Islamic University of Technology

Database Management Systems Lab

CSE 4308 / CSE 4174

Lab 3

Author:
Ishmam Tashdeed
Md. Tariquzzaman
Zannatun Naim Sristy
CSE, IUT

Contents

1 Foreign key

Foreign keys are used to restrict the domain of columns of one table to the values of another table. The statement for declaring a foreign key is as follows:

```
CREATE TABLE table_name
(
  attribute1 datatype [ NULL | NOT NULL ],
  attribute2 datatype [ NULL | NOT NULL ],
  ...,
  [CONSTRAINT constraint_name] FOREIGN KEY (
   foreign_attribute1, ...)
    REFERENCES reference_table_name [ON DELETE CASCADE | SET
   NULL | SET DEFAULT]
                                  [ON UPDATE CASCADE | SET NULL
    | SET DEFAULT]
);
```

For example, the `DEPT_NAME` column of the `COURSE` table can only have values of the departments that are available in the `DEPARTMENT` table.

```
CREATE TABLE DEPARTMENT
(
  DEPT_NAME VARCHAR2(20),
  TITLE VARCHAR2(30),
    EST_YEAR VARCHAR2(4),
  CONSTRAINT PK_DEPARTMENT PRIMARY KEY(DEPT_NAME)
);
```

```
CREATE TABLE COURSE
(
  COURSE_ID VARCHAR2(8),
  TITLE VARCHAR2(30),
    PROGRAM VARCHAR2(5),
  DEPT_NAME VARCHAR2(20),
  CREDITS NUMBER,
  CONSTRAINT PK_COURSE PRIMARY KEY(COURSE_ID, PROGRAM),
  CONSTRAINT FK_COURSE_DEPARTMENT FOREIGN KEY(DEPT_NAME)
   REFERENCES DEPARTMENT(DEPT_NAME) ON DELETE CASCADE
);
```

Here you need to ensure the referenced attribute of referencing table should be of the same data type as referenced attribute of the referenced table. It is preferred to use the primary key (or composite primary key) of the referenced table as the foreign key. **Moreover, the referencing table**

must be created after the referenced table.
In some cases, a referencing table may need to reference itself, this is the concept of self-referencing.

2 Distinct

The `DISTINCT` statement is used to return only distinct (unique) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values. The syntax is following:

```
SELECT DISTINCT attributename
    FROM tablename;
```

For example,

```
SELECT DISTINCT DEPT_NAME FROM EMPLOYEE;
```

3 Range Operator

3.1 BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, strings or dates. The BETWEEN operator is inclusive: begin and end values will be included. However, you have to provide the lower value first then the upper value.

```
    SELECT attribute_1, .....
    FROM tablename
    WHERE attribute_1 BETWEEN l_value AND u_value;
```

For example,

```
    SELECT NAME, DEPT_NAME, SALARY
    FROM EMPLOYEE
    WHERE SALARY BETWEEN 35000 AND 80000;
```

For excluding a range we can use NOT operator before BETWEEN. For example,

```
    SELECT NAME, DEPT_NAME, SALARY
    FROM EMPLOYEE
    WHERE SALARY NOT BETWEEN 35000 AND 80000;
```

3.2 IN Operator

The IN operator allows one to specify multiple values in WHERE clause. It is a shorthand for multiple OR conditions.

```
SELECT attribute_1, .....
  FROM tablename
WHERE attribute_1 IN (value1, value2, ...);
```

For example,

```
SELECT NAME, SALARY
FROM EMPLOYEE
WHERE DEPT_NAME in ('DEV', 'TESTING');
```

Similar to between, we can use NOT to exclude some values.

4 Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name. We can also use it for renaming an expression or an entire query. The primary purpose of using aliases is to make a column or table more readable. For example:

```
SELECT ID, NAME, (SALARY*12) AS ANNUAL_SALARY
FROM EMPLOYEE;
```

Here using `AS` is optional for the expression, table and any query.

5 String Operator

5.1 LIKE Operator

In simple terms, the LIKE operator is used to match substrings in a query. It is used in a WHERE clause to search for a specified pattern in a column. There are two wildcards often used in conjunction with the LIKE operator:

- % to represent any substring.
- _ to represent any single character.

```
SELECT attribute_1, attribute_2 .....
FROM tablename
WHERE attribute_1 LIKE pattern;
```

Here, Patterns are case-sensitive. For example,

```
SELECT NAME
FROM EMPLOYEE
WHERE NAME LIKE 'A%';
```

So, It will show you all the names that start with capital A. For restricting the name size of the resultant name additionally, let's say 4 characters. We can use the following statement:

```
SELECT NAME
FROM EMPLOYEE
WHERE NAME LIKE 'A___';
```

5.2 Concatenation

To concatenate multiple columns or any additional string with any column at the time of retrieving data, SQL supports string operator(||). For instance,

```
SELECT 'Employee NAME: '|| NAME
FROM EMPLOYEE;
```

It will show the term 'Employee NAME: ' as the prefix of any name.

