Jared Yu

HW4

In the first step of the assignment, my goal was to create the tables via the CREATE TABLE commands and the given code to do so (after having created the database itself). An initial problem however was that I couldn’t create the first table, EMPLOYEE, since it had the foreign key constraint that also required the action “ON DELETE SET DEFAULT.” This is referring to the DEPARTMENT table which hadn’t been created yet. Therefore, intuitively I went to try and create the DEPARTMENT table next so that I could go back and create the EMPLOYEE table. However, this DEPARTMENT table similarly had a foreign key requirement on EMPLOYEE, with the same action “ON DELETE SET DEFAULT.” This made it such that neither table could be initialized, since an error would pop up showing that. The error can be seen below in Figure 1. The bottom shows “Error Code 1824…” where it says the reason is that MySQL fails to open the reference table, which makes sense since neither has yet been created.

Timeline

Description automatically generated

Figure The above figure shows the error caused by trying to call CREATE TABLE on EMPLOYEE and DEPARTMENT.

The solution then was to remove the foreign key creation from within the CREATE TABLE command for EMPLOYEE. From there, it was possible to create the EMPLOYEE table as is since there is no reference table. Following this, the DEPARTMENT table could easily be created, since the reference table EMPLOYEE does exist. The third step then is to use “ALTER TABLE” on EMPLOYEE, where the lines removed to create the foreign key in EMPLOYEE were placed in this code. This allows for the creation of both EMPLOYEE and DEPARTMENT, while also allowing for the foreign key to exist. This process can be seen below in Figure 2, where the FOREIGN KEY commands previously in EMPLOYEE were commented out.

A picture containing text

Description automatically generated

Figure The above figure shows the EMPLOYEE and DEPARTMENT tables correctly being created and using ALTER TABLE to add the FK to EMPLOYEE afterward.

There was also a small sub-problem that existed prior to the creation of the DEPARTMENT table. The issue was in the given code provided with the homework prompt. The default value given for “Mgr\_ssn” was ‘88866555,’ but this is only 8 characters long, while the datatype itself is “CHAR(9).” To overcome this, a ‘6’ was added so that the default “Mgr\_ssn” could properly be used. The problem can be seen below in Figure 3.

A picture containing graphical user interface, text

Description automatically generated

Figure The homework prompt shows a requirement of CHAR(9), while the DEFAULT is only of length 8.

The entire DDL process is shown below:

|  |
| --- |
| SHOW DATABASES;  CREATE DATABASE COMPANY;  USE COMPANY;  SHOW TABLES;  CREATE TABLE EMPLOYEE (  Fname VARCHAR(15) NOT NULL,  Minit CHAR,  Lname VARCHAR(15) NOT NULL,  Ssn CHAR(9) NOT NULL,  Bdate DATE,  Address VARCHAR(30),  Sex CHAR,  Salary DECIMAL(10,2),  Super\_ssn CHAR(9),  Dno INT NOT NULL DEFAULT 0,  CONSTRAINT EMP\_PK PRIMARY KEY (Ssn),  CONSTRAINT EMP\_SUPER\_PK FOREIGN KEY (Super\_ssn) REFERENCES EMPLOYEE(Ssn)  ON DELETE SET NULL ON UPDATE CASCADE  /\* CONSTRAINT EMP\_DEPT\_FK FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)  ON DELETE SET DEFAULT ON UPDATE CASCADE \*/  );  CREATE TABLE DEPARTMENT (  Dname VARCHAR(15) NOT NULL,  Dnumber INT NOT NULL,  Mgr\_ssn CHAR(9) NOT NULL DEFAULT '888666555',  Mgr\_start\_date DATE,  CONSTRAINT DEPT\_PK  PRIMARY KEY (Dnumber),  CONSTRAINT DEPT\_UK  UNIQUE (Dname),  CONSTRAINT DEPT\_MGR\_FK  FOREIGN KEY (Mgr\_ssn) REFERENCES EMPLOYEE (Ssn)  ON DELETE SET DEFAULT  ON UPDATE CASCADE  );  ALTER TABLE EMPLOYEE  ADD CONSTRAINT EMP\_DEPT\_FK  FOREIGN KEY (Dno) REFERENCES DEPARTMENT (Dnumber)  ON DELETE SET DEFAULT ON UPDATE CASCADE;  CREATE TABLE DEPT\_LOCATIONS (  Dnumber INT NOT NULL,  Dlocation VARCHAR(15) NOT NULL,  CONSTRAINT DEPT\_LOCATIONS\_PK  PRIMARY KEY (Dnumber, Dlocation),  CONSTRAINT DEPT\_LOCATIONS\_DEPT\_FK  FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT (Dnumber)  ON DELETE CASCADE ON UPDATE CASCADE  );  CREATE TABLE PROJECT (  Pname VARCHAR(15) NOT NULL,  Pnumber INT NOT NULL,  Plocation VARCHAR(15),  Dnum INT NOT NULL,  CONSTRAINT PROJECT\_PK  PRIMARY KEY (Pnumber),  UNIQUE (Pname),  CONSTRAINT PROJECT\_DEPT\_FK  FOREIGN KEY (Dnum) REFERENCES DEPARTMENT (Dnumber)  );  CREATE TABLE WORKS\_ON (  Essn CHAR(9) NOT NULL,  Pno INT NOT NULL,  Hours DECIMAL(3,1),  CONSTRAINT WORKS\_ON\_PK  PRIMARY KEY (Essn, Pno),  CONSTRAINT WORKS\_ON\_EMP\_FK  FOREIGN KEY (Essn) REFERENCES EMPLOYEE (Ssn),  CONSTRAINT WORKS\_ON\_PROJ\_FK  FOREIGN KEY (Pno) REFERENCES PROJECT (Pnumber)  );  CREATE TABLE DEPENDENT (  Essn CHAR(9) NOT NULL,  Dependent\_name VARCHAR(15) NOT NULL,  Sex CHAR,  Bdate DATE,  Relationship VARCHAR(8),  CONSTRAINT DEPENDENT\_PK  PRIMARY KEY (Essn, Dependent\_name),  CONSTRAINT DEPENDENT\_EMP\_FK  FOREIGN KEY (Essn) REFERENCES EMPLOYEE (Ssn)  ); |

An interesting issue appeared after trying to recreate the work done after it was believed that the assignment had been completed. Previously, some “random” commands were tried occasionally in hopes of fixing some bug. Without proper documentation, this is poor work ethic. This became apparent when after having recreated the tables in the DDL, the insertion stage was reattempted. However, an Error 1452 would occur, saying that the issue was that there were some dependency issues related to a FK. However, this issue didn’t directly arise the first time around. It is reasoned that because previously, the following command was used for some other reason “SET foreign\_key\_checks=0” [3]. This makes it possible to insert data into the EMPLOYEE table. It should be set back to 1 afterward so that these errors are avoided. Therefore, to avoid the problem of not being allowed to insert data, it should first be set to 0, then back to 1 after the data are first inserted to the EMPLOYEE and DEPARTMENT tables. This error can be seen below in Figure 4.

Graphical user interface, text, application

Description automatically generated

Figure The above screenshot shows the Error Code: 1452 which can be fixed changing the FK settings.

Another minor issue was during the data insertion step. For the EMPLOYEE and DEPARTMENT tables, the attribute names were “SUPERSSN” rather than “Super\_ssn,” “MGRSSN” rather than “Mgr\_ssn,” and “MGRSTARTDATE” rather than “Mgr\_start\_date.” I am not sure actually if software other than MySQL will allow different column names, but for this assignment I merely renamed all the columns in the INSERT to reflect exactly the attribute names within the tables. A last minor issue is that in the insert data step for DEPT\_LOCATIONS, one of the tuples was missing a value, even though the requirement is set to NOT NULL. Referring back to the textbook, it is apparent that the correct tuple should be (5, ‘Sugarland’) rather than (, ‘Sugarland’).

