Name: VOEUN Rompheu

ID: e20211437

# Exercise Constraint and FD

I. The database movie is defined as follow:

Movies (title, year, length, genre, studioName, #producerC#)
MovieStar (name, address, gender, birthdate)
Starsln (movieTitle, movieYear, starName)
MovieExec (name, address, cert#, netWorth).
Studio (name, address, presC#)

**Movies**

Its key is title and year together. The attribute studioName tells us the studio that owns the movie, and producerC# is an integer that represents the producer of the movie in a way that we shall discuss when we talk about the relation MovieExec below.

**MovieStar**

This relation tells us something about stars. The key is name, the name of the movie star. It is not usual to assume names of persons are unique and therefore suitable as a key. However, movie stars are different; one would never take a name that some other movie star had used. Thus, we shall use the convenient fiction that movie-star names are unique. A more conventional approach would be to invent a serial number of some sort, like social-security numbers, so that we could assign each individual a unique number and use that attribute as the key. We take that approach for movie executives, as we shall see. Another interesting point about the MovieStar relation is that we see two new data types. The gender can be a single character, M or F. Also, birthdate is of type “date,” which might be a character string of a special form.

**Starsln**

This relation connects movies to the stars of that movie, and likewise connects a star to the movies in which they appeared. Notice that movies are represented by the key for Movies — the title and year — although we have chosen different attribute names to emphasize that attributes movieTitle and movieYear represent the movie. Likewise, stars are represented by the key for MovieStar, with the attribute called starName. Finally, notice that all three attributes are necessary to form a key. It is perfectly reasonable to suppose that relation Starsln could have two distinct tuples that agree in any two of the three attributes. For instance, a star might appear in two movies in one year, giving rise to two tuples that agreed in movieYear and starName, but disagreed in movieTitle.

**MovieExec**

This relation tells us about movie executives. It contains their name, address, and networth as data about the executive. However, for a key we have invented "certificate numbers" for all movie executives, including producers (as appear in the relation Movies) and studio presidents (as appear in the relation Studio, below). These are integers; a different one is assigned to each executive.

**Studio**

This relation tells about movie studios. We rely on no two studios having the same name, and therefore use name as the key. The other attributes are the address of the studio and the certificate number for the president of the studio. We assume that the studio president is surely a movie executive and therefore appears in MovieExec.

**Questions**

1. Suggest suitable primary and foreign keys for the relations of the movie database.
2. Declare the following referential integrity constraints for the movie database. (CASCADE, RESTRICT, NULLIFIES)

    a. The producer of a movie must be someone mentioned in MovieExec. Modifications to MovieExec that violate this constraint are rejected.
    b. Repeat (a), but violations result in the producerC# in Movie being set to NULL.
    c. Repeat (a), but violations result in the deletion or update of the offending Movie tuple.
    d. A movie that appears in Starsln must also appear in Movie. Handle violations by rejecting the modification.
    e. A star appearing in Starsln must also appear in MovieStar. Handle violations by deleting violating tuples.

**<u>Answer</u>**

1. The suitable keys for the relations of the movie database

- Movies:
  - PrimaryKey: {title, year}
  - FK: studioName reference on Studio(name)
  - FK: producer# reference on MovieExec(cert#)
- MovieStar:
  - Primary key: name
- Staraln:
  - Primary key: {movieTitle, movieYear, starName}
  - FK: movieTitle reference on Movies(title)
  - FK: movieYear reference on Movies(year)
  - FK: starName reference on MovieStar(name)
- MovieExec:
  - Primary Key: cert#
- Studio:
  - Primary Key: name
  - FK: presC# reference on MovieExecl(cert#)

2. Declare the following referential integrity constraints for the movie database.

a. The producer of a movie must be someone mentioned in MovieExec.

Modifications to MovieExec that violate this constraint are rejected.

- Alter Table: Movies
- Add Constraint: Fk_Producer_Exec
- Foreign Key: (producerC)
- References: MovieExec(cert)

b. Repeat (a), but violations result in the producerC# in Movie being set to NULL.

- Alter Table: Movies
- Add Constraint: Fk_Producer_Exec_Null
- Foreign Key: (producerC)
- References: MovieExec(cert)
- On Update Set: producer = NULL

c. Repeat (a), but violations result in the deletion or update of the offending Movie tuple.

- Alter Table: Movies
- Add Constraint: Fk_Producer_Exec_Delete
- Foreign Key: (producerC)
- References: MovieExec(cert)
- On Delete: CASCADE
- On Update: CASCADE

d. A movie that appears in Starsln must also appear in Movie. Handle violations by rejecting the modification.

