**Project Title**

**Automated Records System for Dreams from the Slum Orphanage**

**By: Victor Oshiomah (oshi9600@stthomas.edu)**

Course: SEIS 630 - Database Management Systems & Design

**Introduction**

With this project my focus is building an automated records system for dreams from the slum orphanage home located in Lagos State, Nigeria to help manage information about **orphans**, **donors**, **volunteers**, and the **home**. However, this database can be replicated and used for any other orphanage home.

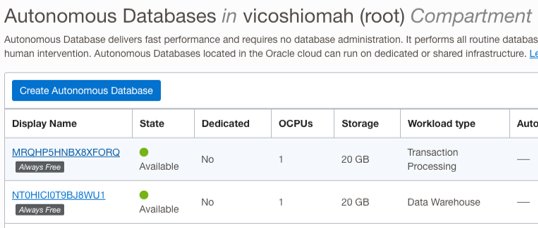
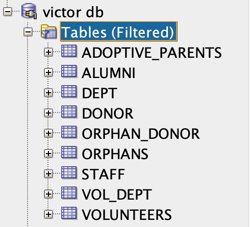
Having volunteered with this home and seen the amount of time spent on paperwork, I have always wanted to create a proper records system to ease the work of the management. I believe that with an efficient automated records system, staff of the home would spend less time on paperwork and devote more time to building personal relationships with these orphans.

I started by exploring the data needs of the home and speaking with some of the staff. The common complaint was that they spent so much time recording information about the children on paper. Sometimes they lose these records and then must start recording all over. We also discussed the entities whose records need to be kept, giving me a broad idea of what I needed to work on.

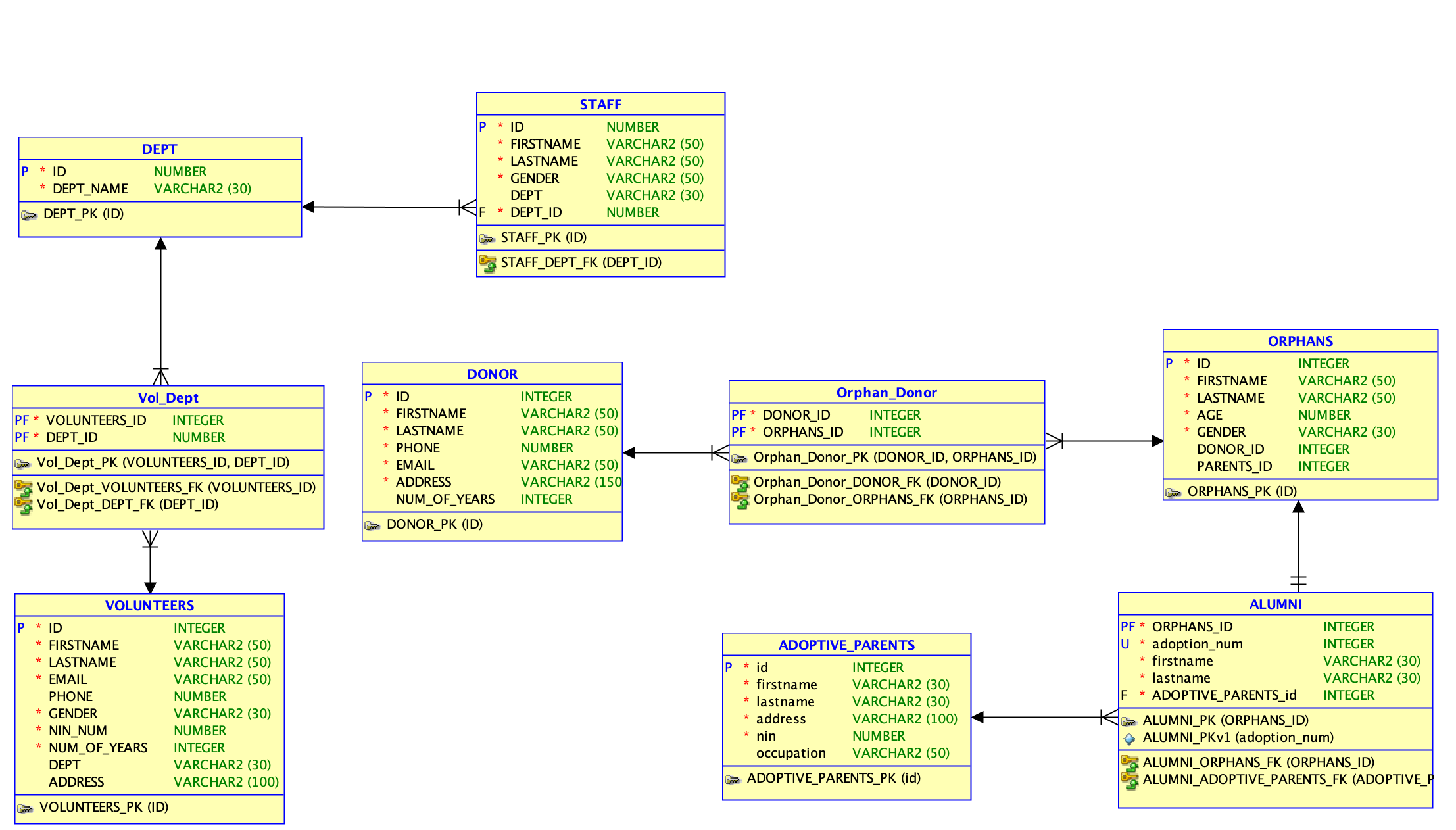
With a better understanding of the problem, I broadly listed orphans, donors, volunteers, staff, department, adoption/alumni, adoptive parents, and class as the entities that would be needed for this database. However, for the first phase of this project, the MVP would primarily focus on orphans, donors, volunteers, staff, and department entities, specifically highlighting the relationship between donor and orphans.

