Final Report MS 1,2,3

Jacob Stringer

2018

Contents

[MileStone 1 5](#_Toc517709802)

[DESCRIPTION OF CONCEPTUAL MODELLING 5](#_Toc517709803)

[Chen Diagram 6](#_Toc517709804)

[2.3 DIAGRAM COMPONENTS 6](#_Toc517709805)

[2.3.1 Entity 6](#_Toc517709806)

[2.3.2 Entity Type 6](#_Toc517709807)

[2.3.3 Entity Instance 7](#_Toc517709808)

[2.3.4 Weak Entity 7](#_Toc517709809)

[2.3.5 Attribute LIST AND DESCRIBE ALL THE ATTRIBUTE TYPES 7](#_Toc517709810)

[2.3.6 Keys LIST AND DESCRIBE ALL THE KEY TYPES 9](#_Toc517709811)

[2.3.7 Relationship 10](#_Toc517709812)

[2.3.8 Optionality 11](#_Toc517709813)

[2.3.9 Cardinality 11](#_Toc517709814)

[Extended EER Notations 12](#_Toc517709815)

[CONCEPTUAL ER MODEL W/ CHEN ERD NOTATION 15](#_Toc517709816)

[DATA DICTIONARY 16](#_Toc517709817)

[Entities 16](#_Toc517709818)

[Relationships 18](#_Toc517709819)

[Attributes 21](#_Toc517709820)

[Rationale 31](#_Toc517709821)

[Contractee 31](#_Toc517709822)

[Contract 31](#_Toc517709823)

[Staff 31](#_Toc517709824)

[Stall 31](#_Toc517709825)

[UserAccount 31](#_Toc517709826)

[SunscriptionType 31](#_Toc517709827)

[DrovingZone 32](#_Toc517709828)

[BTDrone 32](#_Toc517709829)

[DataBaseEntry 32](#_Toc517709830)

[CameraViews 32](#_Toc517709831)

[MaintenanceLog 32](#_Toc517709832)

[Part 32](#_Toc517709833)

[Suppliers 33](#_Toc517709834)

[ASSUMPTIONS ABOUT BUSINESS RULES & REASONS FOR CHOICES 33](#_Toc517709835)

[Business Rules 36](#_Toc517709836)

[References 38](#_Toc517709837)

[Milestone 2 39](#_Toc517709838)

[Executive Summary 39](#_Toc517709839)

[Introduction 39](#_Toc517709840)

[Conceptual to Logical 39](#_Toc517709841)

[Mapping Rules 39](#_Toc517709842)

[Crows Foot Notation 42](#_Toc517709843)

[EER Conceptual to Logical 43](#_Toc517709844)

[Normalisation 43](#_Toc517709845)

[First Normal Form 45](#_Toc517709846)

[Second Normal Form 49](#_Toc517709847)

[Third Normal Form 54](#_Toc517709848)

[Boyce-Codd Normal Form 59](#_Toc517709849)

[Forth Normal Form 65](#_Toc517709850)

[Updated Conceptual Model 71](#_Toc517709851)

[Updates made to conceptual after Milestone 1 Feed Back 72](#_Toc517709852)

[Logical ERD 73](#_Toc517709853)

[Rational & Relations 74](#_Toc517709854)

[Data Dictionary 79](#_Toc517709855)

[Entities 79](#_Toc517709856)

[Attributes 84](#_Toc517709857)

[Derived attributes 92](#_Toc517709858)

[NaLER Analysis 93](#_Toc517709859)

[Step One 93](#_Toc517709860)

[Step Two 93](#_Toc517709861)

[Step Three 93](#_Toc517709862)

[Step 4 101](#_Toc517709863)

[Step 5 105](#_Toc517709864)

[Step 6 106](#_Toc517709865)

[Conclusion 119](#_Toc517709866)

[References 119](#_Toc517709867)

[Milestone 3 120](#_Toc517709868)

[Executive summery 120](#_Toc517709869)

[Introduction 120](#_Toc517709870)

[SQL Statements 120](#_Toc517709871)

[Analyse transactions 121](#_Toc517709872)

[Query A 121](#_Toc517709873)

[Query B 122](#_Toc517709874)

[Query C 124](#_Toc517709875)

[Query D 125](#_Toc517709876)

[Query E 126](#_Toc517709877)

[Query F 127](#_Toc517709878)

[Query G 128](#_Toc517709879)

[Query H 129](#_Toc517709880)

[Query I 130](#_Toc517709881)

[Indexes/Secondary 132](#_Toc517709882)

[File Organizations 133](#_Toc517709883)

[Controlled Redundancy 134](#_Toc517709884)

[Consequences 134](#_Toc517709885)

[Implementation 134](#_Toc517709886)

[Estimate Disk Space Requirements 135](#_Toc517709887)

[Security Mechanisms 142](#_Toc517709888)

[Design User Views 142](#_Toc517709889)

[Design Access Rules 142](#_Toc517709890)

[Monitor and Tune the Operational System 143](#_Toc517709891)

[Physical Diagram 144](#_Toc517709892)

[SQL Queries Purpose 145](#_Toc517709893)

[Query A 145](#_Toc517709894)

[Query B 146](#_Toc517709895)

[Query C 147](#_Toc517709896)

[Query D 148](#_Toc517709897)

[Query E 148](#_Toc517709898)

[Query F 149](#_Toc517709899)

[Query G 150](#_Toc517709900)

[Query H 151](#_Toc517709901)

[Query I 152](#_Toc517709902)

[Issues In counted/Resolutions 153](#_Toc517709903)

[Conclusion 154](#_Toc517709904)

[References 154](#_Toc517709905)

# MileStone 1

## DESCRIPTION OF CONCEPTUAL MODELLING

A Conceptual level of a database is used to describe the highest-level relationships between entities. At this stage you do not have to have any primary keys or foreign keys figured out yet but if there is a natural occurring key this can be marked down my underlining the attribute.

This stage is used to map out a simple structure of the data you have been given so you can give a visual representation for your Boss/Client. This includes Entities, Attributes and relationships while using Chens notation to declare relationships between entities.

The conceptual stage generally contains 9 goals with some level of flexibility dependent on the circumstance:

1. Produce a local conceptual data model
2. Identify your Entities
3. Identify the relationships between these entities
4. Identify attributes and sort them into the correct entity
5. Determine attribute domains
6. Determine the candidate and primary key attributes
7. (Optional) Specialize/Generalize entity types
8. Draw an ER diagram
9. Review this conceptual model with users/clients/boss’s

**Example**



### Chen Diagram

Another Name for a Conceptual Diagram is a Chen Diagram. Chen is the specific Cardinality we use in these diagrams which is marked with 1 of 4 ways. To get more details about these go to Page 8 **Cardinality**.

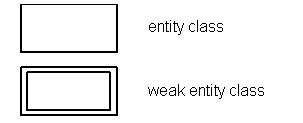
**1-1 1-M M-1 M-N**

## 2.3 DIAGRAM COMPONENTS

### 2.3.1 Entity

An Entity can be anything a person, object, building etc. It is something you are going to store information about E.g. Entity is a Car you may want to store colour, type, rego etc.

Within a conceptual Diagram, Entities are depicted as a rectangle and weak entities are shown as a rectangle within a rectangle as seen below.

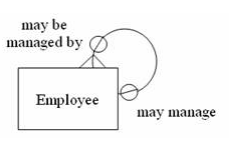


Weak entities put simply are again things of another thing that cannot exist without the Main Thing E.g. The main entity is a Supermarket and weak entities are departments within that supermarket. The departments cannot exist without the supermarket, so they are considered weak.

### 2.3.2 Entity Type

An entity type is the defining characteristic of the dependency of the entities information. There are 4 different applicable Entity types within a Database.

* **Strong Entity**
  + Strong entities are the most common type of entity. This is the entity that can exist by itself unlike the week entity as I mentioned earlier.
* **Weak Entity**
  + Weak entities are always found connected to strong entities. These entities are Things within the strong entity E.g. The main entity is a Supermarket and weak entities are departments within that supermarket. The departments cannot exist without the supermarket, so they are considered weak.
* Recursive Entity
  + Recursive entities are entities that have a relationship with itself



* Composite Entity
  + Composite entities are only used in a Java EE software design pattern.

### 2.3.3 Entity Instance

An entity instance is a manifestation of an entity within that category.

For example, *Cell* could be the entity type, but *Cell\_1*, *Cell\_2*, and *Cell\_3* would represent the actual items within the network.

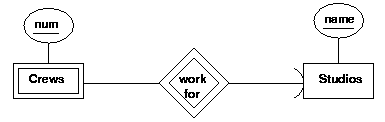
(“IBM Knowledge Center - entity instance,” n.d.)

### 2.3.4 Weak Entity

A weak entity is an entity type which depends on another entity to function within the database.

Another way to put it would be that you have an entity called Computer and then have a weak entity called Computer Parts. This is weak because it is a part of the main entity and could not exist without.

**Example.**



### 2.3.5 Attribute LIST AND DESCRIBE ALL THE ATTRIBUTE TYPES

**Composite Attributes**

A composite attribute is an attribute that can be split up into different parts E.g. Name is the 1st attribute and first name and last name can be attributes coming off the name attribute.



**Single Value Attributes**

A single value attribute is an attribute that only holds one value and does not split off like a composite attribute would E.g. You store an employee’s gender this cannot be split into different parts there’s only the gender.



**Multi Valued Attributes**

Multi valued attributes are attributes that can hold many values unlike a single valued attribute that can only hold one E.g. You may want to store the date and time something was finished or completed.

These attributes are shown a bit differently than the others they are depicted as an oval within an oval as shown below.

https://lh6.googleusercontent.com/YHUoOka273xMs8uqTn23bGnbWpaRPDqqKerMbNM2MNnP_j7J97qrD6dVngXsCmebfZZZx29YTR8pisEbnDwDJCCPLRQVXuxvKmiNJoc53z7_KbSWPUPHlO50t7G8f33OMhxcnVrK

**Derived Attributes**

A derived attribute is an attribute that does not have data in it to begin with. It is an attribute that uses other attributes to calculate the data within this field. E.g. You only ask a customer for their date of birth This then would calculate their age by figuring out the gap between the date provided and todays date and input the data into a derived attribute.

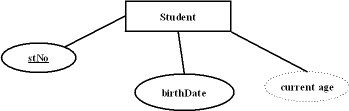
These attributes are shown a bit differently than the others they are still shown as an oval but it’s a dotted oval as shown below.

https://lh6.googleusercontent.com/YhhnXw4KRBgqMr5lHtRVOvWiiyP1Gk94pR9OkALchRxAWOSxJE3yYBe3scCWoFBqbiYcT5b5VLrl7m5zmLC-BNwidf30FeYFFo66rNUGkvfppJhUJFdZKcrQN2E8fbxynfm2FXmc

**Stored Attributes**

Stored attributes are the attributes that store the data that derived attributes pull from to calculate

These attributes are shown just like single valued attributes you can need to make sure that there is always one stored attribute connected to the same entity if there is a derived attribute connected to that entity also.



### 2.3.6 Keys LIST AND DESCRIBE ALL THE KEY TYPES

**Key**

A key is a unique piece of data that can be used to identify each individual entry in a database

**E.g.** A Number or Username

**Candidate Key**

A Candidate key is a key that allows an individual column to identify the uniqueness of a row/tuple of data

**E.g.** If you had an Employee No and a Phone number column you could use them both to identify a particular row/tuple.

**Super Key**

A super key is made when you add another column or attribute to the data base and include it within the primary key.

**E.g.** If you only had Employee No as a primary key and you added in the Phone number you could make phone number another primary key within that table.

**Primary Key**

A primary key is a column or group of columns that you choose to uniquely identify each row/tuple of data within a table

**E.g.** You use Employee No to identify each Different personal detail’s stored in the data base.

**Composite Key**

A composite key occurs when you have a table that cannot be uniquely identified by just one column. You create a composite key when you use two columns in one table to uniquely identify each individual row/tuple.

**E.g.** If an Employee No did not exist you could use a Name and phone number as a composite key to uniquely identify each employee within the data base.

**Alternate Key**

An Alternate key is another column within the table other than your already chosen candidate key that could be used as a primary key if needed.

**E.g.** If you only had Employee No as a primary key you could also use their phone number as a key making it an alternate key.

**Foreign Key**

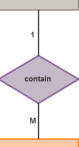
A foreign key can be one or more fields within one row/tuple that will create a relationship with another table. This would allow you to separate data into more manageable tables that can still be linked to each other.

**E.g.** If you had a supermarket instead of keeping each department stored within that table you could create a new table and store them in there using a foreign key, so you still know what supermarket this department belongs to.

### 2.3.7 Relationship

A relationship in data modelling is when you create a link between two different entities. You would do this if you require data to be shared or linked between two different entities

**E.g.** As per the example I gave above in the foreign key section You would have one entity named Supermarket and another entity named Departments You would create a 1-M relationship between them so that 1 supermarket contains many departments.



### 2.3.8 Optionality

In database modelling Optionality is a term used when you want to define if a relationship between two entities are optional or mandatory. Or another way to look at it is if an entity must have a full relationship with an entity or just an optional one.

These are mainly used between strong and weak entities because a weak entity cannot exist without a strong entity so, the relationship between them would always be mandatory or if the relationship does not affect the existence of each entity they can be considered optional because if the other entity it is linked to did not exist it would not affect it in anyway.

E.g. The relationship between Supermarket and Department. The relationship going towards department would be optional because supermarket would still exist without a department. The link from Department to Supermarket would be mandatory because a department cannot exist without a supermarket.

On a diagram a mandatory relationship is indicated by a double line in the relationship and an optional relationship would be integrated by a single line. As seen in the Example diagram below.



### 2.3.9 Cardinality

In database modelling Cardinality is the term used to describe the number of relationships one entity to have to another entity. There are four different possible cardinality relationships you could use between entities.

* One to One (1-1)
  + **E.g.** One Employee is a part of One Department.
* One to Many (1-M)
  + **E.g.** One Supermarket can have Many Departments.
* Many to One (M-1)
  + **E.g.** Many departments must have One Supermarket.
* Many to Many (M-N)
  + **E.g.** Many Departments has Many Employees.

You can see the example diagrams above to get a visual representation of cardinality on a table and below in my Conceptual diagram.

### Extended EER Notations

There are 3 different types of Extended Entity Relationships and they are Disjointed, Overlap and Union.

But first what are EER notations. Extended Entity Relationships can be used within conceptual modelling the whole point for using this technique is to try make your diagram and in the end database easier to manage. This is used by splitting your main entities into multiple separate parts/tables to be able to manage them better.

When used there must be a backwards “C” on each line a part of the EER as seen in the examples below.

**Disjointed**

Disjointed Notations are used when you want to break up one entity and simplify it into categories. For example, if you have a staff entity you can split them off in to General Staff and Management. On the diagram this is marked with a small “d” If this is used you can only be one or the other so as above you can only be a general employee or management not both.

**Overlap**

Overlap Notations are pretty much the same as above you take an entity and split it up into more manageable parts. But unlike Disjointed where you have to be one or the other Overlap you can be a part of the multiple different parts you create.

For example, a person could be both a student and an employee of an educational institute.

When you use an overlap, you make this as an “o” within your diagram.

**Example Diagram.**

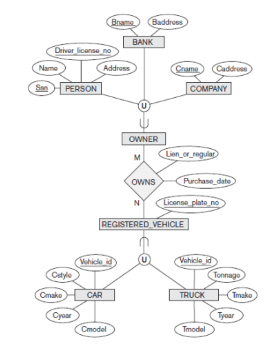
Machine generated alternative text:
Name 
Year 
PERSON 
ALUMNUS 
BirthDate 
STA F F 
Position 
Salary 
EMPLOYEE 
d 
FACULTY 
Rank 
Project 
MajorOept 
STUDENT 
Degrees 
Degree 
percentTime 
STUDENT_ 
ASSISTANT 
Major 
Course 
GRADUATE_ 
STUDENT 
DegreeProgra m 
d 
UNDERGRADUATE_ 
STUDENT 
Class 
RESEARCH ASSISTANT 
TEACHING_ASSISTANT 

**Union**

A Union simply put is when you wish to join multiple entities into one.

For Example, if you had 2 entities for a company car Person and Company and you wanted to combine them into one entity Owner you could use a Union notation to do this. It’s like when you have a many to many relationships in a conceptual diagram and you must create a joining table between the two for the logical stage. That one table in the middle will have all the foreign keys of the other entity’s and for our Owner a ID so some sort to connect It altogether

**Example Diagram.**

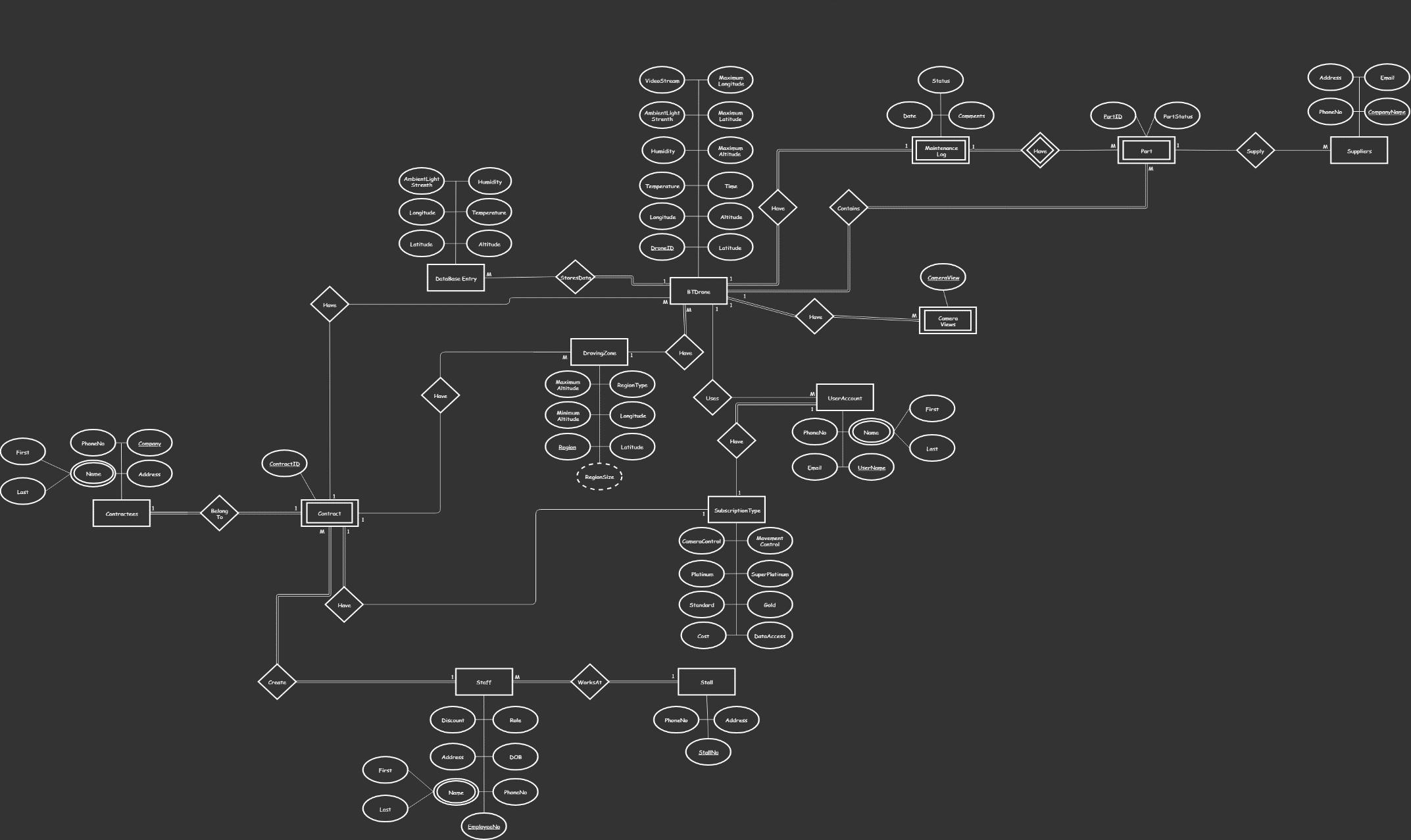


**How I could have incorporated this into my diagram**

Due to time constraints and not clearly understanding the instructions at the beginning I did not incorporate this technique into my current Conceptual Diagram. Here I will explain what I could do to implement this onto my current diagram if I were to do so. Reference the diagram bellow.

* I could of spit up the staff entity using a Disjointed notation. This could have been spit up into 2-3 parts One for staff that work in the store Two for executive staff and management and 3 Engineer staff that would maintain the drones/Order parts.
* For subscription type I could have used a Disjointed Notation. I could have used this buy taking all the different types that are right now attributes and make them into their down entities.
* I think I could have used Overlap on User account by having Contractee and subscriber seeing that they could be both if necessary and then connect Contractee to contract like I do have right now.
* If required I could have split Stall in to two Shop and Offices if they happened to have a mixture of shops/offices around the country. I would have used a disjointed notation because there could possibly be offices above a shop.

## CONCEPTUAL ER MODEL W/ CHEN ERD NOTATION



## DATA DICTIONARY

### Entities

The purpose for this table is to describe the identified entities in more detailed way than an ER diagram can show, including a description of what the entities purpose is and/or what it stores. Aliases is used to list other ways this entity may be referred to within your documentation, occurrence is basically the relationships the entity has although it is more focused on describing the logic of the relationship rather than just the listing the multiplicity constraints.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Name** | **Description** | **Aliases** | **Occurrence** |
| **Contractees** | Basic Company information purchasing a contract | Company | * One Contractee must belong to One Contract |
| **Contract** | Contact for Control and access to data of a BTDrone from a location of the company’s choice |  | * One Contract must belong to One Contractee * Many Contracts must be Created by One staff member * One contract must have One SubscriptionType * One Contract can have Many DrovingZones * One Contract Can have Many BTDrones. |
| **Staff** | Stores Being there staff information | Shop | * One Staff Member Must Create Many Contracts * Many Staff Members Must work at One Stall |
| **Stall** | A shop that will sell subscriptions |  | * One Stall Must Have Many Staff Members |
| **UserAccount** | An account to be used by a subscriber to access BTDrones Cameras |  | * Many UserAccounts can Use One BTDrone * One UserAccount Must Have One SubscriptionType |
| **SubscriptionType** | Different Types of subscriptions customers can purchase |  | * One SubscriptionType Can Have One Contract * One SubscriptionType can have One UserAccount |
| **DrovingZone** | An Area where a BTDrone can move around |  | * One DrovingZone Can have Many BTDrones * Many DrovingZones Can have One Contract |
| **BTDrone** | A BTDrone that stores data of the surrounding area and provides video streams |  | * Many BTDrones Can Have One Contract * One BTDrone Must Store Data in Many DataBase Entry’s * One BTDrone Must Have One Maintenance Log * One BTDrone Must Contain Many Parts * One BTDrone Must Have Many CameraViews * One BtDrone Can be Used by Many User Accounts * Many BTDrones Must Have One Droving Zone |
| **Database Entry** | Data sent by a BTDrone to be stored for use later |  | * Many Data Base Entry’s Is Stored by One BTDrone |
| **CameraView** | Live cameras used by subscribers that show the surrounding area |  | * Many CameraViews Must Have One BTDrone |
| **MaintenanceLog** | Log entry of the status of each drone |  | * One Maintenance Log Must Have One BTDrone * One Maintenance Log Must Have Many Parts |
| **Part** | List of each individual part of a BtDrone |  | * Many Parts Must be A Part of One BTDrone * One Part Can be Supplied by Many Suppliers * Many Parts Can Have One Maintenance Log |
| **Suppliers** | Companies that supply parts for the BTDrones |  | * Many Suppliers Can Supply One Part |

### Relationships

The purpose for this table is to display the relationships and multiplicity between entities in a table format.

