DAT601

Josh Moss

Table of Contents

[Introduction to data modelling 3](#_Toc168074991)

[Conceptual modelling: 3](#_Toc168074992)

[Logical data modelling 3](#_Toc168074993)

[Physical implementation 3](#_Toc168074994)

[Description of conceptual modelling components (Chen’s ERD notation) 4](#_Toc168074995)

[Entity 4](#_Toc168074996)

[4](#_Toc168074997)

[Weak Entity: 4](#_Toc168074998)

[Associative Entity: 4](#_Toc168074999)

[Attributes: 5](#_Toc168075000)

[Key Attribute: 5](#_Toc168075001)

[Partial Key Attribute: 5](#_Toc168075002)

[5](#_Toc168075003)

[Derived Attribute: 6](#_Toc168075004)

[6](#_Toc168075005)

[Relationship: 7](#_Toc168075006)

[ Unary relationship (Recursive) 7](#_Toc168075007)

[ Binary relationship 7](#_Toc168075008)

[ Ternary relationship 7](#_Toc168075009)

[Weak (identifying) Relationship: 7](#_Toc168075010)

[7](#_Toc168075011)

[Optionality: 8](#_Toc168075012)

[8](#_Toc168075013)

[Cardinality: 8](#_Toc168075014)

[Conceptual Data model 9](#_Toc168075015)

[Assumptions made: 9](#_Toc168075016)

[Conceptual Model Documentation 10](#_Toc168075017)

[Table 1: Document Entities 10](#_Toc168075018)

[Table 2: Document Relationships 12](#_Toc168075019)

[Table 3: Attributes 13](#_Toc168075020)

[Assumptions 18](#_Toc168075021)

[**Chapter Two** 19](#_Toc168075022)

[Logical data model 20](#_Toc168075023)

[Logical model Rationale 21](#_Toc168075024)

[Logical Model Documentation 22](#_Toc168075025)

[Mapping rules 22](#_Toc168075026)

[Documented relations 24](#_Toc168075027)

[Document Attributes 25](#_Toc168075028)

[NaLER Analysis 37](#_Toc168075029)

[Description of NaLER 37](#_Toc168075030)

[ Attributes 38](#_Toc168075031)

[ Relationships 41](#_Toc168075032)

[Normalisation 46](#_Toc168075033)

# Introduction to data modelling

In database design, data modelling is a critical aspect. It involves how data is structured within a database system. It encompasses the process of how data is organized and the relationships between this data. Data modelling is important to ensure data is managed accurately and efficiently. According to the “Principal components of relational model” (Vidhya, V, 2016, p, 2.5), there are three basic components the model must follow:

1. Data structure
2. Data integrity
3. Data manipulation

## Conceptual modelling:

In my understanding, conceptual database modelling should encapsulate all information that pertains to a specific enterprise without the consideration of physical implementation. In the conceptual stage, the focus should be on understanding the data structure and relationships between entities within the organization. The conceptual stage can be even more in-depth than the logical model as it encapsulates all ideas and possibilities that could be potentially present in the final database system. Even if some of those ideas are not present in the final system. Chen's ERD notation allows for the creation of diagrams that represent real-world objects, which can be translated into entities with relationships and attributes. This provides stakeholders and designers with a visual representation of the project's scope.

## Logical data modelling

The logical data model takes the elements from the conceptual model and adds more detail on how the data is organized within the entities, attributes, and relationships. It is essentially a low-level outline of how the database is structured. Designers can use logical models to translate requirements from the conceptual model into a structure suitable for the database system's physical implementation.

## Physical implementation

The physical implementation phase would involve translating the data model into a functional database. Tables will be created containing specific datatypes and constraints that correspond to the logical data model. Constraints, such as primary and foreign keys, act as rules to enforce data integrity. This helps ensure the data remains consistent and adheres to the defined structure, contributing to overall data accuracy.

# Description of conceptual modelling components (Chen’s ERD notation)

*Note: All figures and depictions were created using* [*Draw.io*](https://app.diagrams.net/)

# Entity

# **A close-up of a sign Description automatically generated**

Entities represent real-world objects or concepts that you want to capture specific data for. Entities are independently identified from other entities. According to Vidhya “An entity may be concrete, such as person or a book, or it may be abstract, such as a loan, or a holiday” (Vidhya, V., 2016, p, 2.8).

* An example of an entity could be a “customer”.
* Entities are defined by their attributes.

## Weak Entity:

**A black rectangle with black text

Description automatically generated**

An entity is considered weak when it depends on another entity to exist. This means it cannot be uniquely identified by its attributes. This entity exists dependently on another entity’s instance.

## Associative Entity:

**A white rectangle with black text

Description automatically generated**

The associative entity is used when a many to many relationships exists and you want to capture additional specific information within this entity (represents additional table)

# Attributes:

A black text on a white background

Description automatically generated

Attributes help define the characteristics of an entity. Attributes enable you to store information about a particular entity, for example, a customer’s details. A customer entity could have attributes such as “First Name”, “Last Name”, or “Email”. Relationships between entities can share attribute values, either through an identifying attribute or key attribute. Defining relevant attributes is important, as it aids in creating a well-structured database that effectively captures the information about the entities.

* **Simple attributes:** The attribute IS NOT divisible. E.g. Age.
* **Composite attributes:** the attribute IS divisible. E.g. Address: street, city.

**A diagram of a structure

Description automatically generated**

* **Single valued attributes:** Contains only a single value.

## Key Attribute:

A black text on a white background

Description automatically generated

A key attribute is a unique identifier that represents a specific entity instance. There can only be one “identifier” per entity. furthermore, no two instantiated entities of the same type can contain the same key value. A primary key is chosen (or composite key) to represent the retrieval of instantiated entities.

## Partial Key Attribute:

### A close-up of a label Description automatically generated

A partial key attribute provides an identifier for a weak entity by combining with the key attribute of the owner entity.

## Derived Attribute:

### A black text on a white background Description automatically generated

Derived attributes values are not stored, instead their values are calculated from other attributes in the database system. One benefit for using derived attributes is that you avoid storing potentially redundant data.

Multi valued Attribute:

**A black text in a oval shape

Description automatically generated**

“An attribute that can have many values” (Dybka, P. ,2019.)

Multivalued attributes allow an entity to contain multiple values within the same attribute, for example, a “PhoneNumber” attribute could hold a mobile number and home phone number within one attribute.

# A black hexagon with black text Description automatically generatedRelationship:

Relationships exist when an attribute of an entity refers to another entity type. There is a degree off relationship depending on the number of entities that participate in the relationship.

## Unary relationship (Recursive)

If a relationship contains an association within a single entity, then it’s a binary relationship.

