2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #34908

Writing Triggers to Implement Business Rules in a Relational Database

Dr. Reza Sanati-Mehrizy, Utah Valley University

Reza Sanati-Mehrizy is a professor of Computer Science Department at Utah Valley University, Orem, Utah. He received his M.S. and Ph.D. in Computer Science from the University of Oklahoma, Norman, Oklahoma. His research focuses on diverse areas such as: Database Design, Data Structures, Artificial Intelligence, Robotics, Computer Aided Manufacturing, Data Mining, Data Warehousing, and Machine Learning.

Dr. Afsaneh Minaie, Utah Valley University

Afsaneh Minaie is a Professor and Chair of Engineering Department at Utah Valley University. She received her B.S., M.S., and Ph.D. all in Electrical Engineering from University of Oklahoma. Her research interests include gender issues in the academic sciences and engineering fields, Embedded Systems Design, Mobile Computing, Wireless Sensor Networks, Nanotechnology, Data Mining and Databases.

Writing Triggers to Implement Business Rules in a Relational Database

Abstract

Organizations have many business rules (constraints) to implement in their daily operations. This is done mainly by action assertions traditionally implemented in procedural logic buried deeply within user's application program in a form that is virtually unrecognizable, unmanageable, and inconsistent. This approach places a heavy burden on the programmer, who must know all the constraints that an action may violate and must include checks for each of these constraints. An omission, misunderstanding, or error by the programmer will likely leave the database in an inconsistent state.

Entity Relationship (ER) model is a common conceptual database design tool used for relational database design. To enforce the business rules, some business rules can be included in this ER model in the form of constraints. However, including constraints in this graphical model is just a reminder for the programmer to include them in his database implementation. The problem is that constraint is very rigid and the database becomes programmer dependent and there is no grantee that programmer will include them in his implementation.

An alternative solution for this problem is to implement the constraints in the form of triggers. Trigger is a program and is more flexible than a constraint. Trigger has Event, Condition and Action (ECA) property. When an event take place and consequently a condition becomes true, the trigger takes action automatically and modifies the database as needed.

In this paper, we use the ER notation to represent some business rules (constraints) graphically for a simplified university database and write triggers to implement them to ensure the consistency of the data in the database.

Introduction

Database is a collection of data that organizations and businesses may use frequently. It is very important that this date be valid and consistent as organizations and businesses' life depends on this data. To ensure the integrity and consistency of data in a database, the database designers need to consider many rules called business rules or constraints. This is done mainly by action assertions traditionally implemented in procedural logic buried deeply within user's application program in a form that is virtually unrecognizable, unmanageable, and inconsistent. This approach places a heavy burden on the programmer, who must know all the constraints that an action may violate and must include checks for each of these constraints. An omission, misunderstanding, or error by the programmer will likely leave the database in an inconsistent state.

Some constrains can be specified during the definitions of attributes while the tables are being created called column constraint like unique, primary key, null, not null, etc. Some other constraints can be specified right after the definitions of attributes called table constraint like foreign key, composite primary key, etc. But there are some constraints that can not be specifies as column constraint or table constraint.

Entity Relationship (ER) model is a common conceptual database design tool used for relational database design. To guarantee integrity and consistency of data in a database system, business rules need to be enforced. To do so, some business rules [2] can be included in this ER diagram in the form of constraints [1]. However, including constraints in this graphical model are just a reminder for the programmer to be considered during the database implementation but it is better to have them included. The problem is that constraint is very rigid and the database becomes programmer dependent and there is no grantee that programmer will include them in his implementation.

An alternative solution for this problem is to implement the constraints in the form of triggers. Trigger is a program and is more flexible than a constraint. Trigger has Event, Condition and Action (ECA) property. When an event take place and consequently a condition becomes true, the trigger takes action automatically and modifies the database as needed.

In this paper, we use the ER notation to represent some business rules (constraints) graphically for a simplified university database and write triggers to implement them to ensure the consistency of the data in the database. This is a type of assignment that our students will do in our database class.

Background Information

Our University is a state institution with about 40,000 students. This university is located in Utah County which has a population of over 430,000 residents. The Computer Science department at Utah Velley University (UVU) offers a Bachelor’s Degree in Computer Science, a Bachelor’s Degree in Computer Networking, a Bachelor’s Degree in Web Development, a Bachelor’s Degree in Software Engineering, and a Master’s Degree in Computer Science. The Bachelor of Science in Computer Science program was one of the first Bachelor of Science programs implemented at UVU in 1993. The program’s goal has been to provide a quality program that meets accreditation standards while providing the students with a skill set that allows them to succeed in computing careers. The curriculum content for the Computer Science degree is based on the 2001 ACM Curriculum Report. The Computer Science degree at UVU was accredited by Accreditation Board for Engineering and Technology (ABET) in 2002 and currently has more than 1,200 students. Students in this program take core courses until the first semester of their junior year, when they begin choosing their electives for different specialization areas.

Database Engineering is a viable component of Software Engineering. In our Undergraduate Computer Science Degree, there is only one database course where Relational, Object Relational, Object-Oriented and Distributed Databases will be covered. Since the mission of this University is to graduate students with high quality education prepared for the competitive job market, as part of this course work, students work on a set of assignments and implement one/two database(s) of their choice in teams. The contents represented in this paper is an example of an assignment that student need to do in this course.

Entity Relationship for University Database

The following diagram (Figure 1) represents a simplified entity relationship for course assignment of a university database.

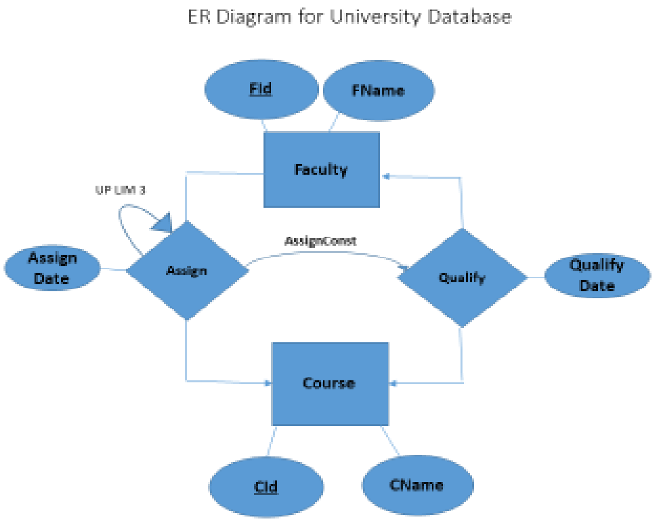


Figure 1: ER diagram for simplified for course assignment

The university has a number of faculty and offers a number of courses. The relationship qualify indicates that a faculty is qualified to teach many courses but not all courses and there are many faculty qualified to teach a course but not everyone. The relationship assign indicates that a faculty will be assigned many courses to teach. In our university, a course is assigned to only one faculty to teach. Therefore, the relationship assign is a one-to-many relationship.