Using this medium it is easier to digest the amount of relationships your ER model contains as well as the type of relationships they have with other entities.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Entity Name** | **Participation** | **Cardinality** | **Relationship** | **Cardinality** | **Participation** | **Entity Name** |
| **Contractees** | Mandatory | 1 | Must Belong To | 1 | Mandatory | **Contract** |
| **Contract** | Mandatory | M | Must Be Created By | 1 | Mandatory | **Staff** |
|  | Mandatory | 1 | Must Have | 1 | Optional | **SubscriptionType** |
|  | Mandatory | 1 | Must Have | 1 | Mandatory | **Contractees** |
|  | Optional | 1 | Can Have | M | Optional | **BTDrone** |
|  | Optional | 1 | Can Have | M | Optional | **DrovingZone** |
| **Staff** | Mandatory | M | Must Work At | 1 | Mandatory | **Stall** |
|  | Mandatory | 1 | Must Create | M | Mandatory | **Contract** |
| **Stall** | Mandatory | 1 | Must Have | M | Mandatory | **Staff** |
| **UserAccount** | Optional | M | Uses | 1 | Optional | **BTDrone** |
|  | Mandatory | 1 | Must Have | 1 | Optional | **SubscriptionType** |
| **SubscriptionType** | Optional | 1 | Can Be Used By | 1 | Mandatory | **UserAccount** |
|  | Optional | 1 | Can Be Used By | 1 | Mandatory | **Contract** |
| **DrovingZone** | Optional | 1 | Can Be Used By | M | Mandatory | **BTDrone** |
|  | Optional | M | Can Be Used By | 1 | Optional | **Contract** |
| **BTDrone** | Mandatory | 1 | Must Store Data In | M | Optional | **DataBaseEntry** |
|  | Optional | M | Can Be A Part Of | 1 | Optional | **Contract** |
|  | Mandatory | M | Must Have | 1 | Optional | **DrovingZone** |
|  | Optional | 1 | Can Be Used By | M | Optional | **UserAccount** |
|  | Mandatory | 1 | Must Have | M | Mandatory | **Cameras** |
|  | Mandatory | 1 | Must Contain | M | Mandatory | **Part** |
|  | Mandatory | 1 | Must Have | 1 | Mandatory | **MaintenanceLog** |
| **DataBaseEntry** | Optional | M | Can Store Data From | 1 | Mandatory | **BTDrone** |
| **CameraView** | Mandatory | M | Must Have | 1 | Mandatory | **BTDrone** |
| **MaintenanceLog** | Mandatory | 1 | Must Have | 1 | Mandatory | **BTDrone** |
|  | Mandatory | 1 | Must Have | M | Optional | **Part** |
| **Part** | Mandatory | M | Must Be A Part Of | 1 | Mandatory | **BTDrone** |
|  | Optional | 1 | Can Be Supplied By | M | Optional | **Suppliers** |
|  | Optional | M | Must Have | 1 | Mandatory | **MaintenanceLog** |
| **Suppliers** | Optional | M | Can Supply | 1 | Optional | **Part** |

### Attributes

The purpose for this table is to go in-depth on the attributes our identified entities will contain, this includes a wealth of detailed information for each attribute allowing a database designer to make a preliminary draft on the physical specifications their data will possess. Doing this type of process early may allow one to discover any errors or consistency early which can be very valuable in the process of creating databases or any information systems in general, recognizing problems early can save a lot of effort down the road.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Description** | **Domain** | **Aliases** | **Composite** | **Derived** | **Nulls** | **Key?** | **Default Value** |
| **Contractees** | Company | Company’s Name | Characters  (30) | - | No | No | No | PK | - |
|  | Address | Company’s Address | Characters  (50) | - | No | No | No | - | - |
|  | Phone No | Company’s Phone number | Integer | - | No | No | No | - | - |
|  | Name | Name of Person Requesting Contract | Characters  (40) | - | Yes  First,  Last | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **Contract** | Contract ID | Unique ID for The Contract | Integer | - | No | No | No | PK | - |
|  |  |  |  |  |  |  |  |  |  |
| **Staff** | Employee No | Unique ID for Each Employee | Integer | - | No | No | No | PK | - |
|  | Name | Employees Name | Characters  (40) | - | Yes,  First,  Last. | No | No | - | - |
|  | Address | Employees Address | Characters  (50) | - | No | No | No | - | - |
|  | DOB | Employee Date of Birth | Date | - | No | No | No | - | - |
|  | Phone No | Employees Contact Number | Integer | - | No | No | No | - | - |
|  | Role | Employees Work Position | Character  (30) | - | No | No | No | - | - |
|  | Discount | Employees Available Discount for Customers | Character  (3) | - | No | Yes | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **Stall** | Stall No | Unique Stall Number | Integer | - | No | No | No | PK | - |
|  | Address | Stalls Address/  Location | Character  (50) | - | No | No | No | - | - |
|  | Phone No | Stall Phone Number | Integer | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **UserAccount** | User Name | Unique Username Decided by the Customer | Character  (30) | - | No | No | No | PK | - |
|  | Name | Customers Name | Character  (40) | - | Yes  First,  Last. | No | No | - | - |
|  | Email | Customers Email Address | Character  (50) | - | No | No | No | - | - |
|  | Phone No | Customer Contact Number | Integer | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **SubscriptionType** | Camera Control | What Cameras Each Level Subscription Can Use | Integer | - | No | No | No | - | - |
|  | Movement Control | What Movement Control Each Level Subscription Can Use | Character  (3) | - | No | No | No | - | - |
|  | Platinum | The Next Best Subscription Level After Gold | Bit | - | No | No | No | - | - |
|  | Super Platinum | The Highest Subscription level | Bit | - | No | No | No | - | - |
|  | Standard | The Lowest Subscription level | Bit | - | No | No | No | - | - |
|  | Gold | The Next Best Subscription Level After Standard | Bit | - | No | No | No | - | - |
|  | Data Access | What Type of Data Access Each Level Subscription Can Use | Bit | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **DrovingZone** | Region | Region Where A BTDrone Is Located and Uses | Character  (30) | - | No | No | No | PK | - |
|  | Region Type | The Type of Location the Region Is | Character  (30) | - | No | No | No | - | - |
|  | Longitude | Measurement of The Width of The Area in Meters | Integer | - | No | No | No | - | - |
|  | Latitude | Measurement of The Length of The Area in Meters | Integer | - | No | No | No | - | - |
|  | Maximum Altitude | Measurement of The Hight of The Area in Meters | Integer | - | No | No | No | - | - |
|  | Minimum Altitude | Measurement of How Far Off the Ground the Area Starts | Integer | - | No | No | No | - | - |
|  | Region Size | Size of Region in Meters Cubed | Integer | - | No | Yes | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **BTDrone** | Drone ID | Unique Identifier for Each Drone | Integer | - | No | No | No | PK | - |
|  | Longitude | Constantly Records location Of Drone | Integer | - | No | No | No | - | - |
|  | Temperature | Records the Current Temperature | Integer | - | No | No | No | - | - |
|  | Humidity | Records the Current Humidity | Character  (30) | - | No | No | No | - | - |
|  | Ambient Light Strength | Records the Current Ambient Light Strength | Character  (30) | - | No | No | No | - | - |
|  | Video Stream | States If this Drone Will Be Used for Live Streams | Bit | - | No | No | Yes | - | - |
|  | Latitude | Constantly Records location Of Drone | Integer | - | No | No | No | - | - |
|  | Altitude | Constantly Records location Of Drone | Integer | - | No | No | No | - | - |
|  | Time | Local Time | Time | - | No | No | No | - | - |
|  | Maximum Altitude | A Set Number Stating How Hight The Drone Can Go | Integer | - | No | No | No | - | -- |
|  | Maximum Latitude | A Set Number Stating How Close the Drone Can Go North/South Before It Must Turn Around | Integer | - | No | No | No | - | - |
|  | Maximum Longitude | A Set Number Stating How Close the Drone Can Go East/West Before It Must Turn Around | Integer | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **DataBaseEntry** | Latitude | Measurement of The Width of The Area in Meters | Integer | - | No | No | No | - | - |
|  | Longitude | Measurement of The Width of The Area in Meters | Integer | - | No | No | No | - | - |
|  | Ambient Light Strength | Records the Current Ambient Light Strength | Character  (30) | - | No | No | No | - | - |
|  | Humidity | Records the Current Humidity | Character  (30) | - | No | No | No | - | - |
|  | Temperature | Records the Current Temperature | Character  (30) | - | No | No | No | - | - |
|  | Altitude | Constantly Records location Of Drone | Integer | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **CameraView** | Camera Number | Camera View Number 1-100 on a Drone | Integer | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **MaintenanceLog** | Date | Date of Last Check Up | Date | - | No | No | No | - | - |
|  | Status | Status of Drone | Character  (30) | - | No | No | No | - | - |
|  | Comments | Any Comments About the Drone | Character  (100) | - | No | No | Yes | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **Part** | Part ID | Unique identifier For Each Part | Integer | - | No | No | No | - | - |
|  | Part Status | Status of That Part | Character  (30) | - | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |
| **Suppliers** | Company Name | Company’s Name | Character  (30) | - | No | No | No | PK | - |
|  | Address | Address of Company | Character  (50) | - | No | No | No | - | - |
|  | Email | Email of Company | Character  (50) | - | No | No | No | - | - |
|  | Phone No | Company Phone Number | Integer | - | No | No | No | - | - |

## Rationale

### Contractee

The Contractee entity is used for storing Company’s information. This Information will be used when creating a contract, so we know who has purchased each unique contract.

### Contract

Each contract is identified by a unique key. There is no single attributes within this entity because all the information required for the contract will be drawn from other tables. The cost is not stored due to my thinking of price will be decided on paper before the details are entered into the database.

### Staff

Stores all staff that work at Being There. This includes basic information and their roles within the business there is also I field that indicates how much of a discount sales people are allowed to give out to customers. Each Employee is Identifies by an Employee No

Only Staff with an Administrative Executive Role can create new contracts

### Stall

A stall is just storing the information about each stall around the county this includes address and phone number

A stall is Identified by a Stall number

### UserAccount

A user account is made by customers that is used to use a drone to watch video streams. Each user account has to have one subscription to allocate the amount of access they have to the drones.

This is identified by a unique username created by the customer

### SunscriptionType

Subscription type stores all the different types of subscriptions that customers can buy. This includes the cost of each one.

Each Subscription type will be identified by the foreign key UserAccount to select witch user account has which subscription level.

### DrovingZone

Droving zone stores geographical data that a drone will roam around in. These Regions can overlap if assigned to a contract. The minimum size for a Region is 100 cubic meters.

Each droving zone is identified by region If multiple drones are in a region E.g. 2 in Antarctica they can be referenced with numbers as well

### BTDrone

BTDrones are the physical drones them self’s. These drones record information while flying around within a region that can then be stored for others to use. These drones are also used by the subscribers, so they are able to watch live video streams so drones that are eligible.

Each drone is uniquely identified by a drone number or serial number.

### DataBaseEntry

This entity will store information sent by each individual drone. Information will be sent to be stored every 10 seconds

This will be identified by the foreign key BTDrone made into a primary key.

### CameraViews

This table shows how many CameraViews there are for each drone.

### MaintenanceLog

This will keep a log of the status of each drone along with each part of the drone. There is a date to keep track on when it was last maintained so it can be done every 5 years. Also, a comment section for if the engineer thinks there is anything to add

The maintenance log is identified by the BTDrone foreign key turned into the primary key.

### Part

A list of each individual part in the drones along with which drone it is a part of.

Each part is identified by their serial number the part should come with from the supplier.

### Suppliers

Suppliers is a list of all the different company’s we would get the parts from for drones There can also be multiple suppliers that can supply one part.

Each supplier is identified by the company name

### ASSUMPTIONS ABOUT BUSINESS RULES & REASONS FOR CHOICES

**Contractee**

In this entity I believed that that the Contractee is the company who requests a contract with Being There. This is a 1-1 Relationship because only one contract per company if they wished to expand upon it like add more drones to the contract you would update the current one not create a new one.

**Contract**

Contract is an important table due to the amount of relationships it has with other tables. Firstly, contract and Contractee has a 1-1 relationship for the reasons stated above. Then we have a M-1 relationship with staff this is here to state who can create a contract and to show that a staff member must create one. Contract to SubscriptionType is a 1-1 relationship I have made it like this because each contract has a super platinum subscription which is then connected to a UserAccount that the Contractee will use. Contract and droving zone has a 1-M relationship. This is like this because a contract can ask and pay for there drones to be in many different droving zones around the world. Lastly, Contract and BTDrone. This makes a link between the BTDrone and contact to show what drones belong to this one contract. Contract is also a weak entity because a contract cannot exist without many different entities linked to it.

**Staff**

Staff is the entity that will keep a record of all the staff working at Being There. This will contain all their basic information and their role in the business this also shows the amount of discount can be given out for each staff member this can be nullable, so new staff members can’t just give discounts out to anyone. Staff and stall have a M-1 Relationship this shows that many staff members must work at a stall or Office, so you would know what branch they work in. Staff and contract have a 1-M relationship. This is showing that a staff member must create a contract and one staff member can create many of the contracts.

**Stall**

Stall is holding information about the different shops/offices around the country. There is a 1-M relationship between Stall and Staff to show what staff works what chat stall.

**UserAccount**

User Account Stores all the customers that have a subscription and you would use this information to login to their own account to access their video streams. User Account has a M-1 relationship with BTDrone. This shows that many different people logged into on there account and access and watch a stream from one drone. User Account and Subscription Type has a 1-1 relationship. This is how I would assign a subscription level to their user account that then states the permission level they have when watching streams.

**SubscriptionType**

Subscription type stores all the information about each subscription level available to customers. Subscription type and User account has a 1-1 relationship. This is used to give out levels of subscriptions to user accounts. Subscription type and Contract have a 1-1 relationship. This is here to link up the super platinum subscription type to a contract because super platinum can only be used by a contract

**DrovingZone**

Droving Zones are made to show a location that a drone will be occupying. This is calculated by the latitude, longitude and altitude these are then calculated to show the meters squared the area is minimum of 100 square meters. Droving Zone and BTDrone has a 1-M relationship. This links that droving zone a drone will occupy It Is also many because droving zones can overlap if they are both apart of the same contract. Droving Zone and Contract have a M-1 relationship. When a contract is made the Contractee can have multiple drones and mutable droving zones a part of that contract.

**BTDrone**

The BTDrone is made thinking that each drone will store its own information on a on board database until it is sent off to another database for clients to use if on a contract. Each drone has 100 different camera streams that is being kept in another table which has a 1-M relationship between them. BTDrone and parts have a 1-M relationship between them to show that 1 drone has many different parts this is then connected to Maintenance log in a 1-1 relationship this then makes sure that all parts are recorded and maintained every 5 years. There is a 1-M relationship between BTDrone and Database entry. This is just showing that a drone will save data into this database many times. You should get a clear understanding how all the other entities work being attached to BTDrone by reading their own explanations about it.

**DataBaseEntry**

This table is used for drones to send all their region data they record so it can be stored to later use or for Contractees to use when they wish. Only all the data specified in the document provided by Being There is shown to be stored within this database.

**CameraViews**

I decided to separate the camera from the drone to make it easier to manage this shows how many camera views each drone has this is stated by their M-1 relationship shown in the diagram.

**MaintenanceLog**

Maintenance log is where each drone status is recorded and when it next needs to be brought in and looked at. This is a weak entity because a record in this table cannot exist without a BTDrone. This has a 1-1 relationship to BTDrone as there should only be one recorded at any given time. We then have a relationship between part with a 1-M relationship. This is like this, so you can keep track if there is a single part that needed fixing and just to list if each part is working or not.

**Part**

Part is the table where we recorded all the parts in a BTDrone. This is then connected to maintenance log, so we know what part is apart of what drone when it’s being maintained. It then also has a 1-M relationship with suppliers, so we know who has supplied each part. Part is also a weak entity because it cannot exist without BTDrone or Suppliers.

**Suppliers**

The Suppliers table list down all the companies that Being There buy their drone parts off. This has a M-1 Relationship with parts to show what part each company sends. It’s declared as many because many companies can supply the same part.

## Business Rules

1. **Contractee**/Contract
   1. **One** Contractee must belong to **One** Contract
   2. **One** Contract Must Belong to **One** Contractee
2. **Contract**/Contractee/Staff/SubscriptionType/DrovingZones/BTDrones
   1. **One** Contract Must Belong to **One** Contractee
   2. **One** Contractee must belong to **One** Contract
   3. **Many** Contracts must be Created by **One** staff member
   4. **One** Staff Member Must Create **Many** Contracts
   5. **One** contract must have **One** SubscriptionType
   6. **One** SubscriptionType Can Have **One** Contract
   7. **One** Contract can have **Many** DrovingZones
   8. **Many** DrovingZones Can have **One** Contract
   9. **One** Contract Can have **Many** BTDrones
   10. **Many** BTDrones Can Have **One** Contract
3. **Staff**/Contract/Stall
   1. **One** Staff Member Must Create **Many** Contracts
   2. **Many** Contracts must be Created by **One** staff member
   3. **Many** Staff Members Must work at **One** Stall
   4. **One** Stall Must Have **Many** Staff Members
4. **Stall**/Staff
   1. **One** Stall Must Have **Many** Staff Members
   2. **Many** Staff Members Must work at **One** Stall
5. **UserAccount**/BTDrone/SubscriptionType
   1. **Many** UserAccounts can Use **One** BTDrone
   2. **One** BTDrone Can be Used by **Many** User Accounts
   3. **One** UserAccount Must Have **One** SubscriptionType
   4. **One** SubscriptionType can have **One** UserAccount
6. **SubscriptionType**/Contract/UserAccount
   1. **One** SubscriptionType Can Have **One** Contract
   2. **One** SubscriptionType can have **One** UserAccount
   3. **One** contract must have **One** SubscriptionType
   4. **One** UserAccount Must Have **One** SubscriptionType
7. **DrovingZone**/BTDrone/Contract
   1. **One** DrovingZone Can have **Many** BTDrones
   2. **Many** DrovingZones Can have **One** Contract
   3. **Many** BTDrones Must Have **One** Droving Zone
   4. **One** Contract can have **Many** DrovingZones
8. **BTDrone**/Contract/DatabaseEntry/MaintenenceLog/Parts/CameraViews/UserAcount/DrovingZone
   1. **Many** BTDrones Can Have **One** Contract
   2. **One** BTDrone Must Store Data in **Many** DataBase Entry’s
   3. **One** BTDrone Must Have **One** Maintenance Log
   4. **One** BTDrone Must Contain **Many** Parts
   5. **One** BTDrone Must Have **Many** CameraViews
   6. **One** BTDrone Can be Used by **Many** User Accounts
   7. **Many** BTDrones Must Have **One** Droving Zone
   8. **One** Contract Can have **Many** BTDrones
   9. **Many** Data Base Entry’s Is Stored by **One** BTDrone
   10. **One** Maintenance Log Must Have **One** BTDrone
   11. **Many** Parts Must be A Part of **One** BTDrone
   12. **Many** CameraViews Must Have **One** BTDrone
   13. **Many** UserAccounts can Use **One** BTDrone
   14. **One** DrovingZone Can have **Many** BTDrones
9. **DataBaseEntry**/BtDrone
   1. **Many** Data Base Entry’s Is Stored by **One** BTDrone
   2. **One** BTDrone Must Store Data in **Many** DataBase Entry’s
10. **CameraView**/BtDrone
    1. **Many** CameraViews Must Have **One** BTDrone
    2. **One** BTDrone Must Have **Many** CameraViews
11. **MaintenanceLog**/BTDrone/Part
    1. **One** Maintenance Log Must Have **One** BTDrone
    2. **One** Maintenance Log Must Have **Many** Parts
    3. **One** BTDrone Must Have **One** Maintenance Log
    4. **Many** Parts Can Have **One** Maintenance Log
12. **Part**/MaintenanceLog/BTDrone/Supplier
    1. **Many** Parts Must be A Part of **One** BTDrone
    2. **One** Part Can be Supplied by **Many** Suppliers
    3. **Many** Parts Can Have **One** Maintenance Log
    4. **One** Maintenance Log Must Have **Many** Parts
    5. **One** BTDrone Must Contain **Many** Parts
    6. **Many** Suppliers Can Supply **One** Part
13. **Supplier**/Part
    1. **Many** Suppliers Can Supply **On**e Part
    2. **Many** Parts Can Have **One** Maintenance Log

## References

IBM Knowledge Center - entity instance. (n.d.). Retrieved April 19, 2018, from https://www.ibm.com/support/knowledgecenter/en/SSBNJ7\_1.4.1/wls\_user\_guide/tnpm\_wls\_user\_guide\_gloss\_entityinstance.html

# Milestone 2

## Executive Summary

This milestone includes a description on how to transform a conceptual model to a logical model, a description of what crow's foot notation is, the logical ERD for Being There’s requested database, and the rationale for each entity in the logical ERD, Normalization,

A data dictionary has been included which describes the number of rows an entity will start with in the database and how much it will be expected to grow over time.  With this is an attribute table which defines how an attribute will be setup within the database in respect to its data type, length, validation rules etc. Primary and foreign keys have been described.

Lastly a table has been included to show the derived attributes that are not included in the database but are calculated from existing attributes.

## Introduction

The purpose of this document is to create a logical model by pulling the data from the conceptual made in the first document provided, As well as a report for the Being There Database.

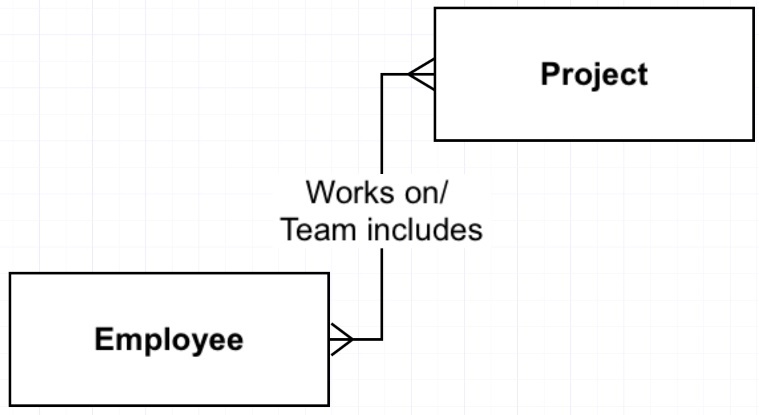
This is to be done by applying logical data mapping rules and normalising the data. Included in this report is a rationale of my decisions, descriptions of normalisation stages including how the data pulled from the conceptual model would look normalised and a data dictionary detailing all the attributes in the logical stage of the database.

