Lab 2.1. SQL Constraints

In The Relational Database Model, you learned that adherence to rules for entity integrity and referential integrity is crucial in a relational database environment. Fortunately, most SQL implementations support both integrity rules. Entity integrity is enforced automatically when the primary key is specified in the CREATE TABLE command sequence. For example, you can create the VENDOR table structure and set the stage for the enforcement of entity integrity rules by using the following:

PRIMARY KEY (V_CODE)

In the PRODUCT table's CREATE TABLE sequence, note that referential integrity has been enforced by specifying the following in the PRODUCT table:

FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE

The foreign key constraint definition ensures that:

• You cannot delete a vendor from the VENDOR table if at least one product row references that vendor. This is the default behavior for the treatment of foreign keys.

• On the other hand, if a change is made in an existing VENDOR table's V_CODE, that change must be reflected automatically in any PRODUCT table V_CODE reference (ON UPDATE CASCADE). That restriction makes it impossible for a V_CODE value to exist in the PRODUCT table if it points to a nonexistent VENDOR table V_CODE value. In other words, the ON UPDATE CASCADE specification ensures the preservation of referential integrity. (Oracle is a strict adherent to the philosophy that the value of a primary key should never change, so it does not support ON UPDATE CASCADE.)

In general, ANSI SQL permits the use of ON DELETE and ON UPDATE clauses to cover CASCADE, SET NULL, or SET DEFAULT.

Besides the PRIMARY KEY and FOREIGN KEY constraints, the ANSI SQL standard also defines the following constraints:

• The NOT NULL constraint ensures that a column does not accept nulls.

• The UNIQUE constraint ensures that all values in a column are unique.

• The DEFAULT constraint assigns a value to an attribute when a new row is added to a table. The end user may, of course, enter a value other than the default value.

• The CHECK constraint is used to validate data when an attribute value is entered. The CHECK constraint does precisely what its name suggests: it checks to see that a specified condition exists. Examples of such constraints include the following:

— The minimum order value must be at least 10.

— The date must be after April 15, 2018.

— If the CHECK constraint is met for the specified attribute (i.e., the condition is true), the data is accepted for that attribute. If the condition is found to be false, an error message is generated and the data is not accepted.

Note that the CREATE TABLE command lets you define constraints in two different places:

• When you create the column definition (known as a column constraint)

• When you use the CONSTRAINT keyword (known as a table constraint)

A column constraint applies to just one column; a table constraint may apply to many columns. Those constraints are supported at varying levels of compliance by enterprise RDBMSs.

Oracle is used to illustrate SQL constraints. For example, note that the following SQL command sequence uses the DEFAULT and CHECK constraints to define the table named CUSTOMER.

```
CREATE TABLE CUSTOMER (
CUS_CODE        NUMBER        PRIMARY KEY,
CUS_LNAME       VARCHAR(15)   NOT NULL,
CUS_FNAME       VARCHAR(15)   NOT NULL,
CUS_INITIAL     CHAR(1),
CUS_AREACODE    CHAR(3)       DEFAULT '615'     NOT NULL
                              CHECK(CUS_AREACODE IN ('615','713','931')),
CUS_PHONE       CHAR(8)       NOT NULL,
CUS_BALANCE     NUMBER(9,2)   DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));
```

In this case, the CUS_AREACODE attribute is assigned a default value of ‘615’. Therefore, if a new CUSTOMER table row is added and the end user makes no entry for the area code, the ‘615’ value will be recorded. Also, the CHECK condition restricts the values for the customer’s area code to 615, 713, and 931; any other values will be rejected.

It is important to note that the DEFAULT value applies only when new rows are added to a table and then only when no value is entered for the customer’s area code. (The default value is not used when an existing row in the table is modified.) In contrast, the CHECK condition is validated whether a customer row is added or modified. However, while the CHECK condition may include any valid expression, it applies only to the attributes in the table being checked. If you want to check for conditions that include attributes in other tables, you must use triggers, as discussed later. Finally, the last line of the CREATE TABLE command sequence creates a unique index constraint (named CUS_UI1) on the customer’s last name and first name. The index will prevent the entry of two customers with the same last name and first name. (This index merely illustrates the process. Clearly, it should be possible to have more than one person named John Smith in the CUSTOMER table.)

In the following SQL command to create the INVOICE table, the DEFAULT constraint assigns a default date to a new invoice, and the CHECK constraint validates that the invoice date is greater than January 1, 2018.

```
CREATE TABLE INVOICE (
INV_NUMBER   NUMBER   PRIMARY KEY,
CUS_CODE     NUMBER   NOT NULL REFERENCES CUSTOMER(CUS_CODE),
INV_DATE     DATE     DEFAULT SYSDATE NOT NULL,
CONSTRAINT INV_CK1 CHECK (INV_DATE > '01-JAN-2018'));
```

In this case, notice the following:

• The CUS_CODE attribute definition contains REFERENCES CUSTOMER (CUS_CODE) to indicate that the CUS_CODE is a foreign key. This is another way to define a foreign key.