Without considering any business rule, it is possible that a faculty be assigned to a course that he is not qualified to teach and imaging what a disaster it will be. Also it will be possible that many courses (more than 3) be assigned to a faculty to teach way beyond what he is supposed to teach.

To prevent such problems, two business rules (constraints) are expressed in this ER diagram, assignConst and Up LIM 3 that the programmer needs to consider during implementation of the underline database. UP LIM 3 indicates that maximum number of courses assigned to faculty can not exceed 3. asssignConst indicates that when we want to assign a course to a faculty to teach, we must make sure that on the date of assignment, the faculty is qualified to teach that course, means the date of qualitied must precede the date of course assignment. Otherwise the assignment should not take place.

These types of constraints can’t be implement as column constraint or table constraint mentioned before. We will write triggers to enforce this type of constraints (business rules).

Why Triggers

An active database is one in which the DBMS monitors the contents in order to prevent illegal states from occurring using constraints and triggers. However triggers are more flexible than constraints and allow a greater variety of actions [3].

Column and table constraints are rigid and can be used only the way DBMS allows. Triggers are flexible and can be written as needed. Triggers and routines are very powerful database objects because they are stored in the database and controlled by the DBMS. The code required to create them is stored in only one location and is administered centrally. This promotes stronger data integrity and consistency of use within the database and code maintenance is simplified [4].

Both routines and triggers are blocks of procedural codes. Routines must be called to operate but triggers will be executed automatically. Triggers have three parts Event, Condition and Action. When an event such as INSERT, UPDATE or DELETE takes place and certain condition becomes true, trigger will be executed (fired) automatically to take on action. Trigger can also cascade, causing other triggers to fire [1]. So, a single action (request) by a client can cause a series of integrity checks to be performed without too much network traffic. Trigger has many applications, can be used to ensure referential integrity, enforce business rules, create audit trail, replicate tables or activate a procedure [5]. Interested readers can get more information about triggers in Oracle 11g at [6].

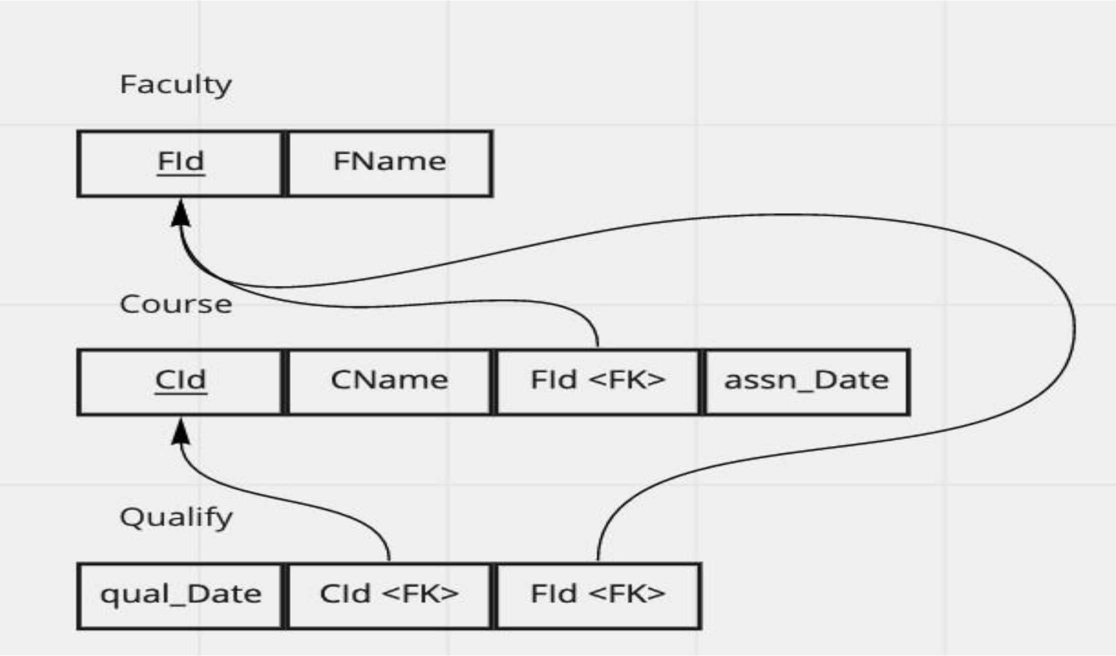


Figure 2: Schema for the Above ER Diagram:

Figure 2 represents the schema for the ER diagram represented in figure 1. Faculty, Qualify and Course become tables. Since the assign relationship is a one-to-many relationship, there is no need for assign table. But the key of faculty table (FId) and date assigned are added to course table. The programmer has created extra table temp to activate the triggers because in Oracle 11g (used in this project), if the trigger is written on update of course table, it does not allow to access the course table in the body of the trigger. But the actual course assignment is done by updating the course table.

Using Oracle Data Base Management System (DBMS), the corresponding tables have been created by the following SQL queries:

Queries to create tables

Note: From now-on Fac_Assn5 means Faculty
Crse_Assn5 means Course
Qual_Assn5 means Qualify
Temp_Assn5 meand Temp

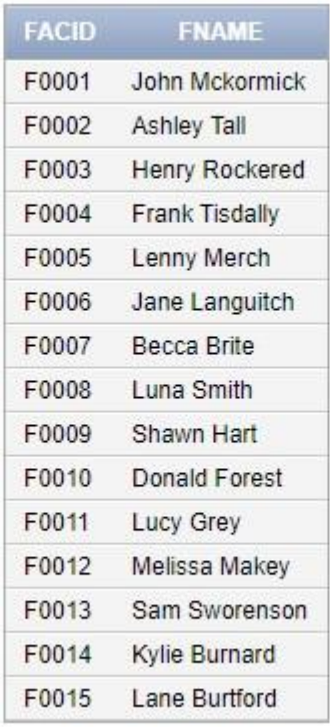
```
CREATE TABLE Fac_Assn5 (
FacId     CHAR(20),
FName      CHAR(20),
CONSTRAINT FacId_pk PRIMARY KEY (FacId)
);

CREATE TABLE Crse_Assn5 (
CrseId      CHAR(20),
CName    CHAR(20),
FId      CHAR(20),
assn_Date  DATE,
CONSTRAINT CId_pk PRIMARY KEY (CrseId),
CONSTRAINT Crsae_FId_fk FOREIGN KEY (FId) REFERENCES Fac_Assn5(FacId) );

CREATE TABLE Qual_Assn5 (
FId      CHAR(20),
CId      CHAR(20),
qual_Date DATE,
CONSTRAINT FId FOREIGN KEY (FId) REFERENCES Fac_Assn5(FacId),
CONSTRAINT CId_fk FOREIGN KEY (CId) REFERENCES Crse_Assn5(CrseId));

CREATE TABLE Temp_Assn5 (
FId     CHAR(20),
CId     CHAR(20),
assn_Date DATE,
CONSTRAINT assn_FId_fk FOREIGN KEY (FId) REFERENCES Fac_Assn5(FacId),
CONSTRAINT assn_CId_fk FOREIGN KEY (CId) REFERENCES Crse_Assn5(CrseId));
```

