08-11-19, group 5

members: Dong Yunxing, Khalisah binte Faroukh,Taylor Trujillo, Thijs Bolscher, Wang Yilin

Lab 5, CZ2007

```
/*create table*/
CREATE TABLE State
(Name VARCHAR(40) PRIMARY KEY,
);

CREATE TABLE City
(Name VARCHAR(40),
StateName VARCHAR(40),
PRIMARY KEY (Name,StateName),
FOREIGN KEY (StateName) REFERENCES State(Name) ON UPDATE CASCADE ON
DELETE SET NULL,
);


CREATE TABLE Laboratory (
LName varchar(40),
LSchool varchar(40),
Location varchar(40),
PRIMARY KEY (LName, LSchool),
CONSTRAINT Location_default  DEFAULT 'Default_Location' FOR Location
);

CREATE TABLE People
(PersonID int PRIMARY KEY,
Address VARCHAR(100) NOT NULL,
School VARCHAR(100),
Email VARCHAR(80),
PhoneNo char(20),
FOREIGN KEY (Email) REFERENCES Email_T(Email) ON UPDATE CASCADE ON
DELETE SET NULL,
FOREIGN KEY (PhoneNo) REFERENCES Phone(PhoneNo) ON UPDATE CASCADE ON
DELETE SET NULL,
FOREIGN KEY (Address) REFERENCES Address_T(Address) ON UPDATE CASCADE ON
DELETE SET NULL

);

CREATE TABLE Address_T
(Address VARCHAR(100) PRIMARY KEY,
Zip INT NOT NULL ,
CityName VARCHAR(40),
StateName VARCHAR(40),
FOREIGN KEY (CityName,StateName) REFERENCES City(Name,StateName) ON
UPDATE CASCADE ON DELETE SET NULL,
CONSTRAINT checkzip CHECK(Zip>99999 AND Zip<1000000),
);

CREATE TABLE Email_T
(Email VARCHAR(80) PRIMARY KEY,
 Name VARCHAR(50) NOT NULL,
);

CREATE TABLE Phone
(PhoneNo CHAR(20) PRIMARY KEY,
Name VARCHAR(50) NOT NULL,
);

CREATE TABLE Equipment (
ID int,
LName VARCHAR(40),
LSchool VARCHAR(40),
Model_No  VARCHAR(40)
DatePurchased DATE,
PRIMARY KEY (ID, LName, LSchool),
FOREIGN KEY (Lname, LSchool) REFERENCES Laboratory(LName, LSchool) ON
UPDATE CASCADE ON DELETE SET NULL,
FOREIGN KEY (Model_No) REFERENCES Model （Model_No） ON UPDATE CASCADE
ON DELETE SET NULL,
);

CREATE TABLE Model(
ModelNo VARCHAR(80) PRIMARY KEY,
Name VARCHAR(50) NOT NULL,
CONSTRAINT ModelName_default DEFAULT 'Default_Model_Name' FOR ModelName;
);

CREATE TABLE ResearchLab (
LName varchar(40),
LSchool varchar(40),
FOREIGN KEY (LName, LSchool) REFERENCES Laboratory(LName, LSchool) ON
UPDATE CASCADE ON DELETE SET NULL,
PRIMARY KEY (LName, LSchool)
);

CREATE TABLE TeachingLab (
LName varchar(40),
LSchool varchar(40),
FOREIGN KEY (LName, LSchool) REFERENCES Laboratory(LName, LSchool) ON
UPDATE CASCADE ON DELETE SET NULL,
PRIMARY KEY (LName, LSchool)
);

CREATE TABLE Student
(PersonID INT PRIMARY KEY,
Major_Minor NCHAR(50) NOT NULL,
Admission_Date DATE,
StudentID int,
);

CREATE TABLE Experiments (
StudentID int,
LName varchar(40),
LSchool varchar(40),
DateAndTime datetime,
Attendance bit,
FOREIGN KEY (LName, LSchool) REFERENCES Laboratory(LName, LSchool) ON
UPDATE CASCADE ON DELETE SET NULL,
PRIMARY KEY (LName, LSchool, StudentID, DateAndTime),
);

CREATE TABLE Undergraduates (
StudentID int NOT NULL,
PRIMARY KEY (StudentID)
);

CREATE TABLE Professors
(PersonID INT PRIMARY KEY,
Field_of_Expertise VARCHAR(40),
FOREIGN KEY (PersonID) REFERENCES Professors(PersonID) ON UPDATE CASCADE
ON DELETE SET NULL,
);

CREATE TABLE Course
(cIndex VARCHAR(40) PRIMARY KEY,
date_time DATETIME,
);

CREATE TABLE Take
(PersonID INT PRIMARY KEY,
cIndex VARCHAR(40),
FOREIGN KEY (PersonID) REFERENCES Student(PersonID) ON UPDATE CASCADE ON
DELETE SET NULL,
FOREIGN KEY (cIndex) REFERENCES Course(cIndex) ON UPDATE CASCADE ON
DELETE SET NULL,
);

Create table Staff
(PersonID int,
Position varchar(40) not null,
DateHired date not null,
Primary key (PersonID),
FOREIGN KEY (PersonID) references People(PersonID) ON UPDATE CASCADE ON
DELETE SET NULL,
);

Create table Admin
(PersonID int PRIMARY KEY,
foreign key (PersonID) references Staff(PersonID) ON UPDATE CASCADE ON DELETE
SET NULL,
);

Create table TechnicalStaff
(PersonID int,
LabName varchar(40) not null,
LabSchool varchar(40) not null,
Primary key (PersonID),
foreign key (PersonID) references Staff(PersonID) ON UPDATE CASCADE ON DELETE
SET NULL,
foreign key (LabName, LabSchool) references Laboratory(LName, LSchool) ON UPDATE
CASCADE ON DELETE SET NULL,
);

Create table Stakeholders
(PersonID int,
primary key (PersonID),
foreign key (PersonID) references People(PersonID) ON UPDATE CASCADE ON DELETE
SET NULL,
);

Create table CommentsSuggestions
(CsID int,
Dateandtime Datetime not null,