A diagram of employee's work flow

Description automatically generated

## Binary relationship

If two entity types are participating in a relationship, then it’s a binary relationship.

A diamond shaped sign with black text

Description automatically generated

## Ternary relationship

If three entities are participating in a relationship, then it’s a ternary relationship. A diagram of relationship

Description automatically generated

### Weak (identifying) Relationship:

### A black hexagon with black text Description automatically generated

This relationship exists when a weak entity relies on a primary key from an owner entity to be uniquely identified.

## Optionality:

* Mandatory relationship

This relationship is shown as a double line. This means the entity must participate in the relationship.

* Optional relationship

Depicted as a single line. This means that the entity does not necessarily need to participate in the relationship.

## Cardinality:

Cardinality represents the number of instantiated entities that can associate with one another. specific characters 1, N or M are used to determine this.

* One to one

One entity instance can relate with no more than one other entity instance.

A black and white hexagon with black text

Description automatically generated

* One to many (many to one)

One entity instance can relate with many instances of another entity.

A black and white hexagon with black text

Description automatically generated

* Many to many

Many instances of an entity instance can relate with many instances of another entity.

A black and white hexagon with black text

Description automatically generated

# A diagram of a diagram Description automatically generatedConceptual Data model

## Assumptions made:

* In this model I have assumed that contracts are bound only to Super Platinum subscribers. However, if each subscription required a contract, I could move the relationship between “Contract” and “Subscription”.
* I am assuming that maintenance contractors are employed by Flight Stream, although this may not be the case, as maintenance teams could be employed by external entities.
* I have assumed that gold subscribers have access to control hardware (Video Cameras) recording the live stream service. Therefore, the relationship will be between the gold subscription and the Data Scoop. However, If the ability was only to control the instance of the Livestream itself. Then perhaps the relationship is between the gold subscription and the live stream.
* Note “Address” is a composite attribute. The depiction may be confusing. Additionally, Name is a multi-valued attribute but can be changed to a composite attribute with “First Name” “Last Name” In logical model if needed.

*Note: Additional assumptions are made at the end of the document.*

# Conceptual Model Documentation

## Table 1: Document Entities

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Name** | **Description** | **Aliases** | **Occurrence** |
| Client | Customer of Flight Stream that utilises their services | - | A client wants to access Flight Stream services |
| Account | A client made account unique to a specific client | - | A client creates an account for Flight Streams services |
| Subscription | A subscription to give clients access to specific Flight Stream services | - | A Client purchases a subscription |
| Contract | A special contract added to a specific subscription granting access to additional data | - | A Client purchases a subscription with a special contract to access data |
| DataScoop | Drone used by Flight Stream to provide services for clients | ds | DataScoop provides live video streams for its viewers |
| Employee | Personal that works for Flight Stream | - | - |
| LiveStream | Live video feed from the DataScoop drone | - | Clients interact with live streaming services provided by a DataScoop |
| Data | Information recorded by the DataScoop for specific clients to access | - | Client accesses recorded data from the DataScoop |
| Maintenance | A record of maintenance carried out by contractors | - | Maintenance contractor records maintenance carried out for a specific DataScoop |
| ScoopMaintenanceContractor | Contractor that carries out maintenance on DataScoops | - | If a DataScoop needs repair/maintenance, then a contractor is sent |
| DroneComponents | Components used to repair/maintain DataScoops | - | DataScoop requires upgrading its components |
| Supplier | Supplier of done components for the repair/maintenance of DataScoops | - | Contractors require parts to repair/maintain DataScoops |
| SalesRepresentative | Employee of Flight Stream who sell subscriptions to clients | - | Sales Representative selling subscriptions to clients |
| Zone | A specific area designated to a DataScoop | - | A DataScoop is transferred to a new zone |
| Administrator | Employee of Flight Stream who oversees contracts | - | An Administrator edits a contract |
| Gold | An upgraded version of a standard subscription that grants additional access | g | Gold subscribers gain the ability to control Data Scoops video streams |
| Platinum | An upgraded version of a gold subscription that grants additional access | p | Platinum subscribers gain access to data collected by the DataScoop |
| Super Platinum | An upgraded version of a platinum subscription that grants additional access | sp | Super Platinum subscribers have an exclusive right to specific DataScoop Zones |

## Table 2: Document Relationships

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Entity Name** | **Cardinality** | **Participation** | **Relationship** | **Participation** | **Cardinality** | **Entity Name** |
| Client | 1 | Mandatory | Creates | Optional | N | Account |
| Account | N | Mandatory | Acquires | Optional | 1 | Subscription |
| Subscription | N | Mandatory | Sells | Optional | 1 | SalesRepresentative |
| Administrator | 1 | Optional | Edits | Mandatory | N | Contract |
| Supplier | 1 | Optional | Supplies | Mandatory | N | DroneComponent |
| ScoopMaintenanceContractor | 1 | Optional | Utilizes | Optional | N | DroneComponent |
| Maintenance | N | Mandatory | Records | Optional | 1 | ScoopMaintenanceContractor |
| ScoopMaintenanceContractor | 1 | Mandatory | Maintained | Optional | N | DataScoop |
| DataScoop | N | Optional | Records | Mandatory | 1 | Data |
| DataScoop | N | Optional | Broadcasts | Mandatory | 1 | LiveStream |
| DataScoop | N | Optional | Resides | Optional | M | Zone |
| SuperPlatinum | 1 | Optional | Owns | Optional | N | Zone |
| Platinum | N | Optional | Views | Mandatory | M | Data |
| Gold | N | Optional | Controls | Optional | 1 | DataScoop |
| Subscription | 1 | Optional | Views | Optional | N | LiveStream |
| Administrator |  |  | Is a |  |  | Employee |
| SalesRepresentative |  |  | Is a |  |  | Employee |
| ScoopMaintenanceContractor |  |  | Is a |  |  | Employee |
| Gold |  |  | Inherits |  |  | Subscription |
| Platinum |  |  | Inherits |  |  | Gold |
| Super Platinum |  |  | Inherits |  |  | Platinum |

| **Entity Name** | **Attributes** | **Description** | **Domain** | **Aliases** | **Composite** | **Derived** | **Nulls** | **Key?** | **Default Value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Client** | Name | Frist Name, Last name of the client | VARCHAR | - | NO | NO | NO | NO | - |
| Address | Location of the client | VARCHAR/INTEGER | - | Country, Zip Code, City, Street Name, Street Number, Suburb | NO | NO | NO | - |
| Phone | Cell phone, home phone numbers belonging to the client | VARCHAR | - | NO | NO | NO | NO | - |
| Email | Email address belonging to the client | VARCHAR | - | NO | NO | NO | NO | - |
| ClientNumber | Unique Identifier for the client within the system | INTERGER | - | NO | NO | NO | PK | - |
| Account | AccountNumber | Unique number issued for a clients account when creating an account | INTEGER | - | NO | NO | NO | PK | - |
| Display Name | Display Name chosen by the client must be unique | VARCHAR | - | NO | NO | NO | CK | - |

## Table 3: Attributes

| **Entity Name** | **Attributes** | **Description** | **Domain** | **Aliases** | **Composite** | **Derived** | **Nulls** | **Key?** | **Default Value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Subscription | SubscriptionID | Identifies an instance of unique subscription | VARCHAR | - | NO | NO | NO | PK | - |
| Discount | A discount applied to a specific subscription instance | DECIAML | - | NO | NO | YES | NO | - |
| Employee | EmployeeID | Unique identifier for an employee | INTEGER | - | NO | NO | NO | PK | - |
| Email | Email address belonging to the employee | VARCHAR | - | NO | NO | NO | NO | - |
| Phone | Cell phone, home phone numbers belonging to the employee | VARCHAR | - | NO | NO | NO | NO | - |
| Name | Frist Name, Last name of the employee | VARCHAR | - | NO | NO | NO | NO | - |
| Address | Location of the employee | VARCHAR/INTEGER | - | Country, Zip Code, City, Street Name, Street Number, Suburb | NO | NO | NO | - |
| SalesRepresentative | SalesRepresentativeID | Uniquely identifies specific sales representatives | INTEGER | - | NO | NO | NO | PK | - |

| **Entity Name** | **Attributes** | **Description** | **Domain** | **Aliases** | **Composite** | **Derived** | **Nulls** | **Key?** | **Default Value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Administrator | AdminID | Uniquely identifies specific administrator | INTEGER | - | NO | NO | NO | PK | - |
| ScoopMaintenanceContractor | MaintenanceContractorID | Uniquely identifies specific MaintenanceContractor | INTEGER | - | NO | NO | NO | PK | - |
| Supplier | SupplierID | Uniquely identifies specific supplier | VARCHAR | - | NO | NO | NO | PK | - |
| Name | Name of the supplier/suppliers off drone components | VARCHAR | - | NO | NO | NO | NO | - |
| DroneComponent | ComponentID | Uniquely identifies specific drone component | VARCHAR | - | NO | NO | NO | PK | - |
| Price | Price of the drone component | DECIMAL | - | NO | NO | NO | NO | - |
| ComponentName | Name of the component | VARCHAR | - | NO | NO | NO | NO | - |
| Quantity | The amount of components received | INTEGER | - | NO | NO | NO | NO | - |
| Maintenance | MaintenanceRecord | Uniquely identifies a specific instance of maintenance | VARCHAR | - | NO | NO | NO | PK | - |
| MaintenanceDiscription | Optional description of completed maintenance | VARCHAR | - | NO | NO | YES | NO | - |
| Date | Date that specific maintenance is done | DATE/TIME | - | NO | NO | NO | NO | - |
| DataScoop | DataScoopID | Uniquely identifies a specific DataScoop | VARCHAR | ds | NO | NO | NO | PK | - |
| Data | RecordID | Uniquely identifies a specific data record from the DataScoop | VARCHEAR | - | NO | NO | NO | PK | - |
| Time | Identifies time of data capture | DATE/TIME | - | NO | NO | NO | NO | - |
| Temperature | Recorded temperature data | VARCHAR | - | NO | NO | NO | NO | - |
| OrganicData | Recorded Organic Data | VARCHAR | - | NO | NO | NO | NO | - |
| AmbientLightStrengtth | Recorded Light strength | VARCHAR | - | NO | NO | NO | NO | - |
| Humidity | Recorded Humidity | VARCHAR | - | NO | NO | NO | NO | - |
| Zone | Climate | Climate of the zone snow, arid etc | VARCHAR | - | NO | NO | NO | CK | - |
| Location | Location of the world LAT/LONG | VARCHAR | - | NO | NO | NO | CK | - |
| LiveStream | StreamID | Uniquely identifies a specific instance of a livestream | VARCHAR | - | NO | NO | NO | PK | - |

| **Entity Name** | **Attributes** | **Description** | **Domain** | **Aliases** | **Composite** | **Derived** | **Nulls** | **Key?** | **Default Value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Contract | ContractID | Uniquely identifies a specific contract | VARCHAR | - | NO | NO | NO | PK | - |
| OrganizationName | The name of the organization who has ownership of the contract | VARCHAR | - | NO | NO | NO | NO | - |
| Gold | Price | Price of the gold subscription | DECIMAL | g | NO | NO | NO | NO | - |
| Platinum | Price | Price of the platinum subscription | DECIMAL | p | NO | NO | NO | NO | - |
| Super Platinum | Price | Price of the super platinum subscription | DECIMAL | sp | NO | NO | NO | NO | - |

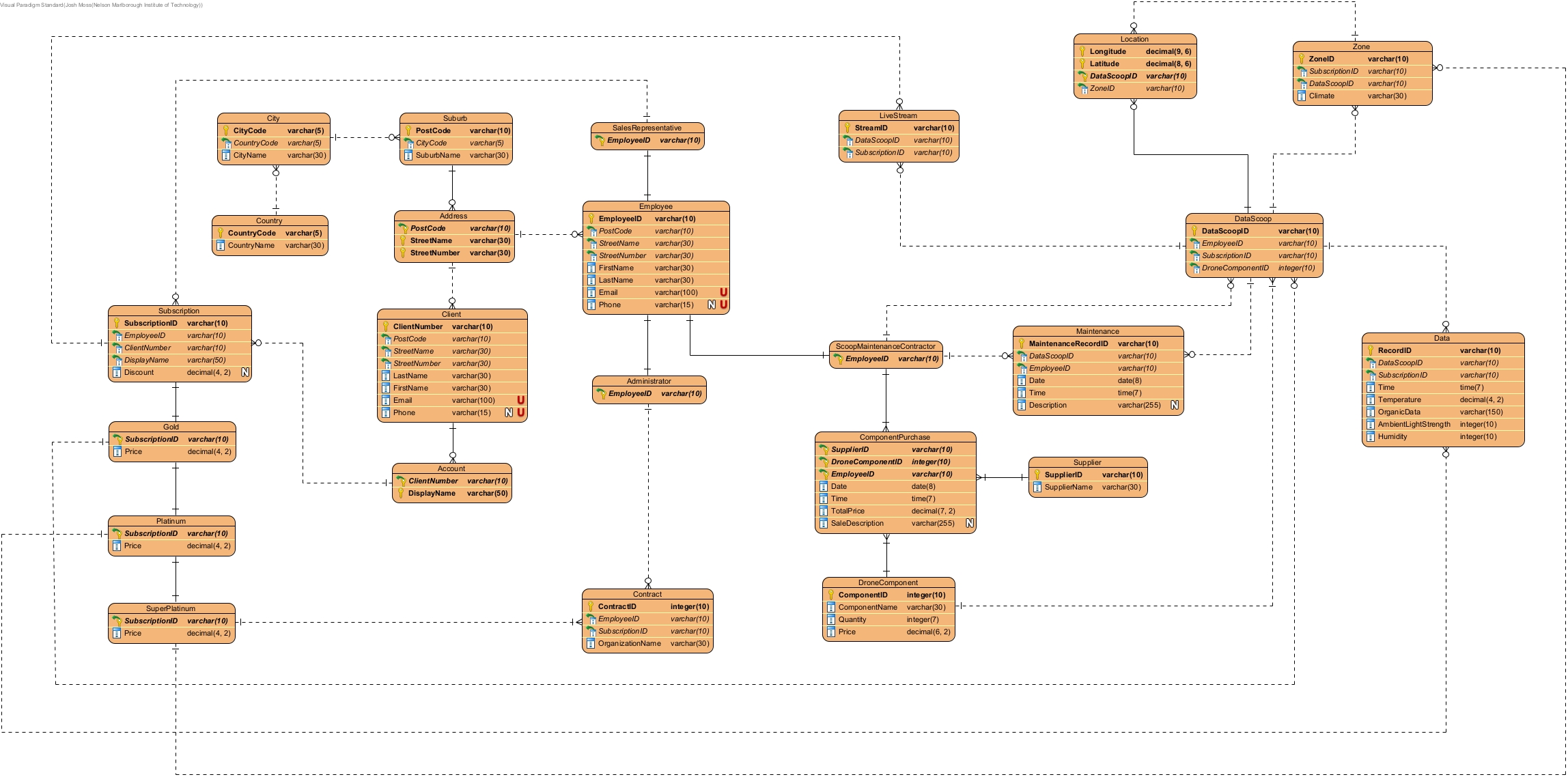
*Note : See assumptions below for reflection and rationale of data dictionary*

# Assumptions

* I assumed implementing an account to act as a middle ground between clients and subscriptions to be a logical choice. I believe this as a client can then create multiple accounts for separate requirements (business, personal). Additionally, clients can create a desired display name to interact with the live stream. This adds a layer of security by not displaying real names on the live stream. As a counterargument, removing the account entity and having a subscriber directly relating to a specific subscription could result in less complexity. However, when moving into a logical model, I feel normalization becomes skewed as you have too many attributes that do not directly correspond to a specific “subscriber”.  I understand, however, that this thought process is potentially outside of the scope of a conceptual model.
* The subscription entity (standard subscription) acts as a parent entity for each subscription instance and is inherited by all subsequent subscription entities. Furthermore, each subsequent subscription tier inherits from the previous. (gold, platinum, super platinum).
* The use of a data dictionary helps define all data elements that pertain to the system in a clear and concise manner. It aids in giving a user a more in-depth understanding of the data system before the transition into a logical model. Without the data dictionary, the transition into the next phases would be vague, lacking in established datatypes and potential validation rules.
* For my data dictionary entities table, I aimed to give a clear but brief overview of the core components providing a short description of each entity, hopefully assisting any external actors in understanding my data model. As for my relationships, this data is crucial for understanding how my data interacts within my system. It defines cardinality and participation between my entities. Lastly, my attributes table. This details each attribute that belongs to an entity within my system. The table contains descriptions, domains, keys, and null values. Detailed views of attributes such as this help ensure data is consistent and accurate throughout the data modelling process.

# **Chapter Two**

# Logical data model



# Logical model Rationale

* The approach I took when modelling the relations for client address pool was to separate them, so the possibility of redundant data (duplicated data) is minimised. However, after reflection on this decision, I have concluded that what I have depicted may not be accurate. This is because the nature of what a city and a suburb is rather subjective and can vary depending on location. Additionally, not every city necessarily has suburbs within it. Perhaps a “Rural” and “Urban area” approach would have made more sense. Additionally, these higher levels of normalisation are outside the scope of the project and end up adding complexity to queries once the system is deployed. Although this helps minimise data redundancy perhaps leaving the address information in Boyce Codd normal form is the best option.
* Clients create accounts within the system which can contain a subscription service which are sold by sales representatives, each subsequent subscription tier inherits access from its pervious tier.
* Sales representatives sell subscriptions to accounts owned by clients. However, I am unsure if this is accurately represented within the model depicted. Perhaps, the relationship is only between “paid” subscriptions (gold and higher) and the “discount” is an attribute belonging to the Sales Representative that could contain a check to determine the discount amount.
* Contracts are created and distributed by administrators of Flight Stream. These contracts are tied to a specific subscription (super platinum) granting special access to services Flight Stream provides. However, I am unsure if this would be correct. Depending on what the business requires, the contract relation could be between any “paid subscription” tier (gold, platinum, super platinum) granting access to special services as stated above. However, this scenario would require a relationship between the “paid subscription” entities and the contract.

Another possibility is that the contracts are tied to specific zones that are managed by higher-level subscribers (super platinum), therefore, the relation would be between the zone and contract.

* An additional table was added for the purchase of drone components. This table helps to capture a specific instance of a purchase and create a record of the sale.
* Added a relationship between component and Data Scoop. This allows for components to be mapped to specific Data Scoops for maintenance and inventory management purposes.
* For Accounts I have made a composite key consisting of DisplayName and ClientNumber, this is so that the Livestream has access to the clients DisplayName though subscription table. The reason I did this was to minimise complex joins and queries once deployed. A relationship between account and livestream is perhaps another option for account names to show in livestream that would not require a composite key. Although, I have assumed that accounts must have some sort of subscription to view live streams.

# Logical Model Documentation

## Mapping rules

* **Composite attribute Address:**

Mapped into four relations:

1. Country

A close-up of a label

Description automatically generated

1. City

A close-up of a code

Description automatically generated

1. Suburb

A diagram of a code

Description automatically generated

1. Address

A close-up of a address

Description automatically generated

* **Location**

Mapped into one relation:

1. Location:

A close-up of a table

Description automatically generated

**Supplies relationship date attribute:**

Mapped into one relation:

1. ComponentPurchase:

A table of a component purchase

Description automatically generated with medium confidence

# Documented relations

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation Name** | **Start Volume** No. of rows loaded at the beginning | **Growth** e.g. no growth / 10% per year | **Comments** |
| Country | 5 | 2% per year | - |
| City | 250 | 2% per year | - |
| Suburb | 20,000 | 4% per year | - |
| Address | 550,000 | 8% per year | - |
| Client | 500,000 | 10% per year | - |
| Account | 500,000 | 10% per year | - |
| Subscription | 500,000 | 10% per year | - |
| Gold | 5000 | 5% per year | - |
| Platinum | 2500 | 2% per year | - |
| SuperPlatinum | 100 | 1% per year | - |
| Contract | 2000 | 1% per year | - |
| employee | 600 | 10% per year | - |
| Administrator | 40 | 2% per year | - |
| SalesRepresentative | 400 | 5% per year | - |
| ScoopMaintenanceContractor | 160 | 5% per year | - |
| LiveStream | 100 | 100% per year | - |
| DataScoop | 50,000 | No growth | - |
| Maintenance | 25,000 | 50% per year | - |
| ComponentPurchase | 150,000 | 30% per year | - |
| Supplier | 20 | No growth |  |
| DroneComponent | 500,000 | No growth | Assuming each drone has 10 components |
| data | 100,000 | 100% per year | - |
| Zone | 100 | 10% per year | - |
| Location | 1000 | 20% per year | Assuming each zone has roughly 10 locations |

# Document Attributes

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Country** | CountryCode | A code given to a specific country to identify it. | varchar | 5 |  | Must be Unique | - | NO | PK | - | The CountryCode must be unique within the table |
| **Country** | CountryName | The name of a specific country | varchar | 30 | - | - | - | NO | - | - | - |
| **City** | CityCode | A code given to a specific city to identify it. | Varchar | 5 | - | Must be unique | - | NO | PK | - | The CityCode must be unique within the table |
| **City** | CountryCode | A code given to a specific country to identify it. | Varchar | 5 | - | Must be unique | - | NO | FK | Country | Foreign key Constraint |
| **Suburb** | PostCode | A code given to a specific suburb to identify it. | Varchar | 10 | - | Must be unique | - | NO | Pk |  | The PostCode must be unique within the table. |
| **Suburb** | CityCode | A code given to a specific country to identify it. | Varchar | 5 | - | - | - | NO | FK | City | Foreign key Constraint |
| **Suburb** | **SuburbName** | Name of a specific suburb | Varchar | 30 | - | - | - | NO | - | - | - |
| **Address** | PostCode | A code given to a specific suburb to identify it. | Varchar | 10 | - | Must be unique |  | NO | CK | Suburb | The PostCode must be unique within the table |
| **Address** | StreetName | Name of the street address | Varchar | 30 | - | - | - | NO | CK | - | - |
| **Address** | StreetNumber | Number of the street address | varchar | 30 | - | - | - | NO | CK | - | - |
| **Client** | ClientNumber | A code given to a specific client to identify them. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The ClientNumber must be unique within the table |
| **Client** | PostCode | A code given to a specific suburb to identify it. | Varchar | 10 | - | - | - | NO | FK | Address | Foreign key Constraint |
| **Client** | StreetName | Name of the client’s street address | Varchar | 30 | - | - | - | NO | FK | Address | Foreign key Constraint |
| **Client** | StreetNumber | Number of the client’s street address | Varchar | 30 | - | - | - | NO | FK | Address | Foreign key Constraint |
| **Client** | LastName | Last name of the client | Varchar | 30 | - | - | - | NO | - | - | - |
| **Client** | FirstName | First name of the client | Varchar | 30 | - | - | - | NO | - | - | - |
| **Client** | Email | Email address of the client | Varchar | 100 | - | Must be unique | - | NO | - | - | The Email must be unique within the table |
| **Client** | Phone | Phone number of the client | Varchar | 15 | - | Must be unique | - | YES | - | - | If phone record exists it must be unique within the table |
| **Account** | ClientNumber | A code given to a specific Account to identify it. | Varchar | 10 | - | Must be unique | - | NO | CK | - | The ClientNumber must be unique within the table |
| **Account** | DisplayName | A unique alias created by a client when creating an account | Varchar | 50 | - | Must be unique | - | NO | CK | - | The DisplayName must be unique within the table |
| **Subscription** | SubscriptionID | A code given to a specific Subscription to identify it. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The SubscriptionID must be unique within the table |
| **Subscription** | EmployeeID | A code given to a specific Employee to identify them. | Varchar | 10 | - | - | - | NO | FK | Employee | Foreign key constraint |
| **Subscription** | ClientNumber | A code given to a specific Account to identify it. | Varchar | 10 | - | - | - | NO | FK | Account | Foreign key constraint |
| **Subscription** | DisplayName | A unique alias created by a client when creating an account | Varchar | 50 | - | - | - | NO | FK | Account | Foreign key constraint |
| **Subscription** | Discount | A discount amount given to a specific subscription | Decimal | 4,2 | - | - | - | YES | - | - | - |
| **Gold** | SubscriptionID | A code given to a specific Subscription to identify it. | Varchar | 10 | - | - | - | NO | PK | Subscription | Foreign key constraint |
| **Gold** | Price | The price of a specific subscription | Decimal | 4,2 | - | - | - | NO | - | - | - |
| **Platinum** | SubscriptionID | A code given to a specific Subscription to identify it. | Varchar | 10 | - | - | - | NO | PK | Gold | Foreign key constraint |
| **Platinum** | Price | The price of a specific subscription | Decimal | 4,2 | - | - | - | NO | - | - | - |
| **SuperPlatinum** | SubscriptionID | A code given to a specific Subscription to identify it. | Varchar | 10 | - | - | - | NO | PK | Platinum | Foreign key constraint |
| **SuperPlatinum** | Price | The price of a specific subscription | Decimal | 4,2 | - | - | - | NO | - | - | - |
| **Employee** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The EmployeeID must be unique within the tabl |
| **Employee** | PostCode | A code given to a specific suburb to identify it. | Varchar | 10 | - | - | - | NO | FK | Address | Foreign key Constraint |
| **Employee** | StreetName | Name of the street address of the employee | Varchar | 30 | - | - | - | NO | FK | Address | Foreign key Constraint |
| **Employee** | StreetNumber | Number of the street address of the employee | Varchar | 30 | - | - | - | NO | FK | Address | Foreign key Constraint |
| **Employee** | LastName | Last name of the employee | Varchar | 30 | - | - | - | NO | - | - | - |
| **Employee** | FirstName | First name of the client | Varchar | 30 | - | - | - | NO | - | - | - |
| **Employee** | Email | Email address of the employee | Varchar | 100 | - | Must be unique | - | NO | - | - | The Email must be unique within the table |
| **Employee** | Phone | Phone number of the employee | Varchar | 15 | - | Must be unique | - | YES | - | - | If phone record exists it must be unique within the table. |
| **SalesRepresentative** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | Must be unique | - | NO | PK | - | Foreign key constraint |
| **Administrator** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | Must be unique | - | NO | PK | - | Foreign key constraint |
| **ScoopMaintenanceContractor** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | Must be unique | - | NO | PK | - | Foreign key constraint |
| **Contract** | ContractID | A code given to a specific contract to uniquely identify it. | Interger | 10 | - | Must be unique | - | NO | PK | - | - |
| **Contract** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | - | - | NO | FK | Administrator | Foreign key constraint |
| **Contract** | SubscriptionID | A code given to a specific Subscription to identify it. | Varchar | 10 | - | - | - | NO | FK | SuperPlatinum | Foreign key constraint |
| **Contract** | OrganizationName | The name of the organization that is tied to the contract | Varchar | 30 | - | - | - | NO | - | - | - |
| **Supplier** | SupplierID | A code given to a specific Supplier to identify it. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The SupplierID must be unique within the table |
| **Supplier** | **SupplierName** | The name of the supplier | Varchar | 30 | - | - | - | NO | - | - | - |
| **DroneComponent** | ComponentID | A code given to a specific component to identify it. | Integer | 10 | - | Must be unique | - | NO | PK | - | The ComponentID must be unique within the table |
| **DroneComponent** | ComponentName | The name of a component | Varchar | 30 | - | - | - | NO | - | - | - |
| **DroneComponent** | Quantity | The quantity of specific drone components | Integer | 7 | - | - | - | NO | - | - | - |
| **DroneComponent** | Price | The price of specific done components | Decimal | 6,2 | - | - | - | NO | - | - | - |
| **ComponentPurchase** | SupplierID | A code given to a specific Supplier to identify it. | Varchar | 10 | - | - | - | NO | PK | Supplier | Foreign key constraint |
| **ComponentPurchase** | ComponentID | A code given to a specific component to identify it. | Integer | 10 | - | - | - | NO | PK | DroneComponent | Foreign key constraint |
| **ComponentPurchase** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | - | - | NO | PK | ScoopMaintenanceContractor | Foreign key constraint |
| **ComponentPurchase** | Date | Date of the specific purchase | Date | 8 | - | YYYY-MM-DD | - | NO | - | - | - |
| **ComponentPurchase** | Time | Time of the specific purchase | Time | 7 | 00:00:00.0000000 through 23:59:59.9999999 | HH:MM:SS.sssssss | - | NO | - | - | - |
| **ComponentPurchase** | TotalPrice | The total cost of the specific purchase of drone components | Decimal | 7,2 | - | - | - | NO | - | - | - |
| **ComponentPurchase** | SalesDescription | An optional description for a specific instance of a purchase | Varchar | 255 | - | - | - | YES | - | - | - |
| **DataScoop** | DataScoopID | A code given to a specific DataScoop to identify it. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The DataScoopID must be unique within the table |
| **DataScoop** | EmployeeID | A code given to a specific employee to identify them. | Varchar | 10 | - | - | - | NO | FK | ScoopMaintenanceContractor | Foreign key constraint |
| **DataScoop** | SubscriptionID | A code given to a specific Subscription to identify it. | Varchar | 10 | - | - | - | NO | FK | Gold | Foreign key constraint |
| **DataScoop** | DroneComponentID | A code given to a specific component to identify it. | Integer | 10 | - | - | - | NO | FK | DroneComponent | Foreign key constraint |
| **Data** | RecordID | A code given to a specific set of data to identify it. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The RecordID must be unique within the table |
| **Data** | DataScoopID | A code given to a specific DataScoop to identify it. | Varchar | 10 | - | - | - | NO | FK | DataScoop | Foreign key constraint |
| **Data** | SubscriptionID | A code given to a specific subscription to identify it. | Varchar | 10 | - | - | - | NO | FK | Platinum | Foreign key constraint |
| **Data** | Time | Time related to the recorded data | Time | 7 | 00:00:00.0000000 through 23:59:59.9999999 | HH:MM:SS.sssssss | - | NO | - | - | - |
| **Data** | Temperature | Temperature related to the recorded data | Decimal | 4,2 | - | - | - | NO | - | - | - |
| **Data** | OrganicData | OrganicData related to the recorded data | Varchar | 150 | - | - | - | NO | - | - | - |
| **Data** | AmbientLightStrength | AmbientLightStrength related to the recorded data | Integer | 10 | - | - | - | NO | - | - | - |
| **Data** | Humidity | Humidity related to the recorded data | Integer | 10 | - | - | - | NO | - | - | - |
| **LiveStream** | StreamID | A code given to a specific LiveStream to identify it. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The StreamID must be unique within the table. |
| **LiveStream** | DataScoopID | A code given to a specific DataScoop to identify it. | Varchar | 10 | - | - | - | NO | FK | DataScoop | Foreign key constraint |
| **LiveStream** | SubscriptionID | A code given to a specific subscription to identify it. | Varchar | 10 | - | - | - | NO | FK | Subscription | Foreign key constraint |
| **Zone** | ZoneID | A code given to a specific zone to identify it. | Varchar | 10 | - | Must be unique | - | NO | PK | - | The ZoneID must be unique within the table |
| **Zone** | SubscriptionID | A code given to a specific subscription to identify it. | Varchar | 10 | - | - | - | NO | FK | SuperPlatinum | Foreign key constraint |
| **Zone** | DataScoopID | A code given to a specific DataScoop to identify it. | Varchar | 10 | - | - | - | NO | FK | DataScoop | Foreign key constraint |
| **Zone** | Climate | The climate of the zone (snow, arid, tropical etc) | Varchar | 30 | - | - | - | NO | - | - | - |
| **Location** | Longitude | Longitude co-ordinates related to a specific location | Decimal | 9,6 | - | - | - | NO | CK | - | - |
| **Location** | Latitude | Latitude co-ordinates related to a specific location | Decimal | 8,6 | - | - | - | NO | CK | - | - |
| **Location** | DataScoopID | A code given to a specific DataScoop to identify it. | Varchar | 10 | - | - | - | NO | FK | DataScoop | Foreign key constraint |
| **Location** | ZoneID | A code given to a specific zone to identify it. | Varchar | 10 | - | - | - | NO | FK | Zone | Foreign key constraint |

Derived attributes:

| **Relation Name** | **Attributes** | **Derived from / calculation** |
| --- | --- | --- |
| **ComponentPurchase** | TotalPrice | Price, Quantity |

# NaLER Analysis

## Description of NaLER

NaLER is A natural language method that enables users to interpret Entity Relationship models.