Queries to populate tables

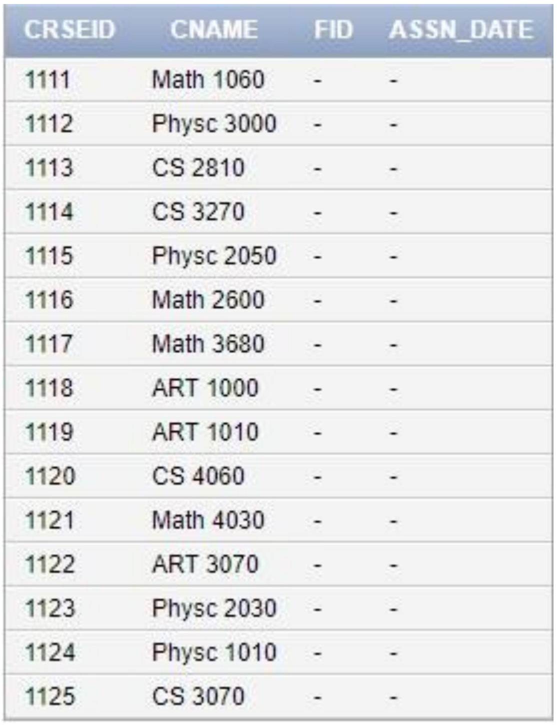
```
INSERT ALL
```

```
INTO Fac_Assn5 VALUES ('F0001', 'John Mckormick')
INTO Fac_Assn5 VALUES ('F0002', 'Ashley Tall')
INTO Fac_Assn5 VALUES ('F0003', 'Henry Rockered')
INTO Fac_Assn5 VALUES ('F0004', 'Frank Tisdally')
INTO Fac_Assn5 VALUES ('F0005', 'Lenny Merch')
INTO Fac_Assn5 VALUES ('F0006', 'Jane Languitch')
INTO Fac_Assn5 VALUES ('F0007', 'Becca Brite')
INTO Fac_Assn5 VALUES ('F0008', 'Luna Smith')
INTO Fac_Assn5 VALUES ('F0009', 'Shawn Hart')
INTO Fac_Assn5 VALUES ('F0010', 'Donald Forest')
INTO Fac_Assn5 VALUES ('F0011', 'Lucy Grey')
INTO Fac_Assn5 VALUES ('F0012', 'Melissa Makey')
INTO Fac_Assn5 VALUES ('F0013', 'Sam Sworenson')
INTO Fac_Assn5 VALUES ('F0014', 'Kylie Burnard')
INTO Fac_Assn5 VALUES ('F0015', 'Lane Burtford')
 SELECT * FROM DUAL;

INSERT ALL
INTO Crse_Assn5 (CrseId, CName) VALUES ('1111', 'Math 1060')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1112', 'Physc 3000')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1113', 'CS 2810')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1114', 'CS 3270')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1115', 'Physc 2050')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1116', 'Math 2600')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1117', 'Math 3680')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1118', 'ART 1000')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1119', 'ART 1010')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1120', 'CS 4060')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1121', 'Math 4030')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1122', 'ART 3070')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1123', 'Physc 2030')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1124', 'Physc 1010')
INTO Crse_Assn5 (CrseId, CName) VALUES ('1125', 'CS 3070')
 SELECT * FROM DUAL;

INSERT ALL
INTO Qual_Assn5 VALUES ('F0001', '1111', TO_DATE('10.05.2012','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0002', '1112', TO_DATE('03.11.2005','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0003', '1113', TO_DATE('13.09.2008','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0004', '1114', TO_DATE('06.07.2015','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0005', '1115', TO_DATE('05.05.2002','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0006', '1116', TO_DATE('06.03.2016','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0007', '1117', TO_DATE('07.06.2001','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0008', '1118', TO_DATE('01.11.2006','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0009', '1119', TO_DATE('02.09.2009','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0010', '1120', TO_DATE('12.04.2007','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0011', '1121', TO_DATE('15.05.2010','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0012', '1122', TO_DATE('25.05.2010','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0013', '1123', TO_DATE('20.06.2011','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0014', '1124', TO_DATE('02.08.2012','DD.MM.YYYY'))
INTO Qual_Assn5 VALUES ('F0015', '1125', TO_DATE('06.09.2005','DD.MM.YYYY'))
SELECT * FROM DUAL;

INSERT ALL
```

```
INTO Temp_Assn5 (CId) VALUES ('1111')
INTO Temp_Assn5 (CId) VALUES ('1112')
INTO Temp_Assn5 (CId) VALUES ('1113')
INTO Temp_Assn5 (CId) VALUES ('1114')
INTO Temp_Assn5 (CId) VALUES ('1115')
INTO Temp_Assn5 (CId) VALUES ('1116')
INTO Temp_Assn5 (CId) VALUES ('1117')
INTO Temp_Assn5 (CId) VALUES ('1118')
INTO Temp_Assn5 (CId) VALUES ('1119')
INTO Temp_Assn5 (CId) VALUES ('1120')
INTO Temp_Assn5 (CId) VALUES ('1121')
INTO Temp_Assn5 (CId) VALUES ('1122')
INTO Temp_Assn5 (CId) VALUES ('1123')
INTO Temp_Assn5 (CId) VALUES ('1124')
INTO Temp_Assn5 (CId) VALUES ('1125')
SELECT * FROM DUAL;
```