## Conceptual to Logical

### Mapping Rules

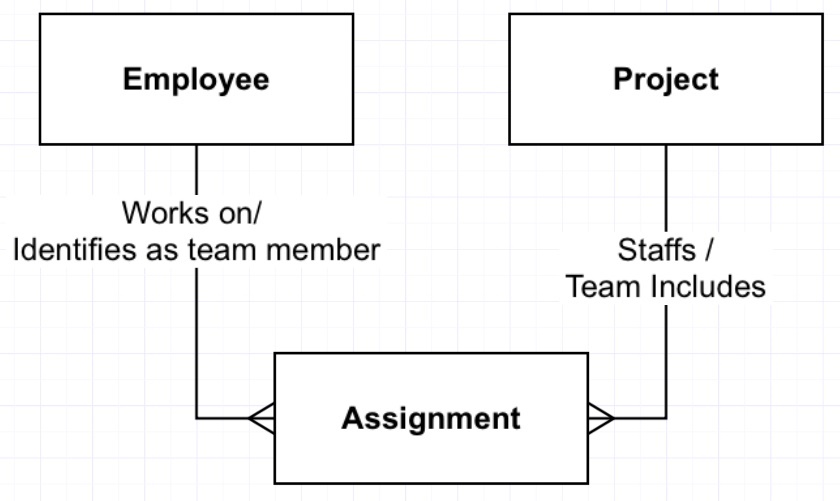
This is a more detailed entity relationship diagram than the conceptual model.  There are a few rules to be observed in this style of modelling, which are:

* Primary and Foreign keys are required for the Logical model
* Composite attributes become entities
* Simple attributes will remain as such
* **Many to Many Relationships**
  + **Many** to **many** relationships do not exist in a database, so there must be a way to model that sort of relationship. **Many** to **many** can exist in a conceptual model but you must remove direct many to many relationships in a logical model. An example of how this might work is a many to many relationships between employee and project.
  + **Many** employees can work on **many** projects and **many** projects can be worked on by **many** employees.
  + What you would do is add a table in between the two tables called Assessment. It will contain the Employee ID and Project ID and additional info as required. The 2 tables will each have a one to many relationships to the Assessments table.
  + **Many to Many**



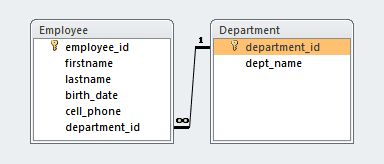
(Lambert, n.d.)

* **Many to Many Fixed**



(Lambert, n.d.)

* As the example shows, a connecting table is added and the many to many relationships is removed. This model allows many employees to be related to many projects and vice versa without the existence of an actual many to many relationships.
* **One to One**
  + When transforming to logical, in a one to one relationship, the ID from the either side of the relationship gets added as a new column to the other side of the relationship as a foreign key.
* **One to Many**
  + When transforming to logical, in a one to many relationship, the ID from the One side of the relationship gets added as a new column to the Many side of the relationship as a foreign key.



(“Hibernate One To Many XML Mapping Tutorial example,” n.d.)

* **Superclass subclass relationships**
  + Create one relation for the super type that holds all the common attributes for the subtypes.
  + Or create one relation for each subtype.
  + Or create one relation for the super type that holds all the common attributes and all the specific attributes of all subtypes. There will be no relations for the subtypes. **Note** that this solution will require that null values are allowed in the attributes related to the subtypes
* **Validate relations using normalisation**
  + Now that we know the basic rules to look out for when making the switch you must also (if not done already) take all the data from the conceptual model and run it though the normalisation process.
    - This can consist of **6** different stages that I will list but we will only be using the first **5** steps. They are First Normal Form **(1NF)**, Second Normal Form **(2NF)**, Third Normal Form **(3NF)**, Boyce-Codd Normal Form **(BCNF)**, Fourth Normal Form**(4NF)** and Fifth Normal Form**(5NF)**.

Each process and each stage will be described and shown below under the relevant headings.

### Crows Foot Notation

Crows foot notation and diagrams are shown a bit differently than the conceptual. In crow’s foot each entity is shown as a table of sorts with a line between each of them to show the relationship.

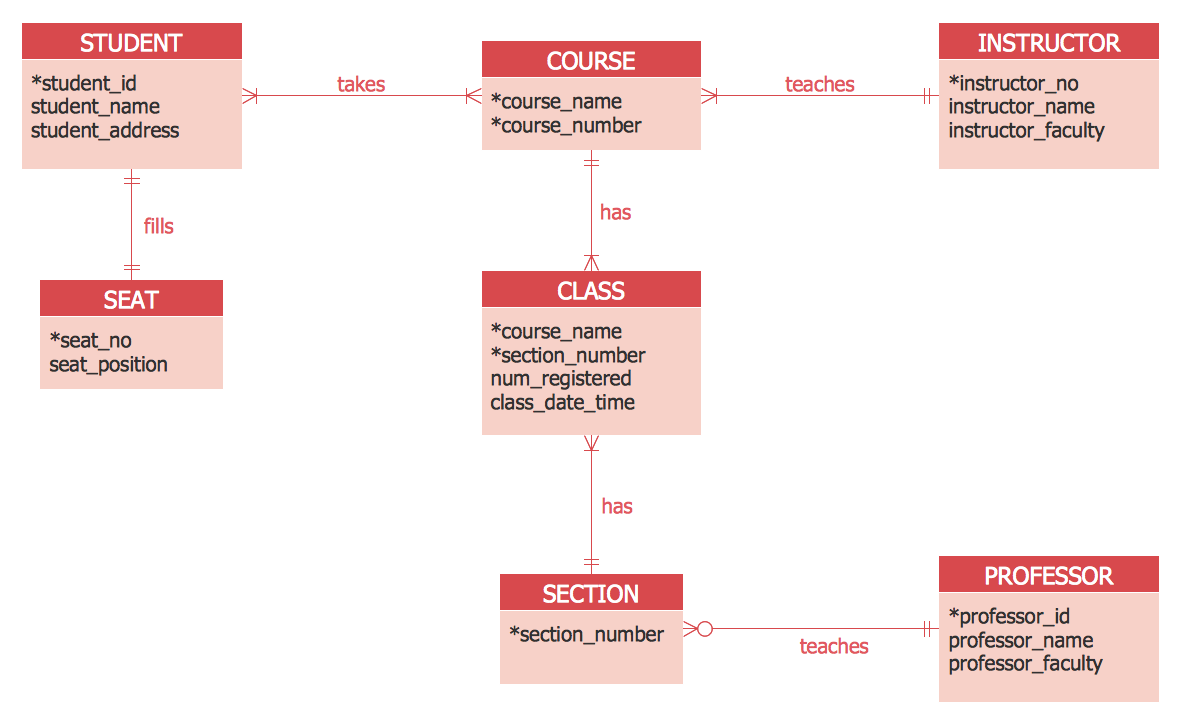
The crow's foot notation does not allow relationships to have attributes, where relationships would usually have attributes, these entities are promoted to existing as entities of their own.

Crows foot notation shows optionality a bit differently. Optional relationships are a dotted line connecting two tables and a mandatory relationship as a solid line connecting the two. It is also shown by how you draw the relationships between the two tables.

There are four different natation that you should know that shows the relationship as well as cab show optionality as well.

* **|O 0 or 1**
* **>| 1 or more**
* **|| Exactly 1**
* **>O 0 or more**
* First off **0 or 1** this means that you may have **0 or 1** of a given entity
* Second **1 or more** means you can have **1 or more** of a given entity
* Third **Exactly 1** means that you can have one and only one of a given entity
* Fourth 0 or more means you can have 0 or more of a given entity

Here is an **example** of crow’s foot notation within a logical diagram.



(“Pinterest,” n.d.)

As you can see here we have 1 or more classes that must have only 1 course. 1 or more courses must have exactly 1 teacher. 1 or more courses are taken by 1 or more students and so on. I don’t know why this model shows a many to many relationships, but this cannot happen within a logical model

### EER Conceptual to Logical

When Extended Entity Relationships are used within a conceptual model there are three different ways they can be turned into a logical model.

**Number 1**

* You Create one giant super class for all of the common attributes and have a relation with each separate subclass which hold more specific attributes

**Number 2**

* You can Create a relation for each subtype and the subtype will hold all the attributes and the relation has a relationship with the super class. This is basically what you do for a many to many relationships by having a middle table.

**Number 3**

* You can Create one Super Class that holds both common and specific attributes which is then has a relationship with the subclasses. If you do it this way you must make sure that all the attributes related to the subclass will have to be made Nullable

## Normalisation

**1NF**

1NF Should follow the following four rules

* It should only have single(atomic) valued attributes/columns.
* Values stored in a column should be of the same domain
* All the columns in a table should have unique names.
* And the order in which data is stored, does not matter

**2NF**

2NF Should follow the following two rules

* It should be in the First Normal form.
* And, it should not have Partial Dependency.

**3NF**

3NF Should follow the following two rules

* It is in the Second Normal form.
* And, it doesn't have Transitive Dependency.

**BCNF**

BCNF Should follow the following two rules

* must be in 3rd Normal Form
* and, for each functional dependency (X → Y), X should be a super Key.

**4NF**

BCNF Should follow the following two rules

* It is in the Boyce-Codd Normal Form.
* And, it doesn't have Multi-Valued Dependency.

### First Normal Form

In first normal from we take the data from the conceptual and we want to separate any repeating data. If there is data that would be repeating multiple times you would separate that data into separate tables.

#### BTDrone

* + 1. DroneID
    2. Longitude
    3. Latitude
    4. Altitude
    5. Time
    6. Temperature
    7. Maximum Altitude
    8. Humidity
    9. Maximum Latitude
    10. Ambient Light Strength
    11. Video Stream
    12. Maximum Longitude

Environment Data

1. DroneID(PK/FK)
2. Longitude
3. Latitude
4. Altitude
5. Temperature
6. Humidity
7. Ambient Light Strength

Maximum Roaming Area

1. DroneID(PK/FK)
2. Maximum Altitude
3. Maximum Latitude
4. Maximum Longitude

BTDrone

1. DroneID(PK)
2. Time
3. Video Stream

#### Sensed Data

1. Altitude
2. Latitude
3. Longitude
4. Temperature
5. Humidity
6. Ambient Light Strength

**I believe this already in 1NF as there is no repeating data**

#### Maintenance Log

1. Status
2. Date
3. Comments

**I believe this already in 1NF as there is no repeating data**

#### Part

1. PartID(PK)
2. Part Status

**I believe this already in 1NF as there is no repeating data**

#### Suppliers

1. Company Name(PK)
2. PhoneNo
3. Address
4. Email

**I believe this already in 1NF as there is no repeating data**

#### Camera Views

1. Camera Views(PK)

**I believe this already in 1NF as there is no repeating data**

#### User Account

1. User Name(PK)
2. Email
3. PhoneNo
4. Name (First, Last)

User Account

1. User Name(PK)

Subscriber Details

1. First Name
2. Last Name
3. Email
4. PhoneNo

#### Subscription Type

1. Camera Control
2. Movement Control
3. Data Access
4. Cost
5. Standard
6. Gold
7. Platinum
8. Super Platinum

**I believe this already in 1NF as there is no repeating data**

#### Droving Zone

1. Region(PK)
2. Region Type
3. Maximum Altitude
4. Minimum Altitude
5. Longitude
6. Latitude
7. Region Size

Droving Zone

1. Region(PK)
2. Region Type

Droving Size

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size

#### Staff

1. EmployeeNo(PK)

**I believe this already in 1NF as there is no repeating data**

#### AdminStaff

1. Name (First, Last)
2. Address
3. Discount
4. DOB
5. PhoneNo

AdminStaff

1. Discount

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo

**Please Note That Staff Details will be One Table Used to store Details of both Admin staff and Sales Staff**

#### SalesStaff

1. Name (First, Last)
2. Address
3. Discount
4. DOB
5. PhoneNo

SalesStaff

1. Discount

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo

**Please Note That Staff Details will be One Table Used to store Details of both Admin staff and Sales Staff**

#### Stall

1. StallNo(PK)
2. PhoneNo
3. Address

**I believe this already in 1NF as there is no repeating data**

#### Contract

1. ContractID(PK)

**I believe this already in 1NF as there is no repeating data**

#### Contractees

1. Company(PK)
2. PhoneNo
3. Address
4. Name (First, Last)

Contractee Details

1. First Name
2. Last Name
3. Address
4. PhoneNo

Contractees

1. Company(PK)

### Second Normal Form

In second normal form we want to make sure all tables have a primary key used as the unique identifier of that table. If there are attributes in a table that are only partially dependent on the current primary key, then they should be separated into their own separate table.

#### BTDrone

Environment Data

1. DroneID(PK/FK)
2. Longitude
3. Latitude
4. Altitude
5. Temperature
6. Humidity
7. Ambient Light Strength

Maximum Roaming Area

1. DroneID(PK/FK)
2. Maximum Altitude
3. Maximum Latitude
4. Maximum Longitude

BTDrone

1. DroneID(PK)
2. Time
3. Video Stream

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### Sensed Data

1. Altitude
2. Latitude
3. Longitude
4. Temperature
5. Humidity
6. Ambient Light Strength

Sensed Data

1. Altitude
2. Latitude
3. Longitude
4. Temperature
5. Humidity
6. Ambient Light Strength
7. BTDroneID (PK, FK)

#### Maintenance Log

1. Status
2. Date
3. Comments

Maintenance Log

1. Status
2. Date
3. Comments
4. BTDroneID (PK, FK)

#### Part

1. PartID(PK)
2. Part Status

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### Suppliers

1. Company Name(PK)
2. PhoneNo
3. Address
4. Email

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### Camera Views

1. Camera Views(PK)

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### User Account

User Account

1. User Name(PK)
2. Email
3. PhoneNo

Subscriber Details

1. First Name
2. Last Name

Subscriber Details

1. First Name
2. Last Name
3. User Name (PK, FK)

#### Subscription Type

1. Camera Control
2. Movement Control
3. Data Access
4. Cost
5. Standard
6. Gold
7. Platinum
8. Super Platinum

Subscription Type

1. Camera Control
2. Movement Control
3. Data Access
4. Cost
5. Standard
6. Gold
7. Platinum
8. Super Platinum
9. User Name (PK, FK)

#### Droving Zone

Droving Zone

1. Region(PK)
2. Region Type

Droving Size

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size

Droving Size

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size
6. Region (PK, FK)

#### Staff

1. EmployeeNo(PK)

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### AdminStaff

AdminStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)

**Please Note That Staff Details will be One Table Used to store Details of both Admin staff and Sales Staff**

#### SalesStaff

SalesStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)

**Please Note That Staff Details will be One Table Used to store Details of both Admin staff and Sales Staff**

#### Stall

1. StallNo(PK)
2. PhoneNo
3. Address

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### Contract

1. ContractID(PK)

**I Believe That this Information is already in 2NF as everything has a Primary key set for tables**

#### Contractees

Contractee Details

1. First Name
2. Last Name
3. Address
4. PhoneNo

Contractees

1. Company(PK)

Contractee Details

1. First Name
2. Last Name
3. Address
4. PhoneNo
5. Company (PK, FK)

### Third Normal Form

To be in third normal from if any single piece of data in a table changed and it affected other table entry’s E.g. Another field would change as well along with the other. Then you would take these fields and separate them into their own table. As seen below in the BTDrone section if time had changed it would also affect the video stream so I have separated them both into there own unique table.

#### BTDrone

Environment Data

1. DroneID(PK/FK)
2. Longitude
3. Latitude
4. Altitude
5. Temperature
6. Humidity
7. Ambient Light Strength

Maximum Roaming Area

1. DroneID(PK/FK)
2. Maximum Altitude
3. Maximum Latitude
4. Maximum Longitude

BTDrone

1. DroneID(PK)
2. Time
3. Video Stream

BTDrone

1. DroneID(PK)

Streaming Details

1. Video Stream ID(PK)
2. Time

#### Sensed Data

Sensed Data

1. Altitude
2. Latitude
3. Longitude
4. Temperature
5. Humidity
6. Ambient Light Strength
7. BTDroneID (PK, FK)

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### Maintenance Log

Maintenance Log

1. Status
2. Date
3. Comments
4. BTDroneID (PK, FK)

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### Part

1. PartID(PK)
2. Part Status

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### Suppliers

1. Company Name(PK)
2. PhoneNo
3. Address
4. Email

Suppliers

1. Company Name(PK)

Supplier Details

1. PhoneNo
2. Email
3. Company Name (PK, FK)

Supplier Addresses

1. Address(PK)

#### Camera Views

1. Camera Views(PK)

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### User Account

User Account

1. User Name(PK)
2. Email
3. PhoneNo

Subscriber Details

1. First Name
2. Last Name
3. User Name (PK, FK)

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### Subscription Type

Subscription Type

1. Camera Control
2. Movement Control
3. Data Access
4. Cost
5. Standard
6. Gold
7. Platinum
8. Super Platinum
9. User Name (PK, FK)

Subscription Type

1. Camera Control
2. Movement Control
3. Data Access
4. Cost
5. Subscription Type(PK)

User Subscription

1. User Name (CK)
2. Subscription Type (CK)

**After looking though this section after I split the tables into two I decided to make the type the primary key rather than have four different attributes.**

#### Droving Zone

Droving Zone

1. Region(PK)
2. Region Type

Droving Size

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size
6. Region (PK, FK)

Droving Size Details

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size
6. Region Size ID(PK)

Droving Size

1. Region Size ID(CK)
2. Region(CK)

#### Staff

1. EmployeeNo(PK)

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### AdminStaff

AdminStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)

Staff Address

1. Address (PK)

**Please Note That Staff Details and Staff Address will be One Table Used to store Details of both Admin staff and Sales Staff**

#### SalesStaff

SalesStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)

Staff Address

1. Address (PK)

**Please Note That Staff Details and Staff Address will be One Table Used to store Details of both Admin staff and Sales Staff**

#### Stall

1. StallNo(PK)
2. PhoneNo
3. Address

Stall Addresses

1. Address

#### Contract

1. ContractID(PK)

**I believe that this is already in 3NF because if one non-key record is deleted or changed none of the others will be affected by this.**

#### Contractees

Contractees

1. Company(PK)

Contractee Details

1. First Name
2. Last Name
3. Address
4. PhoneNo
5. Company (PK, FK)

Contractee Addresses

1. Address(PK)

### Boyce-Codd Normal Form

In BCNF you have basically doing the same as 2NF and 3NF but with only the keys in each table. The best two ways to see signs of this would be if a table has 3 or more keys in it and you can see 2 or more columns with recurring patterns. If this was the case, you would separate these to stop this from happening.

#### BTDrone

Environment Data

1. DroneID(PK/FK)
2. Longitude
3. Latitude
4. Altitude
5. Temperature
6. Humidity
7. Ambient Light Strength

Maximum Roaming Area

1. DroneID(PK/FK)
2. Maximum Altitude
3. Maximum Latitude
4. Maximum Longitude

BTDrone

1. DroneID(PK)
2. Video Stream ID(FK)
3. Region(FK)
4. ContractID(FK)

Streaming Details

1. Video Stream ID(PK)
2. Time

**I Believe that this is already in BCNF because it does have more than 3 keys but the data within them do not have more than one recurring pattern**

#### Sensed Data

Sensed Data

1. Altitude
2. Latitude
3. Longitude
4. Temperature
5. Humidity
6. Ambient Light Strength
7. BTDroneID (PK, FK)
8. SubscriptionType(FK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Maintenance Log

Maintenance Log

1. Status
2. Date
3. Comments
4. BTDroneID (PK, FK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Part

1. PartID(PK)
2. Part Status
3. BTDroneID(FK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Suppliers

Suppliers

1. Company Name(PK)
2. PartID(FK)

Supplier Details

1. PhoneNo
2. Email
3. Company Name (PK, FK)
4. Address(FK)

Supplier Addresses

1. Address(PK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Camera Views

1. Camera Views(PK)
2. BTDroneID(FK)
3. User Name(FK)

**I Believe that this is already in BCNF because it does have more than 3 keys but the data within them do not have more than one recurring pattern**

#### User Account

User Account

1. User Name(PK)
2. Email
3. PhoneNo
4. BTDroneID(FK)

Subscriber Details

1. First Name
2. Last Name
3. User Name (PK, FK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Subscription Type

Subscription Type

1. Cost
2. Subscription Type(PK)
3. BTDroneID(FK)
4. CameraView(FK)
5. CameraView2(FK)

User Subscription

1. User Name (CK)
2. Subscription Type (CK)

**I Believe that this is already in BCNF as there is 3 keys but there would not be any recurring pattern because each CameraView Relationship is for a different purpose and would not be allowed to match up due to the reason for the relationship.**

#### Droving Zone

Droving Zone

1. Region(PK)
2. Region Type
3. ContractID(FK)

Droving Size Details

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size
6. Region Size ID(PK)

Droving Size

1. Region Size ID(CK)
2. Region(CK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Staff

1. EmployeeNo(PK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### AdminStaff

AdminStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)
7. Address(FK)

Staff Address

1. Address (PK)

**Please Note That Staff Details and Staff Address will be One Table Used to store Details of both Admin staff and Sales Staff**

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### SalesStaff

SalesStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)
7. Address (FK)

Staff Address

1. Address (PK)

**Please Note That Staff Details and Staff Address will be One Table Used to store Details of both Admin staff and Sales Staff**

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Stall

1. StallNo(PK)
2. PhoneNo
3. Address(FK)

Stall Addresses

1. Address(PK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

#### Contract

1. ContractID(PK)
2. Company(FK)
3. EmployeeNo(FK)
4. SubscriptionType(FK)

Contract Subscription Type

* + 1. ContractID(CK)
    2. SubscriptionType(CK)

#### Contractees

Contractees

1. Company(PK)

Contractee Details

1. First Name
2. Last Name
3. Address(FK)
4. PhoneNo
5. Company (PK, FK)

Contractee Addresses

1. Address(PK)

**I Believe that this is already in BCNF because there is no table that has three different keys within them**

### Forth Normal Form

It’s very rare to find yourself having to preform fourth normal form but, in the case, you do here’s what you are looking out for. If there are 2 or more many relationships in one table then this must be fixed by making multiple tables.

For example, if you have a person that has many different Skills and can speak many different languages in one table this must be split up so there is only one many in each table.

