Table of Contents

[1. General Description of Problem 2](#_Toc425459144)

[2. Normalization Process 3](#_Toc425459145)

[3. Technologies Used 8](#_Toc425459146)

[4. Difficulties Faced 11](#_Toc425459147)

[5. Database 13](#_Toc425459148)

[6. References 15](#_Toc425459149)

1. General Description of Problem

A relational database is a natural solution for a family tree, as a family tree itself is highly dependent on relationships. A family tree, however, is more than just a list of people and the relationships among them. Other data that may be stored in a family tree may include, but not be limited to, pictures, dates and names of significant events (such as marriages, baptisms, bar mitzvahs, etc.), and sources (such as birth certificates, census records, marriage records, etc).

1. Normalization Process

Mike Chapple of about.com describes the normalization process as “the process of efficiently organizing data in a database” (Chapple, 2015). Normalization is typically stated in forms. First normal form is called 1NF, second normal form is called 2NF, and so on. Part of the idea of normalization is to reduce redundancy and null values in a database. This will typically reduce the size of the database and increase the speed of queries.

For the initial version of my database (v0.1.0), it wasn’t even in 1NF, so I said it was in “0NF.” It consisted of one table with (possibly) many redundant and null values:



While everyone has a father and mother, it is not always necessarily know. Similarly, the place of birth and date may not be known. Also, the person might still be alive, so there could be no date or place of death. Finally, many people may have been born in the same place, or died in the same place, as other people, making the place of birth and death columns possibly redundant.

For the second version (v0.2.0), the database was not brought out of 0NF form, but rather only the parent columns where replaced by simply a number referencing another person row:

The database was finally brought into the world of normalization in the third release (v0.3.0):



(Continued on next page)



Here finally there are multiple tables. According to (Chapple, 2015), I have brought my database into 1NF by removing the columns that are redundant from the person table (namely, the place, date, and parent columns) and moved them into separate tables. I have also made a primary key for each column, another characteristic of 1NF.

With this version, I have also brought my database into 2NF. Again according to (Chapple, 2015), 2NF means eliminating redundant data (rather than just the columns). Each name column in the place table is unique, while in the previous version the places might repeat. The new parent relationship tables (FATHER\_OF and MOTHER\_OF) tables reduce null values that would’ve occurred in the similar columns of the place table in the previous version. Foreign key references match all the tables together.

In order to be in 3NF, there must be no columns that are not dependent on the primary key. In my database, this is true, but it was true of my “0NF” database versions, so this version was only “brought into” 3NF by virtue of each normalization form being required to be in the previous normalization form.

Further versions of the database (v0.6.0, v0.8.0, v1.0.0) did not undergo any further normalization process. When additional tables were added, normalization was kept in mind.

1. Technologies Used
	1. Technologies Stated in Proposal
		1. Java Libraries and Technologies
			1. Java Development Kit (JDK) 1.8
			2. Java Server Faces (JSF) 2.2

JSF is a set of Java libraries that provide wrapper classes that generate html, javascript, and css, as well as provide easy AJAX communication between the client and server.

The html is generated based on xml files (with an extension of xhtml) that define the JSF html components. These components are defined in xml tags of the format “<letters:componentName attribute="value">”. Every tag must be properly terminated, either by an end tag (</letters:componentName>) or simply by a “/” before the “>” (e.g. “<letters:componentName attribute="value"/>”). For example, to output the text “Hello”, use this: <h:outputText value="Hello" />. Standard html works in these files, too, but they must also be properly terminated.

The server is managed by “Managed Beans,” which are Plain Old Java Objects (POJOs) annotated with “@ManagedBean”. The client accesses the server objects by placing the code #{beanName} in the previously described xml files.

* + - 1. PrimeFaces 5.0

PrimeFaces is a JSF library that contains both subclasses of JSF components with advanced skinning (theme) capabilities and other custom components, such as a date entry widget.

PrimeFaces 5.2 exists, but there was nothing I needed from it that PrimeFaces 5.0 didn’t do.

