Vector LTD Database Design Report

Nicholas Leslie

Due: 29/03/2019

Class: BCPR203 - Database Management Systems

Contents

[Final ERD 2](#_Toc4703305)

[Version Control 2](#_Toc4703306)

[Design Process: 3](#_Toc4703307)

[Normalisation 4](#_Toc4703308)

[1NF 4](#_Toc4703309)

[2NF 5](#_Toc4703310)

[3NF 7](#_Toc4703311)

[After Normalisation 8](#_Toc4703312)

[Final ERD 9](#_Toc4703313)

[Entities 10](#_Toc4703314)

[WorkOrder 10](#_Toc4703315)

[TestingDevice 10](#_Toc4703316)

[SignalTester 11](#_Toc4703317)

[Threshold 11](#_Toc4703318)

[Employee 11](#_Toc4703319)

[TestResults 12](#_Toc4703320)

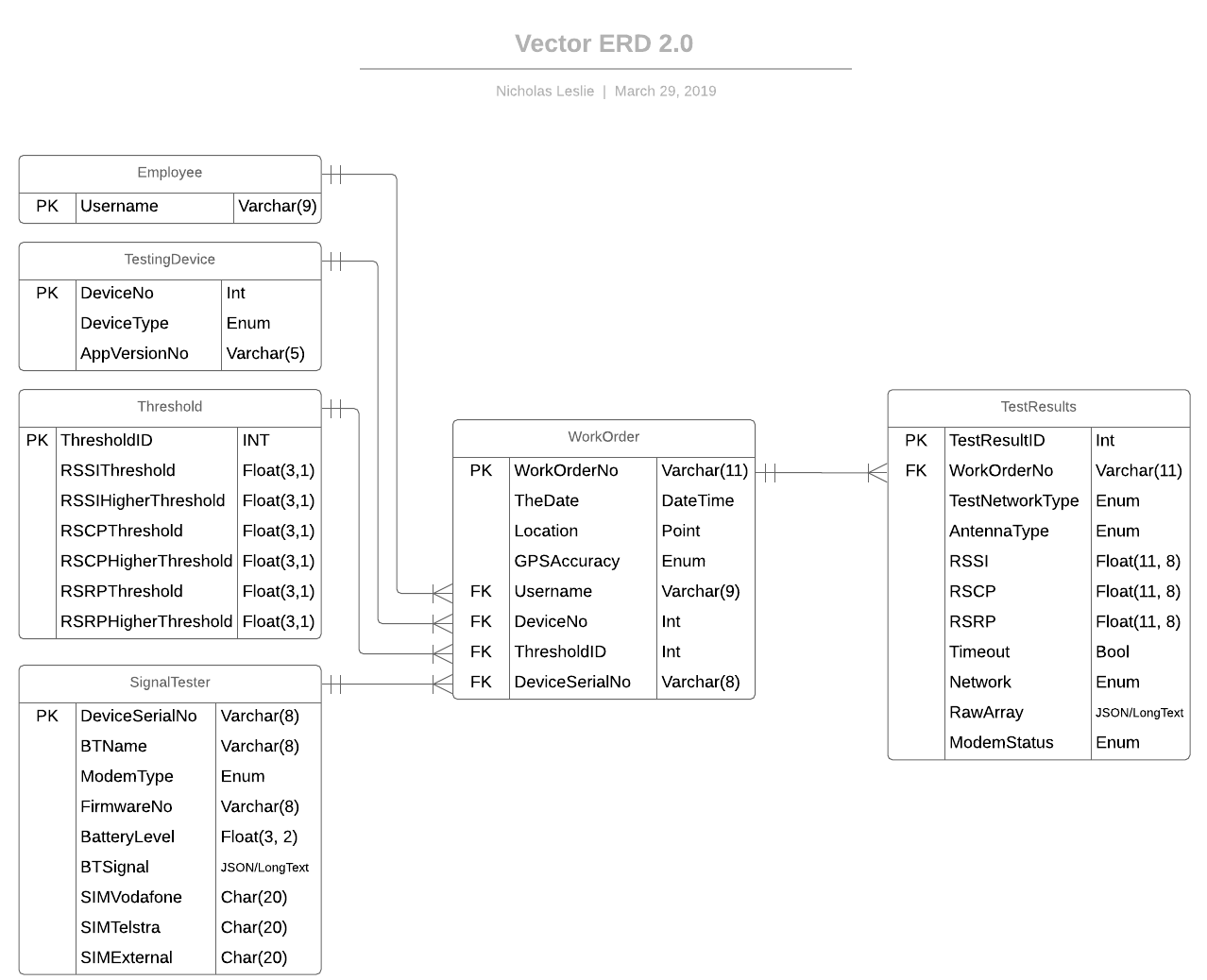
[Testing 13](#_Toc4703321)

[Reverse Engineered 13](#_Toc4703322)

[Final ERD 13](#_Toc4703323)

[Queries 14](#_Toc4703324)

