can create a foreign key in a child table



The mgrssn column in **Example 2** has been designated as the foreign key for the DEPARTMENT table. This foreign key, as you can see, references the SSN column in the EMPLOYEE table. This foreign key ensures that for every mgrssn in the DEPARTMENT table, there is a corresponding SSN in the EMPLOYEE table. This is called a **relationship.**

FOREIGN KEY constraints are also referred to as **referential integrity constraints** and assist in maintaining database integrity. Referential integrity requires that values of a foreign key must correspond to values of a primary key in the table that it references. For example, you cannot enter a row in the project table with a department number unless that department number already exists within one of the department table rows. But what happens if the primary key values change or the row that is referenced is deleted?

In MySQL, the InnoDB storage engine provides foreign key support. The **parent (EMPLOYEE)** is the table that contains the original key values. The **child** (**DEPARTMENT)** is the related table that refers to key values in the parent. Parent table key values are used to associate the two tables.

The syntax to define a foreign key in a child table:

[CONSTRAINT constraintname] FOREIGN KEY [fk_name] REFERENCES tablename

[ON DELETE action] [ON UPDATE action]

action: RESTRICT | CASCADE | SET NULL | NO ACTION

- The CONSTRAINT clause, if given, supplies a name for the foreign key constraint. If you omit it, InnoDB creates a name.
- FOREIGN KEY indicates the columns in the child table that must match values in the parent table. fk_name is the name of the foreign key.
- REFERENCES names the parent table and the columns in that table to which the foreign key in the child table refers.
- ON DELETE enables you to specify what happens to the child table when parent table rows are deleted. The default, if no ON DELETE clause, is to reject any attempt to delete rows in the parent table that have child rows pointing to them.
- ON UPDATE enables you to specify what happens to the child table when parent table rows are updated. The default, if no ON UPDATE clause, is to reject any inserts or updates in the child table that result in foreign key values that do not have any match in the parent table, and to prevent updates to parent table values to which child rows point.

To specify an action value explicitly, use one of the following clauses:

Keyword	What It Means
CASCADE	Delete all records containing references to the deleted key value.
SET NULL	Modify all records containing references to the deleted key value to use a NULL value instead (this can only be used for fields previously marked as NOT NULL).
RESTRICT	Reject the deletion request until all subordinate records using the deleted key value have themselves been manually deleted and no references exist (this is the default setting, and it's also the safest).
NO ACTION	Do nothing.

Table 1 Actions Available in ON DELETE and ON UPDATE Clause

The FOREIGN KEY Constraint on ALTER TABLE

The ALTER TABLE command with the ADD clause can be used to add a FOREIGN KEY constraint to an existing table.

The syntax to add a FOREIGN KEY constraint to a table is

```
ALTER TABLE tablename
ADD [CONSTRAINT constraintname] FOREIGN KEY (columnname)
REFERENCES tablename (columnname);
```

DROP a FOREIGN KEY Constraint

To drop a FOREIGN KEY constraint, use the following SQL:

```
ALTER TABLE department
DROP FOREIGN KEY department_ibfk_1;
```

To drop a foreign key, you must know its name. If you do not know its name, you can use MySQL Workbench.

3.3. Using the UNIQUE Constraint

The UNIQUE constraint is used to ensure that no duplicate values exist in the specified column(s). Unlike the PRIMARY KEY constraint, the UNIQUE constraint will allow NULL values to be entered into the column.

A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it. You can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

- 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
- 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

Example 4: The following SQL creates a UNIQUE constraint on the pname column when the PROJECT table is created:

```
CREATE TABLE project (
  Pname VARCHAR(25) NOT NULL,
  Pnumber INT(4),
  Plocation VARCHAR(15),
  dnum INT(4) NOT NULL,
  PRIMARY KEY (pnumber),
  UNIQUE (pname),
  FOREIGN KEY (dnum) REFERENCES department(dnumber)
)ENGINE = INNODB;
```

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
- 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

The UNIQUE Constraint on ALTER TABLE

The ALTER TABLE command with the ADD clause can be used to add a FOREIGN KEY constraint to an existing table.

The syntax to add a UNIQUE constraint to an existing table.

ALTER TABLE tablename

ADD [CONSTRAINT constraintname] UNIQUE (columnname);

3.4. Using the NOT NULL Constraint

The NOT NULL constraint enforces a column not to accept NULL values. The NOT NULL constraint is a special CHECK constraint with the condition IS NOT NULL. A NOT NULL constraint can only be created using the column level approach.

- 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

Example 5: Create the employee table.

```
CREATE TABLE employee (
      fname VARCHAR(15) NOT NULL,
      minit VARCHAR(1),
      lname VARCHAR(15) NOT NULL,
      ssn CHAR(9),
      bdate DATETIME,
      address VARCHAR(50),
      sex CHAR(10),
      salary DECIMAL(10,2),
      superssn CHAR(9),
      dno INT(4),
      PRIMARY KEY (ssn),
      FOREIGN KEY (superssn) REFERENCES employee(ssn)
      )ENGINE = INNODB;
```

The NOT NULL Constraint on ALTER TABLE

The syntax to add a NOT NULL constraint to an existing table.

```
ALTER TABLE tablename
MODIFY (columnname [CONSTRAINT constraintname] NOT NULL);
```

Use the ALTER TABLE command to add a NOT NULL constraint to the EMPLOYEE table.

```
ALTER TABLE employee
MODIFY COLUMN salary DECIMAL(10,2) NOT NULL;
```

3.5. Using the CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column. A CHECK constraint is used to validate the data being entered into the database.

- The CHECK constraint explicitly defines a condition that must be met
- To satisfy the constraint, each row in the table must make the condition either True or unknown (due to a null)
- The condition of a CHECK constraint can refer to any column in the specified table, but not to columns of other tables

Example 6: Create a special version of the DEPENDENT table and take into account that the gender column may contain only the values M or F.

```
CREATE TABLE dependent_x (
      essn CHAR(9),
      dependent_name VARCHAR(15),
      gender CHAR,
      bdate DATE,
      relationship VARCHAR(8),
      PRIMARY KEY (essn , dependent_name),
      FOREIGN KEY (essn) REFERENCES employee (ssn),
      CHECK (sex IN ('M', 'F')))ENGINE=INNODB;
```

The syntax for adding a CHECK constraint to an existing table

```
ALTER TABLE tablename
ADD [CONSTRAINT constraintname] CHECK (condition);
```

3.6. Using the DEFAULT Constraint

The DEFAULT constraint is used to insert a default value into a column. The default value will be added to all new records if no other value is specified.

Example 7: Creates a special version of the DEPENDENT table with a default value 'M' that will be placed in a gender column.

```
CREATE TABLE dependent_d (
      essn CHAR(9),
      dependent_name VARCHAR(15),
      sex CHAR DEFAULT 'M',
      bdate DATE,
      relationship VARCHAR(8),
      PRIMARY KEY (essn , dependent_name),
      FOREIGN KEY (essn) REFERENCES employee (ssn)
      )ENGINE=INNODB;
```

Key Terms

1. **Check constraint** – a constraint that forces data values stored in a specific column to fall within some acceptable range of values.
2. **Data integrity** – means that the data stored in a column is valid.
3. **Referential integrity** - the enforcement of FOREIGN KEY constraints to ensure that values of a foreign key correspond to values of a primary key in the table that is referenced.
4. **surrogate key** – a unique numeric identifier that is appended to a relation to serve as the primary key
5. **Unique constraint** – a constraint used to enforce uniqueness for a column value that is not a primary key column.