CSC343 Worksheet 5 Solution

June 19, 2020

1. a)

```
CREATE TABLE MovieExec (
    name CHAR(30),
    address VARCHAR(255),
    cert# INT PRIMARY KEY,
    FOREIGN KEY (cert#) REFERENCES Movies(producerC#)
);

```

Example:

- Foreign-key
  - **Syntax 1:** FOREIGN KEY (< attributes >) REFERENCES < table >(< attributes >)
  - **Syntax 2:** REFERENCES < table >(< attributes >)
  - Binds an attribute of one relation to an anttribute in another table
  - Added when creating table

  Example:

```
// Example 1
CREATE TABLE Studio (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    presC# INT REFERENCES MovieExeC(cert#)
);

// Example 2
CREATE TABLE Studio (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    presC# INT,
    FOREIGN KEY (presC#) REFERENCES MovieExec(cert#)
);

```

b)
```
CREATE TABLE Movies (
    title CHAR(30) PRIMARY KEY,
    year INT PRIMARY KEY,
    length INT,
    genre VARCHAR(255),
    studioName VARCHAR(255),
    producerC# PRIMARY KEY
);
```

c) No change required. Violation occurs by the default policy.

```
CREATE TABLE MovieExec (
    name CHAR(30),
    address VARCHAR(255),
    cert# INT PRIMARY KEY,
    FOREIGN KEY (cert#) REFERENCES Movies(producerC#)
);
```

Correct Solution:

```
CREATE TABLE MovieExec (
    name CHAR(30),
    address VARCHAR(255),
    cert# INT PRIMARY KEY,
    FOREIGN KEY (cert#) REFERENCES Movies(producerC#)
        ON UPDATE CASCADE // Correction
        ON DELETE CASCADE // Correction
);
```

Notes:

- Maintaining Referential Integrity
  - Three different types of policies exist on Foreign Key
    1. *The Default Policy: Reject Violating Modifications.*
       * Is default policy
       * Rejects any modification violating referential integrity constant
    2. *The Cascade Policy*
       * Changes to the referenced attributes are mimicked at foreign key.
       * e.g. delete a tuple in **MovieExec**, deletes related referencing tuple(s) from **Studio**
    3. *The Set-Null Policy*
       * When a modification to the referenced relation affects a foreign-key value, the latter is changed to NULL.

* This applies to both UPDATE and DELETE

Example:

```
    CREATE TABLE Movies (
        title CHAR(30) PRIMARY KEY,
        year INT PRIMARY KEY,
        length INT,
        genre VARCHAR(255),
        studioName VARCHAR(255),
        producerC# REFERENCES MovieExec(cert#)
            ON DELETE SET NULL
            ON UPDATE CASCADE
    );
```

d)

```
    CREATE TABLE Movies (
        title CHAR(30) PRIMARY KEY,
        year INT PRIMARY KEY,
        length INT,
        genre VARCHAR(255),
        studioName VARCHAR(255),
        producerC# VARCHAR(255)
        FOREIGN KEY (title) REFERENCES StarsIn(movieTitle)
    );
```

e)

```
    CREATE TABLE StarsIn (
        movieTitle CHAR(30) PRIMARY KEY,
        movieYear INT PRIMARY KEY,
        starName VARCHAR(255) PRIMARY KEY,
        FOREIGN KEY (starName) REFERENCES MovieStar(name)
            ON DELETE CASCADE
    );
```

2. Yes. Set foreign-key constraint on StarsIn's movietitle to Movie's title.

```
    CREATE TABLE Movies (
        title CHAR(30) PRIMARY KEY,
        year INT PRIMARY KEY,
        length INT,
        genre VARCHAR(255),
        studioName VARCHAR(255),
        producerC# VARCHAR(255),
        FOREIGN KEY (title) REFERENCES StarsIn(movieTitle)
    );
```

3.

```sql
CREATE TABLE Product (
    maker CHAR(30),
    model INT PRIMARY KEY,
    type VARCHAR(255)
);

CREATE TABLE PC (
    model INT PRIMARY KEY,
    speed FLOAT,
    ram INT,
    hd INT,
    price FLOAT,
    FOREIGN KEY (model) REFERENCES Product(model)
);

CREATE TABLE Laptop (
    model INT PRIMARY KEY,
    speed FLOAT,
    ram INT,
    hd INT,
    screen INT,
    price FLOAT,
    FOREIGN KEY (model) REFERENCES Product(model)
);

CREATE TABLE Printer (
    model INT PRIMARY KEY,
    color BOOLEAN,
    type VARCHAR(255),
    price FLOAT,
    FOREIGN KEY (model) REFERENCES Product(model)
);


```