The DEFAULT constraint uses the SYSDATE special function. This function always returns today's date.

• The invoice date (INV_DATE) attribute is automatically given today's date (returned by SYSDATE) when a new row is added if no value is given for the attribute.

• A CHECK constraint is used to validate that the invoice date is greater than "January 1, 2018."

The final SQL command sequence creates the LINE table. The LINE table has a composite primary key (INV_NUMBER, LINE_NUMBER) and uses a UNIQUE constraint in INV_NUMBER and P_CODE to ensure that the same product is not ordered twice in the same invoice.

```
CREATE TABLE LINE (
INV_NUMBER      NUMBER          NOT NULL,
LINE_NUMBER     NUMBER(2,0)     NOT NULL,
P_CODE          VARCHAR(10)     NOT NULL,
LINE_UNITS      NUMBER(9,2)     DEFAULT 0.00    NOT NULL,
LINE_PRICE      NUMBER(9,2)     DEFAULT 0.00    NOT NULL,
PRIMARY KEY (INV_NUMBER, LINE_NUMBER),
FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE ON DELETE CASCADE,
FOREIGN KEY (P_CODE) REFERENCES PRODUCT(P_CODE),
CONSTRAINT LINE_UI1 UNIQUE(INV_NUMBER, P_CODE));
```

In the creation of the LINE table, note that a UNIQUE constraint is added to prevent the duplication of an invoice line. A UNIQUE constraint is enforced through the creation of a unique index. Also note that the ON DELETE CASCADE foreign key enforces referential integrity. The use of ON DELETE CASCADE is recommended for weak entities to ensure that the deletion of a row in the strong entity automatically triggers the deletion of the corresponding rows in the dependent weak entity. In that case, the deletion of an INVOICE row will automatically delete all of the LINE rows related to the invoice. In the following section, you will learn more about indexes and how to use SQL commands to create them.

Lab 2.2. Altering Table Structures

Sunday, October 2, 2022, 10:42 PM

Number of replies: 0

In this section, you will learn how to change table structures by changing attribute characteristics and by adding columns.

All changes in the table structure are made by using the ALTER TABLE command followed by a keyword that produces the specific change you want to make. Three options are available: ADD, MODIFY, and DROP. You use ADD to add a column, MODIFY to change column characteristics, and DROP to delete a column from a table. Most RDBMSs do not allow you to delete a column unless the column does not contain any values; otherwise, such an action might delete crucial data used by other tables.

The basic syntax to add or modify columns is:

ALTER TABLE tablename

{ADD | MODIFY} (columnname datatype [{ADD | MODIFY} columnname datatype]);

The ALTER TABLE command can also be used to add table constraints. In those cases, the syntax would be:

ALTER TABLE tablename

ADD constraint [ADD constraint];

You could also use the ALTER TABLE command to remove a column or table constraint. The syntax would be as follows:

ALTER TABLE tablename

DROP {PRIMARY KEY | COLUMN columnname | CONSTRAINT constraintname };

Notice that when removing a constraint, you need to specify it by name, which is onereason you should always name constraints in your CREATE TABLE or ALTER TABLEstatement.

Changing a Column's Data Type

Using the ALTER syntax, the integer V_CODE in the PRODUCT table can be changed to a character V_CODE by using the following command:

ALTER TABLE PRODUCT

MODIFY (V_CODE CHAR(5));

Changing a Column's Data Characteristics

If the column to be changed already contains data, you can make changes in the column's characteristics if those changes do not alter the data type. For example, if you want to increase the width of the P_PRICE column to nine digits, use the following command:

ALTER TABLE PRODUCT

MODIFY (P_PRICE DECIMAL(9,2));

Adding a Column

You can alter an existing table by adding one or more columns. In the following example, you add the column named P_SALECODE to the PRODUCT table. (This column will be used later to determine whether goods that have been in inventory for a certain length of time should be placed on special sale.)

Suppose that you expect the P_SALECODE entries to be 1, 2, or 3. Because no arithmetic will be performed with the P_SALECODE, the P_SALECODE will be classified as a single-character attribute. Note the inclusion of all required information in the following ALTER command:

```
ALTER TABLE PRODUCT
    ADD (P_SALECODE CHAR(1));
```

Adding Primary Key, Foreign Key, and Check Constraints

When you create a new table based on another table, the new table does not include integrity rules from the old table. In particular, there is no primary key. To define the primary key for the new PART table, use the following command:

```
ALTER TABLE PART
ADD         PRIMARY KEY (PART_CODE);
```