The entire DML process is shown below:

|  |
| --- |
| INSERT INTO EMPLOYEE(Fname, Minit,Lname,Ssn,Bdate,Address,Sex,Salary,Super\_ssn,Dno)  VALUES ('John','B','Smith',123456789,'1965-01-09','731 Fondren, Houston, TX','M',30000,333445555,5),  ('Franklin','T','Wong',333445555,'1955-12-08','635 Voss, Houston, TX','M',40000,888665555,5),  ('Alicia','J','Zelaya',999887777,'1968-01-19','3321 Castle, Spring','F',25000,987654321,4),  ('Jennifer','S','Wallace',987654321,'1941-06-20','291 Berry, Bellaire, TX','F',43000,888665555,4),  ('Ramesh','K','Narayan',666884444,'1962-11-15','975 Fire Oak, Humble, TX','M',38000, 333445555,5),  ('Joyce','A','English',453453453,'1972-07-31','5631 Rice Houston','F',25000,333445555,5),  ('Ahmad','V','Jabbar',987987987,'1969-03-29','980 Dallas, Houston','M',25000,987654321,4),  ('James','E','Borg',888665555,'1937-11-10','450 Stone, Houston, TX','M',55000,NULL,1);  INSERT INTO DEPARTMENT(Dname, Dnumber, Mgr\_ssn, Mgr\_start\_date)  VALUES ('Research',5,333445555,'1988-05-22'),  ('Headquarters',1,888665555,'1981-06-19'),  ('Administration',4,987654321,'1995-01-01');  INSERT INTO DEPT\_LOCATIONS(Dnumber, Dlocation) VALUES  (1,'Houston'),  (4,'Stafford'),  (5,'Bellaire'),  (5,'Sugarland'),  (5,'Houston');  INSERT INTO PROJECT(Pname, Pnumber, Plocation, Dnum)  VALUES ('ProductX',1,'Bellaire',5),  ('ProductY',2,'Sugarland',5),  ('ProductZ',3,'Houston',5),  ('Computerization',10,'Stafford',4),  ('Reorganization',20,'Houston',1),  ('Newbenefits',30,'Stafford',4);  INSERT INTO WORKS\_ON(Essn, Pno, Hours)  VALUES (123456789,1,32.5),  (123456789,2,7.5),  (666884444,3,40),  (453453453,1,20),  (453453453,2,20),  (333445555,2,10),  (333445555,3,10),  (333445555,10,10),  (333445555,20,10),  (999887777,30,30),  (999887777,10,10),  (987987987,10,35),  (987987987,30,5),  (987654321,30,20),  (987654321,20,15),  (888665555,20,NULL);  INSERT INTO DEPENDENT(Essn, Dependent\_name, Sex, Bdate, Relationship)  VALUES (123456789,'Alice','F','1988-12-30','Daughter'),  (123456789,'Elizabeth','F','1967-05-05','Spouse'),  (123456789,'Micheal','M','1988-01-04','Son'),  (333445555,'Alice','F','1986-04-05','Daughter'),  (333445555,'Joy','F','1958-05-03','Spouse'),  (333445555,'Theodore','M','1983-10-25','Son'),  (987654321,'Abner','M','1942-02-28','Spouse'); |

The following screenshots show the data being insert successfully into the tables within MySQL.

|  |
| --- |
| Graphical user interface, text, email  Description automatically generated |
| Table  Description automatically generated |
| Table  Description automatically generated |
| Table  Description automatically generated |
| Graphical user interface, table  Description automatically generated |
| Graphical user interface, table  Description automatically generated |

Figure The above screenshots show the data after being inserted to their tables. It shows from top to bottom: EMPLOYEE, DEPARTMENT, DEPT\_LOCATIONS, PROJECT, WORKS\_ON, and DEPENDENT.

The hardest part was performing the DIVISION operation (*Note: I have the international edition and so the pages and figures maybe different.*). I referenced the textbook on pp.285-287 to determine a solution. I noticed that on my platform, MySQL Workbench, that the DIVISION operator is not allowed. However, the textbook indicates a 3-step method to perform the same task. This 3-step task however requires the MINUS/EXCEPT/DIFFERENCE operator which apparently is also not quite possible in MySQL. I tried to consider some alternatives, but I found a trick online [1] to instead use LEFT JOIN to perform the same task as using the DIFFERENCE operator.