Topic varchar(40) not null,
StakeholderID int not null,
primary key (CsID),
foreign key (StakeholderID) references Stakeholders(PersonID) ON UPDATE CASCADE ON
DELETE SET NULL,
);
```

```
CREATE TABLE Graduates(
	StudentID int,
	PRIMARY KEY (StudentID)
);

CREATE TABLE Teach (
	PersonID int NOT NULL,
	Date_Time nchar(50),
	IndexNumber int NOT NULL,
	PRIMARY KEY (PersonID, Date_Time, IndexNumber),
	FOREIGN KEY(PersonID) REFERENCES Professors(PersonID) ON UPDATE
CASCADE ON DELETE SET NULL,
);
CREATE TABLE Research (
	Topic varchar(40),
	ProfessorID int,
	StudentID int,
	LName varchar(40),
	LSchool varchar(40),
	PRIMARY KEY(TOPIC, ProfessorID, StudentID,LName, LSchool),
   FOREIGN KEY (ProfessorID) REFERENCES Professors(PersonID) ON UPDATE
CASCADE ON DELETE SET NULL,
	FOREIGN KEY (LName,LSchool) REFERENCES Laboratory(LName,LSchool) ON
UPDATE CASCADE ON DELETE SET NULL
);


/*trigger part*/
CREATE TRIGGER [dbo].[StudentIDCheck1] on [dbo].[Experiments]
INSTEAD OF INSERT
AS
BEGIN
	IF NOT EXISTS(
	   SELECT *
		FROM inserted
		WHERE StudentID IN (SELECT StudentID FROM [dbo].[Student])
	)
	PRINT 'STUDENT ID IS INVALID'
	ELSE
	INSERT INTO dbo.Experiments SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[StudentIDCheck2] on [dbo].[Undergraduates]
INSTEAD OF INSERT
AS
```

```
BEGIN
        IF NOT EXISTS(
           SELECT *
                FROM inserted
                WHERE StudentID IN (SELECT StudentID FROM [dbo].[Student])
        )
        PRINT 'STUDENT ID IS INVALID'
        ELSE
        INSERT INTO dbo.Undergraduates SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[StudentIDCheck3] on [dbo].[Graduates]
INSTEAD OF INSERT
AS
BEGIN
        IF NOT EXISTS(
           SELECT *
                FROM inserted
                WHERE StudentID IN (SELECT StudentID FROM [dbo].[Student])
        )
        PRINT 'STUDENT ID IS INVALID'
        ELSE
        INSERT INTO dbo.Graduates SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[StudentIDCheck4] on [dbo].[Research]
INSTEAD OF INSERT
AS
BEGIN
        IF NOT EXISTS(
           SELECT *
                FROM inserted
                WHERE StudentID IN (SELECT StudentID FROM [dbo].[Student])
        )
        PRINT 'STUDENT ID IS INVALID'
        ELSE
        INSERT INTO dbo.Research SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[staffidCheck] on [dbo].[Staff]
INSTEAD OF INSERT
AS
BEGIN
   IF EXISTS(
      SELECT StaffID
      FROM (
        SELECT PersonID, StaffID
        FROM dbo.Staff
        UNION ALL
        SELECT PersonID         , StaffID
        FROM INSERTED) AS T
        GROUP BY StaffID
        HAVING COUNT(DISTINCT PersonID)>1
    )
    PRINT 'FD does not permit!'
    ELSE
    INSERT INTO dbo.Staff SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[StudentPersonIDCheck] on [dbo].[Student]
INSTEAD OF INSERT
AS
BEGIN
        IF EXISTS(
                SELECT StudentID
                FROM (
                SELECT StudentID, PersonID
                FROM dbo.Student
                UNION ALL
                SELECT StudentID, PersonID
                FROM INSERTED) AS T
                GROUP BY StudentID
                HAVING COUNT(DISTINCT PersonID)>1
        )
        PRINT 'FD does not permit'
        ELSE
        INSERT INTO dbo.Student SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[TechnicalStaffToOneLab] on [dbo].[TechnicalStaff]
INSTEAD OF INSERT
AS
BEGIN
        IF EXISTS(
                SELECT PersonID
                FROM TechnicalStaff
                INTERSECT
                SELECT PersonID
                FROM inserted
        )
        PRINT 'There is already Technical Staff assigned'
        ELSE
        INSERT INTO dbo.TechnicalStaff SELECT * FROM inserted
END;

CREATE TRIGGER [dbo].[ClassClash] on [dbo].[Take]
INSTEAD OF INSERT
AS
BEGIN
        IF EXISTS(
           SELECT *
                 FROM INSERTED I, Course C
                 WHERE I.cIndex=C.cIndex
                 AND C.date_time IN (SELECT date_time
                 FROM Course c, Take t
                 WHERE c.cIndex=t.cIndex
           AND I.PersonID=t.PersonID)
        )
        PRINT 'class clash'
        ELSE
        INSERT INTO dbo.Take SELECT * FROM inserted
END;
```