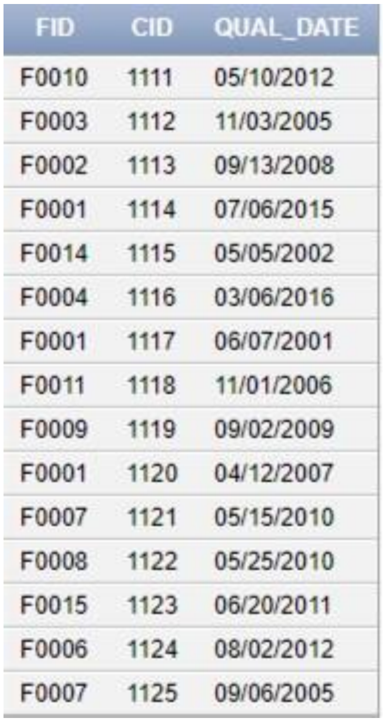
FACID	FNAME
F0001	John Mckormick
F0002	Ashley Tall
F0003	Henry Rockered
F0004	Frank Tisdally
F0005	Lenny Merch
F0006	Jane Languitch
F0007	Becca Brite
F0008	Luna Smith
F0009	Shawn Hart
F0010	Donald Forest
F0011	Lucy Grey
F0012	Melissa Makey
F0013	Sam Sworenson
F0014	Kylie Burnard
F0015	Lane Burtford



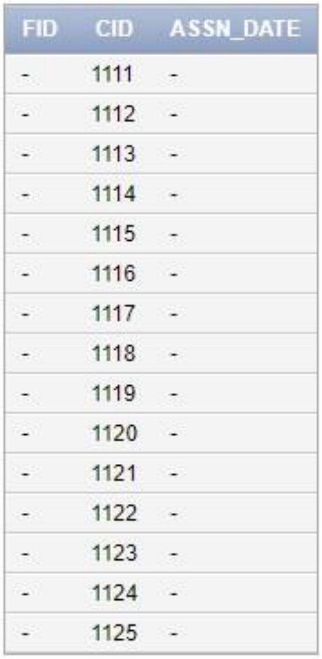
CRSEID	CNAME	FID	ASSN_DATE
1111	Math 1060	-	-
1112	Physc 3000	-	-
1113	CS 2810	-	-
1114	CS 3270	-	-
1115	Physc 2050	-	-
1116	Math 2600	-	-
1117	Math 3680	-	-
1118	ART 1000	-	-
1119	ART 1010	-	-
1120	CS 4060	-	-
1121	Math 4030	-	-
1122	ART 3070	-	-
1123	Physc 2030	-	-
1124	Physc 1010	-	-
1125	CS 3070	-	-

Faculty

Course



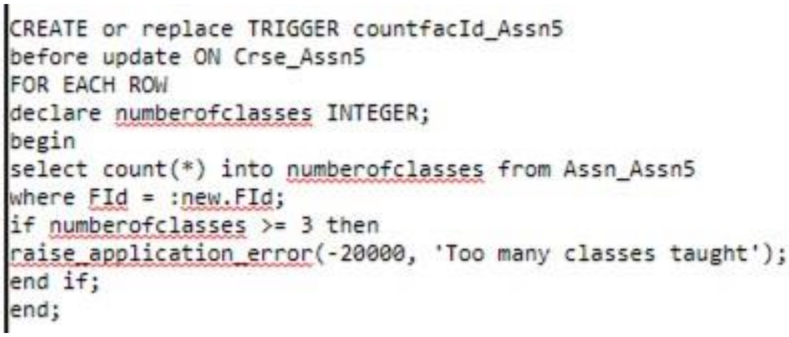
FID	CID	QUAL_DATE
F0010	1111	05/10/2012
F0003	1112	11/03/2005
F0002	1113	09/13/2008
F0001	1114	07/06/2015
F0014	1115	05/05/2002
F0004	1116	03/06/2016
F0001	1117	06/07/2001
F0011	1118	11/01/2006
F0009	1119	09/02/2009
F0001	1120	04/12/2007
F0007	1121	05/15/2010
F0008	1122	05/25/2010
F0015	1123	06/20/2011
F0006	1124	08/02/2012
F0007	1125	09/06/2005



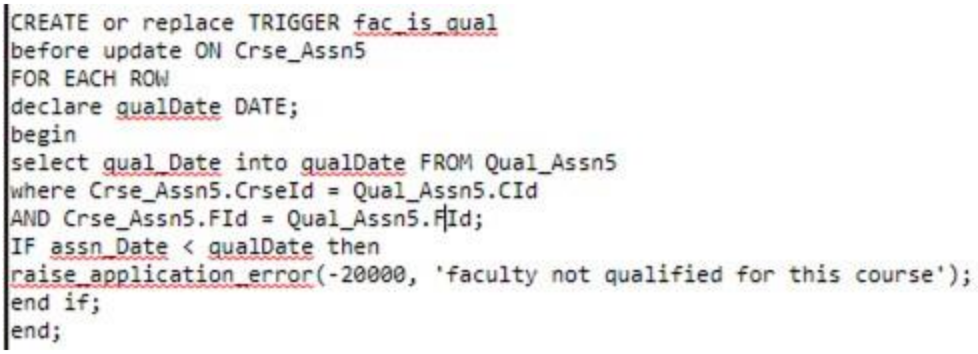
FID	CID	ASSN_DATE
-	1111	-
-	1112	-
-	1113	-
-	1114	-
-	1115	-
-	1116	-
-	1117	-
-	1118	-
-	1119	-
-	1120	-
-	1121	-
-	1122	-
-	1123	-
-	1124	-
-	1125	-

list of courses to be assigned

Triggers



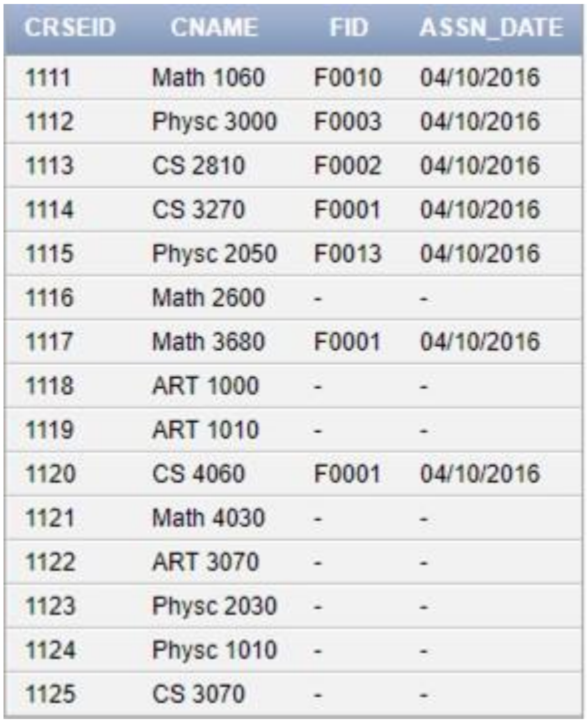
```
CREATE or replace TRIGGER countfacId_Assn5
before update ON Crse_Assn5
FOR EACH ROW
declare numberofclasses INTEGER;
begin
select count(*) into numberofclasses from Assn_Assn5
where FId = :new.FId;
if numberofclasses >= 3 then
raise_application_error(-20000, 'Too many classes taught');
end if;
end;
```



