DMD HW7 - Normal Forms

Nikolay Troshkov

October 2015

1st Normal Form

1

```
SELECT CustName
FROM Customers
  JOIN Rentals
    ON Customers.CustID = Rentals.CustID
WHERE Title1 = ’Die Hard’
      OR Title2 = ’Die Hard’
      OR Title3 = ’Die Hard’;
```

The query is cumbersome because we have to look for the match in 3 attributes. This is unobvious and would cause problems for maintaining the database.

We would encounter problems with determining the number of times a movie was rented in case a customer rents more than one copy of a movie. However, we can assume this is a single rental. In both cases this is an unobvious behavior. Also, in both cases we have to write an overloaded query.

2

In case a customer wants to rent more than 3 movies, we would have to create multiple records for the same rental. It would appear as two or more different rentals, although the date and the customer will be the same.

3

Rentals_2 (RentalID, CustID, CheckOutDate, Title)

Here we have to assume that for a certain movie only one copy can be rented. In order to allow multiple copies, we need to add another attribute with rented copies count. Or, which is better, a separate table.

4

```
CREATE TABLE Rentals_2 (
  RentalID     INT         NOT NULL,
  CustID       INT         NOT NULL,
  CheckOutDate TIMESTAMP   NOT NULL,
  Title        VARCHAR(50) NOT NULL,
  FOREIGN KEY (CustID) REFERENCES Customers (CustID)
  ON DELETE NO ACTION ON UPDATE CASCADE,
  PRIMARY KEY (RentalID, Title)
);
```

5

```
INSERT INTO Rentals_2
  SELECT
    RentalID,
    CustID,
    CheckOutDate,
    Title1
  FROM Rentals;

INSERT INTO Rentals_2
  SELECT
    RentalID,
    CustID,
    CheckOutDate,
    Title2
  FROM Rentals;

INSERT INTO Rentals_2
  SELECT
    RentalID,
    CustID,
    CheckOutDate,
    Title3
  FROM Rentals;
```

6

```sql
SELECT CustName
FROM Customers
  JOIN Rentals_2
    ON Customers.CustID = Rentals_2.CustID
WHERE Title = 'Die Hard';
```

Boyce-Codd Normal Form

7

RentalID → CustID, CheckOutDate
Title → Director, ReleaseType
ReleaseType → Price

8

The table Rentals3 does not meet the requirements of the 3rd normal form by having a FD from a non-key attribute ReleaseType. Also, it is not in the 2nd normal form by having a partial key dependency. Therefore, it does not meet the requirements of the BCNF.

9

Assuming that the only tables we have in the database is those two, Customers and Rentals3, I assume that all available data about movies is stored in the Rentals3 table. Therefore:

Insertion anomaly

We cannot add a new movie without a customer renting this movie.

Deletion anomaly

If the store is no longer provides a certain movie, we cannot remove the information about it without removing the information about renting it.

Modification anomaly

In case we need to update a price for any of the release types, we would have to update all the tuples with this release type in the table.

10

Rentals3 (RentalID, CustID, CheckOutDate)
RentedMovies (RentalID, Title)
ReleaseTypes (ReleaseType, Price)
Movies (Title, Director, ReleaseType)

4th Normal Form

11

A multivalue dependency A ↠ B is a dependency between A and B in case of which all combinations of A, B and some C are present, but B does not depend on C.

12

In the given table we have the following MVD:
Course ↠ Teacher
Course ↠ Book

The minimal key contains all attributes, so the Course attribute is not a superkey. Therefore, the table is not in the 4th normal form.

13

CourseTeachers (Course, Teacher)
CourseBooks (Course, Book)

Normalization: yes or no?

14

We are given the following dependency:

CustCity, CustState → CustZip

At the same time, zip code uniquely identifies city and state, so we have

CustZip → CustCity, CustState

Customer name could be repeated and depends on CustID. Customer address depends on CustID as well, so:

CustID → CustName, CustZip.

I'd say that the PK for the table Customers is CustID. As long as we have dependency from a non-key attribute, the table is not in the BCNF, and not even in the 2nd normal form.

15

We can decompose the table into the following relations:

Customers (CustID, CustName, CustZip)
Addresses (zip, city, state)

But in this case the table Addresses would still be not in the BCNF, so that decomposition doesn't make much sense. And further decomposition will lead to a data and FD loss. Therefore it would be better to just keep it as is.

Although, in the real life city and state do not uniquely identify zip code (so we only have zip → city, state, but not vice versa), and in this case such decomposition would be reasonable.