/*query*/

1. Find all Stakeholders who belong to the public domain.

   ```
   SELECT Email_T.Name
   From People, Stakeholders, Email_T
   Where Stakeholders.domain = 'public' AND Stakeholders.PersonID =
   People.PersonID AND People.Email = Email_T.Email
   ```

2. Find all Stakeholders who have provided at least five comments or suggestions

   ```
   Select DISTINCT Email_T.Name
   From People, Stakeholders, Email_T
   Where People.PersonID IN (
   Select C.PersonID
   From CommentsSuggestions as C
   Group By C.PersonID
   HAVING COUNT(C.PersonID) > 5
   ) AND People.Email = Email_T.Email
   ```

3. Find Graduates who are supervised by more than one professor and assigned to more than one research laboratory.

   ```
   Select Email_T.Name
   From People, Graduates, Student, Email_T
   ```

```
Where People.PersonID = Student.PersonID AND People.Email = Email_T.Email
AND
Graduates.StudentID = Student.StudentID AND
Graduates.StudentID in (
        (Select R.StudentID
        From Research R
        Group By R.StudentID
        Having COUNT (DISTINCT R.ProfessorID) > 1)
        INTERSECT
        SELECT A.StudentID FROM (SELECT DISTINCT StudentID, LName,
LSchool
        FROM Research) AS A
        GROUP BY StudentID
        HAVING COUNT(LName) > 1
)
```

4. Find all Professors who teach more than one courses in the semester.

```
SELECT E.Name
FROM People P, Email_T E,
(SELECT DISTINCT Professor_PersonID AS PersonID
FROM Teach
GROUP BY Professor_PersonID
HAVING COUNT(DISTINCT courseIndex)>1) AS A
WHERE A.PersonID=P.PersonID AND E.Email = P.Email
```

5. List all the Equipment belonging to a particular Laboratory.

```
SELECT E.LName, E.LSchool, E.ID, M.ModelName
FROM  Model AS M JOIN Equipment AS E ON M.Model_No = E.Model_No
WHERE E.LName = 'SWLab1' AND E.LSchool = 'SCSE'
```

6. Find all Undergraduates who have not attended at least one laboratory experiments.

```
    /*if it is refer to both students who don't take attendance and who don't take
lab*/
    SELECT E.Name
    FROM People P, Email_T E
    WHERE  E.Email = P.Email
    AND P.PersonID IN ( SELECT StudentID
                        FROM Undergraduates
                        EXCEPT
                        SELECT DISTINCT StudentID
```

```
                              FROM Experiments
                              WHERE Attendance = 1)

/*if it only refer to students who don't take attendance*/
SELECT E.Name
FROM People P, Email_T E
WHERE  E.Email = P.Email
AND P.PersonID IN (
SELECT DISTINCT StudentID
FROM Experiments
WHERE Attendance = 0 )
```

7. List all Graduates who are doing research and taking courses in the semester

```
SELECT E.Name
FROM People P, Email_T E
WHERE  E.Email = P.Email
AND P.PersonID IN (SELECT StudentID
FROM Research
WHERE StudentID in (SELECT StudentID
FROM Graduates)
INTERSECT
SELECT S.StudentID
FROM Take T, Student S
WHERE T.PersonID = S.PersonID AND S.StudentID in (SELECT StudentID
FROM Graduates))
```