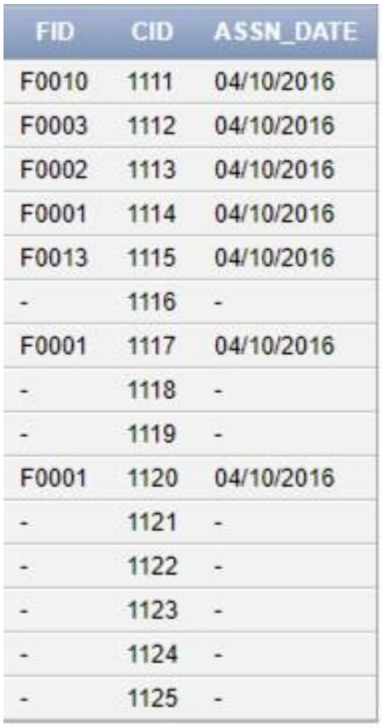
```
CREATE or replace TRIGGER fac_is_qual
before update ON Crse_Assn5
FOR EACH ROW
declare qualDate DATE;
begin
select qual_Date into qualDate FROM Qual_Assn5
where Crse_Assn5.CrseId = Qual_Assn5.CId
AND Crse_Assn5.FId = Qual_Assn5.FId;
IF assn_Date < qualDate then
raise_application_error(-20000, 'faculty not qualified for this course');
end if;
end;
```

Updating tables when faculty gets assigned to teach a course

UPDATE Temp_Assn5 SET FId = 'F0010', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1111';
UPDATE Crse_Assn5 SET FId = 'F0010', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1111';
UPDATE Temp_Assn5 SET FId = 'F0003', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1112';
UPDATE Crse_Assn5 SET FId = 'F0003', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1112';
UPDATE Temp_Assn5 SET FId = 'F0002', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1113';
UPDATE Crse_Assn5 SET FId = 'F0002', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1113';
UPDATE Temp_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1114';
UPDATE Crse_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1114';
UPDATE Temp_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1120';
UPDATE Crse_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1120';
UPDATE Temp_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1117';
UPDATE Crse_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1117';
UPDATE Temp_Assn5 SET FId = 'F0013', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1115';
UPDATE Crse_Assn5 SET FId = 'F0013', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1115';
UPDATE Temp_Assn5 SET FId = 'F0005', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1115';

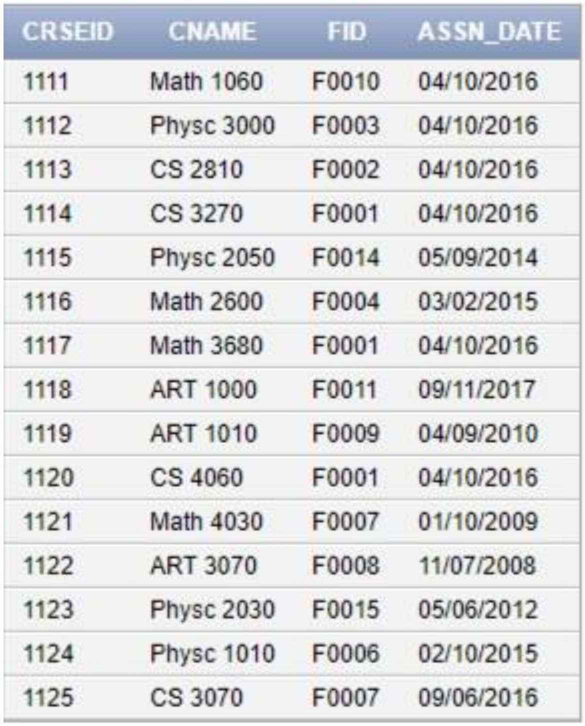


CRSEID	CNAME	FID	ASSN_DATE
1111	Math 1060	F0010	04/10/2016
1112	Physc 3000	F0003	04/10/2016
1113	CS 2810	F0002	04/10/2016
1114	CS 3270	F0001	04/10/2016
1115	Physc 2050	F0013	04/10/2016
1116	Math 2600	-	-
1117	Math 3680	F0001	04/10/2016
1118	ART 1000	-	-
1119	ART 1010	-	-
1120	CS 4060	F0001	04/10/2016
1121	Math 4030	-	-
1122	ART 3070	-	-
1123	Physc 2030	-	-
1124	Physc 1010	-	-
1125	CS 3070	-	-