- Alter Table: StarsIn
- Add Constraint: Fk_Movie_StarsIn
- Foreign Key: (movieTitle, movieYear)

e. A star appearing in Starsln must also appear in MovieStar. Handle violations by deleting violating tuples.

- Alter Table: StarsIn
- Add Constraint: Fk_Star_MovieStar
- Foreign Key: (starName)
- References: MovieStar(name)
- On Delete: CASCADE

II. Consider a relation *R* (A, B, C) and suppose *R* contains the following four tuples:

| **A** | **B** | **C** |
|---|---|---|
| 1 | 2 | 2 |
| 1 | 3 | 2 |
| 1 | 4 | 2 |
| 2 | 5 | 2 |

For each of the following functional dependencies, state whether or not the dependency is satisfied by this relation instance. (Satisfied or Not satisfied)

1) $A \rightarrow B$ is not satisfied
2) $A \rightarrow C$ is satisfied
3) $B \rightarrow A$ is satisfied
4) $B \rightarrow C$ is satisfied
5) $C \rightarrow A$ is not satisfied
6) $C \rightarrow B$ is not satisfied
7) $AB \rightarrow C$ is satisfied
8) $AC \rightarrow B$ is not satisfied
9) $BC \rightarrow A$ is satisfied

III. Consider a relation R (ABCDE). You are given the following FDs: {A -> B, BC -> E, ED -> A}. Find a candidate keys for R.

**Answer**

- R (A, B, C, D, E)
- FDs: A $\longrightarrow$ B, BC $\longrightarrow$ E, and ED $\longrightarrow$ A

- $ABCDE^+$ = {ABCDE} is SK.
- $ACD^+$ = {ABCDE} is SK.
- $A^+$ = {AB} is not SK.
- $C^+$ = {C} is not SK.
- $D^+$ = {D} is not SK.
- $AC^+$ = {ABCE} is not SK.
- $AD^+$ = {ABD} is not SK.
- $CD^+$ = {CD} is not SK.

=> ACD is CK.

=> Prime attribute ACD = A, C, D.

We have ED $\longrightarrow$ A => ADE = CDE

- $CDE^+$ = {ABCDE} is SK.

- $C^+$ = {C} is not SK.
- $D^+$ = {D} is not SK.
- $E^+$ = {E} is not SK.
- $CD^+$ = {CD} is not SK.
- $CE^+$ = {CE} is not SK.
- $DE^+$ = {ABDE}

=> CDE is CK.

=> Prime attribute ACD, CDE = A, B, C, D, E.

We have BC $\longrightarrow$ E => CDE = BCD

- $BCD^+$ = {ABCDE} is SK.

- B+ = {B} is not SK.

- $C^+$ = {C} is not SK.
- $D^+$ = {D} is not SK.
- $BC^+$ = {BCE} is not SK.
- $BD^+$ = {BD} is not SK.
- $CD^+$ = {CD} is not SK.

=> $BCD^+$ is CK.

Thus, the candidate keys of relation are ACD, CDE, BCD.

IV. Consider a relation R (A, B, C, D, E, F). You are given the following FDs: {AB→C, C→DE, E→F, D→A, C→B}. Find all candidate keys for R.

**Answer**

- R (A, B, C, D, E, F)
- FDs: AB ⟶ C, C ⟶ DE, E ⟶ F, D ⟶A, and C ⟶ B

- $ABCDEF^+$ = {ABCDEF} is SK.
- $AB^+$ = {ABCDEF} is SK.
- $A^+$ = {A} is not SK.
- $B^+$ = {B} is not SK.

=> AB is CK.

=> Prime attribute AB = A, B.

We have D ⟶ A => AB = BD

- $BD^+$ = {ABCDEF} is SK.

- $B^+$ = {B} is not SK.
- $D^+$ = {AD} is not SK.

=> $BD^+$ is CK.

=> Prime attribute BD = B, D.

We have C ⟶ B => BD = CD

- $CD^+$ = {ABCDEF} is SK.

- $C^+$ = {ABCDEF} is SK.
- $D^+$ = {AD} is not SK.

=> $CD^+$ is not CK.

Thus, the candidate keys of relation are AB, BD.

