Tutorial 9

Arvind Viswanathan
viswaa2@mcmaster.ca

Normal Forms

- A schema in a “Normal Form” guarantees good properties and no anomalies.
- Normal forms are used in an RDBMS to hold up good design.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy and remove undesired characteristics like anomalies.

Normalization

- The benefits of having normalization is:
  - There is no redundant data.
  - Ensuring that data dependencies make sense.
  - Update and Deletion anomalies are avoided.

Normalization Rule

- The Normalization rule is divided into four normal forms:
  - 1NF
  - 2NF
  - 3NF
  - Boyce Codd NF (BCNF)

First Normal Form

- First normal form (1NF) is a property of a relation in a relational database. A relation is in first normal form if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

First Normal Form

- Key points to note in 1 NF:
  - atomic values - single values should be in each cell.
  - As a whole the tuple will be unique

First Normal Form

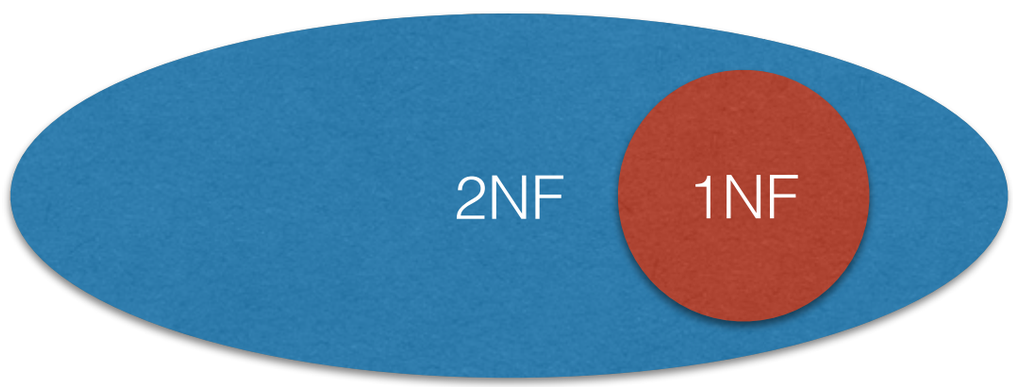
```
CREATE TABLE CUSTOMERS
        ID              INT                     NOT NULL,
        NAME            VARCHAR (20)            NOT NULL,
        AGE             INT                     NOT NULL,
        ADDRESS         CHAR (25),
        ORDERS          VARCHAR(155));
```

So if we populate this table for a single customer having multiple orders, then it would be something as follows:

ID	NAME	AGE	ADDRESS	ORDERS
100	Thomas	24	Lower West Side	Cannon XL-200
100	Thomas	24	Lower West Side	Battery XL-200
100	Thomas	24	Lower West Side	Tripod Large

Second Normal Form

- All the rules of the first normal form are to be met and there must be no partial dependencies of any of the columns on the primary key.
- Basically, whatever is in the second normal form is also in the first normal form as it is a superset of the first normal form as the diagram shows.



Second Normal Form

```
CREATE TABLE CUSTOMERS(

    CUST_ID     INT              NOT NULL,

    CUST_NAME VARCHAR (20)      NOT NULL,

    ORDER_ID    INT              NOT NULL,

    ORDER_DETAIL VARCHAR (20)  NOT NULL,

    SALE_DATE  DATETIME,

    PRIMARY KEY (CUST_ID, ORDER_ID));
```

- There are partial dependencies of primary keys and columns. CUST_NAME is dependent on CUST_ID, and there's no real link between a customer's name and what he purchased. Order detail and purchase date are also dependent on ORDER_ID, but they are not dependent on CUST_ID, because there's no link between a CUST_ID and an ORDER_DETAIL or their SALE_DATE.

