Chapter 14/17 Project

Ben Hanson

University of Minnesota Crookston

Apply the process of normalization to the volunteer project data. Describe in detail the normalization process that was followed.

The process of normalization, as stated by Ramez Elmasri in the textbook Fundamentals of Database Systems, "takes a relation schema through a series of tests to *certify* whether it satisfies a certain normal form." (Elmasri 2017). The process of normalization takes place in a top-down fashion, decomposing relations as necessary. Due to this fact, this process of normalization may also be recognized as relational design by analysis. Ramez Elmasri goes on to state that this process consists of five unique normal forms, all proposed based on different concepts. In most simple terms, the process of normalization is used to organize and separate a database into separate tables and columns. In this process, as the tables satisfy the database normalization form, they become less likely to conform to database modification anomalies, becoming more focused on a particular topic or purpose.

In the Volunteer Project Database, there is lots of information being recorded. Specifically, this database consists of data regarding the project itself, the organization, and volunteers. The first step to the process of normalization in this database, is to get the data to the First Normal Form (1NF). There are several questions in which we must answer in order to do this. We must look at each table, answering whether or not the combination of all columns makes a unique row every single time, as well as determining what field can be used to uniquely identify the row. The answer to the first question is no because there could be the same combination of data, and it would represent a different row. There also could be the same values for this row and it would be a separate row. The answer to the second question takes a little more thought. The field that can be used to uniquely identify a row in this table, is the

PROJECT_ID. The reason for this is that it will be a different, ascending number for each project entry that is recorded into this database. Now that we have answered these questions, our table is now in First Normal Form.

The next step in this process of normalization is to get that data into the Second Normal Form (2NF). The Second Normal Form is based on the concept of full functional dependency. According to Ramez Elmasri, "A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute A from X means that the dependency does not hold anymore; that is, for any attribute $A \varepsilon X$, $(X - \{A\})$ does not functionally determine Y." (Elmasri 2017). Functional dependency means that every field that is not the primary key is determined by that primary key, so it is specific to that record.In other words, the purpose of the Second Normal Form is to Fulfil the requirements of the first normal form, while ensuring that each non-key attribute must be functionally dependent on the primary key. To do this, we must determine whether or not the columns are dependent on and specific to the primary key. In the volunteer project database, the primary key is PROJECT_ID, which represents the project. Let's look at each column:

- PROJECT_NAME: Yes, this is dependent on the primary key because a different PROJECT_ID means a different project name.
- PROJECT_DATE: Yes, this is dependent on the primary key because each project date is unique to the PROJECT_ID.
- ORG_CODE: No, this is not dependent on the primary key because there may be more than one organization working on different projects.

- ORG_NAME: No, this is not dependent on the primary key because there can be multiple organizations working on seperate projects.
- VOL_ID: No, this is not dependent on the primary key because there can be more than one volunteer participating in different projects.
- VOL_LAST: No, same as VOL_ID.
- VOL_FIRST: No, same as VOL_ID.
- VOL_EMAIL: No, same as VOL_ID.
- VOL_STATUS: No, this is not dependent on the primary key because you could be inactive or active, while being assigned to a project.

After going through each column, we can tell that some are dependent on the primary key, PROJECT_ID, while others are not. The solution to the columns that are not dependent on PROJECT_ID, is to simply remove them from the table, and create a new table in which they are dependent on the primary key. For the column ORG_CODE, since it is not dependent on PROJECT_ID, we must remove it from the table and create a new table with ORG_CODE within it. It will then look something like this: ORG (ORG_CODE). Although we have created a new table with ORG_CODE inside of it, it is still not unique due to the fact that there could be two organizations with the same ORG_CODE. In order to fix this, we will simply create a new primary key (ORG_ID) to the newly created ORG table. Now the table looks like this: ORG (ORG_ID, ORG_CODE). Next, we can recognize that ORG_NAME is a property of the organization. With this in mind, we can simply add this to our new table, resulting in the ORG table looking like this: ORG (ORG_ID, ORG_CODE, ORG_NAME). The next step is to do the same process to the next column. The next column that we determined was not unique to the

