Southern New Hampshire University

DAD 220 Module Two Activity Template
Prepared on: 11 Mar., 2022
Prepared for: Prof. Aastha Agarwal
Prepared by: Alexander Ahmann

Preface

Before I may proceed to answer the questions regarding this assignment, I must first create the tables in my `ahmann` MySQL database and then populate them with the test data.

The first part is fairly easy: the module includes two SQL queries to create the databases:

```
CREATE TABLE Employee (
    Employee_ID SMALLINT,
    First_Name VARCHAR(40),
    Last_Name VARCHAR(60),
    Department_ID SMALLINT,
    Classification VARCHAR(10),
    Status VARCHAR(10),
    Salary DECIMAL(7,2)
);
```

and

```
CREATE TABLE Branches (
    Department_ID SMALLINT,
    Department_Name VARCHAR(50)
);
```

As I demonstrate in Figure 1, creating the tables is fairly trivial: it was just a matter of copying and pasting the two given queries after switching to the database that I created in the previous week with the SQL queries "`USE ahmann;`". To

```
mysql> USE ahmann;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE TABLE Employee (
    ->     Employee_ID SMALLINT,
    ->     First_Name VARCHAR(40),
    ->     Last_Name VARCHAR(60),
    ->     Department_ID SMALLINT,
    ->     Classification VARCHAR(10),
    ->     Status VARCHAR(10),
    ->     Salary DECIMAL(7,2)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Branches (
    ->     Department_ID SMALLINT,
    ->     Department_Name VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> DESCRIBE Employee;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| Employee_ID    | smallint(6)  | YES  |     | NULL    |       |
| First_Name     | varchar(40)  | YES  |     | NULL    |       |
| Last_Name      | varchar(60)  | YES  |     | NULL    |       |
| Department_ID  | smallint(6)  | YES  |     | NULL    |       |
| Classification | varchar(10)  | YES  |     | NULL    |       |
| Status         | varchar(10)  | YES  |     | NULL    |       |
| Salary         | decimal(7,2) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> DESCRIBE branches;
ERROR 1146 (42S02): Table 'ahmann.branches' doesn't exist
mysql> DESCRIBE Branches;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Department_ID   | smallint(6) | YES  |     | NULL    |       |
| Department_Name | varchar(50) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Figure 1: Creating tables.

demonstrate that my query worked, I used "`DESCRIBE Employee`" and "`DESCRIBE`

`Branches;`" SQL queries to display the table and its respective properties.[1]

The next step is to populate the tables with the examples. This is only slightly harder than creating the tables, but can easily be solved by someone who has a basic proficiency in the construction of SQL queries. I demonstrate this in Figure 4.[2]

The second module gives the student the following tables for both the `Employee` (Figure 2) and `Branches` (Figure 3) tables. The table can be populated with the `INSERT INTO` SQL statement.

`INSERT INTO` is, in this case, written like:

```
INSERT INTO [table] ([column 1], [column 2], ...  [column n])
       VALUES ([value 1], [value 2], ...  [value n]);
```

Again, figure 4 shows how this works with the given examples, and figure 5 demonstrates that I indeed was able to "`INSERT`" the appropriate rows and data into their respective tables. With this done, I can proceed to work out the problem set.

Employee_ID	First_Name	Last_Name	Department_ID	Classification	Status	Salary
100	John	Smith	1	Exempt	Full-Time	90000
101	Mary	Jones	2	Non-Exempt	Part-Time	35000
102	Mary	Williams	3	Exempt	Full-Time	80000
103	Gwen	Johnson	2	NULL	Full-Time	40000
104	Michael	Jones	3	Non-Exempt	Full-Time	90000

Figure 2: Employee table

[1]The reader may notice that I tried to "`DESCRIBE branches`" with the "branches" being lowercase. An error was shown and this typo demonstrates that SQL queries are case sensitive.

[2]Note that I did not include every query for every individual row to keep my figures nice and short.

Department_ID	Department_Name
1	Accounting
2	Human Resources
3	Information Systems
4	Marketing

Figure 3: Branches table

```
mysql> INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Department_ID, Classification, Status, Salary)
 VALUES (100, "John", "Smith", 1, "Exempt", "Full-Time", 90000);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Department_ID, Classification, Status, Salary)
 VALUES (101, "Mary", "Jones", 2, "Non-Exempt", "Part-Time", 35000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Department_ID, Classification, Status, Salary)
 VALUES (103, "Mary", "Williams", 3, "Exempt", "Full-Time", 80000);
Query OK, 1 row affected (0.02 sec)
```

Figure 4: Using `INSERT` to populate the tables.

```
mysql> SELECT * FROM Employee;
+-------------+------------+-----------+---------------+----------------+-----------+----------+
| Employee_ID | First_Name | Last_Name | Department_ID | Classification | Status    | Salary   |
+-------------+------------+-----------+---------------+----------------+-----------+----------+
|         100 | John       | Smith     |             1 | Exempt         | Full-Time | 90000.00 |
|         101 | Mary       | Jones     |             2 | Non-Exempt     | Part-Time | 35000.00 |
|         102 | Mary       | Williams  |             3 | Exempt         | Full-Time | 80000.00 |
|         103 | Gwen       | Johnson   |             2 | NULL           | Full-Time | 40000.00 |
|         104 | Michael    | Jones     |             3 | Non-Exempt     | Full-Time | 90000.00 |
+-------------+------------+-----------+---------------+----------------+-----------+----------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Branches;
+---------------+---------------------+
| Department_ID | Department_Name     |
+---------------+---------------------+
|             1 | Accounting          |
|             2 | Human Resources     |
|             3 | Information Systems |
|             4 | Marketing           |
+---------------+---------------------+
4 rows in set (0.00 sec)
```

Figure 5: Demonstration with `SELECT` queries.

My solutions to the problem sets

Note that for this section, rather than embedding screenshots, I decided to just copy and paste terminal output to keep things less cluttered—unless I am struggling with rendering the copy/pastes in the final pdf document.

1. How many records are shown in the `Employee` table?

This can easily be worked out with the following query: "`SELECT COUNT(*) FROM Employee;`"—which returns the following:

```
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

So, a total of five (5) rows or employee records.