# Final ERD



# Version Control

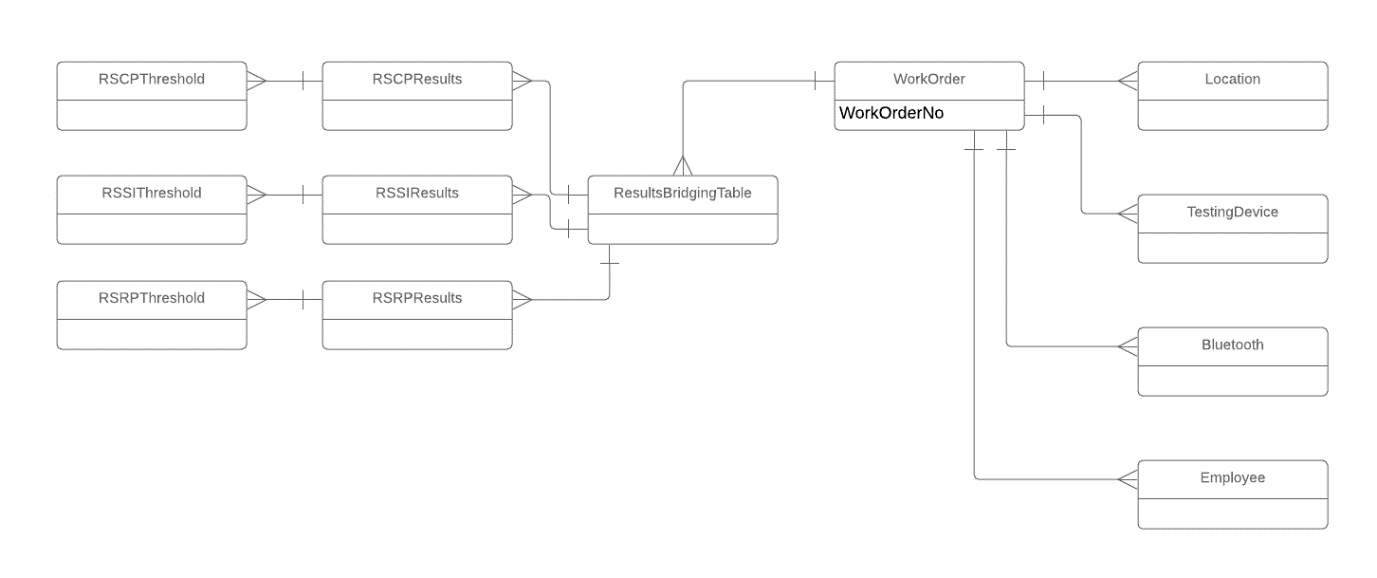
GitHub: <https://github.com/ncpleslie/VectorLTD_SmartMeter_Database>

# Design Process:

The design process started with analysing the log files and attempting to find correlations between all the data within.

I parsed the data through Excel (Using Data from Text/CSV function), and having the data laid out I could see if I could infer any patterns. I created a rudimentary ERD.

This was a quick mock up of how I thought things would connect and relate to each other. This allowed me to make steps towards understanding the data.



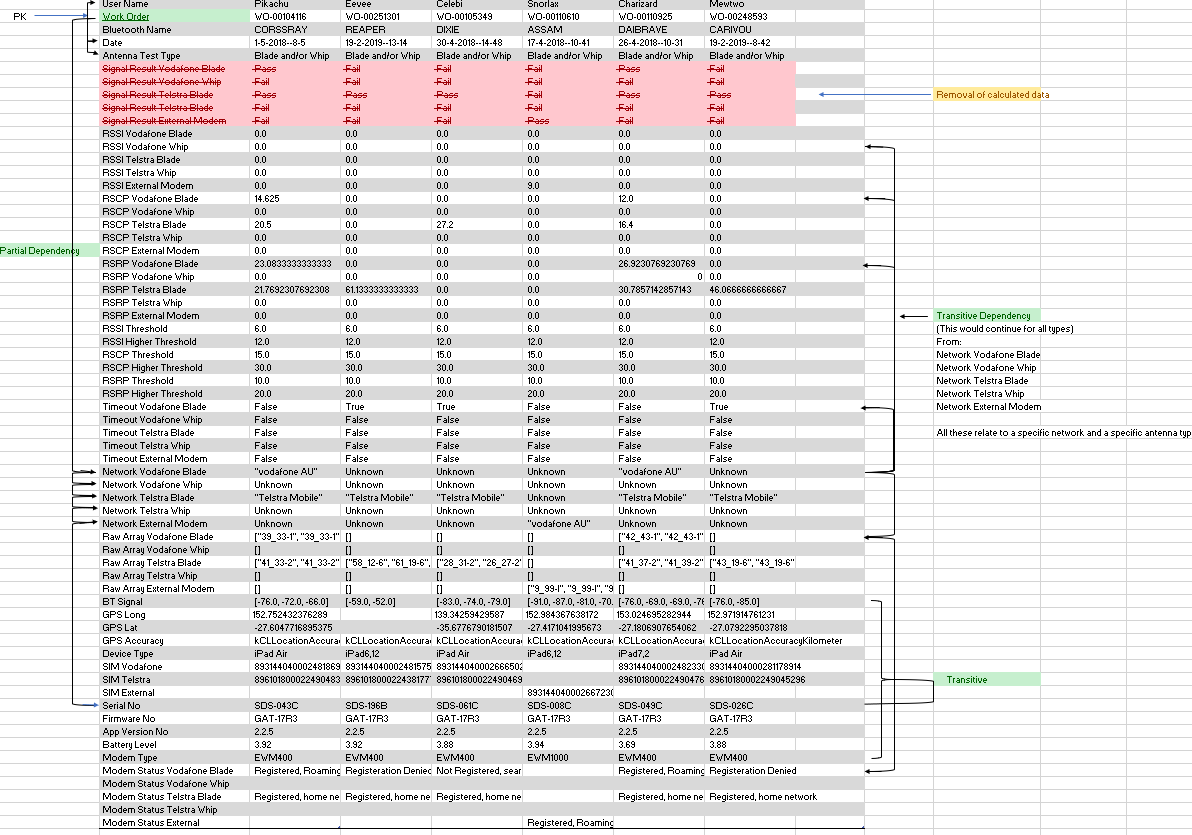
My next step was Normalisation.

# Normalisation

## 1NF

See Document (In same folder): VectorDataInExcel-1NF.xlsx

I removed calculated data from what we were given. I deduced that Signal Result Vodafone Blade, etc, where all calculated data. They were simply comparing the signal test numbers against the signal thresholds that related to those tests. Therefore, this data was removed.

Next step was to find relationships between the data. I documented this in the Excel document titled “VectorDataInExcel-1NF”. This was a growing document that I changed and adapted as I went through the process. E.g. When I found out new information such as the Signal Testing and Testing Device (iPad) were two different things.

I first noticed a unique identifier (WorkOrder) that could be used to determine everything and was unique to every log file. This was to become the primary key for my 1NF.

I could determine that WorkOrder was unique. It could be grouped with Username, date, bluetoothname, AntennaType.

