PQRC Database Relational Schema

1. Introduction

This document will review the relational schema of the PQRC database solution. In particular, it will walk through the functional dependencies in each relation, and subsequently explain why each relation (and thus the entire database) is in 4NF.

2. Relations Overview

Before diving into each relation and its functional dependencies, observe the following overview of our relational schema.

mentors(last_name, first_name, class_year)

courses(id)

softwares(name)

shifts(day, time)

teaches_courses(m_last, m_first, c_id)

teaches_softwares(m_last, m_first, s_name)

works(m_last, m_first, s_day, s_time)

3. Functional Dependency Analysis

Now, let us walk through a functional dependency analysis of all of our relations.

mentors(last_name, first_name, class_year)

Given our mentors data, we can conclude the following functional dependency

last_name → first_name, class_year

As the left side of our functional dependency (*last_name*) is a key for the *mentors* relation, the *mentors* relation is in BCNF. As there are no multivalued dependencies, the *mentors* relation is also in 4NF.

Note that while *last_name* is a sufficient key for our data, we make our primary key *last_name, first_name* with the understanding that it is possible (and moderately likely) for two siblings to work at the PQRC. In this case, *last_name → first_name, class_year* would not hold as *last_name → first_name* would not hold, and *last_name* would not be a key. Thus, we choose *last_name, first_name* as our primary key.

courses(id)

As the *courses* relation has only one attribute, the only functional dependency we can derive is as follows:

id → id

As we can not derive any non-trivial functional dependencies for the *courses* relation, the *courses* relation is in BCNF. As there are no multivalued dependencies in the *courses* relation, the *courses* relation is also in 4NF.

While a relation with only one attribute may seem silly, this is a design decision that reflects an intention to further the work of this database system. For example, future iterations of this database could include a *title* attribute specifying a human-readable course title, or a separate *departments* table to recognize courses cross-listed in multiple departments. Both of these future improvements benefit from having a separate *courses* relation already specified in the database. Additionally, creating a relation for *courses* is more in line with the

entity-relationship model of database design, which is further discussed with regards to *softwares* below.

softwares(name)

As the *softwares* relation has only one attribute, the only functional dependency we can derive is as follows:

name → name

As the *softwares* relation has no non-trivial functional dependencies, the *softwares* relation is in BCNF. Additionally, as the *softwares* relation has no multivalued dependencies, the *softwares* relation is in 4NF.

Again, it may seem foolish to give *softwares* its own relation if it only has one attribute. However, giving softwares its own relation is more in line with the entity-relationship model of database design. While being more space efficient to wrap the *softwares* attribute into the *teaches_softwares* relation (discussed later), it conceptually makes more sense to create a separate *softwares* relation.

shifts(day, time)

The *shifts* relation has only two attributes. The following is the only functional dependency we can draw from our data

day, time → day, time

The *shifts* relation is in BCNF by ‘default’ because it has only two attributes. Additionally, because there are no multivalued dependencies, the *shifts* relation is also in 4NF.

teaches_courses(m_last, m_first, c_id)

As we previously discussed for our *mentors* relation, our data allows us to conclude *m_last → m_first*. However, as we can easily imagine a set of data where this functional dependency does not hold, we will assume it does not hold for our dataset. As a result, there are no nontrivial functional dependencies, and the only functional dependency we can draw is as follows

m_last, m_first, c_id → m_last, m_first, c_id

Because the left side of this functional dependency is a key for our table, the *teaches_courses* relation is in BCNF. Additionally, because there are no multivalued dependencies, the *teaches_courses* relation is also in 4NF.

teaches_softwares(m_last, m_first, s_name)

Similarly to *teaches_courses*, we assume *m_last → m_first* does not hold. As a result, there are no nontrivial functional dependencies, and the only meaningful functional dependency is as follows

m_last, m_first, s_name → m_last, m_first, s_name

Because the left side of this functional dependency is a key for our table, the *teaches_softwares* relation is in BCNF. Additionally, because there are no multivalued dependencies, the *teaches_softwares* relation is also in 4NF

works(m_last, m_first, s_day, s_time)

As with *teaches_courses* and *teaches_softwares*, we assume *m_last → m_first* does not hold. As such, there are no nontrivial functional dependencies, and the only meaningful dependency is as follows

m_last, m_first, s_day, s_time → m_last, m_first, s_day, s_time

As the left side of the functional dependency is a key, the *works* relation is in BCNF. Additionally, because the *works* relation has no multivalued dependencies, the *works* relation is also in 4NF.

As all of the relations in our database are in BCNF and 4NF, our database is in BCNF and 4NF.