- So not in the Second Normal Form

Second Normal Form

- To get it to the second normal form we need to follow three steps:

```
CREATE TABLE CUSTOMERS(
        CUST_ID     INT              NOT NULL,
        CUST_NAME VARCHAR (20)       NOT NULL,
        PRIMARY KEY (CUST_ID));


CREATE TABLE ORDERS(
        ORDER_ID    INT              NOT NULL,
        ORDER_DETAIL VARCHAR (20)  NOT NULL,
        PRIMARY KEY (ORDER_ID));
```

Second Normal Form

Finally, create a third table storing just CUST_ID and ORDER_ID to keep track of all the orders for a customer:

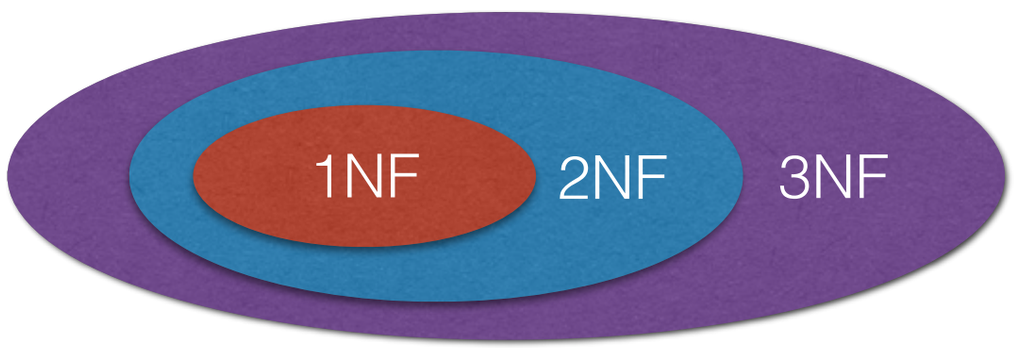
```
CREATE TABLE CUSTOMERORDERS(
    CUST_ID     INT             NOT NULL,
    ORDER_ID    INT             NOT NULL,
    SALE_DATE   DATETIME,
    PRIMARY KEY (CUST_ID, ORDER_ID));
```

Third Normal Form

- For a table to be in third normal form it must satisfy the following properties:

- All non-primary fields are dependent on the primary key.

- It should already be in the second normal form. Hence, we can visualize third normal form as:



Third Normal Form

The dependency of non-primary fields is between the data. For example, in the below table, **street name, city, and state** are unbreakably bound to the **zip code**.

```
CREATE TABLE CUSTOMERS(
    CUST_ID       INT             NOT NULL,
    CUST_NAME     VARCHAR (20)    NOT NULL,
    DOB           DATE,
    STREET        VARCHAR(200),
    CITY          VARCHAR(100),
    STATE         VARCHAR(100),
    ZIP           VARCHAR(12),
    EMAIL_ID      VARCHAR(256),
    PRIMARY KEY (CUST_ID));
```

Third Normal Form

The dependency between zip code and address is called a transitive dependency. To comply with third normal form, all you need to do is move the Street, City, and State fields into their own table, which you can call the Zip Code table:

```
CREATE TABLE ADDRESS(
    ZIP          VARCHAR(12),
    STREET       VARCHAR(200),
    CITY         VARCHAR(100),
    STATE        VARCHAR(100),
    PRIMARY KEY (ZIP));
```

Next, alter the CUSTOMERS table as follows:

```
CREATE TABLE CUSTOMERS(
    CUST_ID      INT             NOT NULL,
    CUST_NAME    VARCHAR (20)    NOT NULL,
    DOB          DATE,
    ZIP          VARCHAR(12),
    EMAIL_ID     VARCHAR(256),
```