PROJECT_ID, was VOL_ID. We can simply remove this column from the original table, creating a new table called VOL. Also, we will add a primary key named "VOL_NUM". The new table will look like so: VOL (VOL_NUM, VOL_ID). Next, looking at the remaining columns that are not unique to PROJECT_ID, we can determine that they are all properties of the volunteer's. With this in mind, we may simply add the remaining columns to our newly created VOL table, with the result of this: VOL (VOL_NUM, VOL_ID, VOL_LAST, VOL_FIRST, VOL_EMAIL, VOL_STATUS). We have now created two more tables to our database, with a total of three tables which go as follows:

PROJECT

PROJECT_ID	PROJECT_NAME	PROJECT_DATE

ORG

ORG_ID	ORG_CODE	ORG_NAME

VOL

VOL_NUM	VOL_ID	VOL_LAST	VOL_FIRST	VOL_EMAIL	VOL_STATUS

Now, to link all of this data together, we will need to focus on creating foreign keys. A foreign key is a column in one table that refers to the primary key in seperate table. Also, it is used to link one record to another based on its unique identifier, without having to store the additional information about the linked record. We must place the primary key from one table into the other table. We will first look at the tables PROJECT and ORG. Due to the fact that in this scenario, an organization can have many projects, we will place the ORG_ID into the PROJECT table. Now, the PROJECT table will look like this, with the new foreign key being italicized: PROJECT (PROJECT_ID, *ORG_ID*, PROJECT_NAME, PROJECT_DATE). Next,

we will look at the relationship between PROJECT and VOL. In this scenario, we have a many-to-many relationship because projects have many volunteers, while volunteers also have many projects. To fix this, we must create a joining table to store the relationships between the two tables. The table will look like this: PROJECT_VOL (*PROJECT_ID, VOL_NUM*). Our final relationship we must focus on is between ORG and VOL. In this scenario, organizations have many volunteers, so we must place the ORG_ID into the VOL table, looking as so: VOL (VOL_NUM, *ORG_ID*, VOL_ID, VOL_LAST, VOL_FIRST, VOL_EMAIL, VOL_STATUS). Our final set of tables go as follows, with the foreign key's being italicized:

PROJECT

PROJECT_ID	*ORG_ID*	PROJECT_NAME	PROJECT_DATE

ORG

ORG_ID	ORG_CODE	ORG_NAME

VOL

VOL_NUM	*ORG_ID*	VOL_ID	VOL_LAST	VOL_FIRST	VOL_EMAIL	VOL_STATUS

PROJECT_VOL

PROJECT_ID	*VOL_NUM*

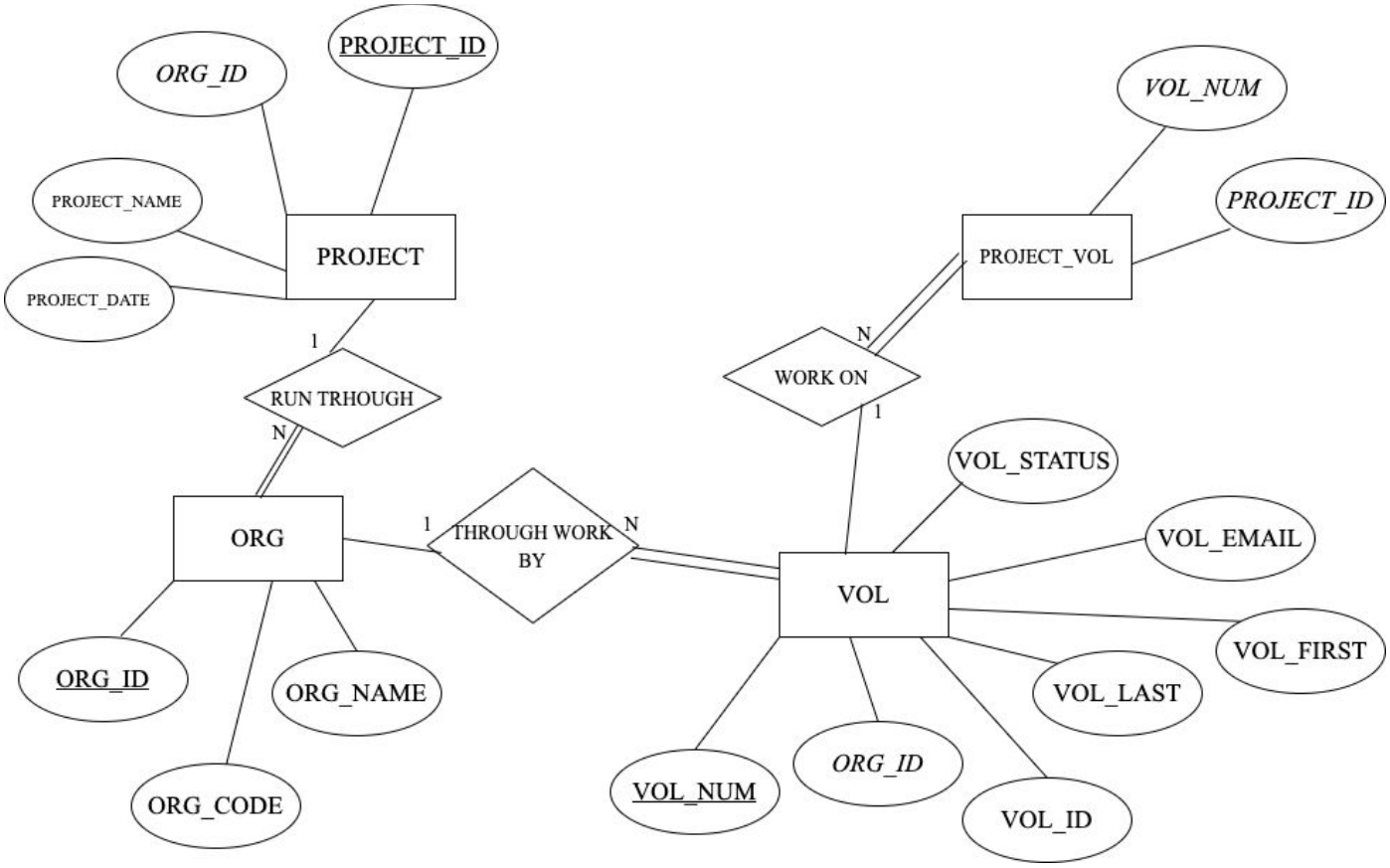
Now that we have completed the Second Normal Form, we will focus on the last stage of the process of normalization.

The Third Normal Form, as stated by Ramez Elmasri, is "is based on the concept of *transitive dependency*. A functional dependency $X \rightarrow Y$ in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R,11 and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold." (Elmasri 2017). In other words, the main

purpose of the Third Normal Form is to fulfill the requirements of theSecond Normal Form, as well as to have no transitive functional dependency. Every non primary key attribute must depend on the primary key and nothing else. First, we will look at the PROJECT table. The table for PROJECT looks like this: PROJECT (PROJECT_ID, *ORG_ID*, PROJECT_NAME, PROJECT_DATE). As we can see for this table, this tabe ha no columns that aren't dependent on the primary key. The Next table, ORG, goes as follows: ORG (ORG_ID, ORG_CODE, ORG_NAME). Again, we can determine that all of the columns in this table depend on the primary key. The last two tables, VOL and PROJECT_VOL, meet this requirement as well, maintaining that all columns depend on the primary key of their tables. Now that we have completed the process of normalization, we can see that we have developed a solid set of relationship rules, greatly improving the data structure from having almost no normalization at all. Some major benefits of the process of normalization in which we have achieved for the database include greater overall efficiency, the prevention of insert anomaly, update anomaly, delete anomaly, and the implementation of accurate data. These are just a few of the many benefits that come with the process of normalization.

Screenshots of the table designs from SSMS showing the tables, fields, and their relationships (draw an ER diagram).

Entity Relationship Diagram



Creating and Connecting to the Database

```
Last login: Sun Apr  5 13:08:00 on ttys000
/Library/PostgreSQL/12/scripts/runpsql.sh; exit
Bens-MacBook-Air:~ benhanson$ /Library/PostgreSQL/12/scripts/runpsql.sh; exit
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (12.2)
Type "help" for help.

postgres=# CREATE DATABASE VOL_PROJECT;
CREATE DATABASE
postgres=# \c vol_project
You are now connected to database "vol_project" as user "postgres".
vol_project=#
```