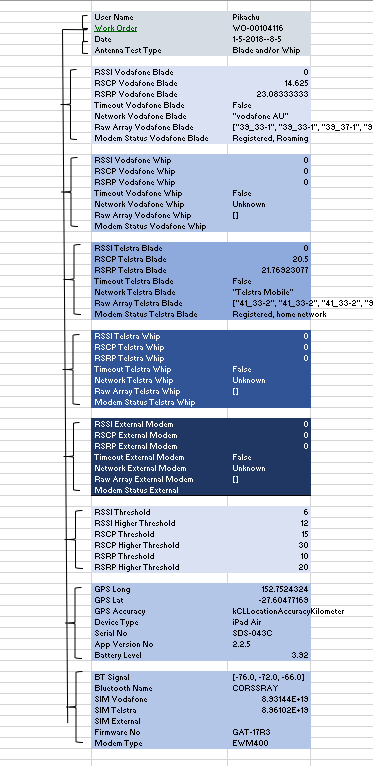
The next grouping I found was each individual networks test. I originally clumped them all together so RSSI results, RSRP results and RSCP results would all contain the respective results from all the Network Providers. From the three results tables would be a threshold table they results tables would connect to. Each results table would connect up to a bridge table which in turn would connect up to the WorkOrder table. As I went through the process, I found that this was a redundant design and would be terrible. But one thing was consistent throughout the entire process. The WorkOrder table. This remained, but its attributes changed MANY times.

I had found how things could connect to the WorkOrder PK, and then I found how things that connected to the WorkOrder PK connected to other parts of the data (Transitive Dependencies). Please see the image for clarification.

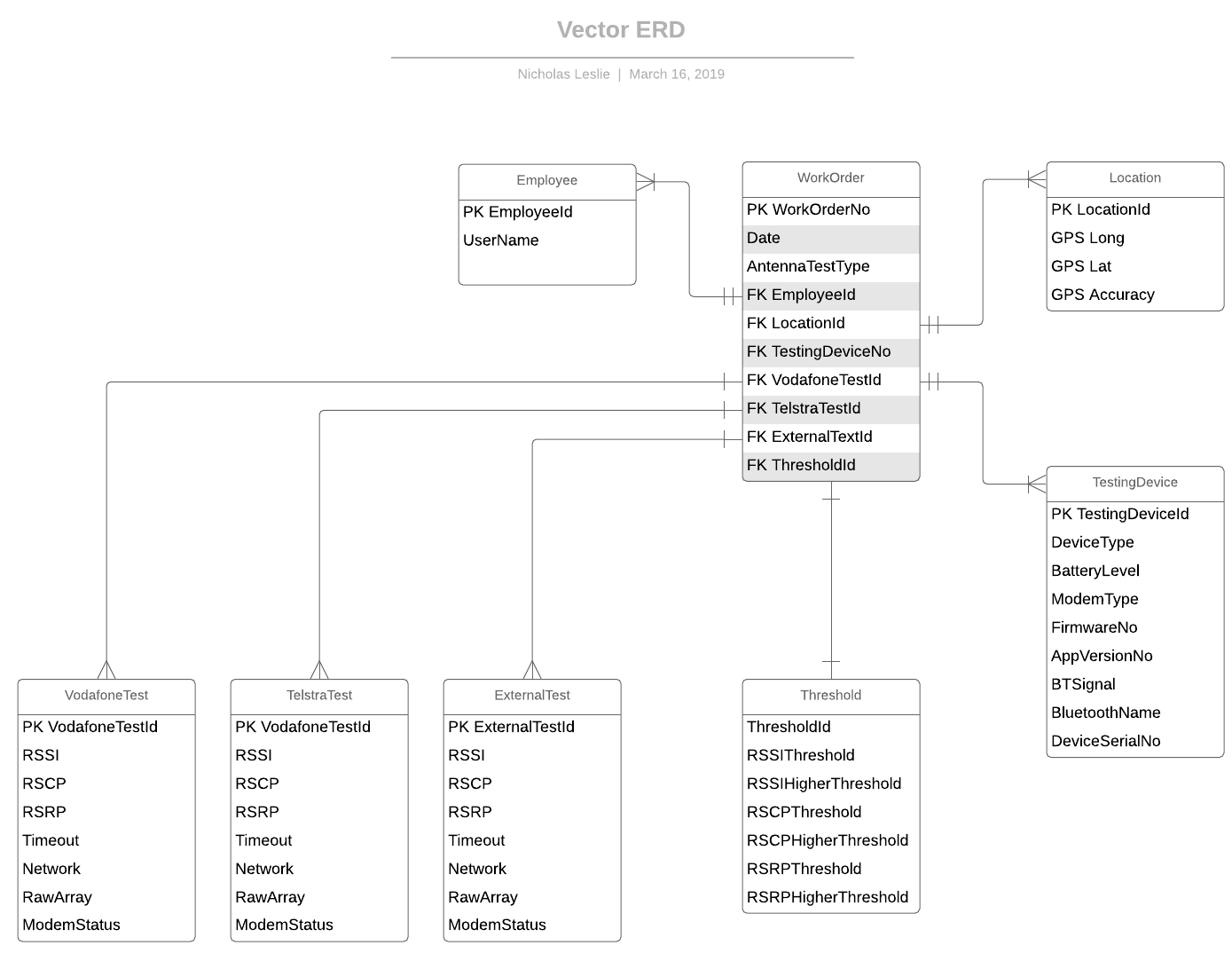
## 2NF

See Document (In same folder): VectorDataInExcel-2NF.xlsx

The 2NF is really where the “growing documentation” happens. Many changes have been made over the life of this project.

All the data was grouped together into how it would appear in the database.

This design is a far cry from what I initially thought the system would turn out to be like. This reinforces the importance of proper planning and design. I mocked up a quick ERD to see how this would play out.