**Topics Explored**

1. **Setting a database on Oracle Cloud/SQL Developer:** I explored using oracle cloud to create an autonomous database and I successfully linked it to SQL Developer using the wallet I generated. I did this because I was having difficulties with exporting the DDL (Data Definition Language) statements from Live SQL. After installing my database on SQL Developer, I re-ran the queries and successfully generated my DDL statements. I also ran my DML (Data Manipulation Language) queries in SQL Developer to save me the hassle of having to run scripts on Live SQL every time I log in.

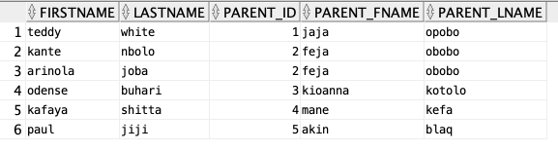
1. **Data Definition Language (DDL):** I have created tables for the 5 entities required for the MVP and 2 extra entities from my ambitious goal using create table statements as learned from this course. The tables were created on SQL Developer. I ensured that primary, and foreign key constraints were properly assigned using commands like **ALTER** to make changes to the structure of the database. I also used DROP statements to drop tables where they needed to be dropped. [Here are the DDL statements for this database](https://uofstthomasmn-my.sharepoint.com/:u:/g/personal/oshi9600_stthomas_edu/EUboWJzaK5JOgZ2aSmjQoTUBRnK35lXpCnPm3akiiQv3oQ?e=HQIbVo).
2. **Data Manipulation Language (DML):** I have populated the tables with some data using insert statements on SQL Developer. The insert statements follow the sequence taught in this course. The DML statements also highlighted the different data types learnt from this course. I also used UPDATE statements to make changes to the data within the tables. [Here are the DML statements for this database](https://uofstthomasmn-my.sharepoint.com/:u:/g/personal/oshi9600_stthomas_edu/Ec2to44v53dDq6hoUTD8bWkBlHoFS8CwvCjlE-IxhfsoNA?e=9eJUSz).
3. **Entity Relationship Diagram:** I have used this to show the relationship between entities and the inherent cardinalities. I did reverse engineering on Oracle Data Modeler using the DDL statements from Live SQL to create the logical model and ensure it has the correct attributes. I used reverse engineering on Oracle Data Modeler to create the logical model from the DDL statements, made updates to the attributes, mapped out the cardinalities, associations and supertype-to-subtype relationship. I then forward engineered the updated logical model to generate a more accurate relational model. See image of relational model below. I also explored **Normalization** here.



1. **View Statements:** I used the knowledge of creating a view to create a table that shows each child (orphan) and their corresponding sponsor. I also intend to create a view to show info about staff and departments. [Here is the query for one of the views I created](https://uofstthomasmn-my.sharepoint.com/:u:/g/personal/oshi9600_stthomas_edu/ETzZOV2KysdEq1B2HRIuhUoBCoLlJOBFA9PzvzvwbiqDug?e=r4g3cQ). See view below for orphans and their corresponding donors.



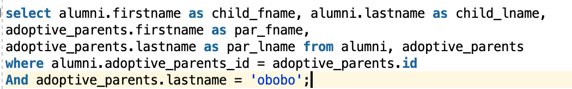
[Here is the query for alumni and their parents view](https://uofstthomasmn-my.sharepoint.com/:u:/g/personal/oshi9600_stthomas_edu/ETiiNbKHmb5Ej4BAhbPiiDABE4p86kueQvHFx3cNjqjg1A?e=7dctEB). See the view image below.