* + - 1. ojdbc7.jar

This file contains the necessary packages and classes to connect to and process queries on the database from another Java class. The proper Java code to connect to the database using this file is:

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection(url, user, pass);

Where url, user, and pass are all String objects.

* + - 1. Apache Tomcat 8.0.15.0

Tomcat is a popular free Java server.

* + 1. Database
			1. Oracle Database 11g Express Edition

Oracle Databases come in one of four editions: Express Edition (sometimes referred to as “XE”), Standard Edition One, Standard Edition, and Enterprise Edition. Express Edition is the only one that can be used for free in commercial, production environments. It’s limitations include a maximum of one CPU, 1 GB of RAM, and a 11 GB database. The other editions do not have a limit of RAM or database size, other than the obvious physical limits. (Oracle Database Comparisons, n.d.)

* + 1. Development Tools
			1. NetBeans IDE 8.0.2

NetBeans is a popular IDE for developing in Java. It is written in Java itself, and is developed by the owners of Java, Oracle. It is also possible to develop in other languages, such as html (or, more specific to this project, the xhtml that defines the JSF client side components), which made NetBeans ideal. (Wikipedia contributors, 2015)

NetBeans can also start Apache Tomcat, and well as handle commits to a version control repository. I used git as my version control repository, as well as GitHub to store and backup my source code, sql code, and other files.

* + - 1. GitHub for Windows 2.14.5

NetBeans only committed to a local repository. I used GitHub for Windows to send my files to GitHub itself.

* + - 1. Oracle SQL Developer 4.1.0.19

SQL Developer is a free tool from Oracle to connect, manage, generate queries, and otherwise develop any version of an Oracle database. SQL Developer was also used to generate the schema diagrams in this document.

* 1. Technologies Not Stated in Proposal
		1. Java Libraries and Technologies
			1. JSTL API 1.2

JSTL is a JSF extension that allows for standard programming statements, such as “if”, “foreach”, etc, to be inserted and properly processed in the xhtml files.

* + - 1. JUnit 4.10

JUnit is a popular testing framework used by Java developers. It allows test methods to be run systematically, and check for proper output, if desired. Tests can also be easily ignored, rather than commented out.

* + 1. Development Tools
			1. ForkHub for GitHub

ForkHub is an Android app that allows access to GitHub. It was mostly used to submit issues (or ideas, as issues) while I did not have access to the internet on a regular computer (for instance, in class).

* + - 1. Notepad++ v6.7.8.2

Notepad++ is a lightweight, syntax highlighting replacement for Windows’ standard Notepad application. It has advanced searching and replacement capabilities as well. It is very useful for reading and editing code when opening the file in the normal IDE would be too tedious and time-consuming.

1. Difficulties Faced
	1. Escaping a single quote

When inserting data, there was a person with an Irish last name, such as O’Brian. When trying to insert him, Oracle generated an error, which didn’t seem to make sense at first. The insert query went something like this:

 **insert** **into** person (name, gender) **values** ('O'Brian', 'Male');

The error generated was ORA-00917: missing comma, which made no sense because the comma is right there. In order to escape the single quote, simply replace the single quote with two single quotes (*not* a double quote):

 **insert** **into** person (name, gender) **values** ('O''Brian', 'Male');

I found the solution to this problem on stackoverflow (Escaping single quote in PLSQL, 2011).

* 1. Too many connections

Initially, for every query I created a new connection, processed my query, and then closed the connection. This worked when I was only processing a few queries. Once I was trying to load over a thousand records, around the four hundredth or so query Oracle would start refusing the connection. To solve this, I made the connection variable static, the connection variable is private and only exists in one class (Database.java in package edu.wright.hendrix11.familyTree.database). When I want to start making queries, I open the connection using the following static method in Database.java:

|  |
| --- |
| **public** **static** void openConnection() **throws** SQLException{  // Load Oracle JDBC Driver **DriverManager**.registerDriver(**new** oracle.jdbc.driver.OracleDriver()); con = **DriverManager**.getConnection(url, user, pass);} |

All queries are eventually handled by Database.java. When I am done with the connection, I close the connection with the following static method:

|  |
| --- |
| **public** **static** void closeConnection() **throws** SQLException{ con.close();} |

* 1. Too many results

Java and Oracle handle many results fine. However, loading the details of over a thousand results into a web browser caused the browser to slow down and sometimes crash. Therefore I had to limit the results sent to the browser.

