Database Design and Implementation Project Report

COMP 6126

Andrew Freisthler

7/23/2015

Table of Contents

Introduction 3

Conceptual Model 3

Entity-Relationship Model 4

Text Annotations 4

Implementation Model 5

Relational Schemas 5

Description of Translation 5

Normalization 5

Text Annotations 7

Database Implementation 7

CREATE TABLE Commands 9

INSERT Commands 9

Source Code 9

Installation and Use Instructions 9

# 

# Introduction

This report is accompanies my submission for my COMP 6126 project for Summer Semester, 2015.

This project was developed using OSX 10.10.4, IntelliJ IDEA 13, and a local MySQL 5.6.22.

# Conceptual Model

## Entity-Relationship Model

## 

## Text Annotations

# Implementation Model

## Relational Schemas

**Worker** (worker\_id, hire\_date, first\_name, last\_name)

**Volunteer** (to\_worker)

**Non\_Medical\_Employee** (to\_worker)

**Staff** (to\_non\_medical\_employee)

**Administrator** (to\_non\_medical\_employee)

**Medical\_Employee** (to\_worker)

**Nurse** (to\_medical\_employee)

**Technician** (to\_medical\_employee)

**Doctor** (to\_medical\_employee, can\_admit)

**Volunteer\_Service** (volunteer\_service\_id, description)

**Volunteer\_Service\_Assignment** (to\_volunteer, to\_volunteer\_service, day)

**Non\_Medical\_Area** (non\_medical\_area\_id, description)

**Staff\_Assignment** (to\_staff, to\_non\_medical\_area)

**Patient** (patient\_id, first\_name, last\_name)

**Room** (room\_id, number)

**Diagnosis** (diagnosis\_id, name)

**Admission** (admission\_id, insurance\_policy, emergency\_contact, to\_room, to\_diagnosis, to\_admitting\_administrator, admission\_date, to\_discharging\_administrator, discharge\_date, to\_primary\_doctor, to\_patient)

**Doctor\_Assigned\_Admission** (to\_admission, to\_doctor)

**Treatment** (treatment\_id, name)

**Outpatient\_Ordered\_Treatment\_Group** (outpatient\_ordered\_treatment\_group\_id, to\_diagnosis)

**Ordered\_Treatment** (ordered\_treatment\_id, to\_treatment, to\_patient, to\_ordering\_doctor, ordered\_date, to\_outpatient\_ordered\_treatment\_group)

**Medical\_Employee\_Administering\_Ordered\_Treatment** (to\_medical\_employee, to\_ordered\_treatment, date)

## Description of Translation

For all relations, the names were changed to Camel Case separated by underscores as is my personal preference for SQL tables. All attributes are lowercase, separated by underscores as required for readability. Above, primary keys are designated by underscores, but types are not shown. They should be intuively obvious given the attribute naming.

The following schema’s are set up in a subclass nature on the diagram and thus all use ‘worker\_id’ as their key: Worker, Volunteer, Non\_Medical\_Employee, Medical\_Employee, Staff, Administrator, Nurse, Technician, Doctor. For the subclasses, the ‘worker\_id’ serves both a primary key for the subclass table as well as a foreign key to the parent table. This ensures the class relationships are maintained.

In order to describe the Volunteer’s assignment to a Volunteer Service, a total of three tables are used. One for the Volunteer themselves, described above as part of the Worker Class network, one for the various volunteer services, and a third to describe the pairing on a certain day, as it is a many to many relationship. That third table, denoted as Volunteer\_Service\_Assignment uses the combination of the foreign keys to the Volunteer and Volunteer\_Service tables along with the day as its primary key, as a Volunteer should only be able to be assigned to a certain service on one day a single time. While it technically could have been possible to use the Volunteer\_Service as its primary key, I have a strong preference for integers as primary keys so have introduced an id attribute. This would allow for the text to change without requiring updates to any foreign key relations.

To capture the Staff’s assignment to non-medical working areas, three tables were used to capure the many to many relationship, same as with the volunteer service. The Non\_Medical\_Area is given an integer id as a primary key instead of using the text and the Staff\_Assignment table uses the combination of the worker\_id (from the Staff table) and the non\_medical\_area\_id as its primary key as expected.

The Patient table is simply an id and their name. In a real system, additional informational personal information, such as their address, would be captured, but these attributes seem to be sufficient for this project. The patient\_id in this table is not just the primary key for the table, but also the unique identifier as described in the project description. It wa clarified during discussion that we could have the system assign this value any time a patient was entered into the system, whether it be an admission or an initial outpatient service, so this implementation makes sense.

Room and Diagnosis are likewise very simple tables. As done elsewhere, and id value is introduced that may not technically be required based on the specification but allows for things in the future such as room renumbering or diagnosis names to be updated without requiring any foreign key rework. This implementation technique is personal preference.

Admissions and their related data are captured in two tables. The first, Admission, uniquely captures the physical admission along with 1 to N relationspis to Room, Diagnosis, both Administorator relationships, Patient, and Doctor for the primary relationship. With the exception of to\_discharging\_administrator and discharge\_date, which will be NULL until the patient is discharged, all other attributes are required. The Doctor\_Assigned\_Admission table captures the many to many relationships where additional doctors can be assigned to patients by the primary doctor.

Treatments and their associated actions are mapped to four schemas. Treatment is a simple table, holding only an id and a treatment name. Ordered\_Treatment does the majority of the heaving lifting, capturing the 1 to N relationships to the ordering doctor, patient, treatment to be administered, and (possibly) the Outpatient\_Ordered\_Treatment\_Group, along with the ordered date and a Boolean indicating whether the treatment is for inpatient or outpatient. Medical\_Employee\_Administering\_Ordered\_Treatment captures actual adminsitrations of the ordered treatment . This is considered a many to many and thus held in a separate table under the assumption that some ordered treatments would require multiple administraions. And finally, we have the Outpatient\_Ordered\_Treatment\_Group which is in place to capture the notion that a patient can have multiple ordered treatments working toward a single diagnosis.

## Normalization

There was only one change made during the translation from my ER diagram into Schemas. Originally, the Ordered\_Treatment Schema was as follows.

**Ordered\_Treatment** (ordered\_treatment\_id, to\_treatment, is\_inpatient, to\_patient, to\_ordering\_doctor, ordered\_date, to\_outpatient\_ordered\_treatment\_group)

However, both is\_inpatient and to\_outpatient\_ordered\_treatment\_group, can be used to determine whether or not an entry ‘is inpatient’. This can be done with to\_outpatient\_ordered\_treatment\_group by testing for NULL. Thus, is\_inpatient is extraneous and can be removed. There was probably a fancy way to do that with dependency descriptions and an algorithm, but I think it was fairly easy by inspection as well.

I believe all my tables are in BCNF

## Text Annotations

For QB2 and QB3 I had to make a distinction between being admitted to the hospital and receiving inpatient services, which looks like the questions may have thought of differently.

A ‘treatment occurance’ was taken to mean the ordered treatment from a doctor which could be administered multiple times.

For D.4, associated diagnosis was taken to mean diagnosis associated with a primary doctor’s admission. For outpatient, there could be multiple doctors.

For D.6 ‘participated’ was interpreted as having actually administered part of the treatment. Ordering alone would not be sufficient to have participated.

I decided not to implement deleting in my interface due to questions on how to handle cascade issues.

# Database Implementation

## CREATE TABLE Commands

## INSERT Commands

## Source Code

## Installation and Use Instructions

For simplicity, all queries were done without using prepared satements. This is not the best for security.