SQL – Part 2

Data Manipulation Language (Insert, Update, Delete)

Data Manipulation Language (DML)

- Data Manipulation Language
  - INSERT
  - UPDATE
  - DELETE
  - SELECT

Data Manipulation Language – Insert, Update, Delete

- The INSERT statement is used to add tuples into a relation.

```
INSERT INTO table_name
            [(attribute_name,...,attribute_name)]
     VALUES (value,...,value),...,(value,...,value);
```

- The UPDATE statement is used to modify attribute values of one or more selected tuples.

```
UPDATE table_name
   SET attribute_name = value,...,attribute_name = value
[WHERE selection_condition];
```

- The DELETE statement is used to remove tuples from a relation.

```
DELETE FROM table_name
     [WHERE selection_condition];
```

Insert - Examples

- The following three ways of inserting tuples into the relation STUDENT are equivalent.

```
INSERT INTO Student
     VALUES (456, 'Tom', '25/01/1988', 'tom@gmail.com'),
            (458, 'Peter', '20/02/1991', 'peter@hotmail.com');

INSERT INTO Student(Name, StudentID, DoB, Email)
     VALUES ('Tom', 456, '25/01/1988', 'tom@gmail.com'),
            ('Peter', 458, '20/02/1991', 'peter@hotmail.com');

INSERT INTO Student
     VALUES (456, 'Tom', '25/01/1988', 'tom@gmail.com');
INSERT INTO Student
     VALUES (458, 'Peter', '20/02/1991', 'peter@hotmail.com');
```

Insert - Primary Key Violation

- Suppose that we have the relation STUDENT with the primary key on StudentID:

StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com
...	...	...	...

- What would happen if we try to recycle Tom's StudentID?

```
INSERT INTO STUDENT(StudentID, Name, DoB, Email)
     VALUES (456, 'Smith', '27/08/1989', 'smith@gmail.com');
```

- DBMSs will not allow two tuples with the same primary key value in STUDENT.

Insert - Foreign Key Violation

- Consider the relations STUDENT, and ENROL with the foreign key [StudentID]$\subseteq$ STUDENT[StudentID].

StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com
459	Fran	11/09/1987	frankk@gmail.com

- If we only have the above three tuples in STUDENT, can we add the following tuple into ENROL?

```
INSERT INTO Enrol(StudentID, CourseNo, Semester, Status)
     VALUES (460, 'COMP2400', '2016 S2', 'active');
```

- Again, DBMSs will not allow a tuple in ENROL which has a student ID not appearing in any tuples of STUDENT due to the foreign key [StudentID]$\subseteq$ STUDENT[StudentID] on ENROL.

Update and Delete - Examples

- If we want to change Tom's email and name stored in the relation STUDENT, then we use

```
UPDATE Student
   SET Name='Tom Lee', Email='tom.lee@yahoo.com'
   WHERE StudentID=456;
```

- If we want to delete Tom's information from the relation STUDENT, we use

```
DELETE FROM Student WHERE StudentID=456;
```

- We can delete all the tuples in the relation STUDENT by using

```
DELETE FROM Student;
```

- **Question:** *What is the difference between the above statement and the following one?*

```
DROP Table Student;
```

- **Answer:** **The table STUDENT (empty) exists after the first statement, but would disappear if applying the second one.**

Update and Delete - Referential Actions

- Referential actions specify what happens in case of deleting or updating referenced tuples (via foreign key constraints).
- SQL offers the following possibilities:
  - `NO ACTION` (default) will throw an error if one tries to delete a row (or update the primary key value) referenced.
  - `CASCADE` will force the referencing tuples to be deleted (or updated with new primary key value).
  - `SET NULL` will force the corresponding values in the referencing tuples to be set to a null value (i.e., unknown).
  - `SET DEFAULT` will force the corresponding values in the referencing tuples to be set to a specified default value.

Referential Actions – Foreign Key

```
CREATE TABLE Student
       ( StudentID  INT PRIMARY KEY,
        Name VARCHAR(50),
        DoB Date,
        Email VARCHAR(100));
CREATE TABLE Course
       (No VARCHAR(20) PRIMARY KEY,
        Cname VARCHAR(50),
        Unit SMALLINT);
CREATE TABLE Enrol
       ( StudentID  INT,
        CourseNo VARCHAR(20),
        Semester VARCHAR(50),
        Status VARCHAR(50),
        FOREIGN KEY(StudentID) REFERENCES Student(StudentID)
        ON DELETE NO ACTION ,
        FOREIGN KEY(CourseNo) REFERENCES Course(No));
```

Referential Actions - Examples

- Consider the following foreign key defined on ENROL:

```
FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)
ON DELETE NO ACTION
```

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP1130	2016 S1	active	25/02/2016
458	COMP1130	2016 S1	active	25/02/2016
456	COMP2400	2016 S2	active	09/03/2016

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com

- The deletion of a student who has enrolled at least one course will throw out an error concerning the foreign key.

Referential Actions - Examples

- Consider the following foreign key defined on ENROL:

```
FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)
ON DELETE CASCADE
```

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP1130	2016 S1	active	25/02/2016
458	COMP1130	2016 S1	active	25/02/2016
456	COMP2400	2016 S2	active	09/03/2016

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com

- Deleting a student in STUDENT will also delete all of his enrolled courses in ENROL. We would have ENROL below after deleting the student 456.

StudentID	CourseNo	Semester	Status	EnrolDate
458	COMP1130	2016 S1	active	25/02/2016