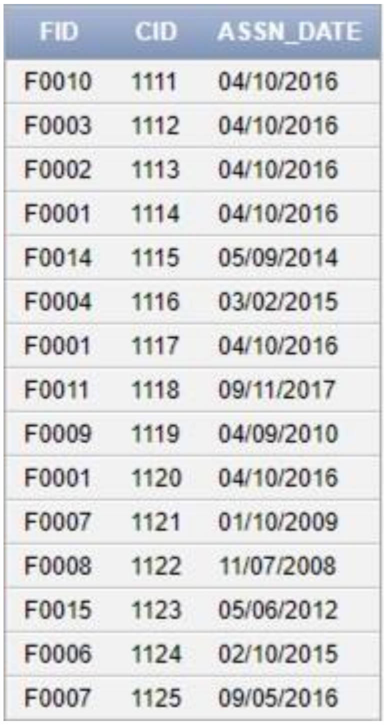


FID	CID	ASSN_DATE
F0010	1111	04/10/2016
F0003	1112	04/10/2016
F0002	1113	04/10/2016
F0001	1114	04/10/2016
F0013	1115	04/10/2016
-	1116	-
F0001	1117	04/10/2016
-	1118	-
-	1119	-
F0001	1120	04/10/2016
-	1121	-
-	1122	-
-	1123	-
-	1124	-
-	1125	-

```
UPDATE Crse_Assn5 SET FId = 'F0005', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1115';
UPDATE Temp_Assn5 SET FId = 'F0015', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1116';
UPDATE Crse_Assn5 SET FId = 'F0015', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1116';
UPDATE Temp_Assn5 SET FId = 'F0014', assn_Date = TO_DATE('05.09.2014') WHERE CId = '1115';
UPDATE Crse_Assn5 SET FId = 'F0014', assn_Date = TO_DATE('05.09.2014') WHERE CrseId = '1115';
UPDATE Temp_Assn5 SET FId = 'F0004', assn_Date = TO_DATE('03.02.2015') WHERE CId = '1116';
UPDATE Crse_Assn5 SET FId = 'F0004', assn_Date = TO_DATE('03.02.2015') WHERE CrseId = '1116';
UPDATE Temp_Assn5 SET FId = 'F0011', assn_Date = TO_DATE('09.11.2017') WHERE CId = '1118';
UPDATE Crse_Assn5 SET FId = 'F0011', assn_Date = TO_DATE('09.11.2017') WHERE CrseId = '1118';
UPDATE Temp_Assn5 SET FId = 'F0009', assn_Date = TO_DATE('04.09.2010') WHERE CId = '1119';
UPDATE Crse_Assn5 SET FId = 'F0009', assn_Date = TO_DATE('04.09.2010') WHERE CrseId = '1119';
UPDATE Temp_Assn5 SET FId = 'F0007', assn_Date = TO_DATE('01.10.2009') WHERE CId = '1121';
UPDATE Crse_Assn5 SET FId = 'F0007', assn_Date = TO_DATE('01.10.2009') WHERE CrseId = '1121';
UPDATE Temp_Assn5 SET FId = 'F0008', assn_Date = TO_DATE('11.07.2008') WHERE CId = '1122';
UPDATE Crse_Assn5 SET FId = 'F0008', assn_Date = TO_DATE('11.07.2008') WHERE CrseId = '1122';
UPDATE Temp_Assn5 SET FId = 'F0015', assn_Date = TO_DATE('05.06.2012') WHERE CId = '1123';
UPDATE Crse_Assn5 SET FId = 'F0015', assn_Date = TO_DATE('05.06.2012') WHERE CrseId = '1123';
UPDATE Temp_Assn5 SET FId = 'F0006', assn_Date = TO_DATE('02.10.2015') WHERE CId = '1124';
UPDATE Crse_Assn5 SET FId = 'F0006', assn_Date = TO_DATE('02.10.2015') WHERE CrseId = '1124';
UPDATE Temp_Assn5 SET FId = 'F0007', assn_Date = TO_DATE('09.05.2016') WHERE CId = '1125';
UPDATE Crse_Assn5 SET FId = 'F0007', assn_Date = TO_DATE('09.06.2016') WHERE CrseId = '1125';
```



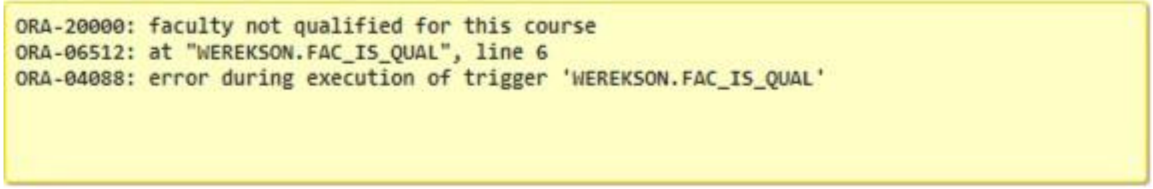
CRSEID	CNAME	FID	ASSN_DATE
1111	Math 1060	F0010	04/10/2016
1112	Physc 3000	F0003	04/10/2016
1113	CS 2810	F0002	04/10/2016
1114	CS 3270	F0001	04/10/2016
1115	Physc 2050	F0014	05/09/2014
1116	Math 2600	F0004	03/02/2015
1117	Math 3680	F0001	04/10/2016
1118	ART 1000	F0011	09/11/2017
1119	ART 1010	F0009	04/09/2010
1120	CS 4060	F0001	04/10/2016
1121	Math 4030	F0007	01/10/2009
1122	ART 3070	F0008	11/07/2008
1123	Physc 2030	F0015	05/06/2012
1124	Physc 1010	F0006	02/10/2015
1125	CS 3070	F0007	09/06/2016



FID	CID	ASSN_DATE
F0010	1111	04/10/2016
F0003	1112	04/10/2016
F0002	1113	04/10/2016
F0001	1114	04/10/2016
F0014	1115	05/09/2014
F0004	1116	03/02/2015
F0001	1117	04/10/2016
F0011	1118	09/11/2017
F0009	1119	04/09/2010
F0001	1120	04/10/2016
F0007	1121	01/10/2009
F0008	1122	11/07/2008
F0015	1123	05/06/2012
F0006	1124	02/10/2015
F0007	1125	09/05/2016

Operations Failed

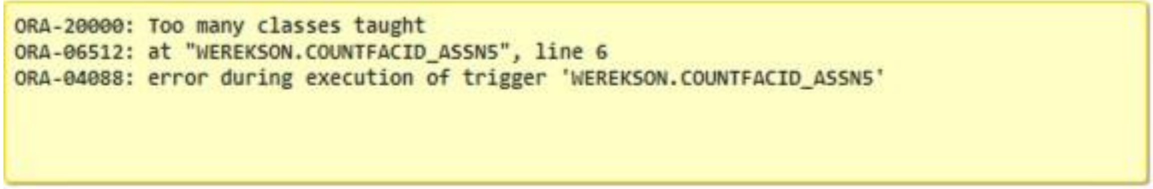
```
UPDATE Temp_Assn5 SET FId = 'F0004', assn_Date = TO_DATE('03.02.1999') WHERE CId = '1114';
UPDATE Crse_Assn5 SET FId = 'F0004', assn_Date = TO_DATE('03.02.1999') WHERE CrseId = '1114';
```



```
ORA-20000: faculty not qualified for this course
ORA-06512: at "WEREKSON.FAC_IS_QUAL", line 6
ORA-04088: error during execution of trigger 'WEREKSON.FAC_IS_QUAL'
```

The above course assignment failed because the faculty is not qualifies to teach course 1114.

```
UPDATE Temp_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CId = '1122';
UPDATE Crse_Assn5 SET FId = 'F0001', assn_Date = TO_DATE('04.10.2016') WHERE CrseId = '1122';
```

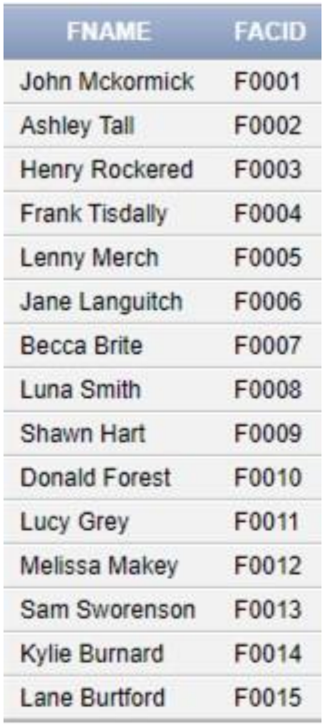


```
ORA-20000: Too many classes taught
ORA-06512: at "WEREKSON.COUNTFACID_ASSN5", line 6
ORA-04088: error during execution of trigger 'WEREKSON.COUNTFACID_ASSN5'
```

The above course assignment failed because the faculty has reached the limit 3.