1. Database
	1. Schema
		1. Tables

Generated from **SELECT** table\_name **FROM** user\_tables;

(Get list of all tables in Oracle?, 2008)

* + - 1. PERSON
			2. GENDER
			3. PLACE
			4. BIRTH
			5. DEATH
			6. MOTHER\_OF
			7. FATHER\_OF
			8. MARRIAGE
			9. SETTINGS
			10. DEFAULT\_PERSON\_TYPE
			11. PERSON\_INFO
			12. PERSON\_INFO\_TYPE
			13. SOURCE
			14. BOOLEAN
			15. BURIAL
		1. Views

Generated from **select** view\_name **from** USER\_VIEWS;

(Database Reference: ALL\_VIEWS, n.d.)

* + - 1. CHILDREN\_VIEW
			2. DEFAULT\_PERSON\_VIEW
			3. LAST\_PERSON\_INSERTED\_VIEW
			4. MARRIAGE\_VIEW
			5. PERSON\_INFO\_VIEW
			6. PERSON\_VIEW
			7. PERSON\_VIEW\_NULL
			8. SETTINGS\_VIEW
		1. Procedures

Generated from

**select** **object\_name** **from** USER\_PROCEDURES **where** OBJECT\_TYPE='PROCEDURE';

(Database Reference: ALL\_PROCEDURES, n.d.)

* + - 1. INSERT\_PLACE\_PROCEDURE
			2. INSERT\_OR\_UPDATE\_DEATH
			3. INSERT\_OR\_UPDATE\_BIRTH
		1. Triggers

Generated from

**select** **object\_name** **from** USER\_PROCEDURES **where** OBJECT\_TYPE='TRIGGER';

 (Database Reference: ALL\_PROCEDURES, n.d.)

* + - 1. MARRIAGE\_VIEW\_INSERT\_TRIGGER
			2. PERSON\_VIEW\_UPDATE\_TRIGGER
			3. PERSON\_SEQ\_TRIGGER
			4. PERSON\_VIEW\_INSERT\_TRIGGER
			5. PLACE\_SEQ\_TRIGGER
			6. PERSON\_VIEW\_DELETE\_TRIGGER
		1. Types (currently unused)
			1. CUSTOM\_DATE
1. References

Chapple, M. (2015). *Database Normalization Basics: Normalizing your Database*. Retrieved from about tech: http://databases.about.com/od/specificproducts/a/normalization.htm

*Database Reference: ALL\_PROCEDURES*. (n.d.). Retrieved from Oracle Help Center: http://docs.oracle.com/cd/B19306\_01/server.102/b14237/statviews\_2025.htm#i1585339

*Database Reference: ALL\_VIEWS*. (n.d.). Retrieved from Oracle Help Center: http://docs.oracle.com/cd/B19306\_01/server.102/b14237/statviews\_2117.htm#REFRN20305

*Escaping single quote in PLSQL*. (2011, July 13). Retrieved from stackoverflow: http://stackoverflow.com/questions/6678478/escaping-single-quote-in-plsql

*Get list of all tables in Oracle?* (2008, October 15). Retrieved from stackoverflow: http://stackoverflow.com/questions/205736/get-list-of-all-tables-in-oracle

*Oracle Database Comparisons*. (n.d.). Retrieved from Oracle: http://www.oracle.com/us/products/database/enterprise-edition/comparisons/index.html

Wikipedia contributors. (2015, July 6). *NetBeans*. Retrieved from Wikipedia, The Free Encyclopedia: https://en.wikipedia.org/w/index.php?title=NetBeans&oldid=670176137