1. **SQL Join:** I used an explicit join to create the table used in the views. With this syntax I was also able to create aliases for existing columns.

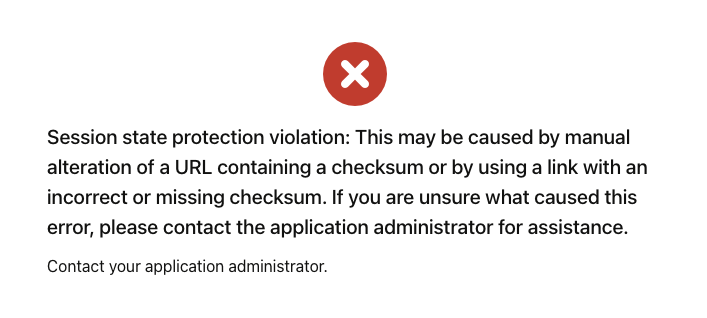


1. **Data Query Language (DQL):** I used SELECT statements to retrieve data about the entities as I worked on this database.

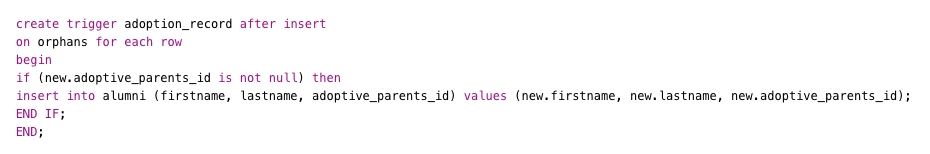


**Challenges**

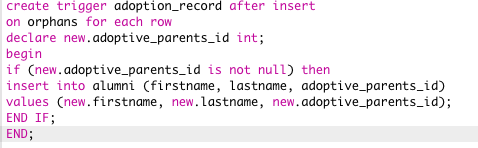
* Exporting DDL from Live SQL- I encountered issues with downloading DDL statements from SQL. When I tried, it showed the error message in the image below. To resolve this, I created my own autonomous database in oracle cloud and linked it to SQL developer. I then ran my DDL queries on SQL Developer to create the entities.



* Creating After Insert Triggers- Even though this was not a part of my MVP, I had an ambitious goal of creating a trigger that would update the Alumni table with adopted child info as soon an adoptive parent id is entered on the orphan table. My attempts so far have not yielded the desired result, so as one of my next steps, I am going to work on ensuring I create a trigger. See the different queries I tried for the trigger.



Then I modified it to this after doing some research,



Still, it was not successful, both on Live SQL and SQL Developer. The common error codes I was getting were;

Errors: TRIGGER ADOPTION\_RECORD Line/Col: 1/7 PL/SQL: Statement ignored Line/Col: 1/14 PLS-00201: identifier 'NEW.ADOPTIVE\_PARENTS\_ID' must be declared

AND

Errors: TRIGGER ADOPTION\_RECORD Line/Col: 1/12 PLS-00103: Encountered the symbol "." when expecting one of the following: constant exception <an identifier> <a double-quoted delimited-identifier> table columns long double ref char time timestamp interval date binary national character nchar

**Conclusion**

This would not be the end of this project for me as I intend to build on it with future classes from my graduate program in software. At the beginning of this class, I was a complete newbie to database design and management, and I knew next to nothing about it. Looking at what I have done with this project. I believe that I am some miles ahead of where I was at the beginning of this class.

This project particularly has helped me gain more knowledge of creating basic database systems and the functionalities that can be explored, and I am quite happy about my progress. I have found that creating the tables and inserting values myself on Live SQL and SQL Developer made it easier for me to understand what I was doing. Overall, I have tried to make use of all the tools (Live SQL, Oracle Cloud, SQL Developer, Oracle Data Modeler) I learnt about on this project, so that I can get a good grasp of how their functionalities.

I was able to achieve my MVP and build 70% of my ambitious goal as well. With an AFTER-INSERT trigger on the Orphan entity, I should have also completed my ambitious goal.

I have not been able to overcome the problem I encountered with creating an AFTER-INSERT TRIGGER, but I am fully motivated to do so as a part of my next steps.

**Next Steps**

* Create a trigger that would update the Alumni table based adopted child info as soon an adoptive parent id is entered on the orphan table.
* As recommended by my colleagues in this class, build a front-end interface for the database using Oracle Apex.
* And a class entity for the classes that each orphan is taking in the home.