Creating the ORG Table

```
vol_project=# CREATE TABLE ORG  (
vol_project(# ORG_ID BIGSERIAL NOT NULL PRIMARY KEY,
vol_project(# ORG_CODE VARCHAR(50) NOT NULL,
vol_project(# ORG_NAME VARCHAR(50) NOT NULL  );
CREATE TABLE
vol_project=#
```

Creating the PROJECT Table

```
CREATE TABLE
vol_project=# CREATE TABLE PROJECT  (
vol_project(#  PROJECT_ID BIGSERIAL NOT NULL PRIMARY KEY,
vol_project(# FOREIGN KEY (ORG_ID) REFERENCES ORG (ORG_ID),
vol_project(# PROJECT_NAME VARCHAR(50) NOT NULL,
vol_project(# PROJECT_DATE DATE NOT NULL  );
```

Creating the VOL Table

```
vol_project=# CREATE TABLE VOL  (
vol_project(# VOL_NUM BIGSERIAL PRIMARY KEY NOT NULL,
vol_project(# FOREIGN KEY (ORG_ID) REFERENCES ORG (ORG_ID),
vol_project(# VOL_ID VARCHAR(20) NOT NULL,
vol_project(# VOL_LAST VARCHAR(50) NOT NULL,
vol_project(# VOL_FIRST VARCHAR(50) NOT NULL,
vol_project(# VOL_EMAIL VARCHAR(75) NOT NULL,
vol_project(# VOL_STATUS VARCHAR(3) NOT NULL  );
```

Creating the PROJECT_VOL Table

```
vol_project=# CREATE TABLE PROJECT_VOL  (
vol_project(# FOREIGN KEY (PROJECT_ID) REFERENCES PROJECT (PROJECT_ID),
vol_project(# FOREIGN KEY (VOL_NUM) REFERENCES VOL (VOL_NUM)  );
```

List of all the Tables, Without Data

```
vol_project=# SELECT* FROM PROJECT;
 project_id | org_id | project_name | project_date
------------+--------+--------------+--------------
(0 rows)

vol_project=# SELECT * FROM ORG;
 org_id | org_code | org_name
--------+----------+----------
(0 rows)

vol_project=# SELECT * FROM VOL;
 vol_num | org_id | vol_id | vol_last | vol_first | vol_email | vol_status
---------+--------+--------+----------+-----------+-----------+------------
(0 rows)

vol_project=# SELECT * FROM PROJECT_VOL;
 project_id | vol_num
------------+---------
(0 rows)

vol_project=#
```

Placing Data Inside PROJECT Table

```
vol_project=# INSERT INTO PROJECT  (
vol_project(# ORG_ID,
vol_project(# PROJECT_NAME,
vol_project(# PROJECT_DATE )
vol_project-# VALUES ('1', 'Greenville Afterschool Tutoring', DATE '2016-02-05');
INSERT 0 1
vol_project=# INSERT INTO PROJECT  (
vol_project(# ORG_ID,
vol_project(# PROJECT_NAME,
vol_project(# PROJECT_DATE )
vol_project-# VALUES ('2', 'Highland Hospital Blood Drive', DATE '2015-11-15');
INSERT 0 1
```

Placing the Remaining Data Inside PROJECT Table

```
[INSERT 0 1
[vol_project=#  INSERT INTO PROJECT  (
[vol_project(# ORG_ID,
[vol_project(# PROJECT_NAME,
 vol_project(# PROJECT_DATE )
[vol_project-# VALUES ('3', 'Madagasgar Retreat', DATE '2016-01-14');
[INSERT 0 1
[vol_project=# INSERT INTO PROJECT  (
[vol_project(# ORG_ID,
[vol_project(# PROJECT_NAME,
[vol_project(# PROJECT_DATE )
[vol_project-# VALUES ('1', 'Homeless Hope Counseling', DATE '2007-02-05');
 INSERT 0 1
```

Placing Data Inside ORG Table

```
vol_project=# INSERT INTO ORG (
vol_project(# ORG_CODE,
vol_project(# ORG_NAME )
vol_project-# VALUES ('UW', 'United Way');
INSERT 0 1
vol_project=# INSERT INTO ORG (
vol_project(# ORG_CODE,
vol_project(# ORG_NAME )
vol_project-# VALUES ('RC', 'Red Cross');
[INSERT 0 1
vol_project=# INSERT INTO ORG (
vol_project(# ORG_CODE,
vol_project(# ORG_NAME )
vol_project-# VALUES ('WF', 'Wildlife Fund');
INSERT 0 1
```