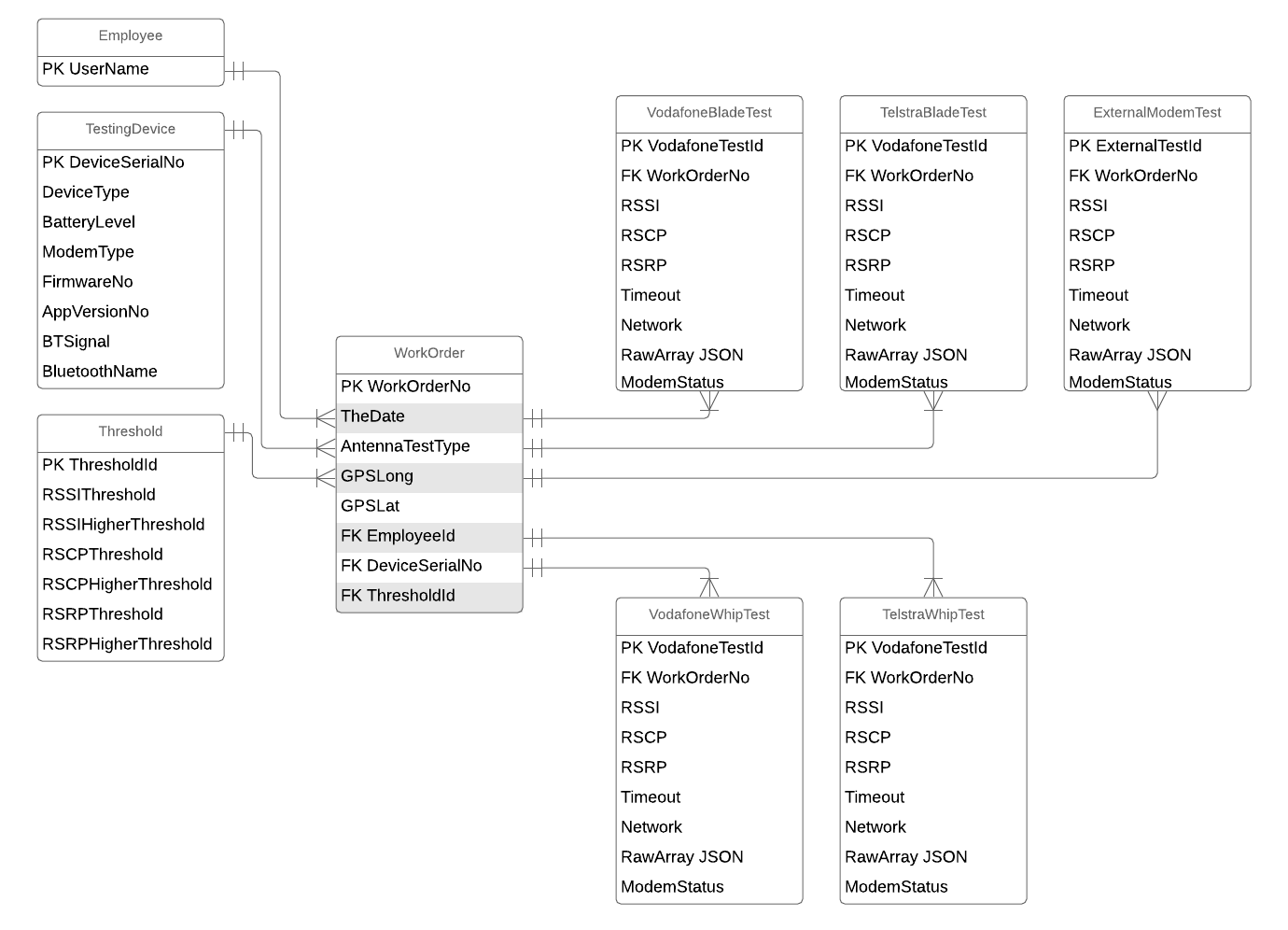


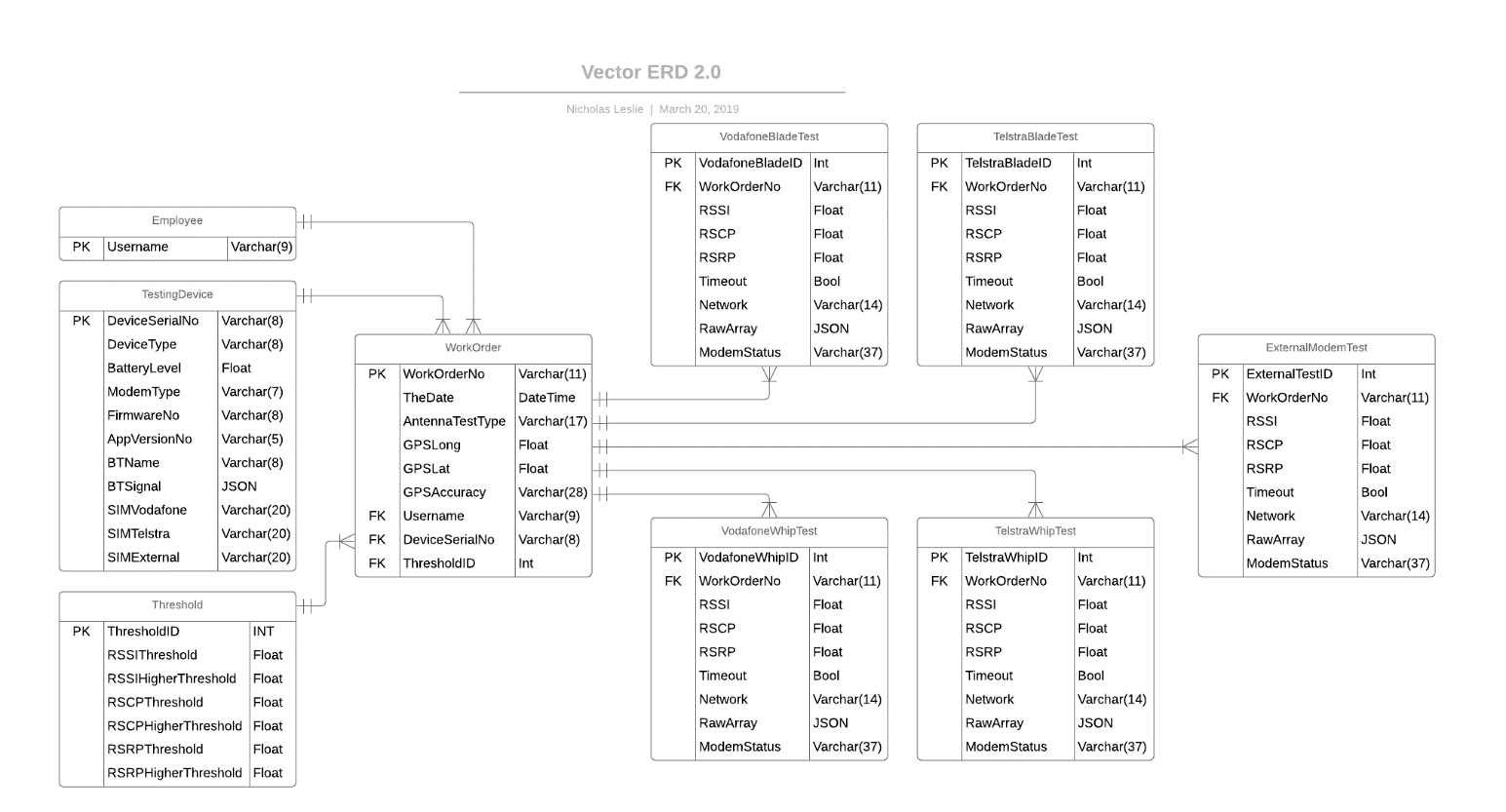
This was starting to come closer to how much final system would finally turn out. But it was still far from the end. I took the time, after creating this ERD, to plug in into MySQL. This is where I hit a roadblock. How would I connect the texting data up with the WorkOrder table? I couldn’t link the testing data up to the WorkOrder with Primary Keys. Keeping track of all the PKs would be a nightmare and add more confusion to the already complicated design. This led into 3NF.

