Database and Big Data

Solution

Normal Forms

Note: The solution below are solutions among others. :)

EXERCISE 1

Let ***movie*** be the following relation:

movie **(customer_id, customer-name, customer-surname, customer-address, customer-startdate-subscription, movie-id, title-movie, price-purchase-movie, name-editor, address-editor, duration-movie, nationality-movie, name-director, year-release-movie, type-movie, date-movie-rental, duration-rental)**

Assumptions :

1. Movie titles are considered unique (no remake ...).
2. There is only 1 dvd per movie.
3. We assume that a movie cannot be borrowed twice in the same day.
4. A movie is rented using a customer card.
5. We want to keep a history of movie rentals.

I added a few assumptions to make the exercise less ambiguous.

QUESTIONS

1. Explain why this relation is not in 2NF.

We have to look for the candidate key: presumably, since no FD were given and since we have the constraint of the history, it seems that **{movie_id, date-movie-rental}** is (part of) a candidate key (which implies that the ***movie*** relation has not a single attribute; therefore it is not necessarily in 2NF).
Logically, we can list at least some functional dependencies (and surely, we could be even more accurate):

- **movie-id → title-movie, price-purchase-movie, name-editor, address-editor, duration- movie, nationality-movie, name-director, year-release-movie, type-movie**
- **movie-id, date-movie-rental → duration-rental, customer-id, customer-name, customer-surname, customer-address, customer-startdate-subscription**

In any case, we have the following FD:

- « **movie-id → title-movie** »

But we do not have « **date-movie-rental → title-movie** ». So **title-movie** is not determined by the whole key, that implies that the relation is not in 2NF.

2. Decompose ***movie*** to get a relation in 2NF while preserving the dependencies. We only ask here the 2NF.

Note: We underline below the candidate keys of each relation.

- R1 (**<u>movie-id</u>,title-movie, price-purchase-movie, name-editor, address-editor, duration-movie, nationality-movie, name-director, year-release-movie, type-movie**)
- R2 (**<u>movie-id, date-movie-rental</u>, duration-rental, customer-id, customer-name, customer-surname, customer-address, customer-startdate-subscription**)

3. Are the obtained relations in 3NF? Explain. If not, modify the schema in order to get relations in 3NF.

Let us consider R1, there is a FD between <u>title-movie</u> and other attributes:

- title-movie $\rightarrow$ name-editor, address-editor, duration-movie, nationality-movie, name-director, year-release-movie

We decompose R1:

- R11 (<u>movie-id</u>,title-movie, price-purchase-movie)
- R12 (<u>title-movie</u>, name-editor, address-editor, duration-movie, nationality-movie, name-director, year-release-movie, type-movie)

But in order to have R12 in 3NF, we still must decompose it

- R121 (<u>title-movie</u>, name-editor, duration-movie, nationality-movie, name-director, year-release-movie, type-movie)
- R122 (name-editor, address-editor)

R11, R121, R122 are in 3NF.

Let us consider now R2, there is a FD between <u>customer-id</u> and other attributes:
customer-id $\rightarrow$ **customer-name, customer-surname, customer-address, customer-startdate-subscription**

We decompose R2:

- R21(<u>movie-id, date-movie-rental</u>, duration-rental, customer-id)

R22(customer-id, **customer-name, customer-surname, customer-address, customer-startdate-subscription**)

R21 et R22 are in 3NF.

4. Are the obtained relations in BCNF? Explain. If not, modify the schema to get relations in BCNF.

We must check whether there are FDs between an attribute that is not part of the candidate key and an attribute that belongs to the candidate key.

After verification, we can conclude that R21, R22, R11, R121, R122 are in BCNF.

To check: Is it lossless? Is it dependency preserving?

EXERCISE 2

Let **R** be the following relation:

R(A, B, C, D, E)

and **F** be the set of FDs in **R**:

A $\rightarrow$ B,
B $\rightarrow$ D,
C $\rightarrow$ A,
AD $\rightarrow$ C,
CB $\rightarrow$ E,
AEC $\rightarrow$ D

<u>QUESTIONS</u>

1. Give a decomposition of R to obtain a 3NF scheme without loss of functional dependencies and without loss of data.

First, let's try to simplify the three FDs having multiple attributes in the left-hand side in order to keep if possible only FDs having a single attribute in the left- and right-hand sides:

AD $\rightarrow$ C

By transitivity A $\rightarrow$ B et B $\rightarrow$ D so we have: A $\rightarrow$ D
Thus: AA $\rightarrow$ C (since « A determines D », we can replace D by A)
And so: <u>A $\rightarrow$ C (that simplifies AD $\rightarrow$ C)</u>

CB $\rightarrow$ E

By transitivity C $\rightarrow$ A et A $\rightarrow$ B so we have: C $\rightarrow$ B
Thus: CC $\rightarrow$ E (since « D determines B », we can replace B by C)
And so: <u>C $\rightarrow$ E (that simplifies CB $\rightarrow$ E)</u>

AEC $\rightarrow$ D

Given the assumptions: C $\rightarrow$ A and C $\rightarrow$ E
Thus: CCC $\rightarrow$ D (since « C determines A », we can replace A by C
and since « C determines E », we can replace E by C)
Then, by simplification, we get: C $\rightarrow$ D which is already a FD deduced by transitivity because of the three following FDs: C $\rightarrow$ A $\rightarrow$ B $\rightarrow$ D

Remark: These simplifications make it possible to suppress some FDs that could be deduced from simpler FDs.

Let us draw a graph of these FDs. That should look like this:

$$\begin{tikzcd}
& & A \arrow[dl] \arrow[dr, bend left] & & \\
& B \arrow[dl] & & C \arrow[ul] \arrow[dr] & \\
D & & & & E
\end{tikzcd}$$

As we see on the graph, A and C are two candidate keys since, from one or the other, we can reach all the other attributes. A or C can become either the primary key of R (A, B, C, D, E).

To be in 3NF we must suppress the following dependence: B $\rightarrow$ D. For that, we decompose R as explained in slide 40 (or slide 48):

- R1 ($\underline{A}$, B, C, E) or R1 (A, B, $\underline{C}$, E)
- R2 ($\underline{B}$, D)

Remark: *I underlined the primary key.*

Remark: *The normalization is done with respect to the candidate keys (not with respect to the primary keys). Since A and C are two candidate keys, we could break the relation to create a relation containing only A and C. However, this would lead to (unnecessarily) over-breaking and requiring additional joins that may affect the performance.*

These relations are in 3NF. Note that all the dependencies mentioned in the graph are present in R1 and R2. Thus, the dependencies are preserved. Moreover, there is no data loss since we decomposed the relation in the same way than in slides 40 and 48 (for those who are interested, that's actually an application of Heath's theorem).