2. How many attributes are there in the `Branches` table?

This can be worked out with the following query: "`DESCRIBE Branches;`—which outputs the following:

```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Department_ID   | smallint(6) | YES  |     | NULL    |       |
| Department_Name | varchar(50) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

So, a total of two (2) attributes in the `Branches` tables.

3. **Which attribute could be a primary key for the `Employee` table?**

To work this question out, I must first list the attributes of the `Employee` table:

```
mysql> DESCRIBE Employee;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
```

```
+----------------+-------------+------+-----+---------+-------+
| Employee_ID    | smallint(6)  | YES  |     | NULL    |       |
| First_Name     | varchar(40)  | YES  |     | NULL    |       |
| Last_Name      | varchar(60)  | YES  |     | NULL    |       |
| Department_ID  | smallint(6)  | YES  |     | NULL    |       |
| Classification | varchar(10)  | YES  |     | NULL    |       |
| Status         | varchar(10)  | YES  |     | NULL    |       |
| Salary         | decimal(7,2) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

I ruled out the `First_Name`, `Last_Name` and `Salary` columns, as it is possible—albeit unlikely—for employees to have the same given and surnames, or identical salaries. I can also rule out `Department_ID`, `Classification` and `Status` as it is not only possible, but very likely to find multiple employees in different departments, different "classifications" and different employment statuses.

Ergo, by process of elimination, I have arrived at `Employee_ID` as the only possible column that can be a qualified candidate for the `PRIMARY KEY`. This makes sense, as it is possible for employees to have identical names, be in identical departments, have identical employment statuses and salaries — but have unique `Employee_ID`s, as this number is independent[3] from other identifying features.

4. How many decimal places (maximum) can be stored in an employee's salary field?

To answer this, I need to reference the `Salary` row from the attributes table that I created in question 3:

```
[... snip ...]
| Salary         | decimal(7,2) | YES  |     | NULL    |       |
[... snip ...]
```

The "`DECIMAL(P, D)`" data type defines the number of significant digits (P) and the number of digits after the decimal point (D). The answer here is the "D" or "2" in this specific case.

5. How many decimal places (maximum) can a `Department_ID` have?

As with question 4, I will reference the attributes table created in question 3:

[3] I use this word in the casual sense as opposed to the specific "data independent" technical term in relational databases

```
[... snip ...]
| Department_ID  | smallint(6)  | YES  |     | NULL    |       |
[... snip ...]
```

The `Department_ID` is a `SMALLINT(n)` datatype, which means that there are no decimal points after it. So, the `Department_ID` has a maximum of zero (0) decimal places after it.

6. What three rules do tables obey?

Tables universally obey these three (3) rules:

i. They are *normalised*, which means that exactly one value exists in each cell.
ii. There are *unique column names* within the same table.
iii. There are no duplicate rows.

7. How do you know that the `Employee` table is or is not normalized?

I had to do a bit of research on this, and came up across an article written by Wenzel (2022)[4] that discusses it.

The command "`DESCRIBE Employee;`" can list all the attributes in the `Employee` table. It can be worked out that the `Employee` is normalised because it follows the relational "R × C" format, which produces a cell. The cells are defined by a data type, which tells us that there is only one possible value for the cell.

8. What is the result of the following query? - `Select sum(Salary) from Employee where Department_ID=3;`

```
mysql> Select sum(Salary) from Employee where Department_ID=3;
+-------------+
| sum(Salary) |
+-------------+
|   170000.00 |
+-------------+
1 row in set (0.00 sec)
```

9. How many rows are returned as a result of the following query? - `Select * from Employee where Classification<> 'Exempt';`

See figure 6 for the answer.

[4]Wenzel, K. (2022). *Database Normalization – in Easy to Understand English*. essentialSQL. Retrieved on Mar. 11, 2022 from: https://www.essentialsql.com/database-normalization/

```
mysql> Select * from Employee where Classification<> 'Exempt';
+-------------+------------+-----------+---------------+----------------+-----------+----------+
| Employee_ID | First_Name | Last_Name | Department_ID | Classification | Status    | Salary   |
+-------------+------------+-----------+---------------+----------------+-----------+----------+
|         101 | Mary       | Jones     |             2 | Non-Exempt     | Part-Time | 35000.00 |
|         104 | Michael    | Jones     |             3 | Non-Exempt     | Full-Time | 90000.00 |
+-------------+------------+-----------+---------------+----------------+-----------+----------+
2 rows in set (0.00 sec)
```

Figure 6: Demonstration with `Answer to question 8.`

10. What is the result of the following query? - `Select max(Salary) from Employee;`

```
mysql> Select max(Salary) from Employee;
+-------------+
| max(Salary) |
+-------------+
|    90000.00 |
+-------------+
1 row in set (0.01 sec)
```