## 3NF

See Document (In same folder): VectorDataInExcel-3NF.xlsx

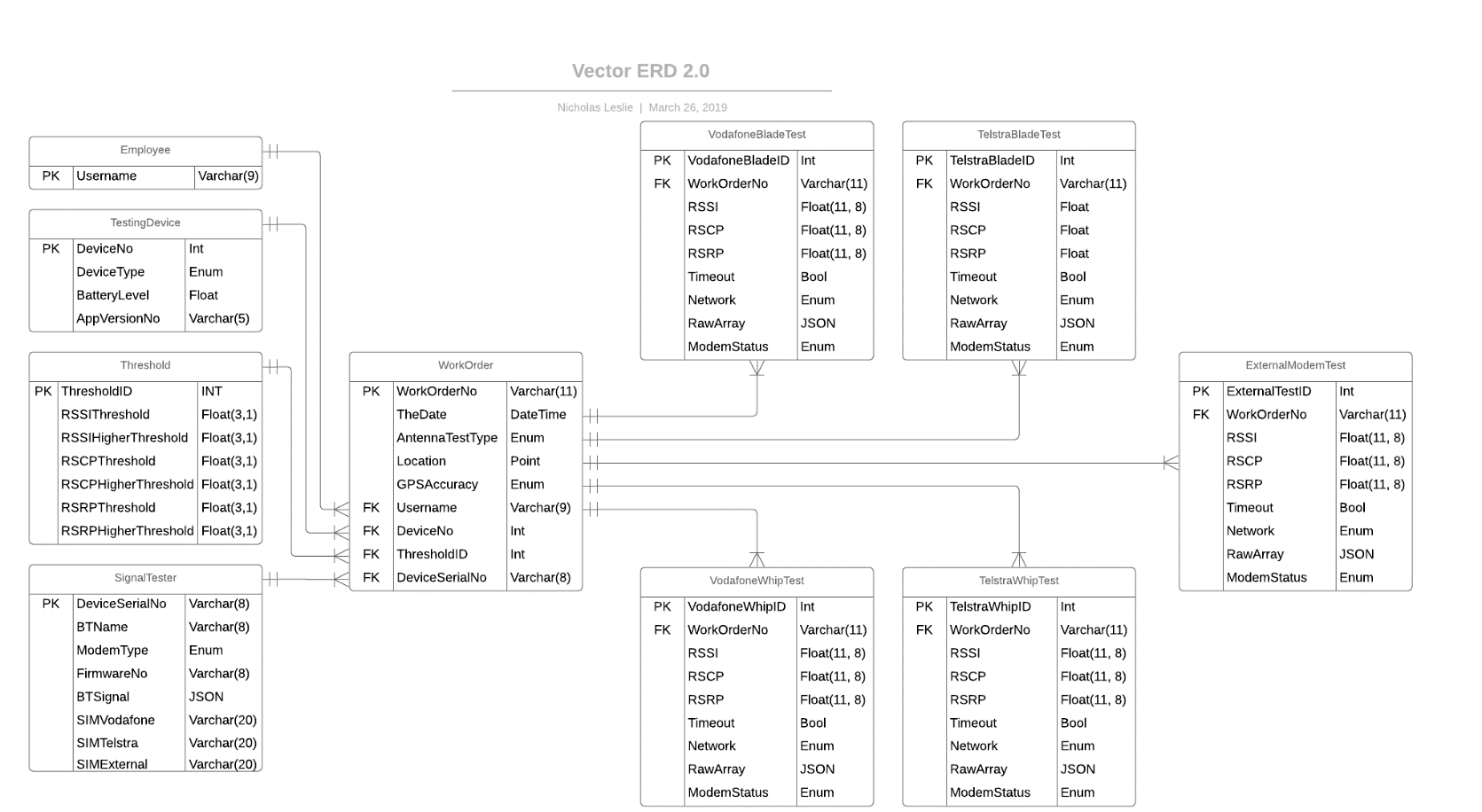
I then switched to the testing data referencing the WorkOrder (By the WorkOrderNo PK).

Breaking up the testing data based on each network, and each test antenna type (Blade or Whip), I was able to get a much clearer design. Also, merging the Location table into the WorkOrder table. My idea with the location table was that it would be able abstract data from the important tables that didn’t need to be referenced. But, I later found that to be pointless.

I also added cardinality and better relationships with proper Crows feet notation (Unlike other ERD that were mostly used just for reference). I then produced an ERD with datatypes to help with the construction of the database.