Placing Data Inside VOL Table

```
vol_project=# INSERT INTO VOL (ORG_ID, VOL_ID, VOL_LAST, VOL_FIRST, VOL_EMAIL, VOL_STATUS)
vol_project-#  VALUES ('1', '10122', 'OLeery', 'Ben', 'benoleery99@gmail.com', 'A');
INSERT 0 1
vol_project=# INSERT INTO VOL (ORG_ID, VOL_ID, VOL_LAST, VOL_FIRST, VOL_EMAIL, VOL_STATUS)
[vol_project-# VALUES ('1,3', '10123', 'Panera', 'Antonio', 'antonp@hotmail.com', 'A');
INSERT 0 1
```

Placing the Remaining Data Inside VOL Table

```
[vol_project=# INSERT INTO VOL (ORG_ID, VOL_ID, VOL_LAST, VOL_FIRST, VOL_EMAIL, VOL_STATUS)
vol_project-# VALUES ('1,2,3', '10121', 'Rogers', 'Kim', 'krogers@gmail.com', 'A');
INSERT 0 1
```

Placing Data Inside PROJECT_VOL Table

```
vol_project=# INSERT INTO PROJECT_VOL (
vol_project(# PROJECT_ID,
vol_project(# VOL_NUM)
vol_project-# VALUES ('1', '1,2');
INSERT 0 1
vol_project=# INSERT INTO PROJECT_VOL (
vol_project(# PROJECT_ID,
vol_project(# VOL_NUM)
vol_project-# VALUES ('2', '3');
INSERT 0 1
vol_project=# INSERT INTO PROJECT_VOL (
vol_project(# PROJECT_ID,
vol_project(# VOL_NUM)
vol_project-# VALUES ('3', '3');
INSERT 0 1
vol_project=# INSERT INTO PROJECT_VOL (
vol_project(# PROJECT_ID,
vol_project(# VOL_NUM)
vol_project-# VALUES ('4', '2,3');
INSERT 0 1
vol_project=#
```

Showing the PROJECT Table (PK = <u>PROJECT_ID</u>, FK = *ORG_ID*)

```
vol_project=# SELECT * FROM PROJECT;
 project_id | org_id |          project_name           | project_date
------------+--------+---------------------------------+--------------
          1 | 1      | Greenville Afterschool Tutoring | 2016-02-05
          2 | 2      | Highland Hospital Blood Drive   | 2015-11-15
          3 | 1      | Homeless Hope Counseling        | 2007-02-05
          4 | 3      | Madagasgar Retreat              | 2016-01-14
(4 rows)
```

Showing the ORG Table (PK = <u>ORG_ID</u>)

```
vol_project=# SELECT * FROM ORG;
 org_id | org_code |   org_name
--------+----------+---------------
      1 | UW       | United Way
      2 | RC       | Red Cross
      3 | WF       | Wildlife Fund
(3 rows)
```

Showing the VOL Table (PK = <u>VOL_NUM</u>, FK = *ORG_ID*)

```
vol_project=# SELECT * FROM VOL;
 vol_num | org_id | vol_id | vol_last | vol_first |       vol_email       | vol_status
---------+--------+--------+----------+-----------+-----------------------+------------
       1 | 1      | 10122  | OLeery   | Ben       | benoleery99@gmail.com | A
       2 | 1,3    | 10123  | Panera   | Antonio   | antonp@hotmail.com    | A
       3 | 1,2,3  | 10121  | Rogers   | Kim       | krogers@gmail.com     | A
(3 rows)
```

Showing the PROJECT_VOL Table (FK = *PROJECT_ID, VOL_NUM*)

```
vol_project=# SELECT * FROM PROJECT_VOL;
 project_id | vol_num
------------+---------
 1          | 1,2
 2          | 3
 3          | 3
 4          | 2,3
(4 rows)
```

Resources

Brumm, B. (2020, March 31). Database Normalization: A Step-By-Step-Guide With Examples. Retrieved from https://www.databasestar.com/database-normalization/

Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of database systems*. Harlow, Essex: Pearson Education.