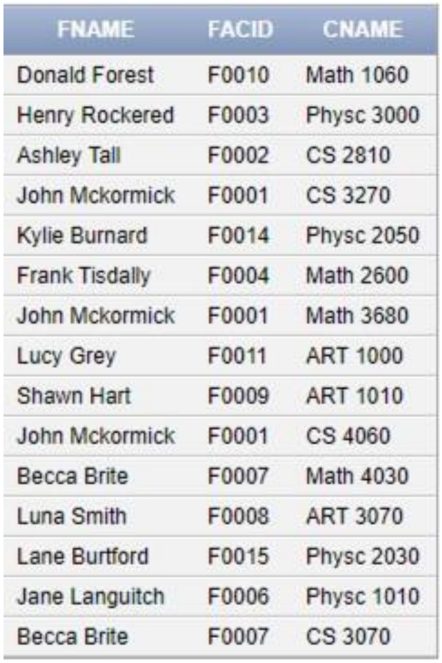
Additional Queries to Show Content of Tables after Course Assignments

```
SELECT F.FName, F.FacId FROM Fac_Assn5 f;
```



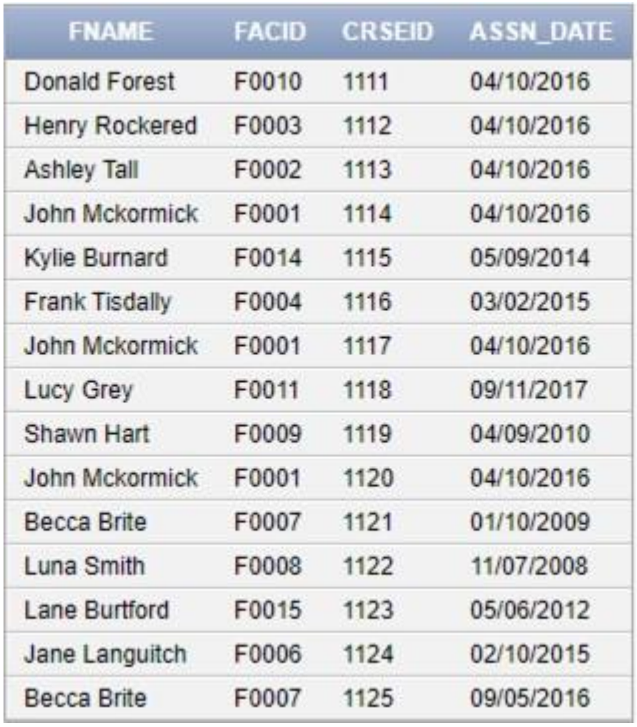
FNAME	FACID
John Mckormick	F0001
Ashley Tall	F0002
Henry Rockered	F0003
Frank Tisdally	F0004
Lenny Merch	F0005
Jane Languitch	F0006
Becca Brite	F0007
Luna Smith	F0008
Shawn Hart	F0009
Donald Forest	F0010
Lucy Grey	F0011
Melissa Makey	F0012
Sam Sworenson	F0013
Kylie Burnard	F0014
Lane Burtford	F0015

```
SELECT F.FName, F.FacId, c.CName FROM Fac_Assn5 f INNER JOIN Crse_Assn5 c ON c.FId = f.FacId;
```



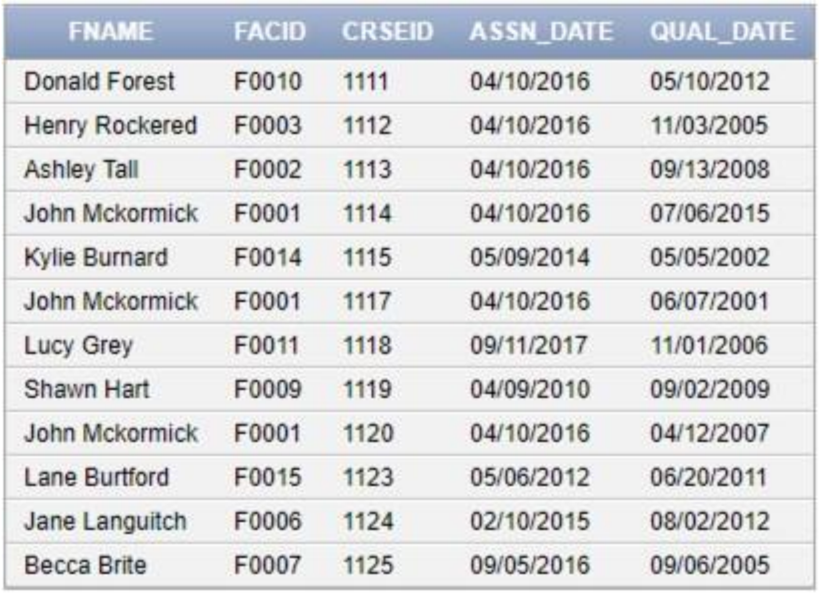
FNAME	FACID	CNAME
Donald Forest	F0010	Math 1060
Henry Rockered	F0003	Physc 3000
Ashley Tall	F0002	CS 2810
John Mckormick	F0001	CS 3270
Kylie Burnard	F0014	Physc 2050
Frank Tisdally	F0004	Math 2600
John Mckormick	F0001	Math 3680
Lucy Grey	F0011	ART 1000
Shawn Hart	F0009	ART 1010
John Mckormick	F0001	CS 4060
Becca Brite	F0007	Math 4030
Luna Smith	F0008	ART 3070
Lane Burtford	F0015	Physc 2030
Jane Languitch	F0006	Physc 1010
Becca Brite	F0007	CS 3070

```
SELECT F.FName, F.FacId, c.CrseId, a.assn_Date FROM Fac_Assn5 f, Crse_Assn5 c,
Temp_Assn5 a WHERE f.FacId = c.FId AND c.CrseId = a.CId;
```