V. Consider the following relational schema and given FDs, find the keys of relation:

a. Person (name, forename, phone, address)
   i. {name, forename} -> phone
   ii. Phone -> address

b. BookOrder (ISBN, title, year, price, unitInstock, authorID, name, orderID, orderDate, quantity, customerId, name, address, phone)
   i. ISBN -> {title, year, price, unitInstock, authorID}
   ii. AuthorID -> name
   iii. OrderId -> {OrderDate, customerId}
   iv. {ISBN, orderID} -> quantity
   v. CustomerID -> {name, address, phone}

**<u>Answer</u>**

Find the keys of relation:

a. Person (name, forename, phone, address)

i. {name, forename} -> phone

ii. Phone -> address

- $\{name, forename, phone, address\}^+$ = {name, forename, phone, address} is SK.
- $\{name, forename\}^+$ = {name, forename, phone, address} is SK.
- $Name^+$ = {name} is not SK.
- $Forename^+$ = {forename} is not SK.

=> {name, forename} is CK.

Thus, the candidate keys of relation is {name, forename}.

PA = name, forename

b. BookOrder (ISBN, title, year, price, unitInstock, authorID, name, orderID, orderDate, quantity, customerId, name, address, phone)

i. ISBN -> {title, year, price, unitInstock, authorID}

ii. AuthorID -> name

iii. OrderId -> {OrderDate, customerId}

iv. {ISBN, orderID} -> quantity

v. CustomerID -> {name, address, phone}

- {ISBN, title, year, price, unitInstock, authorID, name, orderID, orderDate, quantity, customerId, name, address, phone}$^+$ = {ISBN, title, year, price, unitInstock, authorID, name, orderID, orderDate, quantity, customerId, name, address, phone} is SK.
- {ISBN, orderID}$^+$ = {ISBN, title, year, price, unitInstock, authorID, name, orderID, orderDate, quantity, customerId, name, address, phone} is SK.
- {ISBN}$^+$ = {ISBN, title, year, price, unitInstock, authorID, name} is not SK.
- {orderID}$^+$ = {orderID, orderDate, customerID} is not SK.

=> {ISBN, orderID} is CK.

Thus, the candidate keys of relation is {ISBN, orderID, name}.

PA = ISBN, orderID

VI. Compute the closure set of the attributes in the relation R below and find the keys of

| babyID | Bname | dob | tob | gender | motherID | Mname | Maddress | doctorID | Dname | Daddress | NurseID | Nname | Naddress |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b001 | Bopha | 01/04/2010 | 20:13 | F | m012 | Pisey | PP | d345 | Sok | PP | n987 | Sreymao | PP |
| b001 | Bopha | 01/04/2010 | 20:13 | F | m012 | Pisey | PP | d345 | Sok | PP | n432 | Chenda | PP |
| b786 | Pisey | 01/04/2010 | 7:22 | M | m098 | Ratha | PP | d345 | Sok | PP | n432 | Chenda | PP |
| b786 | Pisey | 01/04/2010 | 7:22 | M | m098 | Ratha | PP | d345 | Sok | PP | n032 | Dany | PP |
| b786 | Pisey | 01/04/2010 | 7:22 | M | m098 | Ratha | PP | d345 | Sok | PP | n987 | Sreymao | PP |
| b123 | Jack | 09/08/2014 | 6:00 | M | m012 | Pisey | PP | d001 | Paul | PP | n009 | Kunthea | PP |
| b123 | Jack | 09/08/2014 | 6:00 | M | m012 | Pisey | PP | d001 | Paul | PP | n032 | Dany | PP |

relations.

a. $babyID^+$ = {babyID, Bname, dob, tob, gender, motherID, Mname, Maddress, doctorID, Dname, Daddress} is not SK, not CK

b. $motherID^+$ = {motherID, Mname, Maddress} is not SK, not CK

c. doctorID = {doctorID, Dname, Daddress} is not SK, not CK

d. nurseID = {nurseID, Nname, Naddress} is not SK, not CK

e. {babyID, motherID} = {babyID, Bname, dob, tob, gender, motherID, Mname, Maddress, doctorID, Dname, Daddress} is not SK, not CK

f. $\{babyID, doctorID\}^+$ = {babyID, Bname, dob, tob, gender, motherID, Mname, Maddress, doctorID, Dname, Daddress} is not SK, not CK

g. $\{babyID, motherID, doctorID, nurseID\}^+$ = {babyID, Bname, dob, tob, gender, motherID, Mname, Maddress, doctorID, Dname, Daddress, NurseID, Nname, Naddress} is not SK, not CK

h. {babyID, nurseID} = {babyID, Bname, dob, tob, gender, motherID, Mname, Maddress, doctorID, Dname, Daddress, NurseID, Nname, Naddress} is SK, and CK.

Thus, the candidate key (CK) of relation R is {babyID, nurseID}.