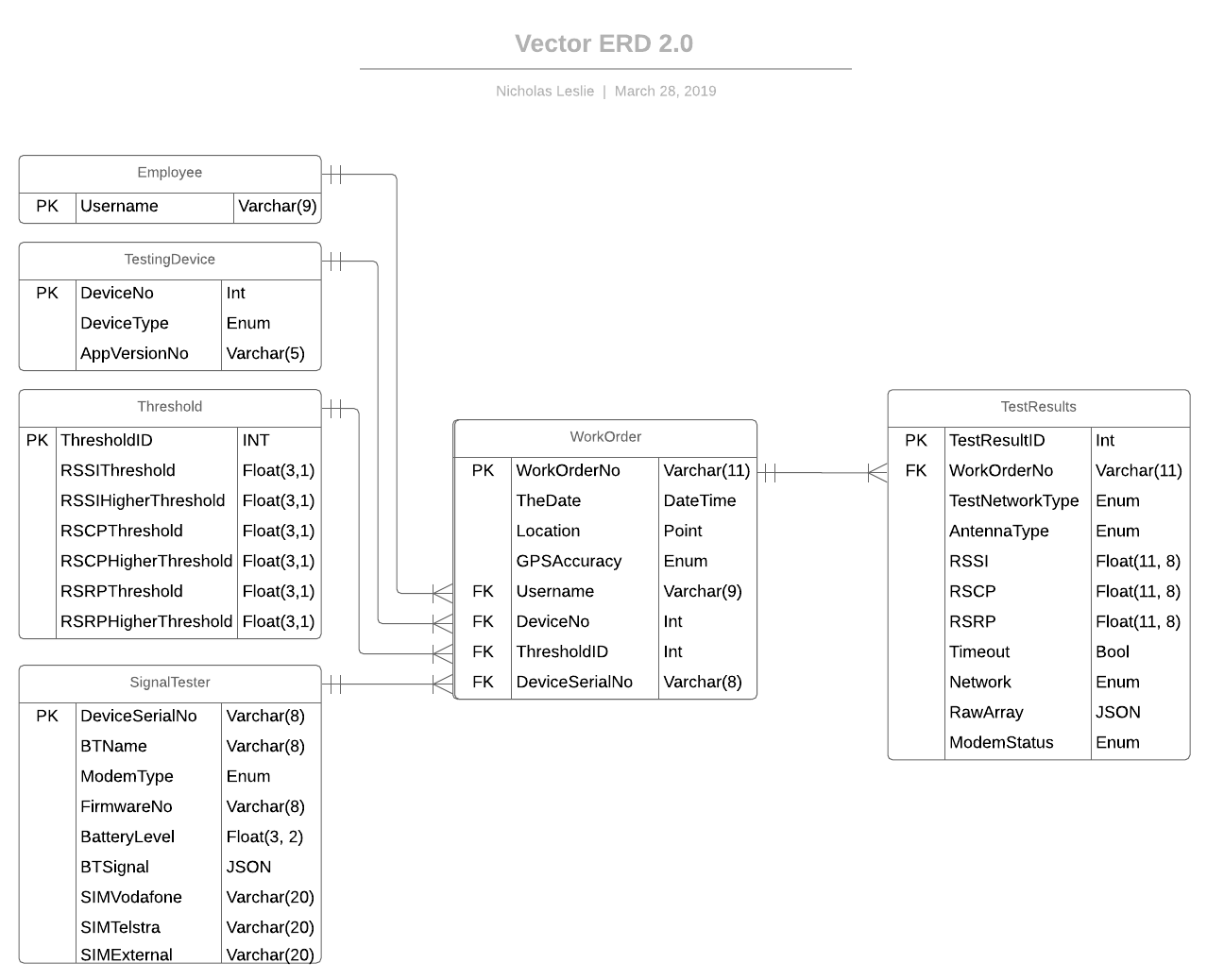
## After Normalisation

Many changes were made after 3NF and the final form. Most were minor. Things like changing datatypes. Network names (and many others. See final ERD for reference) were changed from Varchar to Enum (Only two Networks), change GPS data (Original Float) to Point, Raw Array data was changed from Varchar to Longtext (Was unable to get the JSON datatype to work. MySQL would not recognise it).

As I would find new information, such as finding out that the iPad and Signal Tester were two different devices (Originally thought it was all just part of the app), I could break things up further, and add new tables to reflect this change. At first I thought giving the Signal Tester the PK of BluetoothName would work (Its unique across all the data logs), but upon asking the employee from Vector, he states that the Serial Number relates to the Signal Tester. So, now I removed the Serial Number from the Testing Device (iPad), gave the Testing Device a unique PK, and then gave the Signal Tester the SerialNo as its PK.

As I started building the database, I noticed I was repeating tables with the same properties. One of the fundamentals of programming is not repeating yourself. I changed all the Test Results tables into one table. And I could differentiate between them with an Enum (Declaring the Network name and the Antenna type). Reducing the 5 test results tables into 1 table.

## Final ERD



# Entities

## WorkOrder

This is a table that links all the testing data (Vodafone/Telstra signal results, etc) with the data that is potentially reused (Employees, testing devices like iPads, signal tester, and the minimum thresholds need to pass the test).

WorkOrderNo: Primary Key Varchar(11)

Each job consisted of a unique identifier referred to as Work Order (e.g.: “WO-00251301”). Due to it being unique for every job it became the primary key for the WorkOrder table. Referred to as “WorkOrderNo”.

TheDate: DateTime

The date was stored in the table under the name “theDate”. Date is a reserved name so could not be used. It should be entered with a date formatter to ensure it becomes a DateTime entity.

Example:

DATE\_FORMAT(STR\_TO\_DATE("1-5-2018--8-5", "%d-%m-%Y--%H-%i"),'%Y-%m-%d %H:%i')

Location: Point

Combining GPS Lat and GPS Long into one data point. Datatype is Point. Point datatype is used in case the need to reference other geolocational data

GPSAccuracy: Enum

Denotes the accuracy. Enum is used due to references that GPS accuracy could go to “Metre” accuracy (In log files, accuracy is only to the Kilometre).