#### BTDrone

Environment Data

1. DroneID(PK/FK)
2. Longitude
3. Latitude
4. Altitude
5. Temperature
6. Humidity
7. Ambient Light Strength

Maximum Roaming Area

1. DroneID(PK/FK)
2. Maximum Altitude
3. Maximum Latitude
4. Maximum Longitude

BTDrone

1. DroneID(PK)
2. Video Stream ID(FK)
3. Region(FK)
4. ContractID(FK)

Streaming Details

1. Video Stream ID(PK)
2. Time

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Sensed Data

Sensed Data

1. Altitude
2. Latitude
3. Longitude
4. Temperature
5. Humidity
6. Ambient Light Strength
7. BTDroneID (PK, FK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Maintenance Log

Maintenance Log

1. Status
2. Date
3. Comments
4. BTDroneID (PK, FK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Part

1. PartID(PK)
2. Part Status
3. BTDroneID(FK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Suppliers

Suppliers

1. Company Name(PK)
2. PartID(FK)

Supplier Details

1. PhoneNo
2. Email
3. Company Name (PK, FK)
4. Address(FK)

Supplier Addresses

1. Address(PK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Camera Views

1. Camera Views(PK)
2. BTDroneID(FK)
3. User Name(FK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### User Account

User Account

1. User Name(PK)
2. Email
3. PhoneNo
4. BTDroneID(FK)

Subscriber Details

1. First Name
2. Last Name
3. User Name (PK, FK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Subscription Type

Subscription Type

1. Camera Control
2. Movement Control
3. Data Access
4. Cost
5. Subscription Type(PK)

User Subscription

1. User Name (CK)
2. Subscription Type (CK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Droving Zone

Droving Zone

1. Region(PK)
2. Region Type
3. ContractID(FK)

Droving Size Details

1. Maximum Altitude
2. Minimum Altitude
3. Longitude
4. Latitude
5. Region Size
6. Region Size ID(PK)

Droving Size

1. Region Size ID(CK)
2. Region(CK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Staff

1. EmployeeNo(PK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### AdminStaff

AdminStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)
7. Address(FK)

Staff Address

1. Address (PK)

**Please Note That Staff Details and Staff Address will be One Table Used to store Details of both Admin staff and Sales Staff**

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### SalesStaff

SalesStaff

1. Discount
2. EmployeeNo (PK, FK)

Staff Details

1. FirstName
2. LastName
3. Address
4. DOB
5. PhoneNo
6. EmployeeNo (PK, FK)
7. Address (FK)

Staff Address

1. Address (PK)

**Please Note That Staff Details and Staff Address will be One Table Used to store Details of both Admin staff and Sales Staff**

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Stall

1. StallNo(PK)
2. PhoneNo
3. Address(FK)

Stall Addresses

1. Address(PK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Contract

1. ContractID(PK)
2. Company(FK)
3. EmployeeNo(FK)

Contract Subscription Type

* + 1. ContractID(CK)
    2. SubscriptionType(CK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

#### Contractees

Contractees

1. Company(PK)

Contractee Details

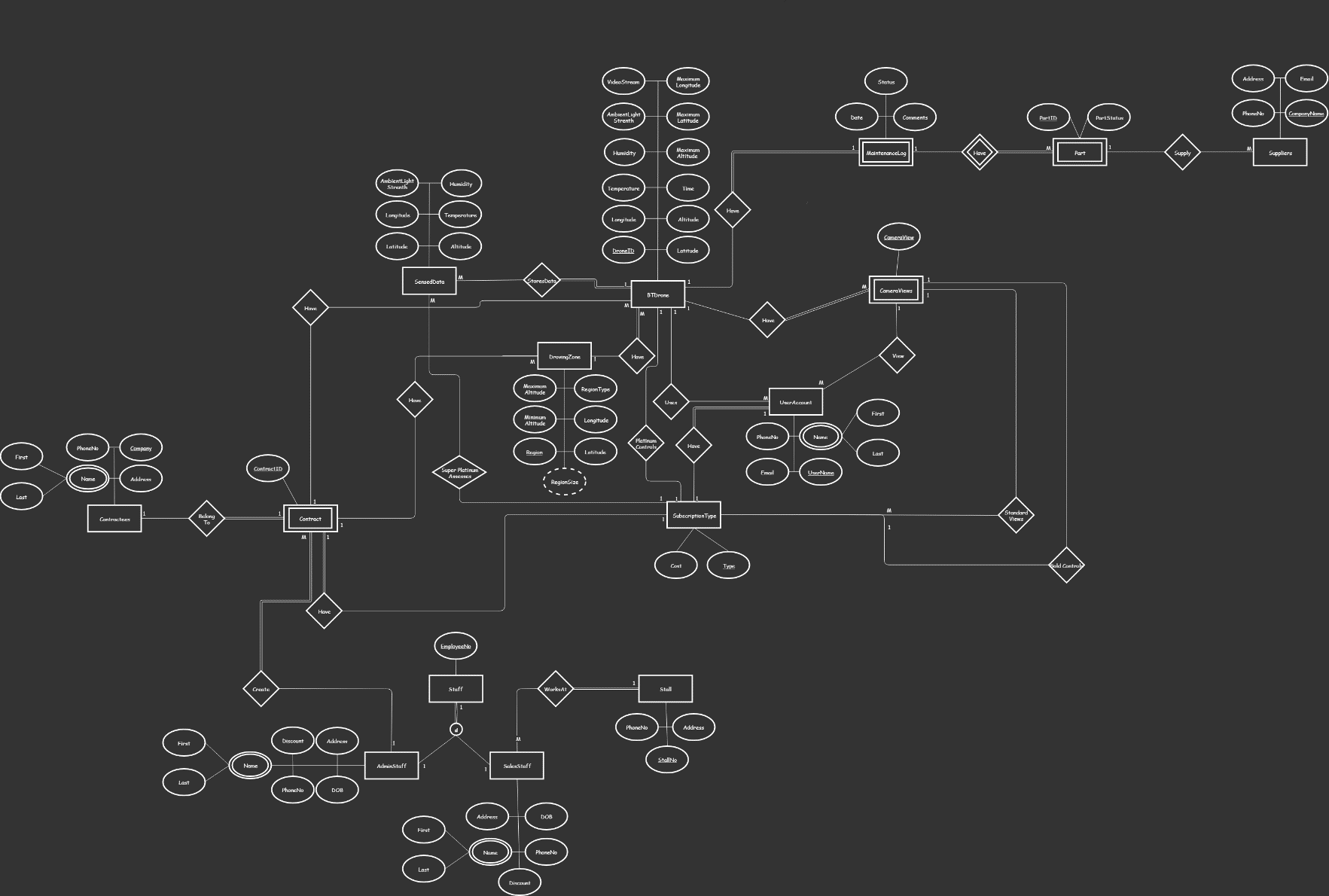
1. First Name
2. Last Name
3. Address(FK)
4. PhoneNo
5. Company (PK, FK)

Contractee Addresses

1. Address(PK)

**I believe that this is already in 4NF because there is only one or no Many relationships in all tables**

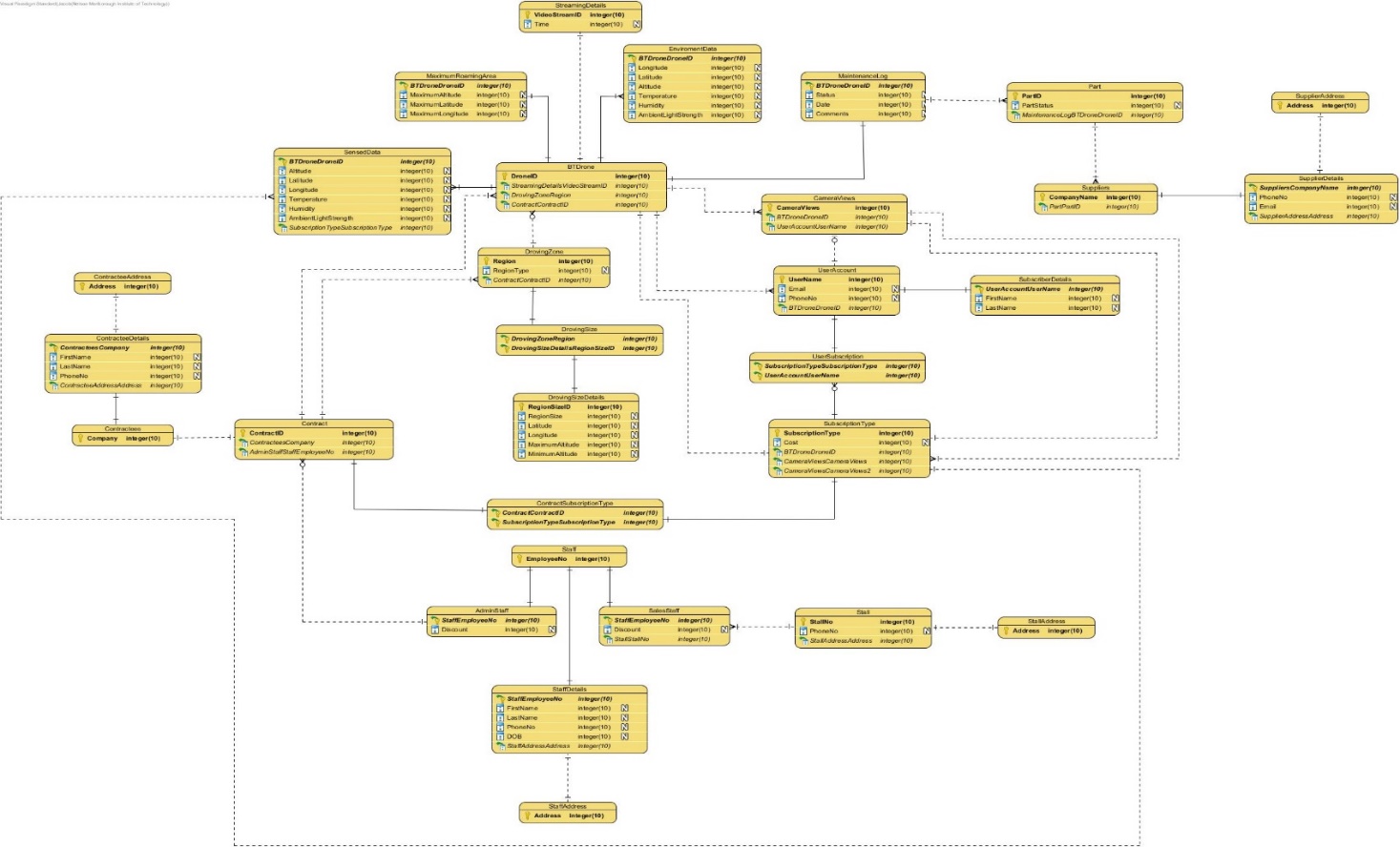
## Updated Conceptual Model



### Updates made to conceptual after Milestone 1 Feed Back

* I have changed the DataBaseEntry name to SensedData.
* I have created a relationship from UserAccount to CameraViews to be able to see exactly who is watching what CameraView.
* The Identifying relationship between BtDrone and Camera View as been fixed by turning the relation from BTDrone to CameraView to optional rather than mandatory like before.
* The identifying relationship for Contract has been decided and fixed.
* The Identifying relationship between BtDrone and MaintenanceLog as been fixed by turning the relation from BTDrone to Maintenance Log to optional rather than mandatory like before.
* The Staff Table was spit using EER to create an AdminStaff and a SalesStaff.
* The SubscriptionType Attributes Standard, Gold, Platinum and Super Platinum have been removed and turned into relationships with the appropriate table.
* The SubscriptionType Attributes CameraControl, MovementControl and DataAccess have been removed and turned into relationships with the appropriate table.

## Logical ERD



### Rational & Relations

#### BTDrone

This Entity stores the unique key of each individual Drone. It then gets all the drones extra details from other tables joined by foreign keys.

This includes EnvironmentData, MaximumRoamingArea and VideoStream. It is also got a relationship with Contract to show if a drone is apart of a contract or not.

Each drone will be identified by a sort of serial number for each individual drone.

#### MaximumRoamingArea

This Entity contains the Maximum area a drone can roam around in. As stated in the report a drone is not allowed to go within 10% of it’s full roaming area and this entity is where we store that full roaming area – the 10%.

This table is identified by the foreign key BTDrone turned into a primary key.

#### StreamingDetails

This entity stores video stream times. Some drones do video streams for customers and this table stores when that stream will be.

This table has its own unique key to show each individual stream.

#### EnvironmentData

This Entity stores all the environmental data that is recorded by the BTDrone. It is stored here until it gets sent off to a more permanent location to be stored for use by others.

This table is identified by the foreign key BTDrone turned into a primary key.

#### SensedData

This Entity is where the environmental data is sent for long term storage. This data can be accessed and used by people with a contract and the Company Being There itself.

This table is identified by the foreign key BTDrone turned into a primary key.

#### DrovingZone

This Entity stores the details of a drones roaming area. This includes the region it’s in and the type of region Desert, Snow etc.

A droving zone can also be assigned to a contract and is done so though a foreign key.

A droving zone is identified by it’s region it is located in.

#### DrovingSize

This Entity is the table used to join the primary key of droving zone up to the primary key of Droving size details. This is used because it was the only way to connect each zone up with it’s size details without any issues with foreign keys.

#### DrovingSizeDetails

This Entity holds the area of an individual droving zone. Four of there columns are store the length, width and Hight of the area while there is one that derives this data to calculate the full area in meters squared.

I had to create a key for this table because I was unable to use the size itself due to the possibility that another zone may be the exact same size.

#### MaintenanceLog

This Entity stores the information of the status of each drone. This includes date when it was last inspected and any general comments about the drone.

A part can also be linked to the maintenance log and the drone though the foreign key from a drone made into a primary key. This can then be used to see what part is apart of what drone and if there are problems with a part it will be recorded as comment and status can be set to needs fixing.

#### Part

This Entity stores all Parts of a BTDrone and knows witch part is a part of which drone though the foreign key

Each part is identified with the parts serial number given by the suppliers of the parts.

#### Suppliers

This Entity stores the companies that supply the parts for the BTDrones. This is shown by having a part foreign key within this table. This will indagate what part is brought from what supplier. Note: that suppliers can supply many of the same part.

#### SupplierDetails

This Entity stores all the details of each individual supplier. Details go from Phone Number to Email addresses.

#### SupplierAddress

This Entity stores all the individual addresses of the suppliers which is connected to Supplier Details though a 1-1 relationship. I did this because I felt that if the address changed for a supplier for any reason it would also affect the phone number, so I decided to separate them both to stop this from happening.

#### CameraViews

This entity shows all the different camera views each drone has and who is watching them currently. Who’s watching them is shown by a relationship between user account and camera views.

Each camera view would have it’s own unique ID to show what drone it’s from and which camera view it is “1-100”.

#### UserAccount

This entity is used and created by clients using Being There’s services. A client can create an account along with buying a subscription allowing them to watch the many video streams provided.

Each user name is dubbed the primary key and must be unique.

#### UserSubscription

This Entity is being used to allocate a user account with the subscription the client has decided to purchase. It is done like this because it makes it so the SubscriptionType table only must have One entry of each type which is then assigned to numerous clients.

#### SubscriberDetails

This entity stores some details about the Client. This only stores the First Name and Last Name because in my conceptual this was composite attribute and had to be made into its own entity. Email and phone number is not included in this because it was not necessary to separate them from the main UserAccount table.

This Entity is identified by the foreign key from UserAccount tuned into a primary key.

#### SubscriptionType

This entity stores the four types of subscriptions available. By using the join table between This entity and UserAccount I am able to only create each subscription once.

The unique identifier Type is basically what is says it will have Standard Gold etc.

The three foreign keys seen are relationships formed because of what each SubscriptionType can do. The connection to BTDrone allows platinum to control BTDrones. The two CameraView keys are both there for different reasons. One is for Standard subscribers allowing them to only View CameraViews and the other is there for Gold subscribers allowing them to move CameraViews.

#### ContractSubscriptionType

This Entity is the same as the join table between SubscriptionType and UserAccount. But this links a subscription types up to a contract.

Once again this allows me to only have to make one entry in the SubscriptionType table and just link up the SubscriptionType to the contract in question.

#### Contract

This Entity Holds all the details about a contract. Most of the extra details are found in joint tables which includes company and details.

Note: Only a admin staff member is allowed to make a contract who does this is shown by there employee number as a foreign key.

#### Contractees

This Entity just stores the name of the company that has a contract. The company is also the primary key and their details are kept separate in another table.

#### ContracteeDetails

This Entity stores all the details of each individual company and who has there name down as the person requested the contract.

This Entity is identified by the foreign key from the Contractee table turned into a primary key.

#### ContracteeAddress

This Entity stores all the individual addresses of the Contractees which is connected to Contractee Details though a 1-1 relationship. I did this because I felt that if the address changed for the Contractee for any reason it would also affect the phone number, so I decided to separate them both to stop this from happening.

#### AdminStaff

This Entity is where Admin staff are stored. This has been separated like this because only admin staff can create a contact and there for it is easier to allocate it by sorting them this way.

Each staff member also has a discount they can give out to clients this is also stored here. This discount can be up to 3% off a subscription.

This table is uniquely identified by the EmployeeNo foreign key made into a primary key.

#### Staff

This Entity stores all the EmployeeNo of each individual staff member. This is then split up into both AdminStaff and SalesStaff to help separate the two. All staff details are kept in another table though a 1-1 relationship.

#### SalesStaff

This Entity is where Sales staff are stored. It has been separated like this because sales staff cannot create contracts so separating the two staff types helps not to get these two mixed up and have a sales member create a contract by mistake.

Each staff member also has a discount they can give out to clients this is also stored here. This discount can be up to 3% off a subscription.

#### StaffDetails

This Entity stores all the details of each individual Employee. This ranges from name to phone number and date of birth.

This is uniquely identified by the foreign key EmployeeNo turned into a primary key.

#### StaffAddress

This Entity stores all the individual addresses of the Employees which is connected to Staff Details though a 1-1 relationship. I did this because I felt that if the address changed for an Employee for any reason it would also affect the phone number, so I decided to separate them both to stop this from happening.

#### Stall

This Entity Stores all the individual stores around the Globe/Town that subscriptions can be sold at and SalesStaff can work at.

There is not much information stored here only a PhoneNo and an address kept in a separate table.

#### StallAddress

This Entity stores all the individual addresses of the Stalls which is connected to Stall though a 1-1 relationship. I did this because I felt that if the address changed for a Stall for any reason it would also affect the phone number, so I decided to separate them both to stop this from happening.

## Data Dictionary

### Entities

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation Name** | **Start Volume** | **Growth** | **Comments** |
| BTDrone | Over 50 Thousand | However, many Drones are required | Project brief mentions over 50 thousand drones |
| EnvironmentData | 0 | 6 | Each drone records 6 different pieces od data and when sent off to be stored they will be deleted from the drone itself. |
| MaximumRoamingArea | Over 50 Thousand | For However many drones need one | 1 Maximum Area for each drone |
| StreamingDetails | 48,000 | As more drone exist more details exist. | It’s said in the brief that 2000 drones are contracts and would not be available for stream usage. |
| SensedData | 0 | 15552000000 Every year Approx. | Each Drone will send around 311040 sets of data per year |
| MaintenanceLog | Over 50 Thousand | As more drone exist more logs exist. | Each drone would have a Maintenance log |
| Part | Over 5 Million | As more drones exist more parts will | No where in the brief does it state how many parts there are per drone I have guessed at least 100 per drone. |
| Suppliers | Multiple | However, many suppliers needed | There can be many different suppliers. |
| SupplierDetails | Multiple | For each supplier existing | One set of details per supplier in the suppliers table. |
| SupplierAddress | Multiple | For each supplier existing | One address per supplier in the suppliers table. |
| DrovingZone | Over 50 Thousand | For However many drones need one | Each drone will have one droving zone |
| DrovingSize | Over 50 Thousand | For how many Droving zones exist | Join table to connect the Droving zone details up to a droning zone. |
| DrovingSizeDetails | Over 50 Thousand | For how many Droving zones exist | Contains the size of the droving zone |
| CameraViews | Over 5 Million | 100 more for each drone that exists | Each drone has a set of 100 different camera Views. |
| UserAccount | 502,000 | 5% per year | As there is no information on Being there’s growth rate I am guessing 5% per year that adds up to 25,100 per year |
| UserSubscription | 502,000 | 5% per year | This is a Join Table Each User Account will have one subscription |
| SubscriberDetails | 500,000 | 5% per year | Each Subscriber User Account not including the contracts will have their own details |
| SubscriptionType | 4 | None | There are only 4 different types of subscriptions |
| ContractSubscriptionType | 2000 | 1% | This is a join table that gives the contact a subscription type. |
| Contract | 2000 | 1% | As there are no details on the growth for contracts in the brief I have guessed a 1% increase per year |
| Contractees | 2000 | 1% | As the contracts grow so will the Contractees |
| ContracteeDetails | 2000 | 1% | As the contracts grow so will the Contractees |
| ContracteeAddress | 2000 | 1% | As the contracts grow so will the Contractees |
| Staff | Many | When staff are required | As there is no mention of how many staff currently work at Being There I have just stating there as bring many. |
| AdminStaff | Many | When staff are required | As there is no mention of how many Admin staff currently work at Being There I have just stating there as bring many. |
| SalesStaff | Many | When staff are required | As there is no mention of how many Sales staff currently work at Being There I have just stating there as bring many. |
| StaffDetails | Many | As more staff are hired | As there is no mention of how many staff currently work at Being There I have just stating there as bring many. 1 set of details for each staff member |
| StaffAddress | Many | As more staff are hired | As there is no mention of how many staff currently work at Being There I have just stating there as bring many. 1 address for each staff member. |
| Stall | Many | Added as Being There Grows | There is no mention of how many stalls there are currently apart of Being There, so I have just stated it as many. |
| StallAddress | Many | Added as more stalls open | 1 Address for each stall. |