6 Data Sorting

The ORDER BY keyword is used to sort the result-set in ascending or descending order. If one does not specify the sorting order, the ORDER BY keyword sorts the records in ascending order by default. Otherwise, to specify ascending or descending order of sorting, we use the keywords 'ASC' and 'DESC' respectively.

```
SELECT attribute_1, attribute_2 .....
  FROM tablename
ORDER BY attribute_1 [ASC/DESC], .....;
```

For example,

```
SELECT NAME, SALARY
FROM EMPLOYEE
ORDER BY DEPT, SALARY DESC;
```

7 Sub-query

SQL provides a mechanism for nesting sub-queries. A sub-query is a `SELECT-FROM-WHERE` expression that is nested within another query. It can be placed at two places:

- In the WHERE clause of a SELECT statement. Example,

```
SELECT DEPT
FROM EMPLOYEE
WHERE SALARY>
    (SELECT avg(SALARY)
     FROM EMPLOYEE);
```

8 Lab Task

Column	Data Type	Description
rest_id	INT	Primary key
name	VARCHAR(50)	Name of the restaurant
city	VARCHAR(30)	City where the restaurant is located
cuisine	VARCHAR(30)	Type of cuisine (e.g., Italian, Chinese)

Table 1: `restaurant` Table Schema

Column	Data Type	Description
dish_id	INT	Primary key
name	VARCHAR(50)	Name of the dish
cuisine	VARCHAR(30)	Type of cuisine
price	DECIMAL(5,2)	Price of the dish
taste	VARCHAR(20)	Taste profile (e.g., Spicy, Sweet)
rest_id	INT	Foreign key referencing `restaurant(rest_id)`

Table 2: `dish` Table Schema

Column	Data Type	Description
cust_id	INT	Primary key
first_name	VARCHAR(30)	First name of the customer
last_name	VARCHAR(30)	Last name of the customer
city	VARCHAR(30)	City where the customer resides

Table 3: `customer` Table Schema

1. Create Tables and Insert the Data

rest_id	name	city	cuisine
1	Bella Italia	New York	Italian
2	Dragon Palace	San Francisco	Chinese
3	Spice Route	Chicago	Indian
4	Sushi World	Los Angeles	Japanese
5	Taco Fiesta	Houston	Mexican

Table 4: `restaurant` Table Data

dish_id	name	cuisine	price	taste	rest_id
1	Margherita Pizza	Italian	12.99	Savory	1
2	Spaghetti Carbonara	Italian	14.99	Savory	1
3	Sweet and Sour Pork	Chinese	10.99	Sweet	2
4	Kung Pao Chicken	Chinese	11.99	Spicy	2
5	Butter Chicken	Indian	13.50	Spicy	3
6	Tandoori Chicken	Indian	14.00	Spicy	3
7	California Roll	Japanese	8.99	Savory	4
8	Salmon Sashimi	Japanese	15.99	Savory	4
9	Chicken Tacos	Mexican	9.99	Spicy	5
10	Beef Burrito	Mexican	11.50	Spicy	5

Table 5: `dish` Table Data

cust_id	first_name	last_name	city
1	John	Smith	New York
2	Mary	Johnson	Los Angeles
3	Robert	Brown	Chicago
4	Linda	Davis	Houston
5	Michael	Miller	Phoenix

Table 6: `customer` Table Data

2. Write a query to display the different types of cuisines available in the `dish` table. Ensure that each cuisine type is listed only once.

3. List all dishes whose `price` is between $10 and $15, inclusive.

4. Find all dishes whose names start with the word 'Chicken'.

5. Find all dishes whose names contain 'Roll' anywhere in their names.

6. Find all dishes whose names have exactly 3 words.

7. Create a query to display the full names (first name and last name concatenated) of all customers, along with their city.

8. List all dishes sorted first by `cuisine` in ascending order and then by `price` in descending order.

9. Suppose you need to track which customer ordered which dishes. To do this, create a new table called `customer_dish` with the following columns:

   - `cust_id INT`
   - `dish_id INT`
   - `order_date DATE`

   Write a SQL statement to create the `customer_dish` table.

10. Add a foreign key constraint on `cust_id` referencing `customer(cust_id)` and another on `dish_id` referencing `dish(dish_id)`.