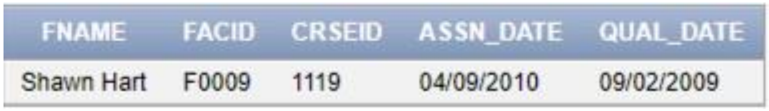
FNAME	FACID	CRSEID	ASSN_DATE
Donald Forest	F0010	1111	04/10/2016
Henry Rockered	F0003	1112	04/10/2016
Ashley Tall	F0002	1113	04/10/2016
John Mckormick	F0001	1114	04/10/2016
Kylie Burnard	F0014	1115	05/09/2014
Frank Tisdally	F0004	1116	03/02/2015
John Mckormick	F0001	1117	04/10/2016
Lucy Grey	F0011	1118	09/11/2017
Shawn Hart	F0009	1119	04/09/2010
John Mckormick	F0001	1120	04/10/2016
Becca Brite	F0007	1121	01/10/2009
Luna Smith	F0008	1122	11/07/2008
Lane Burtford	F0015	1123	05/06/2012
Jane Languitch	F0006	1124	02/10/2015
Becca Brite	F0007	1125	09/05/2016

```
SELECT F.FName, F.FacId, c.CrseId, a.assn_Date, q.qual_Date FROM Fac_Assn5 f,
Crse_Assn5 c, Temp_Assn5 a, Qual_Assn5 q
WHERE f.FacId = c.FId AND c.CrseId = a.CId AND c.CrseId = q.CId;
```



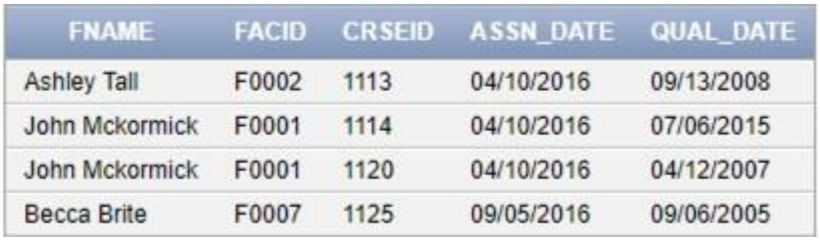
FNAME	FACID	CRSEID	ASSN_DATE	QUAL_DATE
Donald Forest	F0010	1111	04/10/2016	05/10/2012
Henry Rockered	F0003	1112	04/10/2016	11/03/2005
Ashley Tall	F0002	1113	04/10/2016	09/13/2008
John Mckormick	F0001	1114	04/10/2016	07/06/2015
Kylie Burnard	F0014	1115	05/09/2014	05/05/2002
John Mckormick	F0001	1117	04/10/2016	06/07/2001
Lucy Grey	F0011	1118	09/11/2017	11/01/2006
Shawn Hart	F0009	1119	04/09/2010	09/02/2009
John Mckormick	F0001	1120	04/10/2016	04/12/2007
Lane Burtford	F0015	1123	05/06/2012	06/20/2011
Jane Languitch	F0006	1124	02/10/2015	08/02/2012
Becca Brite	F0007	1125	09/05/2016	09/06/2005

```
SELECT F.FName, F.FacId, c.CrseId, a.assn_Date, q.qual_Date FROM Fac_Assn5 f,
Crse_Assn5 c, Temp_Assn5 a, Qual_Assn5 q
WHERE f.FacId = c.FId AND c.CrseId = a.CId AND c.CrseId = q.CId
AND f.FacId = 'F0009';
```



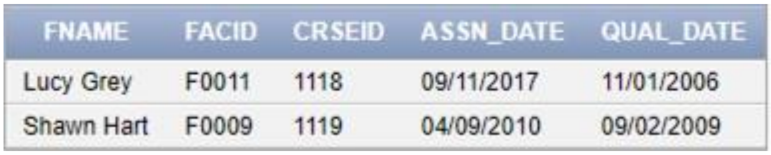
FNAME	FACID	CRSEID	ASSN_DATE	QUAL_DATE
Shawn Hart	F0009	1119	04/09/2010	09/02/2009

```
SELECT F.FName, F.FacId, c.CrseId, a.assn_Date, q.qual_Date FROM Fac_Assn5 f,
Crse_Assn5 c, Temp_Assn5 a, Qual_Assn5 q
WHERE f.FacId = c.FId AND c.CrseId = a.CId AND c.CrseId = q.CId
AND c.CName  LIKE '%CS%';
```



FNAME	FACID	CRSEID	ASSN_DATE	QUAL_DATE
Ashley Tall	F0002	1113	04/10/2016	09/13/2008
John Mckormick	F0001	1114	04/10/2016	07/06/2015
John Mckormick	F0001	1120	04/10/2016	04/12/2007
Becca Brite	F0007	1125	09/05/2016	09/06/2005

```
SELECT F.FName, F.FacId, c.CrseId, a.assn_Date, q.qual_Date FROM Fac_Assn5 f,
Crse_Assn5 c, Temp_Assn5 a, Qual_Assn5 q
WHERE f.FacId = c.FId AND c.CrseId = a.CId AND c.CrseId = q.CId
AND c.CName  LIKE '%ART%';
```



FNAME	FACID	CRSEID	ASSN_DATE	QUAL_DATE
Lucy Grey	F0011	1118	09/11/2017	11/01/2006
Shawn Hart	F0009	1119	04/09/2010	09/02/2009

Conclusion

In this paper, we have used the ER Diagram to represent some business rules graphically at conceptual level in a relational data model to enforce database consistency. The constraints that have been represented in this paper have so far focused on the relationships between entities and could be characterized as existence constraints. These constraints make sure that faculty will be assigned only courses that he/she is qualified to teach and he/she will not be assigned more than 3 courses to teach. Triggers are written for course assignments and have been tested to make sure these constraints will not be violated. Yet, business rules are not limited to only these types of constraints. The future of this work will be to examine other types of constraints to ensure that there is a method of representing and enforcing the other type of constraint.

Our students in our database course get a number of hands-on assignments of this kind. They find them a little bit challenging but they really enjoy doing this kind of work.

References:

1. J. A. Hoffer, M. B. Prescott and F. R. McFadden, “Modern Database Management”, Seventh Edition, Prentice Hall, 2005.
2. A. Perkins, “Business Rules = Meta Data”, the proceedings of the: Technology of Object-Oriented Languages and Systems, IEEE, 2000.
3. Catherine M. Ricardo, “Database Illuminated”, Third Edition, Jones & Bartlett Learning, 2017.
4. Mullins C. S. 1995, “The procedural DBA.” *Database Programming & Design 812 (December)*: 40-45.
5. Rennhackkamp, M 1996, “Trigger Happy.” DBMS9,5 (May): 89-91, 95.
6. “Using Triggers and Compound Triggers in Oracle 11g”, https://dbanotes.com/using-triggers-and-compound-triggers-in-oracle-11g-3ebb050867cc.