Several other scenarios could leave you without entity and referential integrity. For example, you might have forgotten to define the primary and foreign keys when you created the original tables. Or, if you imported tables from a different database, you might have discovered that the importing procedure did not transfer the integrity rules. In any case, you can re-establish the integrity rules by using the ALTER command. For example, if the PART table's foreign key has not yet been designated, it can be designated by:

```
ALTER TABLE   PART
ADD           FOREIGN KEY (V_CODE) REFERENCES VENDOR;
```

Just as primary key and foreign key constraints can be added after the table structure is created, check constraints can also be applied to the table. For example, if there should be a constraint that the part price cannot be a negative value (i.e., it should be greater than or equal to 0), then a check constraint can be designated by:

```
ALTER TABLE   PART
ADD           CHECK (PART_PRICE >= 0);
```

Alternatively, if the PART table's primary key, foreign key, or check constraint has not been designated, you can incorporate all three changes at once:

```
ALTER TABLE   PART
ADD           PRIMARY KEY (PART_CODE)
ADD           FOREIGN KEY (V_CODE) REFERENCES VENDOR
ADD           CHECK (PART_PRICE >= 0);
```

Even composite primary keys and multiple foreign keys can be designated in a single SQL command. For example, if you want to enforce the integrity rules for the LINE table shown in Figure 8.1, you can use:

```
ALTER TABLE   LINE
ADD           PRIMARY KEY (INV_NUMBER, LINE_NUMBER)
ADD           FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE
ADD           FOREIGN KEY (P_CODE) REFERENCES PRODUCT;
```

Dropping a Column

Occasionally, you might want to modify a table by deleting a column. Suppose that you want to delete the V_ORDER attribute from the VENDOR table. You would use the following command:

```
ALTER TABLE VENDOR
    DROP COLUMN V_ORDER;
```

Deleting a Table from the Database

A table can be deleted from the database using the DROP TABLE command. For example, you can delete the PART table you just created with the following command:

```
DROP TABLE PART;
```

Lab 2.3. Updating Table Rows

Sunday, October 2, 2022, 10:42 PM

Number of replies: 0

Use the UPDATE command to modify data in a table. The syntax for this command is as follows:

```
UPDATE     tablename
SET        columnname = expression [, columnname = expression]
[WHERE     conditionlist ];
```

For example, if you want to change P_INDATE from December 13, 2017, to January 18, 2018, in the second row of the PRODUCT table (see Figure 8.2), use the primary key (13-Q2/P2) to locate the correct row. Therefore, type:

```
UPDATE     PRODUCT
SET        P_INDATE = '18-JAN-2018'
WHERE      P_CODE = '13-Q2/P2';
```

If more than one attribute is to be updated in the row, separate the corrections with commas:

```
UPDATE     PRODUCT
SET        P_INDATE = '18-JAN-2018', P_PRICE = 17.99, P_MIN = 10
WHERE      P_CODE = '13-Q2/P2';
```

Lab 2.4. Deleting Table Rows

Sunday, October 2, 2022, 10:42 PM

Number of replies: 0

It is easy to delete a table row using the DELETE statement. The syntax is:

```
DELETE FROM    tablename
[WHERE         conditionlist ];
```

For example, if you want to delete the product you added earlier whose code (P_CODE) is BRT-345, use the following command:

```
DELETE FROM   PRODUCT
WHERE         P_CODE = 'BRT-345';
```

In this example, the primary key value lets SQL find the exact record to be deleted from the PRODUCT table. However, deletions are not limited to a primary key match; any attribute may be used. For example, in your PRODUCT table, you will see several products for which the P_MIN attribute is equal to 5. Use the following command to delete all rows from the PRODUCT table for which the P_MIN is equal to 5:

```
DELETE FROM   PRODUCT
WHERE         P_MIN = 5;
```

Lab 2.5. Saving Table Changes

Sunday, October 2, 2022, 10:42 PM

Number of replies: 0

Any changes made to the table contents are not saved on disk until you close the database, close the program you are using, or use the COMMIT command. If the database is open and a power outage or some other interruption occurs before you issue the COMMIT command, your changes will be lost and only the original table contents will be retained. The syntax for the COMMIT command is:

```
COMMIT [WORK]
```

The COMMIT command permanently saves all changes—such as rows added, attributes modified, and rows deleted—made to any table in the database. Therefore, if you intend to make your changes to the PRODUCT table permanent, it is a good idea to save those changes by using the following command:

COMMIT;

However, the COMMIT command's purpose is not just to save changes. In fact, the ultimate purpose of the COMMIT and ROLLBACK commands is to ensure database update integrity in transaction management.

Lab 2.6. Restoring Table Contents

Sunday, October 2, 2022, 10:42 PM

Number of replies: 0

If you have not yet used the COMMIT command to store the changes permanently in the database, you can restore the database to its previous condition with the ROLLBACK command. ROLLBACK undoes any changes since the last COMMIT command and brings all of the data back to the values that existed before the changes were made. To restore the data to its "prechange" condition, type:

ROLLBACK;

and then press Enter.