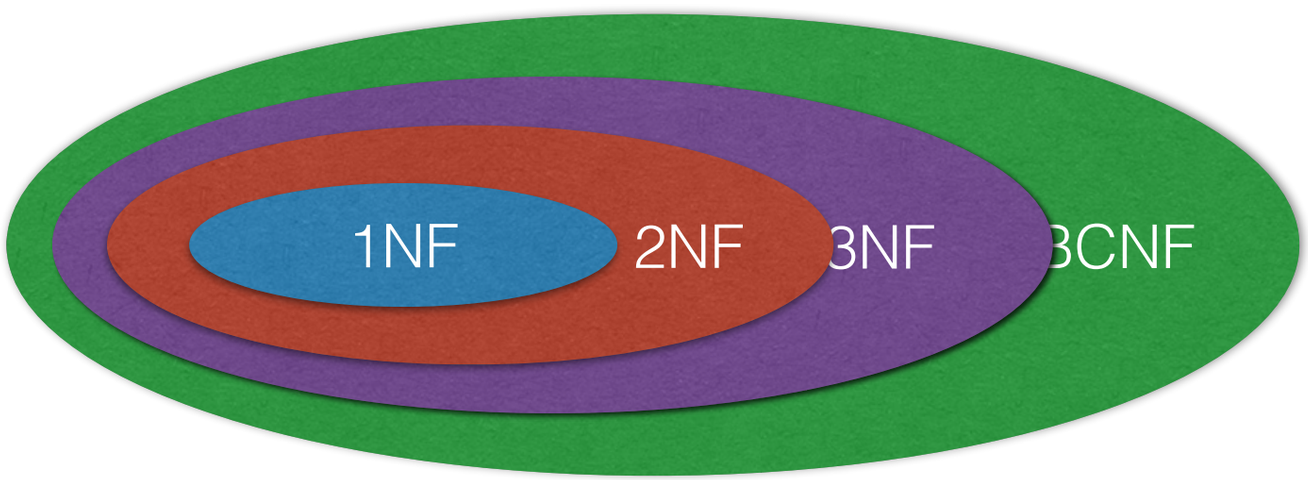
Third Normal Form

- The advantages of removing transitive dependencies are mainly twofold. First, the amount of data duplication is reduced and therefore your database becomes smaller.
- The second advantage is data integrity. When duplicated data changes, there's a big risk of updating only some of the data, especially if it's spread out in a number of different places in the database. For example, if address and zip code data were stored in three or four different tables, then any changes in zip codes would need to ripple out to every record in those three or four tables.

Boyce Codd Normal Form

- A database table is in BCNF if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key.

- BCNF is also sometimes referred to as 3.5NF, or 3.5 Normal Form.



Boyce Codd Normal Form

- BCNF is really an extension of 3rd Normal Form (3NF). For this reason it is frequently termed 3.5NF. 3NF states that all data in a table must depend only on that table's primary key, and not on any other field in the table. At first glance it would seem that BCNF and 3NF are the same thing. However, in some rare cases it does happen that a **3NF table is not BCNF-compliant. This may happen in tables with two or more overlapping composite candidate keys.**

Decomposition

- What are the factors we expect from decomposition?
- Lossless Join : we get back the original tuples
- No Anomalies
- Dependency preservation : The original FD’s should be satisfied.

Boyce Codd Normal Form Decomposition

- Though the ideal scenario would be to get all three properties of decomposition, it is not the case.

- In BCNF only two of the properties are satisfied and hence we can differentiate it from a 3NF decomposition.

- Lossless Join : Satisfied!

- No Anomalies: Satisfied!

- Dependency Preservation: NOT Satisfied!

3NF Decomposition

- Though the ideal scenario would be to get all three properties of decomposition, it is not the case.

- In BCNF only two of the properties are satisfied and hence we can differentiate it from a 3NF decomposition.

- Lossless Join : Satisfied!

- No Anomalies: NOT Satisfied!

- Dependency Preservation: Satisfied!

References

- http://www.tutorialspoint.com/sql/third-normal-form.htm

- http://www.cas.mcmaster.ca/~fchiang/courses/db3/W15/slides/DBDesign_part5.pdf

- http://www.studytonight.com/dbms/database-normalization.php

- http://www.techopedia.com/definition/5642/boyce-codd-normal-form-bcnf

Questions?

Thank you!

viswaa2@mcmaster.ca

Thursday: 3-4 pm. ITB 128