To figure out the entire process, I tried to build the queries step-by-step like the textbook. I first queried SMITH as in the textbook:

|  |
| --- |
| /\*SMITH\*/  SELECT \*  FROM EMPLOYEE  WHERE Fname = 'John' AND Lname = 'Smith'; |

Then I queried SMITH\_PNOS:

|  |
| --- |
| /\*SMITH\_PNOS\*/  SELECT Pno  FROM WORKS\_ON AS table1, (SELECT \* FROM EMPLOYEE WHERE Fname = 'John' AND Lname = 'Smith') AS table2  WHERE table1.Essn = table2.Ssn; |

Then I queried SSN\_PNOS:

|  |
| --- |
| /\*SSN\_PNOS\*/  (SELECT Pno, Essn  FROM WORKS\_ON); |

The difficult part comes next with the division operator, dividing SSN\_PNOS by SMITH\_PNOS. This is broken down using relational algebra as follows:

T1 is also quite simple and is as follows:

|  |
| --- |
| /\*T1\*/  SELECT Essn  FROM (SELECT Essn, Pno FROM WORKS\_ON) AS table0; |

T2 is more complicated. First I found :

|  |
| --- |
| (SELECT \*  FROM (SELECT Pno  FROM WORKS\_ON AS table1, (SELECT \* FROM EMPLOYEE WHERE Fname = 'John' AND Lname = 'Smith') AS table2  WHERE table1.Essn = table2.Ssn) AS table1  CROSS JOIN (SELECT Essn  FROM (SELECT Essn, Pno FROM WORKS\_ON) AS table0) AS table2); |

Then I took the difference of the cross join and SSN\_PNOS by using the LEFT JOIN trick. This step also makes it easy to include the projection of only Essn:

|  |
| --- |
| /\*T2\*/  (SELECT table1.Essn  FROM (SELECT \*  FROM (SELECT Pno  FROM WORKS\_ON AS table1, (SELECT \* FROM EMPLOYEE WHERE Fname = 'John' AND Lname = 'Smith') AS table2  WHERE table1.Essn = table2.Ssn) AS table1  CROSS JOIN (SELECT Essn  FROM (SELECT Essn, Pno FROM WORKS\_ON) AS table0) AS table2) AS table1  LEFT JOIN (SELECT Pno, Essn FROM WORKS\_ON) AS table2  ON table1.Pno = table2.Pno AND table1.Essn = table2.Essn  WHERE table2.Pno IS NULL); |

After finding T2, I only need to repeat the LEFT JOIN difference trick once again to get the final result:

|  |
| --- |
| /\*SSNS(Ssn)\*/  (SELECT DISTINCT table1.Essn  FROM (SELECT Essn  FROM (SELECT Essn, Pno FROM WORKS\_ON) AS table0) AS table1  LEFT JOIN (SELECT table1.Essn /\*Here it is doing the projection\*/  FROM (SELECT \*  FROM (SELECT Pno  FROM WORKS\_ON AS table1, (SELECT \* FROM EMPLOYEE WHERE Fname = 'John' AND Lname = 'Smith') AS table2  WHERE table1.Essn = table2.Ssn) AS table1  CROSS JOIN (SELECT Essn  FROM (SELECT Essn, Pno FROM WORKS\_ON) AS table0) AS table2) AS table1  LEFT JOIN (SELECT Pno, Essn FROM WORKS\_ON) AS table2  ON table1.Pno = table2.Pno AND table1.Essn = table2.Essn  WHERE table2.Pno IS NULL) AS table2  ON table1.Essn = table2.Essn  WHERE table2.Essn IS NULL); |

This gives the final result that the only Ssn’s matching the condition are ‘123456789,’ and ‘453453453.’ This can be seen in the screenshot below.

Graphical user interface, text, application

Description automatically generated

Reference:

[1] <https://explainextended.com/2009/09/18/not-in-vs-not-exists-vs-left-join-is-null-mysql/>

[2] <https://www.mysqltutorial.org/mysql-cross-join/>

[3] <https://stackoverflow.com/questions/1253459/mysql-error-1452-cannot-add-or-update-a-child-row-a-foreign-key-constraint-fa>