Username: Foreign Key Varchar(9)

Foreign Key reference to the Employee table

DeviceNo: Foreign Key Int

Foreign Key reference to the TestingDevice table

ThresholdID: Foreign Key Int

Foreign Key reference to the Threshold table

DeviceSerialNo: Foreign Key Varchar(8)

Foreign Key reference to the SignalTester table

## TestingDevice

The table for the iPad that connects via Bluetooth to the SignalTester

DeviceNo: Primary Key Int

Used to link the testing device (iPad) up to the WorkOrder table.

DeviceType: Enum

This is a list of different iPads. I used Enum due to if you are wanting to search by iPad type, it would be quicker than search by varchar. Enum is used to ensure data validation.

AppVersionNo: Varchar(5)

The version the app is on. Used varchar, even though its all number, because it has multiple decimals.

## SignalTester

This relates to information received from the Signal Testing box

DeviceSerialNo: Primary Key Varchar(8)

The serial number is unique too all signal testers, so is used as a primary key, and links up with the WorkOrder table

BTName: Varchar(8)

Name the TestingDevice (iPad) connects up to (In real life. Not in this table)

ModemType: Enum

Only two modems are used but more could be added. Enum is used to ensure data validation

FirmwareNo: Varchar(8)

A string of letters and numbers. Consistent length across all log files.

BatteryLevel: Float(3,2)

Even though the data is with a redundant “3” in the front (Example: 3.98), I used float in case the “3” had an importance I couldn’t infer. The battery percentage is probably what is after the decimal.

BTSignal: JSON/LONGTEXT

The signal received back from the SignalTester (to the iPad). Comes back in array format. Longtext is used, but JSON would be preferred. JSON would not work on my version of MySQL. Longtext has similar functionality to JSON

SIMVodafone + SIMTelstra + SIMExternal: Char(20)

This is large number. I used char instead of BIGINT because, other than no math or calculations being performed on these numbers, is that the numbers are above the limit of 64bit unsigned numbers of “18,446,744,073,709,551,615”.

Other number data types were considered but with the possibility of truncation/rounding, I didn’t want to risk the data changing. The SIM numbers would be worthless without exact data certainty.

## Threshold

All threshold data repeats throughout the log files. Data is only entered into this table once.

Float(3,1) is used for all attributes in this table (Apart from the Primary Key. It uses INT)

Links up to the workOrder table.

## Employee

This table could be expanded if Vector need to add more data relating to employees (Name, branch, etc)

Username: Primary Key Varchar(9)

Used as the primary key due to username being unique throughout all log files.

## TestResults

This table relates to all the test results: Vodafone Blade, Vodafone Whip, Telstra Blade, Telstra Whip, External Modem. Also, all the results they each receive: RSSI, RSCP, RSRP, Raw Array. Each test can be determined by the enum’s stating the Antenna type and Network Name.

TestResultID: Primary Key Int

Primary key for each testResult table

WorkOrderNo: Foreign Key Varchar(11)

Foreign Key reference to WorkOrder table

TestNetworkType: Enum

An enum for each network, plus to determine if using an external modem. This is reference to the desired network to connect to and each individual network’s test

Values will be Telstra, Vodafone, or External Modem.

AntennaType: Enum

Used to denote if its testing for with Whip or Blade. All log files show that they only test with Blade. Enum is used with the intention that Whip testing could happen.

RSSI + RSCP = RSRP: Float(11,8)

Float was used, over decimal, due to the double-precision decimal offers isn’t important when we are comparing the results (Which could have up to 8 decimal places) to a number that only goes to 1 decimal place. So, exact values aren’t important and the rounding that could happen shouldn’t affect the results. I have set the precision to (11, 8), so the option is there if Vector determine if they need the accuracy. Otherwise, I would recommend altering this if accuracy isn’t important.

Timeout: Bool

If test times out. Bool due to all the data being True or False

Network: Enum

The network name that the test receives back from the carrier. Only "vodafone AU", "Unknown", "Telstra Mobile".

RawArray: JSON/LONGTEXT

Originally meant to be JSON but was unable to get it to work on my version of MySQL. MySQL documentation states that it is similar to LONGTEXT, so I have set to LONGTEXT. JSON could be selected if needed.

ModemStatus: Enum

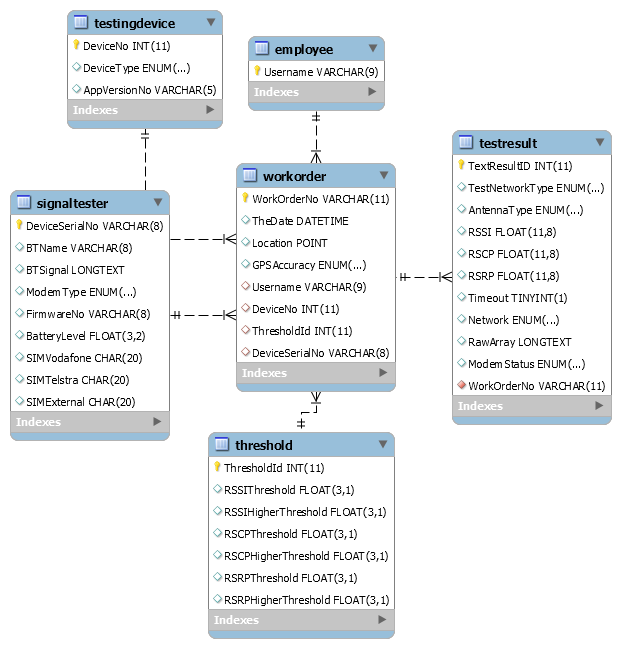
Status that is determined from the results of the test: "Registered, Roaming", "Registered, home network", "Not Registered, searching for network", "Registeration Denied". Data is copied exactly as seen in logs, including all spelling to ensure cohesiveness with all other data (If more is add beyond this assignment).