**each entity** is constructed into a sentence consisting of the primary key attribute(s) in the format:

* “Each <entity name> is uniquely identified by <Primary key>” ([NaLER.ppt](https://ecampus.nmit.ac.nz/moodle/pluginfile.php/2145828/mod_resource/content/2/NaLER.ppt))

**Each attribute** that is not a primary or foreign key is constructed in this format:

* “One <E-name> identified by <PK> must have one <attribute name>” ([NaLER.ppt](https://ecampus.nmit.ac.nz/moodle/pluginfile.php/2145828/mod_resource/content/2/NaLER.ppt))

Each relationship that the entity participates in is in this format:

* “One <E-name1> identified by <PK> <optionality><relationship-name><cardinality><E-name2>identified by <foreign key>” ([NaLER.ppt](https://ecampus.nmit.ac.nz/moodle/pluginfile.php/2145828/mod_resource/content/2/NaLER.ppt))

### Attributes

S1: Each Country is uniquely identified by one CountryCode

S2: Each Country (CountryCode) must have one CountryName

S3: Each City is uniquely identified by one CityCode

S4: Each City (CityCode) must have one CountryCode

S4: Each City (CityCode) must have one CityName

S5: Each Suburb is uniquely identified by one PostCode

S6: Each Suburb (PostCode) must have one CityCode

S7: Each Suburb (PostCode) must have one SuburbName

S8: Each Address is uniquely identified by PostCode, StreetName, StreetNumber

S9: One Client is uniquely identified by one ClientNumber

S10: One Client (ClientNumber) must have one PostCode

S11: One Client (ClientNumber) must have one StreetName

S12: One Client (ClientNumber) must have one StreetNumber

S13: One Client (ClientNumber) must have one LastName

S14: One Client (ClientNumber) must have one FirstName

S15: One Client (ClientNumber) must have one unique Email

S16: One Client (ClientNumber) may have one unique Phone

S17: One Employee is uniquely identified by EmployeeID

S18: One Employee (EmployeeID) must have one PostCode

S19: One Employee (EmployeeID) must have one StreetName

S20: One Employee (EmployeeID) must have one StreetNumber

S21: One Employee (EmployeeID) must have one FirstName

S22: One Employee (EmployeeID) must have one LastName

S23: One Employee (EmployeeID) must have one unique Email

S24: One Employee (Employee) may have one unique Phone

S25: Each Administrator is uniquely identified by EmployeeID

S26: Each SalesRepresentative is uniquely identified by EmployeeID

S27: Each ScoopMaintenanceContractor is uniquely identified by EmployeeID

S28: Each Account is uniquely identified by ClientNumber, DisplayName

S29: Each Subscription is uniquely identified by SubscriptionID

S30: One Subscription (SubscriptionID) must have one EmployeeID

S31: One Subscription (SubscriptionID) must have one ClientNumber

S32: One Subscription (SubscriptionID) must have one DisplayName

S33: One Subscription (SubscriptionID) may have one Discount

S34: Each Gold is uniquely identified by SubscriptionID

S35: One Gold (SubscriptionID) must have one Price

S36: Each Platinum is uniquely identified by SubscriptionID

S37: One Platinum (SubscriptionID) must have one Price

S38: Each SuperPlatinum is uniquely identified by SubscriptionID

S39: One SuperPlatinum (SubscriptionID) must have one Price

S40: Each Contract is uniquely identified by ContractID

S41: One Contract (ContractID) must have one EmployeeID

S42: One Contract (ContractID) must have one SubsciptionID

S43: One Contract (ContractID) must have one OrganizationName

S44: Each LiveStream is uniquely identified by StreamID

S45: One LiveStream (StreamID) must have one DataScoopID

S46: One LiveStream (StreamID) must have one SubscriptionID

S47: Each DataScoop is uniquely Identified by one DataScoopID

S48: One DataScoop (DataScoopID) must have one EmployeeID

S49: One DataScoop (DataScoopID) must have one SubscriptionID

S50: One DataScoop (DataScoopID) must have one DroneComponentID

S51: Each ComponentPurchase is uniquely identified by SupplierID, DroneComponentID, EmployeeID

S52: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) must have one Date

S53: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) must have one Time

S54: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) must have one TotalPrice

S55: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) may have one SalesDescription

S56: Each Supplier is uniquely identified by SupplierID

S57: One Supplier (SupplierID) must have one SupplierName

S58: Each DroneComponent is uniquely identified by ComponentID

S59: One DroneComponent (ComponentID) must have one ComponentName

S60: One DroneComponent (ComponentID) must have one Quantity

S61: One DroneComponent (ComponentID) must have one Price

S62: Each Maintenance is uniquely identified by MaintenanceRecordID

S63: One Maintenance (MaintenanceRecordID) must have one DataScoopID

S64: One Maintenance (MaintenanceRecordID) must have one EmployeeID

S65: One Maintenance (MaintenanceRecordID) must have one Date

S66: One Maintenance (MaintenanceRecordID) must have one Time

S67: One Maintenance (MaintenanceRecordID) may have one Description

S68: Each Location is uniquely identified by Longitude, Latitude, DataScoopID

S69: One Location (Longitude, Latitude, DataScoopID) must have one ZoneID

S70: Each Zone is uniquely identified by ZoneID

S71: One Zone (ZoneID) must have one SubscriptionID

S72: One Zone (ZoneID) must have one DatascoopID

S73: One Zone (ZoneID) must have one Climate

S74: Each Data is uniquely identified by RecordID

S75: One Data (RecordID) must have one DataScoopID

S76: One Data (RecordID) must have one SubscriptionID

S77: One Data (RecordID) must have one Time

S78: One Data (RecordID) must have one Temperature

S79: One Data (RecordID) must have one OrganicData

S80: One Data (RecordID) must have one AmbientLightStrength

S81: One Data (RecordID) must have one Humidity

### Relationships

S82: Each Country (CountryCode) may have zero or more Cities (CityCode)

S83: One City (CityCode) must belong to one Country (CountryCode)

S84: One City (CityCode) may have zero or more Suburbs (PostCode)

S85: One Suburb (PostCode) must belong to one City (CityCode)

S86: One Suburb (PostCode) may have zero or more address’s (PostCode, StreetName, StreetNumber)

S87: One Address (PostCode, StreetName, StreetNumber) must belong to one Suburb (PostCode)

S88: Each Address (PostCode, StreetName, StreetNumber) may have zero or more Client/s (ClientNumber)

S89: One Client (ClientNumber) must reside at an Address (PostCode, StreetName, StreetNumber)

S90: Each Address (PostCode, StreetName, StreetNumber) may have zero or many Employee/s (EmployeeID)

S91: One Employee (EmployeeID) must reside at an Address (PostCode, StreetName, StreetNumber)

S92: One Client (ClientNumber) may have zero or more Account/s (ClientNumber, DisplayName)

S93: One Account (ClientNumber, DisplayName) must relate to one Client (ClientNumber)

S94: Each Account (ClientNumber, DisplayName) may have zero or more Subscription/s (SubscriptionID)

S95: One subscription (SubscriptionID) must relate to one Account (ClientNumber, DisplayName)

S96: One subscription (SubscriptionID) is a Gold (SubscriptionID)

S97: One Gold (SubscriptionID) must be a Subscription (SubscriptionID)

S98: One gold (SubscriptionID) is a Platinum (SubscriptionID)

S99: One Platinum (SubscriptionID) must be a Gold (SubscriptionID)

S100: One Platinum (SubscriptionID) is a SuperPlatinum (SubscriptionID)

S101: One SuperPlatinum (SubscriptionID) must be a Platinum (SubscriptionID)

S102: One SuperPlatinum (SubscriptionID) may have one or more Contract/s (ContractID)

S103: One Contract (ContractID) Must relate to one SuperPlatinum (SubscriptionID)

S104: One Subscription (SubscriptionID) may view zero or more LiveStream/s (StreamID)

S105: Each LiveStream (StreamID) requires a Subscription to view (SubscriptionID)

S106: Each Gold (SubscriptionID) may control zero or more DataScoop/s (DataScoopID)

S107: One DataScoop (DataScoopID) must relate to one Gold (SubsciptionID)

S108: Each Platinum (SubscriptionID) may view zero or more Data (RecordID)

S109: One Data (RecordID) must relate to one Platinum (SubscriptionID)

S110: One SuperPlatinum (SubscriptionID) may own zero or more Zone/s (ZoneID)

S111: One Zone (ZoneID) must relate to one SuperPlatinum (SubscriptionID)

S112: One SalesRepresentative (EmployeeID) may subscribe zero or many subscriptions (SubscriptionID)

S113: One Subscription (SubscriptionID) must relate to one SalesRepresentative (EmployeeID)

S114: One SuperPlatinum (SubscriptionID) may have one or more Contract/s (ContractID)

S115: One Contract (ContractID) must relate to one SuperPlatinum (SubscriptionID)

S116: One Employee (EmployeeID) is a Administrator (EmployeeID)

S117: One Administrator (EmployeeID) must relate to one Employee (EmployeeID)

S118: One Administrator (EmployeeID) may edit zero or many Contracts (ContractID)

S119: One Contract (ContractID) must relate to one Employee (EmployeeID)

S120: One Employee (EmployeeID) is a SalesRepresentative (EmployeeID)

S121: One SalesRepresentative (EmployeeID) must relate to one Employee (EmployeeID)

S122: One Employee (EmployeeID) is a ScoopMaintenanceContractor (EmployeeID)

S123: One ScoopMaintenanceContractor (EmployeeID) must relate to one Employee (EmployeeID)

S124: One ScoopMaintenanceContractor (EmployeeID) may purchase one or more components (SupplierID, DroneComponentID, EmployeeID)

S125: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) must relate to one ScoopMaintenanceContractor (EmployeeID)

S125: One Supplier (SupplierID) may supply one or more components (SupplierID, DroneComponentID, EmployeeID)

S126: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) must relate to one Supplier (SupplierID)

S127: One DroneComponent (ComponentID) must contribute one or more components for ComponentPurchase (SupplierID, DroneComponentID, EmployeeID)

S128: One ComponentPurchase (SupplierID, DroneComponentID, EmployeeID) must relate to one DroneComponent (ComponentID)

S129: One ScoopMaintenanceContractor (EmployeeID) may carry out zero or more Maintenance (MaintenanceRecordID)

S130: One maintenance (MaintenanceRecordID) must relate to one ScoopMaintenanceContractor (EmployeeID)

S131: One DroneComponent (ComponentID) may relate to one or more DataScoop/s (DataScoopID)

S132: One DataScoop (DataScoopID) must relate to one component (ComponentID)

S133: One ScoopMaintenanceContractor (EmployeeID) may be assigned zero or more DataScoop/s (DataScoopID)

S134: One DataScoop (DataScoopID) must relate to one ScoopMaintenanceContractor (EmployeeID)

S135: One DataScoop (DataScoopID) may have zero or more Maintenance (MaintenanceRecordID)

S136: One Maintenance (MaintenanceRecordID) must relate to one DataScoop (DataScoopID)

S137: One DataScoop (DataScoopID) may have zero or more LiveStream/s (StreamID)

S138: One LiveStream (StreamID) must relate to one DataScoop (DataScoopID)

S139: One DataScoop (DataScoopID) may have zero or many Zone (ZoneID)

S140: One Zone (ZoneID) must relate to one DataScoop (DataScoop)

S141: Each DataScoop (DataScoopID) may have zero or many Location (Longitude, Latitude, DataScoopID)

S142: One Location (Longitude, Latitude, DataScoopID) must relate to one DataScoop (DataScoopID)

S142: One Zone (ZoneID) may have zero or many Location/s (Longitude, Latitude, DataScoopID)

S143 One Location (Longitude, Latitude, DataScoopID) must relate to one Zone (ZoneID)

# Normalisation

Normalisation helps to minimise redundancy and dependencies within a database system. There are 6 levels of normalization:

**First Normal Form:**

* Attributes that can be calculated are removed.
* A primary key is assigned to the relation.
* Any repeating groups are removed.

Second Normal Form:

* Full functional dependency on the primary is required for all non-key attributes.
* New relations are created for attributes that are partially dependent on the primary key.

Third normal form:

* Identify any relationships between non-key attributes, including alternate keys which are also considered key attributes.
* Create a separate table for attributes that indirectly depend on the primary key.

Boyce Codd Normal Form:

* Find all sets of attributes that uniquely identify each row in the table.
* Identify any relationships between attributes in the table, where the value of one attribute determines the value of another.
* If some data dependencies rely on attributes that aren't unique identifiers, remove those dependent attributes and create a separate table.

Fourth Normal Form:

* The transition of BCNF to 4NF is the removal of multi-valued dependency from a specific relation and moving the attributes to a new table.

References

Dybka, P. (2019, February 13). Chen notation. Vertabelo Data Modeler. <https://vertabelo.com/blog/chen-erd-notation/>

Vidhya, V., et al. *Database Management Systems*, Alpha Science International, 2016.*ProQuest Ebook Central*, <https://ebookcentral.proquest.com/lib/lnmit/detail.action?docID=5248352.>

Learn Computer Science. (2020, December 11). What Is Entity Relationship Diagram ( ERD ) | ER Model Explained In DBMS With Examples. YouTube. <https://www.youtube.com/watch?v=BaN7Ng7a82Y&ab_channel=LearnComputerScience>

<https://www.youtube.com/watch?v=VGxGELh5n2U&ab_channel=BrianFinnegan>

[NaLER.ppt](https://ecampus.nmit.ac.nz/moodle/pluginfile.php/2145828/mod_resource/content/2/NaLER.ppt) from logical model resources on moodle