### Attributes

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BTDrone** | DroneID | Unique Drone Serial Number | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  | VideoStreamID | Unique Number of Stream | int | (30) | 0-9 | - | - | Yes | - | VideoStream | FK |
|  | Region | Unique Location of Drone | VarChar | (30) | A-Z  0-9 | - | - | No | - | DrovingZone | FK |
|  | ContractID | Contract Drone Belongs To | int | (30) | 0-9 | - | - | Yes | - | Contract | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Environment**  **Data** | Longitude | The length of the area | int | (30) | 0-9 | - | - | Yes | - | - | - |
|  | Latitude | The Width of the Area | int | (30) | 0-9 | - | - | Yes | - | - | - |
|  | Altitude | The Hight of the Area | int | (30) | 0-9 | - | - | Yes | - | - | - |
|  | Temperature | How Hot/Cold the Area Is | Decimal | (5,2) | 0-9 | - | - | Yes | - | - | - |
|  | Humidity | The Amount of Moisture in The Air | int | (30) | 0-9 | - | - | Yes | - | - | - |
|  | Ambient Light  Strength | How Bright the Area is | int | (30) | 0-9 | - | - | Yes | - | - | - |
|  | BTDrone | Unique Drone Serial Number | int | (30) | 0-9 | - | - | No | PK, FK | BTDrone | PK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Streaming**  **Details** | VideoStreamID | Unique Number of Stream | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  | Time | Time Stream Starts | Time | (7) | 0-9 | HH:MM:SS | - | Yes | - | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Maximum**  **RoamingArea** | Maximum  Altitude | Maximum Hight Drone can go | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Maximum  Latitude | Maximum Width Drone can go | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Maximum  Longitude | Maximum Length Drone can go | int | (30) | 0-9 | - | - | No | - | - | - |
|  | BTDrone | Unique Drone Serial Number | int | (30) | 0-9 | - | - | No | PK, FK | BTDrone | PK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **SensedData** | Longitude | The length of the area | int | (30) | 0-9 | - | - | No |  | - | - |
|  | Latitude | The Width of the Area | int | (30) | 0-9 | - | - | No |  | - | - |
|  | Altitude | The Hight of the Area | int | (30) | 0-9 | - | - | No |  | - | - |
|  | Temperature | How Hot/Cold the Area Is | int | (30) | 0-9 | - | - | No |  | - | - |
|  | Humidity | The Amount of Moisture in The Air | int | (30) | 0-9 | - | - | No |  | - | - |
|  | Ambient Light  Strength | How Bright the Area is | int | (30) | 0-9 | - | - | No |  | - | - |
|  | BTDrone | Unique Drone Serial Number | int | (30) | 0-9 | - | - | No | PK, FK | BTDrone | PK, FK |
|  | Subscription  Type | SubscriptionType Allowed to access the data | VarChar | (30) | A-Z | - | - | No | FK | Subscription  Type | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Maintenance**  **Log** | Status | Condition drone is in | VarChar | (30) | A-Z  0-9 | - | - | No | - | - | - |
|  | Date | Date drone was maintained | Date | (4) | 0-9 | YYYY,MM,DD | - | No | - | - | - |
|  | Comments | Comments made my user | VarChar | (250) | A-Z  0-9 | - | - | Yes | - | - | - |
|  | BTDrone | Unique Drone Serial Number | int | (30) | 0-9 | - | - | No | PK, FK | BTDrone | PK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Part** | PartID | Unique Serial Number of each Part | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  | PartStatus | Condition of the Part | VarChar | (50) | A-Z | - | - | No |  | - | - |
|  | BTDrone | Unique Drone Serial Number | int | (30) | 0-9 | - | - | No | FK | MaintenanceLog/BTDrone | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Suppliers** | CompnayName | Name of Company | VarChar | (30) | A-Z | - | - | No | PK |  | PK |
|  | PartID | Part Company Supplies | int | (30) | 0-9 | - | - | No | FK | Part | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Supplier**  **Details** | PhoneNo | Company Phone Number | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Email | Company Email Address | VarChar | (50) | A-Z  0-9 | - | - | No | - | - | - |
|  | Address | Company Address | VarChar | (50) | A-Z  0-9 | - | - | No | FK | Supplier  Address | FK |
|  | CompanyName | Name of Company | VarChar | (30) | A-Z | - | - | No | PK | Supplier | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Supplier**  **Address** | Address | Company Address | VarChar | (50) | A-Z  0-9 | - | - | No | PK | - | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **DrovingZone** | Region | Region Drone is in | VarChar | (50) | A-Z  0-9 | - | - | No | PK | - | PK |
|  | RegionType | Type of Climate | VarChar | (50) | A-Z | - | - | No |  | - | - |
|  | ContractID | Used if Droving zone is a part of a contract | int | (30) | 0-9 | - | - | Yes | FK | Contract | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **DrovingSize** | Region | Region Drone is in | VarChar | (50) | A-Z  0-9 | - | - | No | CK, FK | DrovingZone | CK, FK |
|  | RegionSizeID | Unique Number for Region Size | int | (30) | 0-9 | - | - | No | CK, FK | DrovingSizeDetails | CK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **DrovingSize**  **Details** | RegionSizeID | Unique Number for Region Size | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  | RegionSize | Total Size of Area in M3 | VarChar | (30) | A-Z  0-9 | - | - | No | - | - | - |
|  | Latitude | Width of Area | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Longitude | Length of Area | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Maximum  Altitude | Maximum Altitude allowed | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Minimum  Altitude | Minimum Altitude allowed | int | (30) | 0-9 | - | - | No | - | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **CameraViews** | CameraView | Camera Serial Number | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  | BTDrone | Drone camera is A part of | int | (30) | 0-9 | - | - | No | FK | BTDrone | FK |
|  | UserName | UserAccount Using/Watching Drone | VarChar | (30) | A-Z  0-9 | - | - | Yes | FK | UserAccount | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **UserAccount** | UserName | UserName of Subscriber | VarChar | (30) | A-Z  0-9 | - | - | No | PK | - | PK |
|  | Email | Email of Subscriber | VarChar | (50) | A-Z  0-9 | - | - | No | - | - | - |
|  | PhoneNo | PhoneNo of Subscriber | int | (30) | 0-9 | - | - | No | - | - | - |
|  | BTDrone | What Drone User is Using/Watching | int | (30) | 0-9 | - | - | Yes | FK | BTDrone | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Subscriber**  **Details** | FirstName | FirstName of Subscriber | VarChar | (30) | A-Z | - | - | No | - | - | - |
|  | LastName | LastName of Subscriber | VarChar | (30) | A-Z | - | - | No | - | - | - |
|  | UserName | Account This Name is Linked to | VarChar | (30) | A-Z  A-Z | - | - | No | PK, FK | UserAccount | PK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **User**  **Subscription** | UserName | User Account with this Subscription | VarChar | (30) | A-Z  0-9 | - | - | No | CK, FK | UserAccount | CK, FK |
|  | Subscription  Type | Subscription of this Account | VarChar | (30) | A-Z | -- | - | No | CK, FK | Subscription  Type | CK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Subscription**  **Type** | Subscription  Type | Name of Subscription | VarChar | (30) | A-Z | - | - | No | PK | - | PK |
|  | Cost | Cost of Subscription | Money | (8) | 0-9 | - | - | No | - | - | - |
|  | BTDrone | This SubscriptionType can Control a Drone | int | (30) | 0-9 | - | - | Yes | FK | BTDrone | FK |
|  | CameraViews | This SubscriptionType can View Cameras | VarChar | (30) | A-Z  0-9 | - | - | Yes | FK | Camera  Views | FK |
|  | CameraViews2 | This SubscriptionType can Control Cameras | VarChar | (30) | A-Z  0-9 | - | - | Yes | FK | Camera  Views | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Contract**  **Subscription**  **Type** | ContractID | Contract With this Subscription | int | (30) | 0-9 | - | - | No | CK, FK | Contract | CK, FK |
|  | Subscription  Type | Subscription With this Contract | VarChar | (30) | A-Z | - | - | No | CK, FK | Subscription  Type | CK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Contract** | ContractID | Unique ID for Contract | int | (30) | 0-9 | - | - | No | PK | - | FK |
|  | Company | Company Registered to this Contract | VarChar | (30) | A-Z  0-9 | - | - | No | FK | Contractee | FK |
|  | EmployeeNo | Staff member who Created Contract | int | (30) | 0-9 | - | - | No | FK | AdminStaff | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Contractees** | Company | Name of Company | VarChar | (30) | A-Z  0-9 | - | - | No | PK | - | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Contractee**  **Details** | FirstName | Contractees FirstName | VarChar | (30) | A-Z | - | - | No | - | - | - |
|  | LastName | Contractees LastName | VarChar | (30) | A-Z | - | - | No | - | - | - |
|  | PhoneNo | Contractees PhoneNo | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Company | Company Client Belongs to | VarChar | (30) | A-Z  0-9 | - | - | No | PK, FK | Contractee | PK, FK |
|  | Address | Companies Address | VarChar | (50) | A-Z  0-9 | - | - | No | FK | Contractee  Address | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Contractee**  **Address** | Address | Address of a Company | VarChar | (50) | A-Z  0-9 | - | - | No | PK | - | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **AdminStaff** | Discount | Discount Staff Member can give | VarChar | (3) | 0-9  % | - | - | No | - | - | - |
|  | EmployeeNo | Unique identifier of employee | int | (30) | 0-9 | - | - | No | PK, FK | Staff | PK, FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Staff** | EmployeeNo | Unique identifier of employee | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **SalesStaff** | Discount | Discount Staff Member can give | VarChar | (3) | 0-9  % | - | - | No | - | - | - |
|  | EmployeeNo | Unique identifier of employee | int | (30) | 0-9 | - | - | No | PK, FK | Staff | PK, FK |
|  | StallNo | Stall Staff Member Works at | int | (30) | 0-9 | - | - | No | FK | Stall | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **StaffDetails** | FirstName | FirstName of Employee | VarChar | (30) | A-Z | - | - | No | - | - | - |
|  | LastName | LastName of Employee | VarChar | (30) | A-Z | - | - | No | - | - | - |
|  | PhoneNo | Phone No of Employee | int | (30) | 0-9 | - | - | No | - | - | - |
|  | DOB | DOB of Employee | Date | (4) | 0-9 | YYYY,MM,DD | - | No | - | - | - |
|  | EmployeeNo | Employee Details belong to | int | (30) | 0-9 | - | - | No | PK, FK | Staff | PK, FK |
|  | Address | Address of Employee | VarChar | (50) | A-Z  0-9 | - | - | No | FK | StaffAddress | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **StaffAddress** | Address | Address of a Staff Member | VarChar | (50) | A-Z  0-9 | - | - | No | PK | - | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **Stall** | StallNo | Unique Number for Stall | int | (30) | 0-9 | - | - | No | PK | - | PK |
|  | PhoneNo | PhoneNo of Stall | int | (30) | 0-9 | - | - | No | - | - | - |
|  | Address | Address of Stall | VarChar | (50) | A-Z  0-9 | - | - | No | FK | StallAddress | FK |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **StallAddress** | Address | Address of a Stall | VarChar | (50) | A-Z  0-9 | - | - | No | PK | - | PK |
|  |  |  |  |  |  |  |  |  |  |  |  |

### Derived attributes

| **Relation Name** | **Attributes** | **Derived from / calculation** |
| --- | --- | --- |
| **Droving**  **Size**  **Details** | RegionSizeID |  |
|  | RegionSize | Longitude X Latitude X MaximumAltitude = RegionSizeM3 |
|  | Latitude |  |
|  | Longitude |  |
|  | MaximumAltitude |  |
|  | MinimumAltitude |  |

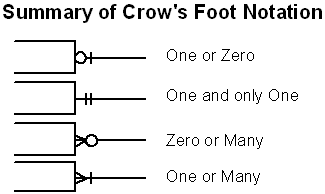
## NaLER Analysis

### Step One

**1st** You would Identify the notation used in creating the logical model and data dictionary and make a list to show other people that may read/use this how to read your diagram.

Crow Foot Notation Has ben used to create the logical diagram and Data Dictionary’s

This is the legend of how to read the logical table.



(Stewart, n.d.)

### Step Two

**2nd Step** is to check your notations and check if they correct I.E. You can't have many to many relationships within a logical diagram.

I have looked over my entire table and feel that all cardinality relationships are correct and there are no M-M Many to Many relationships.

### Step Three

#### 3.1

**3.1** you would write out sentences about how the entity name relates to the primary key If they made sense you are good to go. If not, you may have to make some changes.

Each **BTDrone** Is uniquely identified by **DoneID**

Each **EnvironmentData** Is uniquely identified by **DroneID**

Each **Maximum Roaming Area** Is uniquely identified by **DroneID**

Each **Streaming Details** Is uniquely identified by **VideoStreamID**

Each **Sensed Data** Is uniquely identified by **DroneID**

Each **Maintenance Log** Is uniquely identified by **DroneID**

Each **Part** Is uniquely identified by **DroneID**

Each **Supplier** Is uniquely identified by **CompanyName**

Each **Supplier** **Details** Is uniquely identified by **CompanyName**

Each **Supplier Address** Is uniquely identified by **Address**

Each **Droving Zone** Is uniquely identified by **Region**

Each **Droving Zone Size** Is uniquely identified by **Region and RegionSizeID**

Each **Droving Size Details** Is uniquely identified by **Region Size ID**

Each **Camera Views** Is uniquely identified by **CameraView**

Each **User Account** Is uniquely identified by **UserName**

Each **User Subscription** Is uniquely identified by **UserName and SubscriptionType**

Each **Subscriber Details** Is uniquely identified by **UserName**

Each **Subscription Type** Is uniquely identified by **SubscriptionType**

Each **Contract Subscription Type** Is uniquely identified by **ContractID and SubscriptionType**

Each **Contract** Is uniquely identified by **ContractID**

Each **Contractee** Is uniquely identified by **Company**

Each **Contractee Details** Is uniquely identified by **Company**

Each **Contractee Address** Is uniquely identified by **Address**

Each **Staff** Is uniquely identified by **EmployeeNo**

Each **Admin Staff** Is uniquely identified by **EmployeeNo**

Each **Sales Staff** Is uniquely identified by **EmployeeNo**

Each **Staff Details** Is uniquely identified by **EmployeeNo**

Each **Staff Address** Is uniquely identified by **Address**

Each **Stall** Is uniquely identified by **StallNo**

Each **Stall Address** Is uniquely identified by **Address**

#### 3.2

**3.2** You would write out sentences about how the entity name relates to the primary key and an attribute. If they made sense you are good to go. If not, you may have to make some changes.

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **EnvironmentData** identified by **DroneID** May have one **Longitude**

One **EnvironmentData** identified by **DroneID** May have one **Latitude**

One **EnvironmentData** identified by **DroneID** May have one **Altitude**

One **EnvironmentData** identified by **DroneID** May have one **Temperature**

One **EnvironmentData** identified by **DroneID** May have one **Humidity**

One **EnvironmentData** identified by **DroneID** May have one **AmbientLightStrength**

One **SensedData** identified by **DroneID** May have one **Longitude**

One **SensedData** identified by **DroneID** May have one **Latitude**

One **SensedData** identified by **DroneID** May have one **Altitude**

One **SensedData** identified by **DroneID** May have one **Temperature**

One **SensedData** identified by **DroneID** May have one **Humidity**

One **SensedData** identified by **DroneID** May have one **AmbientLightStrength**

One **MaximumRoamingArea** Identified by **DroneID** Must have one **MaximumAltitude**

One **MaximumRoamingArea** Identified by **DroneID** Must have one **MaximumLatitude**

One **MaximumRoamingArea** Identified by **DroneID** Must have one **MaximumLongitude**

One **StreamingDetails** Identified by **VideoStreamID** May have one **Time**

One **MaintenanceLog** Identified by **DroneID** Must have one **Status**

One **MaintenanceLog** Identified by **DroneID** Must have one **Date**

One **MaintenanceLog** Identified by **DroneID** Must have one **Comments**

One **Part** Identified by **PartID** Must have one **PartStatus**

One **Part** Identified by **PartID** Must have one **DroneID**

One **Suppliers** Identified by **CompanyName** Must have one **PartID**

One **SupplierDetails** Identified by **CompanyName** Must have one **PhoneNo**

One **SupplierDetails** Identified by **CompanyName** Must have one **Email**

One **SupplierDetails** Identified by **CompanyName** Must have one **Address**

One **DrovingZone** Identified by **Region** Must have one **RegionType**

One **DrovingZone** Identified by **Region** May have one **ContractID**

One **DrovingSizeDetails** Identified by **RegionSizeID** May have one **RegionSize**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **Latitude**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **Longitude**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **MaximumAltitude**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **MinimumAltitude**

One **CameraViews** Identified by **CameraView** May have one **DroneID**

One **CameraViews** Identified by **CameraView** May have one **UserName**

One **UserAccount** Identified by **UserName** Must have one **Email**

One **UserAccount** Identified by **UserName** Must have one **PhoneNo**

One **UserAccount** Identified by **UserName** May have one **DroneID**

One **SubscriberDetails** identified by **UserName** Must have one **FirstName**

One **SubscriberDetails** identified by **UserName** Must have one **LastName**

One **SubscriptionType** identified by **SubscriptionType** Must have one **Cost**

One **SubscriptionType** identified by **SubscriptionType** May have one **DroneID**

One **SubscriptionType** identified by **SubscriptionType** May have many **CameraView**

One **Contract** Identified by **ContractID** Must have one **Company**

One **Contract** Identified by **ContractID** Must have one **EmployeeNo**

One **ContracteeDetails** Identified by **Company** Must have one **FirstName**

One **ContracteeDetails** Identified by **Company** Must have one **LastName**

One **ContracteeDetails** Identified by **Company** Must have one **PhoneNo**

One **ContracteeDetails** Identified by **Company** Must have one **Address**

One **AdminStaff** Identified by **EmployeeNo** May have one **Discount**

One **SalesStaff** Identified by **EmployeeNo** May have one **Discount**

One **SalesStaff** Identified by **EmployeeNo** Must have one **StallNo**

One **StaffDetails** Identified by **EmployeeNo** Must have one **FirstName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **LastName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **PhoneNo**

One **StaffDetails** Identified by **EmployeeNo** Must have one **DOB**

One **StaffDetails** Identified by **EmployeeNo** Must have one **Address**

One **Stall** Identified by **StallNo** Must have one **PhoneNo**

One **Stall** Identified by **StallNo** Must have one **Address**

#### 3.3

**3.3** You would write out sentences about how the entity name relates to the primary key and a Foreign Key. If they made sense you are good to go. If not, you may have to make some changes. This is written also like business rules It Must be written in both directions/ways.

One **BTDrone** identified by **DroneID** Must Record one or more **EnvironmentData** Identified by **DroneID**

One **EnvironmentData** identified by **DroneID** Must Belong to one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** May Stream one **StreamingDetails** Identified by **VideoStreamID**

One **StreamingDetails** Identified by **VideoStreamID** May be Streamed by one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** Must have one **MaximumRoamingArea** Identified by **DroneID**

One **MaximumRoamingArea** identified by **DroneID** Must have one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** Must have one or more **SensedData** Identified by **DroneID**

One **SensedData** identified by **DroneID** Must Belong to one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** Must have one **MaintenanceLog** Identified by **DroneID**

One **MaintenanceLog** identified by **DroneID** Must Belong to one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** May have one or more **CameraViews** Identified by **CameraView**

One **CameraViews** identified by **CameraView** May Belong to one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** May be watched by one or more **UserAccount** Identified by **UserName**

One **UserAccount** identified by **UserName** May watched one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** May be Controlled by one **SubscriptionType** Identified by **SubscriptionType**

One **SubscriptionType** identified by **SubscriptionType** May Control one **BtDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** Must have one **DrovingZone** Identified by **Region**

One **DrovingZone** identified by **Region** May have one **BTDrone** Identified by **DroneID**

One **BTDrone** identified by **DroneID** May be a part of one **Contract** Identified by **ContractID**

One **Contract** identified by **ContractID** May have one or more **BTDrone** Identified by **DroneID**

One **MaintenanceLog** Identified by **DroneID** Must have one or more **Parts** identified by **PartID**

One **Part** Identified by **PartID** May be a part of one **MaintenanceLog** identified by **DroneID**

One **Part** Identified by **PartID** May be supplied by one or more **Suppliers** identified by **CompanyName**

One **Supplier** Identified by **CompanyName** May be supply one **Part** identified by **PartID**

One **Supplier** Identified by **CompanyName** must have one **SupplierDetails** identified by **CompanyName**

One **SupplierDetails** Identified by **CompanyName** must have one **SupplierAddress** identified by **Address**

One **SupplierAddress** Identified by **Address** must belong to one **SupplierDetails** Identified by **CompanyName**

One **CameraViews** Identified by **CameraView** May be Viewed by one **UserAccount** Identified by **UserName**

One **UserAccount** Identified by **UserName** May View one **CameraViews** Identified by **CameraView**

One **CameraViews** Identified by **CameraView** May be Viewed by one or more **SubscriptionType** Identified by **SubscriptionType**

One **CameraViews** Identified by **CameraView** May be Controlled by one **SubscriptionType** Identified by **SubscriptionType**

One **SubscriptionType** Identified by **SubscriptionType** May View one or more **CameraViews** Identified by **CameraView**

One **SubscriptionType** Identified by **SubscriptionType** May Control one **CameraViews** Identified by **CameraView**

One **UserAccount** Identified by **UserName** Must have one **SubscriberDetails** Identified by **UserName**

One **SubscriberDetails** Identified by **UserName** Must belong to one **UserAccount** Identified by **UserName**

One **UserAccount** Identified by **UserName** Must have one **UserSubscription** Identified by **UserName** and **SubscriptionType**

One **UserSubscription** Identified by **UserName** and **SubscriptionType** Must have one **UserAccount** Identified by **UserName**

One **UserSubscription** Identified by **UserName** and **SubscriptionType** Must have one **SubscriptionType** Identified by **SubscriptionType**

One **SubscriptionType** Identified by **SubscriptionType** May have one or more **UserSubscription** Identified by **UserName** and **SubscriptionType**

One **SubscriptionType** Identified by **SubscriptionType** May have one or more **ContractSubscriptionType** Identified by **ContractID** and **SubscriptionType**

One **ContractSubscriptionType** Identified by **ContractID** and **SubscriptionType** Must have one **SubscriptionType** Identified by **SubscriptionType**

One **SubscriptionType** Identified by **SubscriptionType** May access one or more **SencedData** Identified by **DroneID**

One **SensedData** Identified by **DroneID** May be accessed by one **SubscriptionType** Identified by **SubscriptionType**

One **ContractSubscriptionType** Identified by **ContractID** and **SubscriptionType** Must have one **Contract** Identified by **ContractID**

One **Contract** Identified by **ContractID** Must have one **ContractSubscriptionType** Identified by **ContractID** and **SubscriptionType**

One **Contract** Identified by **ContractID** Must have one **Contractee** Identified by **Company**

One **Contractee** Identified by **Company** Must belong to one **Contract** Identified by **ContractID**

One **Contract** Identified by **ContractID** May have one or more **DrovingZones** Identified by **Region**

One **Contract** Identified by **ContractID** Must be created by one **AdminStaff** Identified by **EmployeeNo**

One **AdminStaff** Identities by **EmployeeNo** May Create a **Contract** Identified by **ContractID**

One **Contractee** Identified by **Company** Must have one **ContracteeDetails** Identified by **Company**

One **ContracteeDetails** Identified by **Company** Must have one **Contract** Identified by **Company**

One **ContracteeDetails** Identified by **Company** Must have one **ContracteeAddress** Identified by **Address**

One **ContracteeAddress** Identified by **Address** Must have one **ContracteeDetails** Identified by **Company**

One **AdminStaff** Identified by **EmployeeNo** Must have one **Staff** Identified by **EmployeeNo**

One **Staff** Identified by **EmployeeNo** May be a part of one **AdminStaff** Identified by **EmployeeNo**

One **Staff** Identified by **EmployeeNo** May be a part of one **SalesStaff** Identified by **EmployeeNo**

One **SalesStaff** Identified by **EmployeeNo** Must have one **Staff** Identified by **EmployeeNo**

One **Staff** Identified by **EmployeeNo** Must have one **StaffDetails** Identified by **EmployeeNo**

One **StaffDetails** Identified by **EmployeeNo** Must have one **Staff** Identified by **EmployeeNo**

One **StaffDetails** Identified by **EmployeeNo** Must have one **StaffAddress** Identified by **Address**

One **StaffAddress** Identified by **Address** Must have one **StaffDetails** Identified by **EmployeeNo**

One **SalesStaff** Identified by **EmployeeNo** Must work at one **Stall** Identified by **StallNo**

One **Stall** Identified by **StallNo** Must have one or more **SalesStaff** Identified by **EmployeeNo**

One **Stall** Identified by **StallNo** Must have one **StallAddress** Identified by **Address**

One **StallAddress** Identified by **Address** Must have one **Stall** Identified by **StallNo**

One **DrovingZone** Identified by **Region** Must have one **DrovingSize** Identified by **Region** and **RegionSizeID**

One **DrovingSize** Identified by **Region** and **RegionSizeID** Must have one **DrovingZone** Identified by **Region**

One **DrovingSize** Identified by **Region** and **RegionSizeID** Must have one **DrovingSizeDetails** Identified by **RegionSizeID**

One **DrovingSizeDetails** Identified by **Region** Must have one **DrovingSize** Identified by **Region** and **RegionSizeID**

### Step 4

**Step 4** You would Populate the sentences from the 2 steps above with relevant data and write them out to see if they make sense. If they do your good to go.

**BTDrone** **001** May have **VideoStream 001**

**BTDrone** **001** Must have **Region Amazon Forest**

**BTDrone 001** May belong to **Contract 246**

**EnvironmentData 001** May have **Longitude of 1092**

**EnvironmentData 001** May have **Latitude of 1762**

**EnvironmentData 001** May have **Altitude of 5684**

**EnvironmentData 001** May have **Temperature of 25**

**EnvironmentData 001** May have **Humidity of 14**

**EnvironmentData 001** May have **AmbientLightStrength of 56**

**SensedData** **001** May have **Longitude 1092**

**SensedData** **001** May have **Latitude 1762**

**SensedData** **001** May have **Altitude 5684**

**SensedData** **001** May have **Temperature 25**

**SensedData** **001** May have **Humidity 14**

**SensedData** **001** May have one **AmbientLightStrength**

**MaximumRoamingArea** **001** Must have **MaximumAltitude of 568**

**MaximumRoamingArea 001** Must have **MaximumLatitude of 450**

**MaximumRoamingArea 001** Must have **MaximumLongitude of 340**

**StreamingDetails 78** May start at **3pm**

**MaintenanceLog** **001** May be **Good**

**MaintenanceLog 001** was maintained on **5/6/2018**

**MaintenanceLog 001** **is Good and has no problems**

**Part** **24567 is in working condition**

**Part** **14678** belongs to Drone **001**

**Supplier Bobs Parts** supplies part **12576**

**SupplierDetails Bobs Parts** Must have a **PhoneNo**

**SupplierDetails Bobs Parts** Must have an **Email**

**SupplierDetails Bobs Parts** Must have an **Address**

**DrovingZone** **Amazon** is a **Forest Region**

**DrovingZone** **Amazon** is a part of the **Contract 12763**

**DrovingSizeDetails** **12345** May be **100m2**

**DrovingSizeDetails 12345** has a latitude of **1764**

**DrovingSizeDetails 12345** has a longitude of **5689**

**DrovingSizeDetails 12345** has a MaximumAltitude of **1500**

**DrovingSizeDetails 12345** has a MinimumAltitude of **1000**

**CameraView 002** May belong to Drone **001**

**CameraView 002** May be used my **Harry28**

**UserAccount** **Harry28** has the email **Harry28@gmail.com**

**UserAccount Harry28** has the PhoneNo **5440000**

**UserAccount Harry28** May use Drone **001**

**SubscriberDetails** **Harry28** has the FirstName **Harry**

**SubscriberDetails** **Harry28** has the LastName **Jones**

**SubscriptionType** **Standard** Costs **$15** per month

**SubscriptionType** **Platinum** May Control Drone **001**

**SubscriptionType** **Standard** May View CameraView **002**

**SubscriptionType** **Gold** May Control CameraView **002**

**Contract** **001** belongs to **Hight’s Data**

**Contractee** **Hight’s Data** belongs to **Contract** **001**

**Contract** **001** Must be created by EmployeeNo **1285**

**AdminStaff** **1285** May Create the **Contract** **001**

**ContracteeDetails** **bobs parts** is managed by FirstName **Mark**

**ContracteeDetails** **bobs parts** is managed by LastName **Lemming**

**ContracteeDetails** **bobs parts** has the PhoneNo **5442211**

**ContracteeDetails** **bobs parts** has an address **14 Newman’s road nelson**

**AdminStaff** **0056** can give out a discount of **3%**

**SalesStaff** **0058** can give out a discount of **1%**

**SalesStaff** **0058** Must work at stall **41**

**StaffDetails** **0058** has the first Name **Peter**

**StaffDetails** **0058** has the last Name **Hall**

**StaffDetails** **0058** has the PhoneNo **5448888**

**StaffDetails** **0058** has a DOB of **5/9/1994**

**StaffDetails** **0058** has an Address of **78 Knox road Wakefield**

**Stall** **98** has the PhoneNo **5461298**

**Stall** **98** has the address **67 Mayfair lane Auckland**

**BTDrone** **001** Records **EnvironmentData** 11,22,33,44,55,66,**001**

**EnvironmentData** **001** Belong to **BTDrone** **001**

**BTDrone 001** Has **StreamingDetails** **4754**

**StreamingDetails** **4754** is streamed by **BTDrone** **001**

**BTDrone 001** has the **MaximumRoamingArea** **001**

**MaximumRoamingArea 001** is used by **BTDrone 001**

**BTDrone** **001** records **SensedData** **001**

**SensedData** **001** is recorded by **BTDrone 001**

**BTDrone** **001** has the **MaintenanceLog** **001**

**MaintenanceLog** **001** belongs to **BTDrone** **001**

**BTDrone** **001** Has **CameraViews** **1-100**

**CameraViews** **1-100** belongs to **BTDrone** **001**

**BTDrone** **001** can be watched by **UserAccount Harry28**

**UserAccount** **Harry28** can use **BTDrone 001**

**BTDrone** 001 can be controlled by **SubscriptionType Platinum**

**SubscriptionType** **Platinum** can control **BTDrone 001**

**BTDrone** **001** Uses **DrovingZone Egypt**

**DrovingZone** **Egypt** can be used by **BTDrone 001**

**BTDrone** **001** can be a part of **Contract 7854**

**Contract** **7854** has **BTDrone 001**

**MaintenanceLog** **001** has **Part** **1579**

**Part** **1579** is a part of the **MaintenanceLog 001**

**Part** **1579** comes from the **Supplier Bobs Parts**

**Supplier** **bobs** parts supply’s **Part 1579**

**Supplier** **bobs parts** has one **SupplierDetails bobs parts**

**SupplierDetails** **bobs parts** has the **Address 45 Aron road Richmond**

**CameraView 25** can be Viewed by **UserAccount Harry28**

**UserAccount Harry28** can view **CameraView 25**

**CameraView 25** May be Viewed by the **SubscriptionType Standard**

**CameraViews 45** May be Controlled by **SubscriptionType Gold**

**UserAccount Harry28** has **SubscriberDetails** **Harry28**

**SubscriberDetails Harry28** belongs to **UserAccount Harry28**

**UserAccount Harry28** has the **UserSubscription** **Harry28, Gold**

**UserSubscription** **Harry28, Gold** is the subscription for **UserAccount** **Harry28**

**UserSubscription** **Harry28, Gold** has the **SubscriptionType** **Gold**

**SubscriptionType** **Gold** has a **UserSubscription** **Harry28, Gold**

**SubscriptionType** **SuperPlatinum** has a **ContractSubscriptionType** **SuperPlatinum, 5874**

**ContractSubscriptionType** **SuperPlatinum, 5874** has one **SubscriptionType** **SuperPlatinum**

**SubscriptionType** **SuperPlatinum** can access **SensedData** **001**

**SensedData 001** can be accessed by **SubscriptionType** **SuperPlatinum**

**ContractSubscriptionType** **SuperPlatinum, 1287** is used by **Contract** **1287**

**Contract** **1287** has one **ContractSubscriptionType** Identified by **SuperPlatinum, 1287**

**Contract** **1287** has one **Contractee** **Marks Data**

**Contract** **1287** is assigned one **DrovingZone** **Egypt**

**DrovingZone** **Egypt** May belong to **Contract** **001**

**Contract** **1287** is created by **AdminStaff** **1985**

**Contractee** **Marks Data** has one **ContracteeDetails** **Marks Data**

**ContracteeDetails** **Marks Data** is a part of **Contractee** **Marks Data**

**ContracteeDetails** **Marks Data** has one **ContracteeAddress** **78 bob lane Wellington**

**ContracteeAddress** **78 bob lane Wellington** is a part of **ContracteeDetails** **Marks Parts**

**AdminStaff** **1896** is a part of **Staff** **1896**

**Staff** **1896** has the position **AdminStaff** **1896**

**Staff** **1987** has the position **SalesStaff** **1987**

**SalesStaff** **1987** is a part of **Staff** **1987**

**Staff** **1987** has a set of **StaffDetails** **1987**

**StaffDetails** **1987** are for the **Staff** **1987**

**StaffDetails** **1987** Must have one **StaffAddress** **89 Matter road Hope**

**StaffAddress** **89 Matter road Hope** Must belong to **StaffDetails** **1987**

**SalesStaff 1987** works at **Stall** **12**

**Stall** **12** has **SalesStaff** **1987**

**Stall** **12** has one **StallAddress** **78 cow lane Springfield**

**StallAddress** **78 cow lane Springfield** belongs to the **Stall** **12**

**DrovingZone** **Egypt** has one **DrovingSize** **Egypt,1234**

**DrovingSize** **Egypt,1234** has one **DrovingZone** **Egypt**

**DrovingSize** **Egypt,1234** has **DrovingSizeDetails** **1234**

**DrovingSizeDetails** **1234** has one **DrovingSize** **Egypt,1234**

### Step 5

#### 5A

**5A** You would write out sentences with Entities and Primary keys. This one is used if you have a composite key within a table. Note- This one is only used if there are only primary keys/attributes on the table.

One **UserAccount** Identified by **UserName** Must have one **UserSubscription** Identified by **UserName** which Must have one **SubscriptionType** Identified by **SubscriptionType**

One **Contract** Identified by **ContractID** Must have one **ContractSubscriptionType** Identified by **ContractID** which Must have one **SubscriptionType** Identified by **SubscriptionType**

Once **DrovingZone** Identified by **Region** Must have one **DrovingSize** Identified by **Region** which Must have one **DrovingSizeDetails** Identified by **RegionSizeID**

#### 5B

**5B** You would write out sentences with Entities and Primary keys and foreign keys. This one is used if you have a composite key within a table. Note- This one is only used if there are only primary keys/foreign keys and attributes on the table.

**As I do not have any tables with all the requirements above I have skipped this Step**

### Step 6

**Step 6**The last step is simple You compile all your sentences from the previous steps and format them appropriately.

#### BTDrone/MaximumRoamingData

Each **Maximum Roaming Area** Is uniquely identified by **DroneID**

One **MaximumRoamingArea** Identified by **DroneID** Must have one **MaximumAltitude**

One **MaximumRoamingArea** Identified by **DroneID** Must have one **MaximumLatitude**

One **MaximumRoamingArea** Identified by **DroneID** Must have one **MaximumLongitude**

One **MaximumRoamingArea** identified by **DroneID** Must have one **BTDrone** Identified by **DroneID**

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** Must have one **MaximumRoamingArea** Identified by **DroneID**

#### BTDrone/StreamingDetails

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** identified by **DroneID** May Stream one **StreamingDetails** Identified by **VideoStreamID**

Each **Streaming Details** Is uniquely identified by **VideoStreamID**

One **StreamingDetails** Identified by **VideoStreamID** May have one **Time**

One **StreamingDetails** Identified by **VideoStreamID** May be Streamed by one **BTDrone** Identified by **DroneID**

#### BTDrone/EnvironmentData

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

Each **EnvironmentData** Is uniquely identified by **DroneID**

One **EnvironmentData** identified by **DroneID** May have one **Longitude**

One **EnvironmentData** identified by **DroneID** May have one **Latitude**

One **EnvironmentData** identified by **DroneID** May have one **Altitude**

One **EnvironmentData** identified by **DroneID** May have one **Temperature**

One **EnvironmentData** identified by **DroneID** May have one **Humidity**

One **EnvironmentData** identified by **DroneID** May have one **AmbientLightStrength**

One **BTDrone** identified by **DroneID** Must Record one or more **EnvironmentData** Identified by **DroneID**

One **EnvironmentData** identified by **DroneID** Must Belong to one **BTDrone** Identified by **DroneID**

#### BTDrone/SensedData

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** Must have one or more **SensedData** Identified by **DroneID**

Each **Sensed Data** Is uniquely identified by **DroneID**

One **SensedData** identified by **DroneID** May have one **Longitude**

One **SensedData** identified by **DroneID** May have one **Latitude**

One **SensedData** identified by **DroneID** May have one **Altitude**

One **SensedData** identified by **DroneID** May have one **Temperature**

One **SensedData** identified by **DroneID** May have one **Humidity**

One **SensedData** identified by **DroneID** May have one **AmbientLightStrength**

One **SensedData** identified by **DroneID** Must Belong to one **BTDrone** Identified by **DroneID**

#### SensedData/SubscriptionType

Each **Sensed Data** Is uniquely identified by **DroneID**

One **SensedData** identified by **DroneID** May have one **Longitude**

One **SensedData** identified by **DroneID** May have one **Latitude**

One **SensedData** identified by **DroneID** May have one **Altitude**

One **SensedData** identified by **DroneID** May have one **Temperature**

One **SensedData** identified by **DroneID** May have one **Humidity**

One **SensedData** identified by **DroneID** May have one **AmbientLightStrength**

One **SensedData** Identified by **DroneID** May be accessed by one **SubscriptionType** Identified by **SubscriptionType**

Each **Subscription Type** Is uniquely identified by **SubscriptionType**

One **SubscriptionType** identified by **SubscriptionType** Must have one **Cost**

One **SubscriptionType** identified by **SubscriptionType** May have one **DroneID**

One **SubscriptionType** identified by **SubscriptionType** May have many **CameraView**

One **SubscriptionType** Identified by **SubscriptionType** May access one or more **SensedData** Identified by **DroneID**

#### BTDrone/Contract

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** May be a part of one **Contract** Identified by **ContractID**

Each **Contract** Is uniquely identified by **ContractID**

One **Contract** Identified by **ContractID** Must have one **Company**

One **Contract** Identified by **ContractID** Must have one **EmployeeNo**

One **Contract** identified by **ContractID** May have one or more **BTDrone** Identified by **DroneID**

#### BTDrone/DrovingZone

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** Must have one **DrovingZone** Identified by **Region**

Each **Droving Zone** Is uniquely identified by **Region**

One **DrovingZone** Identified by **Region** Must have one **RegionType**

One **DrovingZone** Identified by **Region** May have one **ContractID**

One **DrovingZone** identified by **Region** May have one **BTDrone** Identified by **DroneID**

#### BTDrone/SubscriptionType

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** May be Controlled by one **SubscriptionType** Identified by **SubscriptionType**

Each **Subscription Type** Is uniquely identified by **SubscriptionType**

One **SubscriptionType** identified by **SubscriptionType** Must have one **Cost**

One **SubscriptionType** identified by **SubscriptionType** May have one **DroneID**

One **SubscriptionType** identified by **SubscriptionType** May have many **CameraView**

One **SubscriptionType** identified by **SubscriptionType** May Control one **BtDrone** Identified by **DroneID**

#### BTDrone/UserAccount

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** May be watched by one or more **UserAccount** Identified by **UserName**

Each **User Account** Is uniquely identified by **UserName**

One **UserAccount** Identified by **UserName** Must have one **Email**

One **UserAccount** Identified by **UserName** Must have one **PhoneNo**

One **UserAccount** Identified by **UserName** May have one **DroneID**

One **UserAccount** identified by **UserName** May watched one **BTDrone** Identified by **DroneID**

#### BTDrone/CameraViews

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** May have one or more **CameraViews** Identified by **CameraView**

Each **Camera Views** Is uniquely identified by **CameraView**

One **CameraViews** Identified by **CameraView** May have one **DroneID**

One **CameraViews** Identified by **CameraView** May have one **UserName**

One **CameraViews** identified by **CameraView** May Belong to one **BTDrone** Identified by **DroneID**

#### BTDrone/MaintenanceLog

Each **BTDrone** Is uniquely identified by **DoneID**

One **BTDrone** Identified by **DoneID** May have one **VideoStream**

One **BTDrone** Identified by **DoneID** Must have one **Region**

One **BTDrone** Identified by **DoneID** May belong to one **Contract**

One **BTDrone** identified by **DroneID** Must have one **MaintenanceLog** Identified by **DroneID**

Each **Maintenance Log** Is uniquely identified by **DroneID**

One **MaintenanceLog** Identified by **DroneID** Must have one **Status**

One **MaintenanceLog** Identified by **DroneID** Must have one **Date**

One **MaintenanceLog** Identified by **DroneID** Must have one **Comments**

One **MaintenanceLog** identified by **DroneID** Must Belong to one **BTDrone** Identified by **DroneID**

#### MaintenanceLog/Part

Each **Part** Is uniquely identified by **DroneID**

One **Part** Identified by **PartID** Must have one **PartStatus**

One **Part** Identified by **PartID** Must have one **DroneID**

One **Part** Identified by **PartID** May be a part of one **MaintenanceLog** identified by **DroneID**

Each **Maintenance Log** Is uniquely identified by **DroneID**

One **MaintenanceLog** Identified by **DroneID** Must have one **Status**

One **MaintenanceLog** Identified by **DroneID** Must have one **Date**

One **MaintenanceLog** Identified by **DroneID** Must have one **Comments**

One **MaintenanceLog** Identified by **DroneID** Must have one or more **Parts** identified by **PartID**

#### Part/Suppliers

Each **Part** Is uniquely identified by **DroneID**

One **Part** Identified by **PartID** Must have one **PartStatus**

One **Part** Identified by **PartID** Must have one **DroneID**

One **Part** Identified by **PartID** May be supplied by one or more **Suppliers** identified by **CompanyName**

Each **Supplier** Is uniquely identified by **CompanyName**

One **Suppliers** Identified by **CompanyName** Must have one **PartID**

One **Supplier** Identified by **CompanyName** May be supply one **Part** identified by **PartID**

#### Suppliers/SupplierDetails

Each **Supplier** Is uniquely identified by **CompanyName**

One **Suppliers** Identified by **CompanyName** Must have one **PartID**

One **Supplier** Identified by **CompanyName** must have one **SupplierDetails** identified by **CompanyName**

Each **Supplier** **Details** Is uniquely identified by **CompanyName**

One **SupplierDetails** Identified by **CompanyName** Must have one **PhoneNo**

One **SupplierDetails** Identified by **CompanyName** Must have one **Email**

One **SupplierDetails** Identified by **CompanyName** Must have one **Address**

One **SupplierDetails** Identified by **CompanyName** must have one **Address** identified by **Address**

#### SupplierDetails/SupplierAddress

Each **Supplier** **Details** Is uniquely identified by **CompanyName**

One **SupplierDetails** Identified by **CompanyName** Must have one **PhoneNo**

One **SupplierDetails** Identified by **CompanyName** Must have one **Email**

One **SupplierDetails** Identified by **CompanyName** Must have one **Address**

One **SupplierDetails** Identified by **CompanyName** must have one **SupplierAddress** identified by **Address**

Each **Supplier Address** Is uniquely identified by **Address**

One **SupplierAddress** Identified by **Address** must belong to one **SupplierDetails** Identified by **CompanyName**

#### CameraViews/SubscriptionType

Each **Camera Views** Is uniquely identified by **CameraView**

One **CameraViews** Identified by **CameraView** May have one **DroneID**

One **CameraViews** Identified by **CameraView** May have one **UserName**

One **CameraViews** Identified by **CameraView** May be Viewed by one or more **SubscriptionType** Identified by **SubscriptionType**

One **CameraViews** Identified by **CameraView** May be Controlled by one **SubscriptionType** Identified by **SubscriptionType**

Each **Subscription Type** Is uniquely identified by **SubscriptionType**

One **SubscriptionType** identified by **SubscriptionType** Must have one **Cost**

One **SubscriptionType** identified by **SubscriptionType** May have one **DroneID**

One **SubscriptionType** identified by **SubscriptionType** May have many **CameraView**

One **SubscriptionType** Identified by **SubscriptionType** May View one or more **CameraViews** Identified by **CameraView**

One **SubscriptionType** Identified by **SubscriptionType** May Control one **CameraViews** Identified by **CameraView**

#### CameraViews/UserAccount

Each **Camera Views** Is uniquely identified by **CameraView**

One **CameraViews** Identified by **CameraView** May have one **DroneID**

One **CameraViews** Identified by **CameraView** May have one **UserName**

One **CameraViews** Identified by **CameraView** May be Viewed by one **UserAccount** Identified by **UserName**

Each **User Account** Is uniquely identified by **UserName**

One **UserAccount** Identified by **UserName** Must have one **Email**

One **UserAccount** Identified by **UserName** Must have one **PhoneNo**

One **UserAccount** Identified by **UserName** May have one **DroneID**

One **UserAccount** Identified by **UserName** May View one **CameraViews** Identified by **CameraView**

#### UserAccount/SubscriberDetails

Each **User Account** Is uniquely identified by **UserName**

One **UserAccount** Identified by **UserName** Must have one **Email**

One **UserAccount** Identified by **UserName** Must have one **PhoneNo**

One **UserAccount** Identified by **UserName** May have one **DroneID**

One **UserAccount** Identified by **UserName** Must have one **SubscriberDetails** Identified by **UserName**

Each **Supplier** **Details** Is uniquely identified by **CompanyName**

One **SupplierDetails** Identified by **CompanyName** Must have one **PhoneNo**

One **SupplierDetails** Identified by **CompanyName** Must have one **Email**

One **SupplierDetails** Identified by **CompanyName** Must have one **Address**

One **SubscriberDetails** Identified by **UserName** Must belong to one **UserAccount** Identified by **UserName**

#### UserAccount/UserSubscription

Each **User Account** Is uniquely identified by **UserName**

One **UserAccount** Identified by **UserName** Must have one **Email**

One **UserAccount** Identified by **UserName** Must have one **PhoneNo**

One **UserAccount** Identified by **UserName** May have one **DroneID**

One **UserAccount** Identified by **UserName** Must have one **UserSubscription** Identified by **UserName** and **SubscriptionType**

Each **User Subscription** Is uniquely identified by **UserName and SubscriptionType**

One **UserSubscription** Identified by **UserName** and **SubscriptionType** Must have one **UserAccount** Identified by **UserName**

#### UserSubscription/SubscriptionType

Each **User Subscription** Is uniquely identified by **UserName and SubscriptionType**

One **UserSubscription** Identified by **UserName** and **SubscriptionType** Must have one **SubscriptionType** Identified by **SubscriptionType**

Each **Subscription Type** Is uniquely identified by **SubscriptionType**

One **SubscriptionType** identified by **SubscriptionType** Must have one **Cost**

One **SubscriptionType** identified by **SubscriptionType** May have one **DroneID**

One **SubscriptionType** identified by **SubscriptionType** May have many **CameraView**

One **SubscriptionType** Identified by **SubscriptionType** May have one or more **UserSubscription** Identified by **UserName** and **SubscriptionType**

#### SubscriptionType/ContractSubscriptionType

Each **Subscription Type** Is uniquely identified by **SubscriptionType**

One **SubscriptionType** identified by **SubscriptionType** Must have one **Cost**

One **SubscriptionType** identified by **SubscriptionType** May have one **DroneID**

One **SubscriptionType** identified by **SubscriptionType** May have many **CameraView**

One **SubscriptionType** Identified by **SubscriptionType** May have one or more **ContractSubscriptionType** Identified by **ContractID** and **SubscriptionType**

Each **Contract Subscription Type** Is uniquely identified by **ContractID and SubscriptionType**

One **ContractSubscriptionType** Identified by **ContractID** and **SubscriptionType** Must have one **SubscriptionType** Identified by **SubscriptionType**

#### Contract/AdminStaff

Each **Contract** Is uniquely identified by **ContractID**

One **Contract** Identified by **ContractID** Must have one **Company**

One **Contract** Identified by **ContractID** Must have one **EmployeeNo**

One **Contract** Identified by **ContractID** Must be created by one **AdminStaff** Identified by **EmployeeNo**

Each **Admin Staff** Is uniquely identified by **EmployeeNo**

One **AdminStaff** Identified by **EmployeeNo** May have one **Discount**

One **Contract** Identified by **ContractID** Must be created by one **AdminStaff** Identified by **EmployeeNo**

One **AdminStaff** Identities by **EmployeeNo** May Create a **Contract** Identified by **ContractID**

#### Contract/DrovingZone

Each **Contract** Is uniquely identified by **ContractID**

One **Contract** Identified by **ContractID** Must have one **Company**

One **Contract** Identified by **ContractID** Must have one **EmployeeNo**

One **Contract** Identified by **ContractID** May have one or more **DrovingZones** Identified by **Region**

Each **Droving Zone** Is uniquely identified by **Region**

One **DrovingZone** Identified by **Region** Must have one **RegionType**

One **DrovingZone** Identified by **Region** May have one **ContractID**

One **DrovingZone** Identified by **Region** May belong to one **Contract** Identified by **ContractID**

#### Contract/Contractee

Each **Contract** Is uniquely identified by **ContractID**

One **Contract** Identified by **ContractID** Must have one **Company**

One **Contract** Identified by **ContractID** Must have one **EmployeeNo**

One **Contract** Identified by **ContractID** Must have one **Contractee** Identified by **Company**

Each **Contractee** Is uniquely identified by **Company**

One **Contractee** Identified by **Company** Must belong to one **Contract** Identified by **ContractID**

#### Contractee/ContracteeDetails

Each **Contractee** Is uniquely identified by **Company**

One **Contractee** Identified by **Company** Must have one **ContracteeDetails** Identified by **Company**

Each **Contractee Details** Is uniquely identified by **Company**

One **ContracteeDetails** Identified by **Company** Must have one **FirstName**

One **ContracteeDetails** Identified by **Company** Must have one **LastName**

One **ContracteeDetails** Identified by **Company** Must have one **PhoneNo**

One **ContracteeDetails** Identified by **Company** Must have one **Address**

One **ContracteeDetails** Identified by **Company** Must have one **Contract** Identified by **Company**

#### ContracteeDetails/ContracteeAddress

Each **Contractee Details** Is uniquely identified by **Company**

One **ContracteeDetails** Identified by **Company** Must have one **FirstName**

One **ContracteeDetails** Identified by **Company** Must have one **LastName**

One **ContracteeDetails** Identified by **Company** Must have one **PhoneNo**

One **ContracteeDetails** Identified by **Company** Must have one **Address**

One **ContracteeDetails** Identified by **Company** Must have one **ContracteeAddress** Identified by **Address**

Each **Contractee Address** Is uniquely identified by **Address**

One **ContracteeAddress** Identified by **Address** Must have one **ContracteeDetails** Identified by **Company**

#### AdminStaff/Staff

Each **Staff** Is uniquely identified by **EmployeeNo**

One **Staff** Identified by **EmployeeNo** May be a part of one **AdminStaff** Identified by **EmployeeNo**

Each **Admin Staff** Is uniquely identified by **EmployeeNo**

One **AdminStaff** Identified by **EmployeeNo** May have one **Discount**

One **AdminStaff** Identified by **EmployeeNo** Must have one **Staff** Identified by **EmployeeNo**

#### Staff/StaffDetails

Each **Staff** Is uniquely identified by **EmployeeNo**

One **Staff** Identified by **EmployeeNo** Must have one **StaffDetails** Identified by **EmployeeNo**

Each **Subscriber Details** Is uniquely identified by **UserName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **FirstName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **LastName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **PhoneNo**

One **StaffDetails** Identified by **EmployeeNo** Must have one **DOB**

One **StaffDetails** Identified by **EmployeeNo** Must have one **Address**

One **StaffDetails** Identified by **EmployeeNo** Must have one **Staff** Identified by **EmployeeNo**

#### Staff/SalesStaff

Each **Staff** Is uniquely identified by **EmployeeNo**

One **Staff** Identified by **EmployeeNo** May be a part of one **SalesStaff** Identified by **EmployeeNo**

Each **Sales Staff** Is uniquely identified by **EmployeeNo**

One **SalesStaff** Identified by **EmployeeNo** May have one **Discount**

One **SalesStaff** Identified by **EmployeeNo** Must have one **StallNo**

One **SalesStaff** Identified by **EmployeeNo** Must have one **Staff** Identified by **EmployeeNo**

#### StaffDetails/StaffAddress

Each **Staff Address** Is uniquely identified by **Address**

One **StaffAddress** Identified by **Address** Must have one **StaffDetails** Identified by **EmployeeNo**

Each **Subscriber Details** Is uniquely identified by **UserName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **FirstName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **LastName**

One **StaffDetails** Identified by **EmployeeNo** Must have one **PhoneNo**

One **StaffDetails** Identified by **EmployeeNo** Must have one **DOB**

One **StaffDetails** Identified by **EmployeeNo** Must have one **Address**

One **StaffDetails** Identified by **EmployeeNo** Must have one **StaffAddress** Identified by **Address**

#### SalesStaff/Stall

Each **Sales Staff** Is uniquely identified by **EmployeeNo**

One **SalesStaff** Identified by **EmployeeNo** May have one **Discount**

One **SalesStaff** Identified by **EmployeeNo** Must have one **StallNo**

One **SalesStaff** Identified by **EmployeeNo** Must work at one **Stall** Identified by **StallNo**

Each **Stall** Is uniquely identified by **StallNo**

One **Stall** Identified by **StallNo** Must have one **PhoneNo**

One **Stall** Identified by **StallNo** Must have one **Address**

One **Stall** Identified by **StallNo** Must have one or more **SalesStaff** Identified by **EmployeeNo**

#### Stall/StallAddress

Each **Stall** Is uniquely identified by **StallNo**

One **Stall** Identified by **StallNo** Must have one **PhoneNo**

One **Stall** Identified by **StallNo** Must have one **Address**

One **Stall** Identified by **StallNo** Must have one **StallAddress** Identified by **Address**

Each **Stall Address** Is uniquely identified by **Address**

One **StallAddress** Identified by **Address** Must have one **Stall** Identified by **StallNo**

#### DrovingZone/DrovingSize

Each **Droving Zone** Is uniquely identified by **Region**

One **DrovingZone** Identified by **Region** Must have one **RegionType**

One **DrovingZone** Identified by **Region** May have one **ContractID**

One **DrovingZone** Identified by **Region** Must have one **DrovingSize** Identified by **Region** and **RegionSizeID**

Each **DrovingSize** Is uniquely identified by **Region and RegionSizeID**

One **DrovingSize** Identified by **Region** and **RegionSizeID** Must have one **DrovingZone** Identified by **Region**

#### DrovingSize/DrovingSizeDetails

Each **DrovingSize** Is uniquely identified by **Region and RegionSizeID**

One **DrovingSize** Identified by **Region** and **RegionSizeID** Must have one **DrovingSizeDetails** Identified by **RegionSizeID**

Each **Droving Size Details** Is uniquely identified by **Region Size ID**

One **DrovingSizeDetails** Identified by **RegionSizeID** May have one **RegionSize**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **Latitude**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **Longitude**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **MaximumAltitude**

One **DrovingSizeDetails** Identified by **RegionSizeID** Must have one **MinimumAltitude**

One **DrovingSizeDetails** Identified by **Region** Must have one **DrovingSize** Identified by **Region** and **RegionSizeID**

## Conclusion

This Milestone has been a long one, but I can understand why. When I think about it the logical stage is probably the most important stage of the data modelling process as it sorts out how everything relates to each other and allows you to organize data into more manageable chunks.

Because I did not have time to add in EER into my Milestone one I was inclined to do it here. Now that I have done it I understand why it’s so helpful and important that this step is made when necessary in the conceptual stage.

On another note I did not realise that I did not have to go all the way up to 4th normal form instead of just 3rd normal form. But I can say that doing so I think gave me a deeper understanding of what each step really is and what I had to look out for in my own conceptual model.

With everything I have done I am happy with the result and I am in high hopes that there will not be any more necessary changes.

## References

Hibernate One To Many XML Mapping Tutorial example. (n.d.). Retrieved May 30, 2018, from https://viralpatel.net/blogs/hibernate-one-to-many-xml-mapping-tutorial/

Lambert, B. (n.d.). Resolve your Many-to-Manys for Accurate Requirements. Retrieved May 30, 2018, from http://www.captechconsulting.com/blogs/resolve-your-many-to-manys-for-accurate-requirements

Pinterest. (n.d.). Retrieved May 30, 2018, from https://www.pinterest.nz/pin/590956782325902944/

Stewart, J. (n.d.). Crow’s Feet Are Best. Retrieved May 31, 2018, from http://tdan.com/crows-feet-are-best/7474

# Milestone 3

## Executive summery

This milestone includes a set of transactional analysis diagrams for each query requested. Along with a list of secondary indexes and why they were chosen. A discussion on what controlled redundancy is and when it should be used. A set of tables showing the estimated size of the database.

A discussion on what security could be used as well as what views are and how they would be used when security is concerned. Talking about monitoring the database once deployed and what I would monitor if it were to be deployed. A new and updated physical diagram of the database.

A list of all the query’s requested and how they work. Lastly, a list of problems I had and how I resolved them.

## Introduction

The purpose of this document is to take the information from the previous section (Milestone 2) and create a physical diagram out of it along with creating a real working database within SQL Server with querys.

This is done by using SQL Server by creating tables along with test data, so I am able to run all nine querys that have been

## SQL Statements

My SQL statements come in **three different .SQL files**. The 1st file will create the database along with all the procedures. The second file will run the querys that create the database and lastly a file will run each query with data ready to be handed in.

Create Statements: **Start from Line 81**

Implementation of foreign keys: **Start From line 302**

Insert Statements: **Start From line 380**

Constraints: **Start From line 1993**

Indexes: **Start From line 2338**

Querys: **Start from line 2515**

## Analyse transactions

### Query A

A sales person subscribes a new standard subscription to a BTDrone. The transaction receives the sales person Id, a discount %, all subscriber details, and a BTDrone ID



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | Sales Staff |  |  | 1 |  |
|  |  | EmployeeNo | R(E) |  |  |
|  |  | Discount | R |  |  |
| SalesStaff | Subscription |  |  | 1 |  |
|  |  | EmployeeNo | I(E)\* |  |  |
|  |  | SubscriptionType | I |  |  |
|  |  | Cost | I |  |  |
|  |  | ReceiptNumber | I |  |  |
| SubscriptionType | UserSubscription |  |  | 1 |  |
|  |  | ReceiptNumber | I(E) |  |  |
|  |  | UserName | I |  |  |
|  |  |  |  |  |  |
| UserSubscription | UserAccount |  |  | 1 |  |
|  |  | UserName | I(E) |  |  |
|  |  | Email | I |  |  |
|  |  | PhoneNo | I |  |  |
|  |  | DroneID | I |  |  |
| UserAccount | SubscriberDetails |  |  | 1 |  |
|  |  | UserName | I(E) |  |  |
|  |  | FirstName | I |  |  |
|  |  | LastName | I |  |  |

### Query B

For each sales person list the subscribers they have sold a subscription to. The transaction receives the sales person's name as input, and presents each subscriber's name, address, and the % they were discounted



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | Staff |  |  | 2500 | Estimate of number of staff |
|  |  | EmployeeNo | R |  |  |
|  |  | FirstName | R(E) |  |  |
|  |  | LastName | R(E) |  |  |
| Staff | SalesStaff |  |  | 2000 |  |
|  |  | EmployeeNo | R(E) |  |  |
|  |  | Discount | R |  |  |
| SalesStaff | Subscriptions |  |  | 502,000 |  |
|  |  | EmployeeNo | R(E)\* |  |  |
|  |  | SubscriptionType | R |  |  |
|  |  | ReceiptNumber | R |  |  |
| SubscriptionType | UserSubscription |  |  | 502,000 |  |
|  |  | ReceiptNumber | R(E) |  |  |
|  |  | UserName | R |  |  |
| UserSubscription | UserAccount |  |  | 502,000 |  |
|  |  | UserName | R(E) |  |  |
|  |  | Email | R |  |  |
|  |  | PhoneNo | R |  |  |
| UserAccount | SubscriberDetails |  |  | 502,000 |  |
|  |  | UserName | R(E) |  |  |
|  |  | FirstName | R |  |  |
|  |  | LastName | R |  |  |
|  |  | LetterBoxNumber | R |  |  |
|  |  | Street | R |  |  |
|  |  | Town | R |  |  |
|  |  | City | R |  |  |
|  |  | PostCode | R |  |  |
|  |  |  |  |  |  |

### Query C

List the location in latitude, longitude coordinates, of each BTDrone that is currently in a contract. The transaction presents the Contracting organisation's name, a BTDrone ID, a Latitude, and a Longitude



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | Contract |  |  | 2000 |  |
|  |  | ContractID | R(E) |  |  |
|  |  | Company | R |  |  |
| Contract | BTDrone |  |  | 50,000 |  |
|  |  | ContractID | R(E)\* |  |  |
|  |  | DroneID | R |  |  |
| BTDrone | EnvironmentalData |  |  | 2000 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | Longitude | R |  |  |
|  |  | Latitude | R |  |  |

### Query D

For a contract list all the data collected. The transaction receives the contracting organisation's name and presents for each collected data record, the contracting organisation's name, a BTDrone ID, Temperature, Humidity and Ambient light strength



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | Contract |  |  | 1 |  |
|  |  | ContractID | R |  |  |
|  |  | Company | R(E) |  |  |
| Contract | BTDrone |  |  | 1 |  |
|  |  | ContractID | R(E)\* |  |  |
|  |  | DroneID | R |  |  |
| BTDrone | EnvironmentalData |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | Temperature | R |  |  |
|  |  | Humidity | R |  |  |
|  |  | AmbientLightStrength | R |  |  |

### Query E

For each BTDrone present the list of subscribers who are viewing a live 3D video stream. The transaction lists BTDroneID, Subscriber Name, Stream ID.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | CameraViews |  |  | 100 |  |
|  |  | CameraView | R(E) |  |  |
|  |  | DroneID | R |  |  |
|  |  | UserName | R |  |  |
| CameraViews | BTDrone |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | VideoStreamID | R |  |  |

### Query F

For a given BTDrone list all the suppliers of parts. The transaction receives the BTDrone ID, and presents the Supplier Name and, Part Name



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | BTDrone |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
| BTDrone | MaintenanceLog |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
| MaintenanceLog | Part |  |  | 100 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | PartID | R |  |  |
|  |  | PartName | R |  |  |
| Part | PartSupplier |  |  | 50 |  |
|  |  | PartID | R(E)\* |  |  |
|  |  | CompanyName | R |  |  |
| PartSupplier | Supplier |  |  |  |  |
|  |  | CompanyName | R(E) |  |  |

### Query G

Update the location and region of a BTDrone. The transaction receives the BTDrone ID, a location and a region expressed as a list of coordinates in latitude, longitude pairs. It updates the location of the BTDrone and its corresponding region. (This transaction may require more than one update query.)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | EnvironmentalData |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | Latitude | U |  |  |
|  |  | Longitude | U |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | B |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | BTDrone |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | Region | U |  |  |
| BTDrone | DrovingZone |  |  | 1 |  |
|  |  | Region | U(E) |  |  |
|  |  | RegionType | U |  |  |

### Query H

Delete the data collected for a given Contract. The transaction receives a Contract ID, the data collected for a Contract is deleted.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  |  |  | When Requested |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | SensedData |  |  | 311,040 | All keys can be found in one table delete can happen with just one table |
|  |  | ContractID | R - D(E)\* |  |  |
|  |  | SensedDataID | D |  |  |
|  |  | DroneID | D |  |  |
|  |  | CompanyName | D |  |  |
|  |  | Altitude | D |  |  |
|  |  | Latitude | D |  |  |
|  |  | Longitude | D |  |  |
|  |  | Humidity | D |  |  |
|  |  | AmbientLightStrength | D |  |  |

### Query I

Insert data from a BTDrone to its stored data. The transaction receives the BTDrone ID.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Transaction | A |  | EXAMPLE |  |  |
| Peak (avg) | Day | Time | Number of runs per hour. |  |  |
|  | 24/7 | Every 10 seconds | 360 |  |  |
| From Relation | To Relation | Attributes | Access | Number of Times Accessed | Notes |
| - | EnvironmentalData |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
|  |  | Longitude | R |  |  |
|  |  | Latitude | R |  |  |
|  |  | Altitude | R |  |  |
|  |  | Temperature | R |  |  |
|  |  | Humidity | R |  |  |
|  |  | AmbientLightStrength | R |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| EnvironmentalData | BTDrone |  |  | 1 |  |
|  |  | DroneID | R(E) |  |  |
| BTDrone | SensedData |  |  | 1 |  |
|  |  | DroneID | I(E) |  |  |
|  |  | SensedDataID | I |  |  |
|  |  | Longitude | I |  |  |
|  |  | Latitude | I |  |  |
|  |  | Altitude | I |  |  |
|  |  | Temperature | I |  |  |
|  |  | Humidity | I |  |  |
|  |  | AmbientLightStrength | I |  |  |
|  |  | ContractID | I |  | If Drone is a part of a contract |
|  |  | Company | I |  | If Drone is a part of a contract |

## Indexes/Secondary

Here is a list of all secondary indexes that I have created and why I decided that is was necessary to index them.

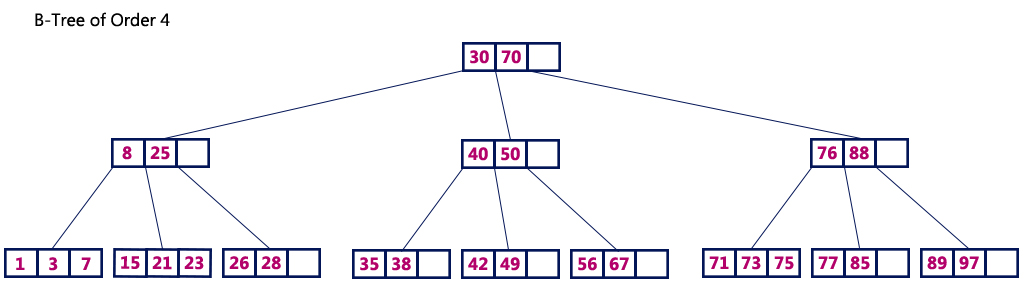
* First Name
  + The first name columns though most of my database have been indexed because it is a term that would most commonly be used to search something up related to a person. This could be a customer and their details or an employee and their details.
* LastName
  + The last name columns though most of my database have been indexed because it is a term that would most commonly be used to search something up related to a person. This along with the first name could be a customer and their details or an employee and their details.
* Town
  + The town columns though most of the database have been indexed because it could be used to show all businesses or even employees what town they are located in.
* City
  + The City columns though most of the database have been indexed because it could be used to show all businesses or even employees what City they are located in.
* PostCode
  + The PostCode columns though most of the database have been indexed because it could be used show all businesses or even employees that are located in an entire Region.
* Part Name
  + The Part Name column in my Parts table is indexed because I feel that it would be a lot easier to search up parts this way rather than using the part ID itself.
* Region
  + The Region column in my Droving Zone table is indexed because it will allow you to search up all drones within a given region which would make things easier to find.
* Region Type
  + The Region Type column in my Droving Zone table is indexed because it will allow you to look up all drones that a found within a certain region type E.g. Snow or Forest. It will make looking up locations of drones a lot easier.
* PhoneNo
  + The PhoneNo columns though most of the database have been indexed because it will allow for a quick search up option. What was going though my head was that most businesses if you call up or something may ask and use your phone number to find your personal info so that is why I feel like it is important.

## File Organizations

File organisations are the format as in which you store all of your data within your database. There are six types of file organisations I could find.

* Sequential File Organizations
* Heap File Organizations
* Hash/Direct File Organisations
* Indexed Sequential Access Method(ISAM)
* B+ Tree File Organizations
* Cluster File Organizations

The Organization that I feel I would use would be a B tree and I will explain in more detail what a b tree is down below.

(“B - Trees :: Data Structures,” n.d.)

The diagram above is a visual representation of a b tree and what it looks like. You can think of this like an upside town tree the top table being the trunk or roots the second tables being the branches and the third tables being the leaf’s.

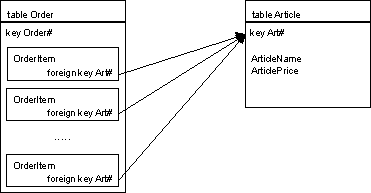
The top table would be the attribute or key that has been indexed and will allow you to search on that particular attribute. It then searches using a true false sort of system to allow it to navigate though each stage of the tree.

Once it has reaches the bottom of the tree it will pull that data out and display it on screen. This makes searching a lot faster due to the query not having to read all data within the table

## Controlled Redundancy

Controlled Redundancy is a technique to use redundant fields in a physical database in order to speed up reading database access (“Controlled Redundancy,” n.d.)

Controlled redundancy is seen when there is a single piece of data held in two sperate tables. This can be used to help speed up reading database access yes, but it can also be used to store backups of data as well. For example, to keep a back up of an order that has been made just in case something happens to the original.



(“Controlled Redundancy,” n.d.)

### Consequences

* Time: You can considerably reduce the number of database pages accessed for read operations by using Controlled Redundancy. In case of write operations to the parent entity (Article) you will need additional database accesses as you have to update more than one table to write the same (redundant) fact. (“Controlled Redundancy,” n.d.)
* Space: You need more database space for redundant data. (“Controlled Redundancy,” n.d.)
* Code complexity: Controlled Redundancy results in more complex code as you pull application level considerations down to the database level. A database user must know that he or she also has to update the Order table if the name of an article changes. In any case Controlled Redundancy should be shielded and controlled by a Physical Access Layer. Never clutter code that accesses such data into the application kernel. (“Controlled Redundancy,” n.d.)
* Queries: Redundancy does not impair the understandability of a physical data model. It might even improve understandability. (“Controlled Redundancy,” n.d.)

### Implementation

You should only use this if you have data that is fixed and will not change when new entries are made. This could be someone’s name/Company name or a price od a product if you know it wont change.

## Estimate Disk Space Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| ContracteeAddress | ContracteeCompany | varchar | 30 | 32 |
|  | LetterBoxNumber | int |  | 4 |
|  | Street | char | 30 | 30 |
|  | Town | char | 30 | 30 |
|  | City | char | 30 | 30 |
|  | PostCode | int |  | 4 |
|  |  |  | Total | 130 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| Contractees | Company | varchar | 30 | 32 |
|  | FirstName | varchar | 30 | 32 |
|  | LastName | varchar | 30 | 32 |
|  | PhoneNo | int |  | 4 |
|  |  |  | Total | 100 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| Contract | ContractID | int |  | 4 |
|  | ContracteesCompany | varchar | 30 | 32 |
|  | AdminStaffStaffEmployeeNo | int |  | 4 |
|  |  |  | Total | 40 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| AdminStaff | StaffEmployeeNo | int |  | 4 |
|  | Discount | varchar | 3 | 5 |
|  |  |  | Total | 9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| Staff | EmployeeNo | int |  | 4 |
|  | FirstName | varchar | 30 | 32 |
|  | LastName | varchar | 30 | 32 |
|  | PhoneNo | int |  | 4 |
|  | DOB | datetime |  | 8 |
|  |  |  |  | 80 |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| StaffAddress | StaffEmployeeNo | int |  | 4 |
|  | LetterBoxNumber | int |  | 4 |
|  | Street | char | 30 | 30 |
|  | Town | char | 30 | 30 |
|  | City | char | 30 | 30 |
|  | PostCode | int |  | 4 |
|  |  |  | Total | 102 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| SalesStaff | StaffEmployeeNo | int |  | 4 |
|  | Discount | varchar | 3 | 5 |
|  | StallStallNo | int |  | 4 |
|  |  |  | Total | 13 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| Stall | StallNo | int |  | 4 |
|  | PhoneNo | int |  | 4 |
|  |  |  | Total | 8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| StallAddress | StallStallNo | int |  | 4 |
|  | LetterBoxNumber | int |  | 4 |
|  | Street | char | 30 | 30 |
|  | Town | char | 30 | 30 |
|  | City | char | 30 | 30 |
|  | PostCode | int |  | 4 |
|  |  |  | Total | 102 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Relation Name | | Attribute | | Data Type | | Length | | Total Bytes Per Row | |
| ContractSubscriptionType | | ContactContractID | | int | |  | | 4 | |
|  | | ContractContracteesCompany | | varchar | | 30 | | 32 | |
|  | | SubscriptionsReciptNumber | | int | |  | | 4 | |
|  | |  | |  | | Total | | 40 | |
| Relation Name | Attribute | | Data Type | | Length | | Total Bytes Per Row | |
| Subscriptions | ReceiptNumber | | int | |  | | 4 | |
|  | SubscriptionNumber | | varchar | | 30 | | 32 | |
|  | Cost | | money | |  | | 8 | |
|  | BTDroneDroneID | | int | |  | | 4 | |
|  | CameraViewsCameraViews | | int | |  | | 4 | |
|  | CameraViewsCameraViews2 | | int | |  | | 4 | |
|  | SalesStaffStaffEmployeeNo | | int | |  | | 4 | |
|  |  | |  | | Total | | 60 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| UserSubscription | UserAccountUserName | varchar | 30 | 32 |
|  | SubscriptionsReciptNumber | int |  | 4 |
|  |  |  | Total | 36 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| UserAccount | UserName | varchar | 30 | 32 |
|  | Email | varchar | 50 | 52 |
|  | PhoneNo | int |  | 4 |
|  | BTDroneDroneID | int |  | 4 |
|  |  |  | Total | 92 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| SubscriptionDetails | UserAccountUserName | varchar | 30 | 32 |
|  | FirstName | varchar | 30 | 32 |
|  | LastName | varchar | 30 | 32 |
|  | LetterBoxNo | int |  | 4 |
|  | Street | char | 30 | 30 |
|  | Town | char | 30 | 30 |
|  | City | char | 30 | 30 |
|  | PostCode | int |  | 4 |
|  |  |  | Total | 194 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| CameraViews | CameraView | int |  | 4 |
|  | BTDroneDroneID | int |  | 4 |
|  | UserAccountUserName | varchar | 30 | 32 |
|  |  |  | Total | 40 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| BTDrone | DroneID | int |  | 4 |
|  | StreamingDetailsVideoStreamID | int |  | 4 |
|  | ContractContractID | int |  | 4 |
|  | ContractContracteesCompany | varchar | 30 | 32 |
|  | DrovingZoneRegionID | int |  | 4 |
|  |  |  | Total | 48 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| DrovingZone | RegionID | int |  | 4 |
|  | Region | varchar | 50 | 52 |
|  | RegionType | varchar | 50 | 52 |
|  | ContractContractID | int |  | 4 |
|  | ContractContracteeCompnay | varchar | 30 | 32 |
|  |  |  | Total | 144 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| DrovingSize | DrovingSizeDetailsRegionSizeID | int |  | 4 |
|  | DrovingZoneRegionID | int |  | 4 |
|  |  |  | Total | 8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| DrovingSizeDetails | RegionSizeID | int |  | 4 |
|  | Latitude | int |  | 4 |
|  | Longitude | int |  | 4 |
|  | MaximumAltitude | int |  | 4 |
|  | MinimumAltitude | int |  | 4 |
|  |  |  | Total | 20 |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| MaximumRoamingArea | BTDroneDroneID | int |  | 4 |
|  | MaximumAltitude | int |  | 4 |
|  | MaximumLatitude | int |  | 4 |
|  | MaximumLongitude | int |  | 4 |
|  |  |  | Total | 16 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| SensedData | SensedDataID | int |  | 4 |
|  | Altitude | int |  | 4 |
|  | Latitude | int |  | 4 |
|  | Longitude | int |  | 4 |
|  | Temperature | Decimal | 5,2 | 5 |
|  | Humidity | int |  | 4 |
|  | AmbientLightStrength | int |  | 4 |
|  | BTDroneDroneID | int |  | 4 |
|  | ContractContractID | int |  | 4 |
|  | ContractContracteesCompany | varchar | 30 | 32 |
|  |  |  | Total | 69 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| StreamingDetails | VideoStreamID | int |  | 4 |
|  | Time | datetime |  | 8 |
|  |  |  | Total | 12 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| EnviromentData | BTDroneDroneID | int |  | 4 |
|  | Longitude | int |  | 4 |
|  | Latitude | int |  | 4 |
|  | Altitude | int |  | 4 |
|  | Temperature | decimal | 5,2 | 5 |
|  | Humidity | int |  | 4 |
|  | AmbientLightStrength | int |  | 4 |
|  |  |  | Total | 29 |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| MaintenanceLog | BTDroneDroneID | int | 4 | 4 |
|  | Status | varchar | 30 | 32 |
|  | Date | datetime |  | 5 |
|  | Comments | varchar | 250 | 252 |
|  |  |  | Total | 296 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| Suppliers | CompanyName | varchar | 30 | 32 |
|  | PhoneNo | int |  | 4 |
|  | Email | varchar | 50 | 52 |
|  |  |  | Total | 88 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| Part | PartID | int |  | 4 |
|  | PartName | varchar | 50 | 52 |
|  | PartStatus | varchar | 50 | 52 |
|  | MaintenaceLogBTDroneDroneID | int |  | 4 |
|  |  |  | Total | 112 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| PartSupplier | PartPartID | int |  | 4 |
|  | SupliersCompanyName | varchar | 30 | 32 |
|  |  |  | Total | 36 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data Type | Length | Total Bytes Per Row |
| SupplierAddress | SuppliersCompanyName | varchar | 30 | 32 |
|  | LetterBoxNumber | int |  | 4 |
|  | Street | char | 30 | 30 |
|  | Town | char | 30 | 30 |
|  | City | char | 30 | 30 |
|  | PostCode | int |  | 4 |
|  |  |  | Total | 130 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Relation Name** | **Total Per Row** | **Max Size** | **Total Relation Size** | **Pages Per Relation** | **New Relation Size** |
| ContracteeAddress | 130 | 5,000 | 650000 | 83 | 664000 |
| Contractees | 100 | 5,000 | 500000 | 64 | 512000 |
| Contract | 40 | 5,000 | 200000 | 26 | 208000 |
| AdminStaff | 9 | 2,000 | 18000 | 3 | 24000 |
| Staff | 80 | 10,000 | 800000 | 102 | 816000 |
| StaffAddress | 102 | 10,000 | 1020000 | 130 | 1040000 |
| SalesStaff | 13 | 8,000 | 104000 | 14 | 112000 |
| Stall | 8 | 1,000 | 8000 | 2 | 16000 |
| StallAddress | 102 | 1,000 | 102000 | 13 | 104000 |
| ContractSubscriptionType | 40 | 5,000 | 200000 | 26 | 208000 |
| Subscriptions | 60 | 105,000 | 6300000 | 798 | 6384000 |
| UserSubscription | 36 | 100,000 | 3600000 | 456 | 3648000 |
| UserAccount | 92 | 100,000 | 9200000 | 1164 | 9312000 |
| SubscriptionDetails | 194 | 100,000 | 19400000 | 2455 | 19640000 |
| CameraViews | 40 | 10,000,000 | 400000000 | 50608 | 404864000 |
| BTDrone | 48 | 100,000 | 4800000 | 608 | 4864000 |
| DrovingZone | 144 | 100,000 | 14400000 | 1822 | 14576000 |
| DrovingSize | 8 | 100,000 | 800000 | 102 | 816000 |
| DrovingSizeDetails | 20 | 100,000 | 2000000 | 254 | 2032000 |
| SensedData | 69 | 15,552,000,000 | 1073088000000 | 135765183 | 1086121464000 |
| MaximumRoamingArea | 16 | 100,000 | 1600000 | 203 | 1624000 |
| StreamingDetails | 12 | 98,000 | 1176000 | 149 | 1192000 |
| EnviromentData | 29 | 100,000 | 2900000 | 367 | 2936000 |
| MaintenanceLog | 296 | 100,000 | 29600000 | 3745 | 29960000 |
| Part | 112 | 10,000,000 | 1120000000 | 141701 | 1133608000 |
| PartSupplier | 36 | 10,000,000 | 360000000 | 45547 | 364376000 |
| Suppliers | 88 | 1,000 | 88000 | 12 | 96000 |
| SupplierAddress | 130 | 1,000 | 130000 | 17 | 136000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Total Database Size | | | |
| Total Bytes | Total Kilobytes | Total Megabytes | Total Gigabytes |
| 1088125096000 | 1062622164 | 1037716.96 | 1013.4 |

## Security Mechanisms

### Design User Views

Views within a database are temporary tables created to display data from within the database. This is good for security because the database administrator can decide what each person can see when it comes to data. This will be good for when there is information stored about your customer that s meant to be kept private and you can set it up, so the sales staff/admin staff cannot see this information.

### Design Access Rules

The access rules are rules setup within the database to determine if a request is approved or denied. As explained above this could be rules for views, who can see what data within a database and assigning that view to a group of users. This can also include rules like is able to update or is not able to update data within the system.

Below is a list of access rules a believe would be put in place for this current database.

* Standard subscribers can only view one camera view
* Gold subscribers can view any camera view and is also able to move the camera
* Platinum subscribers can control the drone’s movement
* User Accounts standard – platinum cannot see Drones Apart of a contract
* Admin Staff are the only people with the option to create a contract
* Sales staff cannot update discounts
* Contractees can only View Data collected within the Sensed Data Table.
* Sales Staff can create new subscriptions

These are just some rules that I could come up with but I’m sure that there would be a lot more if this was a database being used by clients and staff.

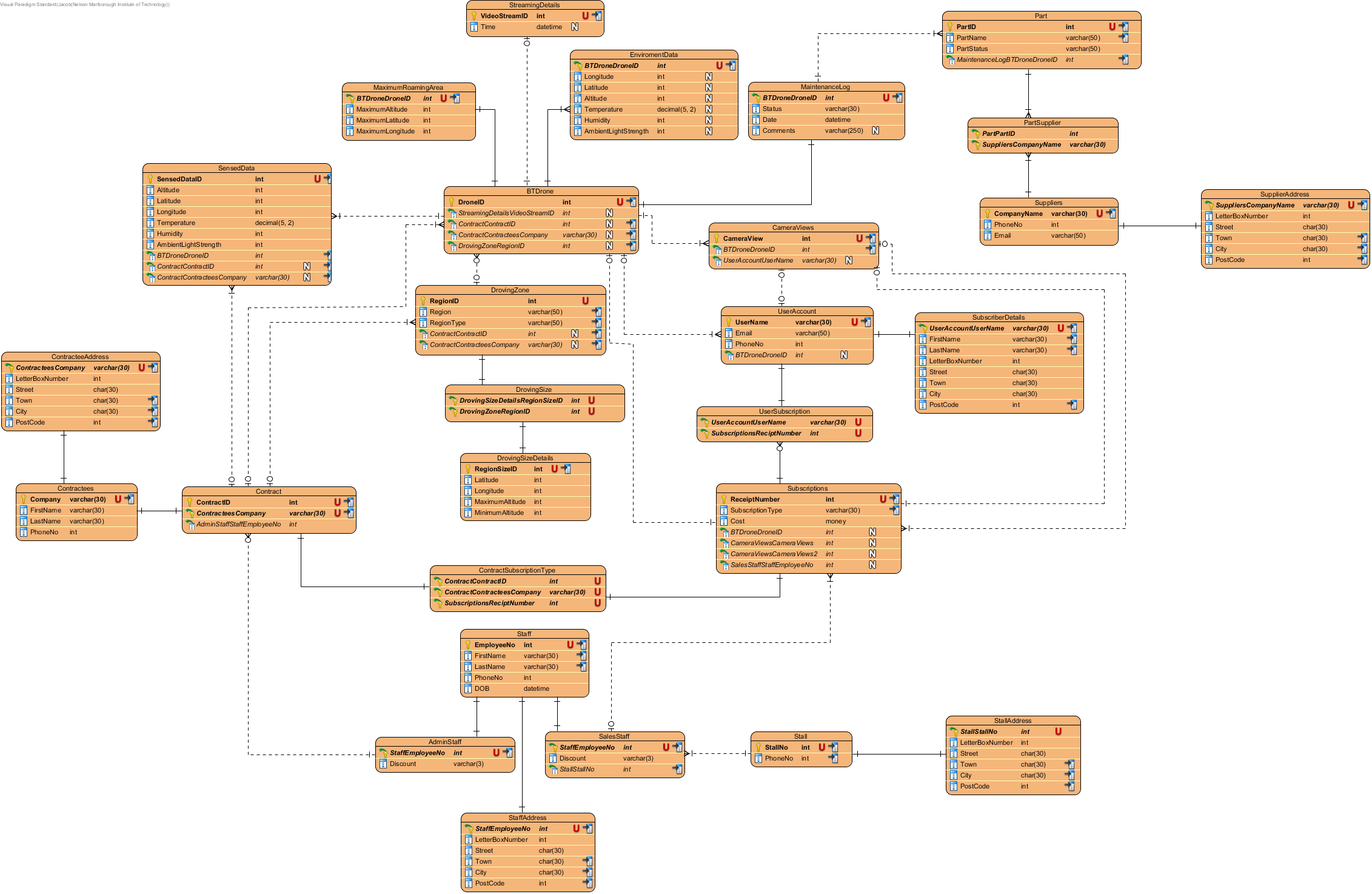
## Monitor and Tune the Operational System

The objective of this step is to monitor the database after it has been deployed and to improve on it if the need arises.

Below I will state some areas within my database I feel that would need to be monitored if it were to be deployed as it is now.

* The relationships between Subscriptions and camera views to make sure that each relationship is working correctly and is giving each subscription level the correct benefits.
* Watching the sensed data table to see how long the data should be held for before deletion right now it’s every year but that may be too long
* Watch to make sure my check constraint in camera views works properly as I am not too sure currently.
* Monitor to see if there are any more constraints that are needed to be added.

## Physical Diagram



# SQL Queries Purpose

The query code has been placed here as well so it would be easier to read my description along with the code above it. These procedures are still automatically created within the .SQL file I have handed in

## Query A

Create Procedure Create\_New\_Subscriber

@prSalesID int,

@prUserName varchar (30),

@prEmail varchar (50),

@prPhoneNo int,

@prFirstName varchar (30),

@prLastName varchar (30),

@prLetterBoxNo int,

@prStreet char (30),

@prTown char (30),

@prCity char (30),

@prPostCode int,

@prDroneID int

As

Begin

Insert Into dbo.Subscriptions (SubscriptionType,Cost,BTDroneDroneID,CameraViewsCameraViews,CameraViewsCameraViews2,SalesStaffStaffEmployeeNo)

Values ('Standard',$25.00,null,2,1,@prSalesID);

Insert Into dbo.UserAccount(UserName,Email,PhoneNo,BTDroneDroneID)

Values (@prUserName,@prEmail,@prPhoneNo,@prDroneID);

Insert Into dbo.UserSubscription(UserAccountUserName,SubscriptionsReciptNumber)

Values (@prUserName,11);

Insert Into dbo.SubscriberDetails(UserAccountUserName,FirstName,LastName,LetterBoxNumber,Street,Town,City,PostCode)

Values (@prUserName,@prFirstName,@prLastName,@prLetterBoxNo,@prStreet,@prTown,@prCity,@prPostCode);

End

Go

In this query we are creating a new subscriber and adding them to the relevant tables. I am passing in all the relevant data for a new user from address to user name. I then insert the data into the correct tables using the variables I have handed in.

## Query B

Create Procedure Sold\_Subscriptions

@prFirstName varchar(30),

@prLastName varchar(30)

As

Begin

SELECT SalesStaff.Discount, SubscriberDetails.FirstName, SubscriberDetails.LastName, SubscriberDetails.LetterBoxNumber, SubscriberDetails.Street, SubscriberDetails.Town, SubscriberDetails.City, SubscriberDetails.PostCode

FROM Staff INNER JOIN

SalesStaff ON Staff.EmployeeNo = SalesStaff.StaffEmployeeNo INNER JOIN

Subscriptions ON SalesStaff.StaffEmployeeNo = Subscriptions.SalesStaffStaffEmployeeNo INNER JOIN

UserSubscription ON Subscriptions.ReceiptNumber = UserSubscription.SubscriptionsReciptNumber INNER JOIN

UserAccount ON UserSubscription.UserAccountUserName = UserAccount.UserName INNER JOIN

SubscriberDetails ON UserAccount.UserName = SubscriberDetails.UserAccountUserName

Where Staff.FirstName = @prFirstName And Staff.LastName = @prLastName

End

Go

In this query we are listing all the subscriptions sold by **One** Staff member. Here I am passing in the first and last name of the sales staff and searching for it using that. The rest of the statement you see is me joining all the forging keys together though out each table until all the data I have selected at the beginning of the statement is available to select display on screen.

## Query C

Create Procedure Contract\_Drone\_Locations

As

Begin

SELECT Contract.ContracteesCompany AS Company, BTDrone.DroneID, EnviromentData.Longitude, EnviromentData.Latitude

FROM Contract INNER JOIN

BTDrone ON Contract.ContractID = BTDrone.ContractContractID AND Contract.ContracteesCompany = BTDrone.ContractContracteesCompany INNER JOIN

EnviromentData ON BTDrone.DroneID = EnviromentData.BTDroneDroneID

End

Go

This next query is where we are asked to list all the locations of each drone that is apart of a contract. We were not required to pass any variables in this time as it is just a list of all drones. Here I am selected the company’s name drone id latitude and longitude and displaying them though all the joins created. I also added on an alias onto contracteecompany to make it easier to read when displayed on screen.

## Query D

Create Procedure Contract\_Drone\_Data

@prCompany varchar(30)

As

Begin

SELECT Contract.ContracteesCompany AS Company, BTDrone.DroneID, EnviromentData.Temperature, EnviromentData.Humidity, EnviromentData.AmbientLightStrength

FROM Contract INNER JOIN

BTDrone ON Contract.ContractID = BTDrone.ContractContractID AND Contract.ContracteesCompany = BTDrone.ContractContracteesCompany INNER JOIN

EnviromentData ON BTDrone.DroneID = EnviromentData.BTDroneDroneID

WHERE (Contract.ContracteesCompany = @prCompany)

ORDER BY Company, BTDrone.DroneID, EnviromentData.Temperature, EnviromentData.Humidity, EnviromentData.AmbientLightStrength

End

Go

This query lists all the data collected for **One** contract this is found my passing in the company name. This statement selects all the data the required and then joins all the tables that contain the selected data. Then in the Where clause it only selected the data where the Company name = the variable I passed in. I also tried to add in an order by to try make it a bit more readable otherwise it would have displayed on screen a bit differently.

## Query E

Create Procedure Drone\_Viewers

As

Begin

SELECT BTDrone.DroneID, BTDrone.StreamingDetailsVideoStreamID AS StreamID, CameraViews.UserAccountUserName AS UserName

FROM BTDrone INNER JOIN

CameraViews ON BTDrone.DroneID = CameraViews.BTDroneDroneID

ORDER BY BTDrone.DroneID, UserName, StreamID

End

Go

This Query creates a list of subscribers that are viewing stream. It shows the User name of the subscriber what stream they are watching and what drone they are watching. This is a simple statement that is just selecting all users that are watching a stream and orders them my DroneID Username and stream ID.

## Query F

Create Procedure Drone\_Parts

@prDroneID int

As

Begin

SELECT BTDrone.DroneID, Part.PartName, Suppliers.CompanyName

FROM BTDrone INNER JOIN

MaintenanceLog ON BTDrone.DroneID = MaintenanceLog.BTDroneDroneID INNER JOIN

Part ON MaintenanceLog.BTDroneDroneID = Part.MaintenanceLogBTDroneDroneID INNER JOIN

PartSupplier ON Part.PartID = PartSupplier.PartPartID INNER JOIN

Suppliers ON PartSupplier.SuppliersCompanyName = Suppliers.CompanyName

WHERE (BTDrone.DroneID = @prDroneID)

End

Go

This query list all the parts that is supplied for **One** Drone. You must hand in the drone ID and it brings up the part name and the suppliers company name. I select all the fields required and connect them though the required foreign keys. I then only display the parts for where the DroneID = the ID I Passed it so it only brings up the information for one drone.

## Query G

Create Procedure Update\_Location\_And\_Reagion\_Of\_Drone

@prDroneID int,

@prLongitude int,

@prLatitude int,

@prRegion varchar (50),

@prRegionType varchar (50)

As

Begin

DECLARE @RegionID int

SELECT @RegionID = DrovingZoneRegionID FROM BTDrone Where DroneID = @prDroneID

Update DrovingZone

SET Region = @prRegion , RegionType = @prRegionType

Where RegionID = @RegionID;

Update EnviromentData

SET Latitude = @prLatitude , Longitude = @prLongitude

Where BTDroneDroneID = @prDroneID;

End

Go

This statement updates the locations and region of a Drone. It first gets passed in all the new data and the drone ID of the drone that needs to be updated. I then create a variable that will grab the region ID from my BTDrone table where the Drone ID = the DroneID passed in. I did this because drone ID is not a foreign key within my droving zone table. I can then update and set all the columns in my droving zone table where the ID = the variable I created in this statement. The environment data update was a lot easier because a drone foreign key already existed so all I had to do was update the longitude and latitude of where the Drone ID = the DroneID passed in.

## Query H

Create Procedure Delete\_Contract\_Saved\_Data

@prContractID int

As

Begin

DELETE FROM SensedData

Where ContractContractID = @prContractID

End

This statement deletes data from the sensed data table for a given contract. You start by passing in the ContractID and from there you just have to delete from sensed data where the foreign key ContractContractID = the ID I passed in

## Query I

Create Procedure Store\_Data

@prDroneID int

As

Begin

DECLARE @Longitude int

DECLARE @Latitude int

DECLARE @Altitude int

DECLARE @Temperature decimal(5,2)

DECLARE @Humidity int

DECLARE @AmbientLightStrength int

SELECT @Longitude = Longitude FROM EnviromentData Where BTDroneDroneID = @prDroneID

SELECT @Latitude = Latitude FROM EnviromentData Where BTDroneDroneID = @prDroneID

SELECT @Altitude = Altitude FROM EnviromentData Where BTDroneDroneID = @prDroneID

SELECT @Temperature = Temperature FROM EnviromentData Where BTDroneDroneID = @prDroneID

SELECT @Humidity = Humidity FROM EnviromentData Where BTDroneDroneID = @prDroneID

SELECT @AmbientLightStrength = AmbientLightStrength FROM EnviromentData Where BTDroneDroneID = @prDroneID

Insert INTO SensedData (BTDroneDroneID,Altitude,Latitude,Longitude,Temperature,Humidity,AmbientLightStrength)

Values(@prDroneID,@Altitude,@Latitude,@Longitude,@Temperature,@Humidity,@AmbientLightStrength)

End

Go

This statement inserts all data from the Environment data time within the drone to the Sensed Data table to be more permanently stored. Firstly, I must declare a lot of variables to make them = the data stored with the drone, so I can then use them to insert the data into the Sensed data table. As you can see this is just a bunch of select statements that make the variable = the data in the environment table. Then I just have to insert it into the sensed data table along with the drone ID that is also passed in.

## Issues In counted/Resolutions

* The first issue I in counted was where I could not figure out how to drop my database, so I could just run all my queries with a single press of a button. I solved this by not dropping the database itself but only dropping the tables, so everything is deleted. I did this by using your basic drop statements for my tables and when you first run the queries I have an if exists on the database itself, so it will not try creating the database twice and will skip past it once it has been created the first time.
* The next issue I has was the fact that you had to insert data in a specific order once my foreign keys were implemented. I solved this by inserting all my data first and then creating all the foreign key constraints. This saved me considerable time as I did not have to try figure out the order in witch the data has to be inserted.
* While creating my insert statements I ran into a few problems. One was I could not keep my primary key unique over the whole table. This table being Sensed data where at first, I was only using my BTDrone foreign key as it’s primary key. To fix this I had to implement a new primary key called SencedDataID so that each piece of data inserted would be unique in this way. The second problem I has was I would not get a unique key within my Supplier Address table because of the PartID that flowed into it due to the relationship. I then concluded that this must have been a many to many relationship and I had never noticed until trying to create it, so I created a new table called PartSupplier to join the two tables together nicely and with this I was able to keep all data in my Supplier Address table unique because now there was only one primary key that flowed into this table.
* Another problem I had was not to big and was my own fault really was that some of my insert statements did not work because I got the foreign keys mixed up. I just had to spend the time and go back to make sure they lined up with their relationships correctly.
* Another problem I had was never solved really it just seemed to fix itself. Once I arrived at tech to start work again I had a problem where I could not get my tables to drop correctly. I have no idea how or why this was happening, and it held me up for longer than I would like to admit. I then saw that SQL Server had an update available and tried that with no success. Lastly, I tried executing queries separately from the procedure I has created for them and it seemed to work. After that everything just seemed to work again my procedures worked and the over creation of everything worked fine, with a sigh of relief I just started on my database for the day.
* Another problem I had was when I was creating my constraints. The code itself was correct but the names were not. I did not know that you could not have duplicate names when it comes to check statements. I just went though and for every table I added on the name of the table it was on and where there was multiple of these instances I added on the field it was in. This solved this problem right away with all constraints running smoothly.

* Small problem I has with creating query number 6 was that I never has a part name only a PartID and PartStatus. I just had to quickly add a new column into my database and update my insert statements. Also, being it a name of a part I also decided to index it as well.
* Another Problem I has was during query nine where I was trying to insert data into my sensed data table. I kept getting an error saying that it key already existed which was odd because the duplicate key was a foreign primary key, turns out I had set up an index as Unique, so it conflicted with the index I just had to go back and remove the unique part of the index and it now works fine.

## Conclusion

This Milestone has been a tough one due to time constraints, but it has been a lot of fun. I do enjoy the more hands on assessments we get to do like being able to code all this stuff up and creating the database from scratch.

I do feel that I and some others did have a bit of an advantage because I know how to wright procedures and to do what is required and hand in information to get something back it requires a procedure to do so.

Overall this has been a fun assessment and has shown me that if put under pressure I can get anything done if I set my mind to it.

## References

B - Trees :: Data Structures. (n.d.). Retrieved June 23, 2018, from http://btechsmartclass.com/DS/U5\_T3.html

Controlled Redundancy. (n.d.). Retrieved June 25, 2018, from http://www.objectarchitects.de/ObjectArchitects/orpatterns/Performance/ControlledRedundancy/