# Testing

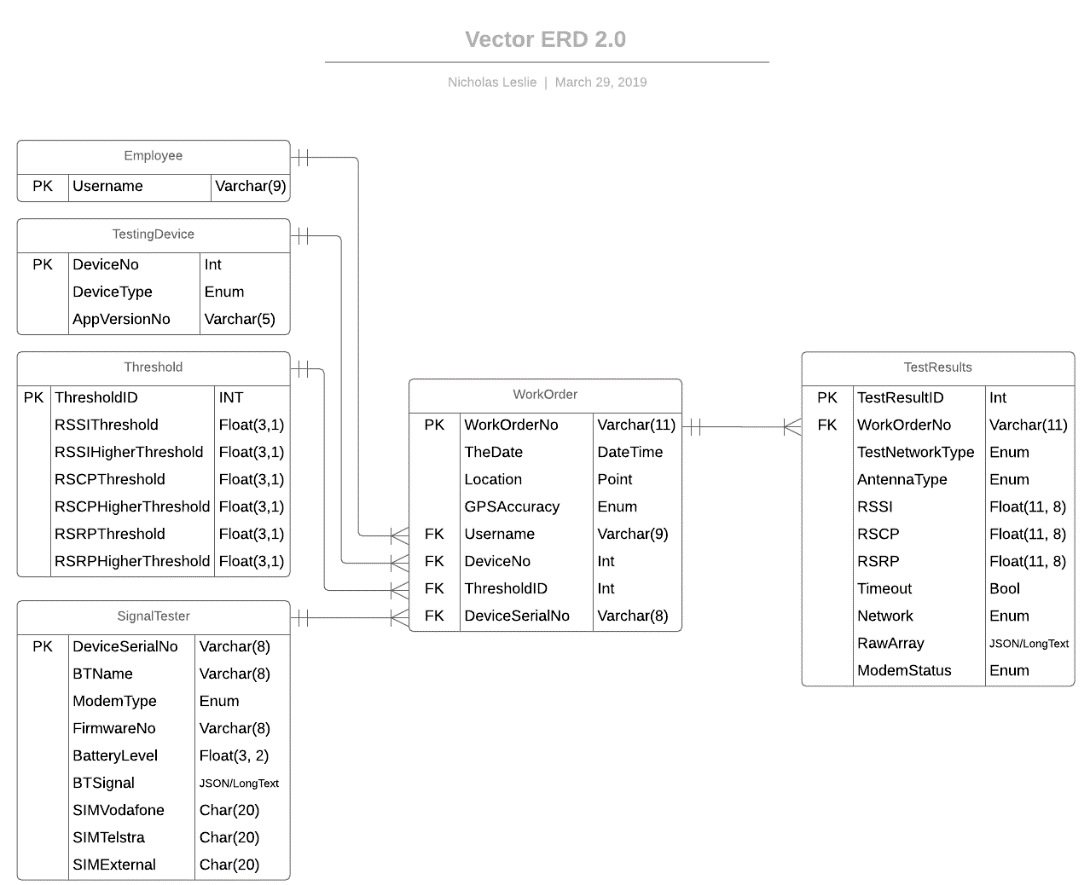
Comparing ERD to reverse engineering

As you can see, the ERD diagram I created and the final ERD of the database that I created are the same. This means that the database was created correctly and to specifications.

## Reverse Engineered



## Final ERD



# Queries

1. How many blade and whip antennas have been passed in the signal test for each

network provider? (If both blade and whip passed the test for the same Work Order

Number only consider that as blade)

SET sql\_mode=(SELECT REPLACE(@@sql\_mode,'ONLY\_FULL\_GROUP\_BY',''));

SELECT

testResult.network as "Network Name",

testResult.antennaType as "Antenna Type (Blade/Whip)",

CASE

WHEN testResult.antennaType = "Blade" AND testResult.antennaType = "Whip" THEN 0

WHEN testResult.antennaType = 'Blade' THEN SUM(1)

WHEN testResult.antennaType = 'Whip' THEN 0

ELSE COUNT(testResult.antennaType) END as "Number of Passed Antennas"

FROM

threshold,

workorder,

testResult

WHERE

testResult.TestNetworkType NOT LIKE 'External Modem' -- Excluding External Modem since out of scope of question(Neither Blade or whip)

AND testResult.workOrderNo = workOrder.workOrderNo

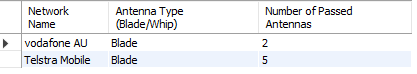
AND (testResult.RSSI >= threshold.RSSIThreshold

OR testResult.RSCP >= threshold.RSCPThreshold

OR testResult.RSRP >= threshold.RSRPThreshold)

GROUP BY testResult.network;

Result:



6. Count the number of signal log files per Bluetooth Names and print out each of the locations.

Assumptions:

Signal Log Files refers to the text documents received from Vector

SELECT

signalTester.BTName,

COUNT(workOrder.workOrderNo) AS 'Log File relating to Bluetooth Name',

ST\_X(workOrder.location) AS 'X Location',

ST\_Y(workOrder.location) AS 'Y Location'

FROM

signalTester,

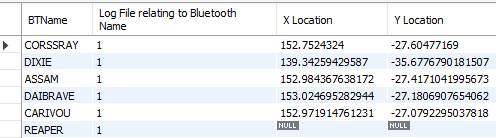
workOrder

WHERE

workOrder.DeviceSerialNo = SignalTester.DeviceSerialNo

GROUP BY workOrder.workOrderNo;

Result:



7. count the number of log files without GPS coordinates and sort them by iPad models

SELECT

COUNT(workOrder.workOrderNo) as 'Log Count',

workOrder.location,

testingDevice.deviceType

FROM

workOrder,

testingDevice

WHERE

testingDevice.DeviceNo = workOrder.DeviceNo

AND ST\_X(location) IS NULL

AND ST\_Y(location) IS NULL

GROUP BY workorder.workorderNo

ORDER BY testingDevice.deviceType;

Result:

8. count the number of whip antenna tested vs blade antenna tested

SELECT

COUNT(testResult.antennaType) as 'Antenna Count',

testResult.antennaType

FROM

testResult,

workOrder

WHERE

testResult.antennaType IS NOT NULL

AND

testResult.modemStatus IS NOT NULL

AND

testResult.workOrderNo = workOrder.workOrderNo

GROUP BY testResult.antennaType;

Result:



9. Count the number of different modem types used during the test

SELECT COUNT(DISTINCT modemType) as "No. of modem types" FROM SignalTester;

Result:



10. Count the number of number of different installers

SELECT COUNT(DISTINCT username) as "No. of installers" FROM employee;

Result:

