DAT601

Table of Contents

[Conceptual Database Design 3](#_Toc170276343)

[Introduction to data modeling in information systems 3](#_Toc170276344)

[A description of conceptual modelling 4](#_Toc170276345)

[A Conceptual Entity-Relationship Diagram 11](#_Toc170276346)

[Data Dictionary 12](#_Toc170276347)

[Table 1: Document Entities 12](#_Toc170276348)

[Table 2: Document Relationship 13](#_Toc170276349)

[Table 3: Document Attributes 15](#_Toc170276350)

[Business Assumptions 27](#_Toc170276351)

[Conceptual to Relational Logical model 28](#_Toc170276352)

[Transform to logical model 28](#_Toc170276353)

[Logical Model 30](#_Toc170276354)

[Retionale for ERD 31](#_Toc170276355)

[Normalization 32](#_Toc170276356)

[Description of Normalization 32](#_Toc170276357)

[First Normal Form (1NF) 32](#_Toc170276358)

[Second Normal Form (2NF) 32](#_Toc170276359)

[Third Normal Form (3NF) 32](#_Toc170276360)

[Boyce-Codd Normal Form 33](#_Toc170276361)

[Fourth Normal Form (4NF) 33](#_Toc170276362)

[Normalization for FlightStream Organization 33](#_Toc170276363)

[Data Dictionary 35](#_Toc170276364)

[Document Relations 35](#_Toc170276365)

[Document Attributes 37](#_Toc170276366)

[Derived Attributes 48](#_Toc170276367)

[NaLER Analysis 49](#_Toc170276368)

[Introduction to NaLER Analysis 49](#_Toc170276369)

[NaLER Analysis – for my ERD 49](#_Toc170276370)

[NaLER - Cardinality 50](#_Toc170276371)

[NaLER - Attribute Sentences: 51](#_Toc170276372)

[NaLER – Relationship Sentences: 52](#_Toc170276373)

[Conclusion 53](#_Toc170276374)

[Physical database design and implementation 53](#_Toc170276375)

[Physical Database Design 53](#_Toc170276376)

[Translate global logical data model for target DBMS 53](#_Toc170276377)

[Design Base relation for target DBMS 53](#_Toc170276378)

[Design Enterprise constraints for DBMS 59](#_Toc170276379)

[Design physical Representation 60](#_Toc170276380)

[Analyze Transactions 60](#_Toc170276381)

[Choose File Organizations 60](#_Toc170276382)

[Choose Secondary Indexes 60](#_Toc170276383)

[Consider the introduction of controlled redundance 61](#_Toc170276384)

[Estimate disk space requirements 61](#_Toc170276385)

[Physical Design for FlightStream 64](#_Toc170276386)

[Issues & Resolutions 65](#_Toc170276387)

[Conclusion 66](#_Toc170276388)

[References 67](#_Toc170276389)

# Conceptual Database Design

## Introduction to data modeling in information systems

* Data modeling in information systems is the process of creating a visual representation that defines the structure and relationship between data objects within the information systems of organizations and businesses *(IBM, n.d.).* To visualize data modelling, ‘Entity-Relationship Diagrams’ or ERD are commonly used which makes it easier to organize the collected information by which then can be used to build effective information systems that meets the organizations and business’s needs.
* There are 8 key principles in data management - Accountability, Transparency, Integrity, Protection, Compliance, Availability, Retention, and Disposition. These principles are known as Generally Acceptable Recordkeeping Principles (GARP) and it describes how data should be maintained. Incorporating these principles and implementing them in the development of information systems will allows for a better and effective data management *(Salahuddin, 2021).*
* There are three stages in database design methodology – Conceptual, Logical and Physical design. Each stage refers to a different level or phases of a database design process.
* Conceptual database design is the process of understanding the organizations requirement. This stage defines the overall structure of the database and focus on identifying the relationship of one data to another *(Visual Paradigm, 2023)* .
* Logical database design takes the previous phase, conceptual database design into more detail. This stage defines the relationship and the constraints between the data which helps ensure data integrity within the database. Logical database design process involves defining tables, entities, attributes, keys and the data types of the database.  
  This stage is independent on specific Database Management System (DBMS) and should align with the requirements and needs of the organization. To visualize the logical database design process, a data model such as an Entity-Relationship Diagram (ERD) is commonly used, which helps in understanding the relationship between the entities or data much clearer.
* Physical database design is the last stage of the database design process. This stage builds on the previous phases conceptual and logical design and begin the implementation on the chosen database management system (DBMS) and create a database.  
  The Physical design process involves considering the database storage, performance optimization, security, and access controls.

## A description of conceptual modelling

* Enhanced Entity-Relationship diagram or EER is used when the complexity of data modelling rises, and the Entity-Relationship diagram (ERD) could no longer be used to visualize in conceptual modelling due to the complicated structure of organization *(javatpoint, n.d.)*.
* Conceptual modelling is used to represent the organization’s structure without independent of all physical considerations. It’s the stage of understanding the organization’s objects or entities, relationships, attributes and visualizing it using diagrams such as Chen Entity-Relationship (ERD) and Enhanced Entity-Relationship diagram (EER).
* Chen Entity-Relationship diagram (ERD) as well as the Chen Extended Entity-Relationship diagram (EER) is a widely used diagrams in representing an organization’s structure visually. There are various components, and each are represented with different shapes and symbols *(Chen Notation & Crow’s Foot | Gleek Diagram Maker, n.d.)*.
* Components of Chen ERDs includes:
* **Entities:** entities are real-world objects or concepts. They can either be physical as well as non-physical.
  + In Chen ERD, entities are distinguished into three types – Strong entity, Weak entity, and Associative entity.
    - Strong entity are entities that does not dependent on any other entity and always have a primary key. It is represented by a single ‘rectangle’ shape *(GreeksforGreeks, 2023).*
    - Weak entity are entities that depend on a strong entity to ensure its existence. They don’t have a primary key and is represented by a double ‘rectangle’ shape *(GreeksforGreeks, 2023).*
    - Associative entity are entities that is used to define the relationship between entities, often in many-to-many relationships. It is represented as a diamond with a surrounding rectangle *(fuchs, 2021).*

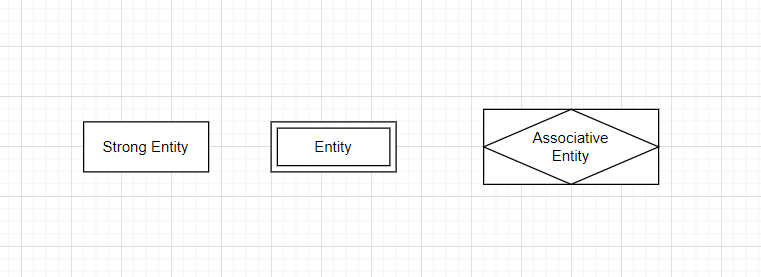


Figure 1Diagram representations of Strong Entity, Weak Entity and Associative Entity

**Attributes**:

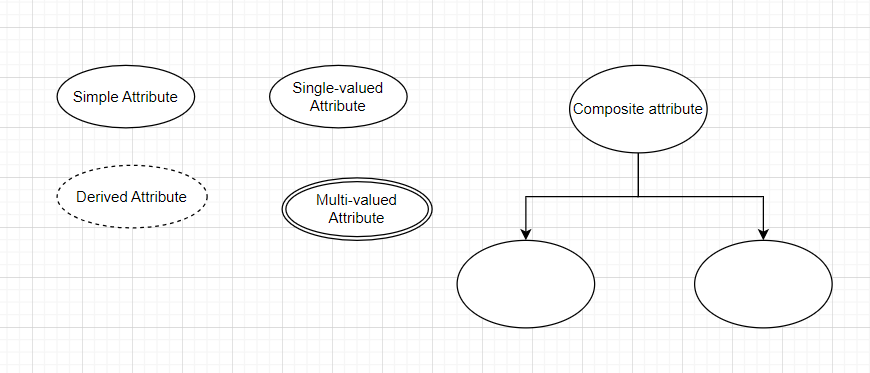
* Attributes are properties that describe the entity. They are represented with an ‘oval’ shape and are connected to its entity with a single line.
  + There are many types of attributes and each type have a different meaning in the context of a database. The attributes are categorized into 6 different attributes – Simple attribute, Composite attribute, Single valued attribute, Multi valued attribute, Derived attribute, and Stored attribute.
    - Simple attribute are atomic values that cannot be broken down further.
    - Composite attribute are attributes that can be further divided into smaller subparts.
    - Single valued attribute are attributes that hold just one value for particular entity.
    - Multi valued attribute are attributes that can store more than one value for an entity.
    - Derived attribute have its values derived from another attribute and is dependent on other attributes for its values *(Jain, 2022).*
    - Stored attribute are attributes that other attributes values are derived from.

Figure 2 Diagram representations of the different attributes

* Simple attribute and Single-valued attribute are both represented using the same shape ‘Oval/Ellipse’ shape.
* **Stored attribute** are attributes from which the values of other attributes are derived – something like ‘Date of Birth’ and the derived attribute would be ‘age’

**Relationships:**

* In Chen ERD, we can define the degree of a relationship into four different types - Binary, Ternary, n-array, quaternary and unary – also known recursive relationships.
  + Binary relationship refers to when two entities are involved and have a relationship.
  + In ternary relationship, there are three entities involved.
  + There can also be quaternary relationships when four entities are involved together in a relationship.
  + There is also an N-ary relationship when the number of entities involved in a relationship is many and is not define.
  + Unary relationships also known as ‘recursive relationship’ refers to when one entity plays more than one role in a relationship.

- This way of clarifying the relationships between entities helps with effective database design and management.

A diagram of a medicine

Description automatically generatedA diagram of a graph

Description automatically generated

Figure 3 Shows Ternary-Relationship

Figure 4 Shows Binary-Relationship

A diagram of a course

Description automatically generated

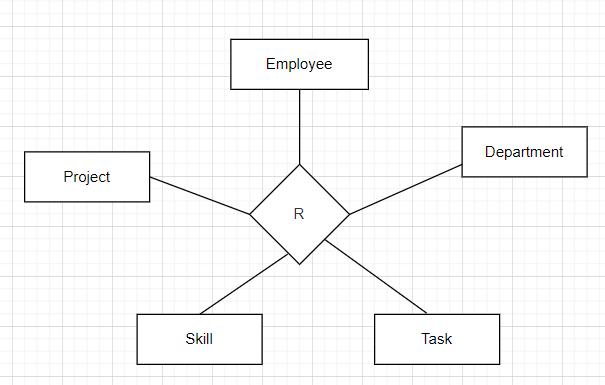


Figure 5 Shows Quaternary-Relationship

A diagram of a company

Description automatically generated

Figure 6 Shows N-ary Relationship

Figure 7 Shows Unary-Relationship

* To define the types of relationship between entities, there are four possible ways to identify in Chen ERD’s notation: one-to-one, one-to-many, many-to-one, and many-to-many relationships. It is also known

as ‘Cardinality’.

* In Chen ERD’s notation, the ‘one’ side of relationship is represented as ‘1’ while the ‘many’ said of the relationship is represented as ‘N’.
* There is also another widely used notation called ‘Crow’s foot notation’ and it uses two graphic symbols on relationship lines to represent cardinality.

A diagram of a graph

Description automatically generated

Figure 8 Diagrams: Types of Relationship

**Identifier/Key Attributes**

* Identifier attributes also known as key attributes are used to uniquely identify each entity and therefore is mandatory *(Relationaldbdesign, n.d.)*. Key attributes in CHEN’s ERD are represented by underlining the attributes. In database design, the key attributes are known as the primary key – which can be either combination of attributes set as unique identifier, or a specific attribute chosen.
* There can be more than one identifier in each entity -such as candidate key, super key, composite key, and foreign key.
  + Super key refers to an attribute or set attributes that uniquely identify an entity within a table.
  + Candidate key is a set of attributes that uniquely identified a table. A table can have multiple candidate keys.
  + Composite key is a candidate key that contains two or more attributes and uniquely identifies any record.
  + A Foreign key refers to an attribute or set of attributes that references the primary key in another table.

Extended Components (EER)

* Enhanced entity-relationship diagram (EER) is an extended version of entity-relationship diagram (ERD). Enhanced entity-relationship diagram is a high-level model and is used to represent complexities in complex databases *(Lucidchart, n.d.).*
* **Subclass and super class**
  + Super class is an entity that can be divided into further smaller subgroup while subclass are entity or instances of the super class, that inherited the properties and attribute of the super class (geeksforgeeks, 2024a)*.*

A diagram of a dog and animal

Description automatically generated

Figure 9 Superclass and Subclasses

* **Specialization and Generalization**
  + Specialization is the process of breaking down an entity into a smaller subset that share characteristics of the main entity.
  + Generalization refers to the process of identifying common features of several entities and grouping them into a single entity – a super class. This process of generalization is identifying the single parent entity of several entities or children (geeksforgeeks, 2024a)*.*

A diagram of a car

Description automatically generatedA diagram of a vehicle

Description automatically generated

Figure 11 Specialization

* **Categorization**

Figure 10 Generalization

* + Categorization also known as Union is when a subset has relationship with more than one super class *(Barnes, 2023).*

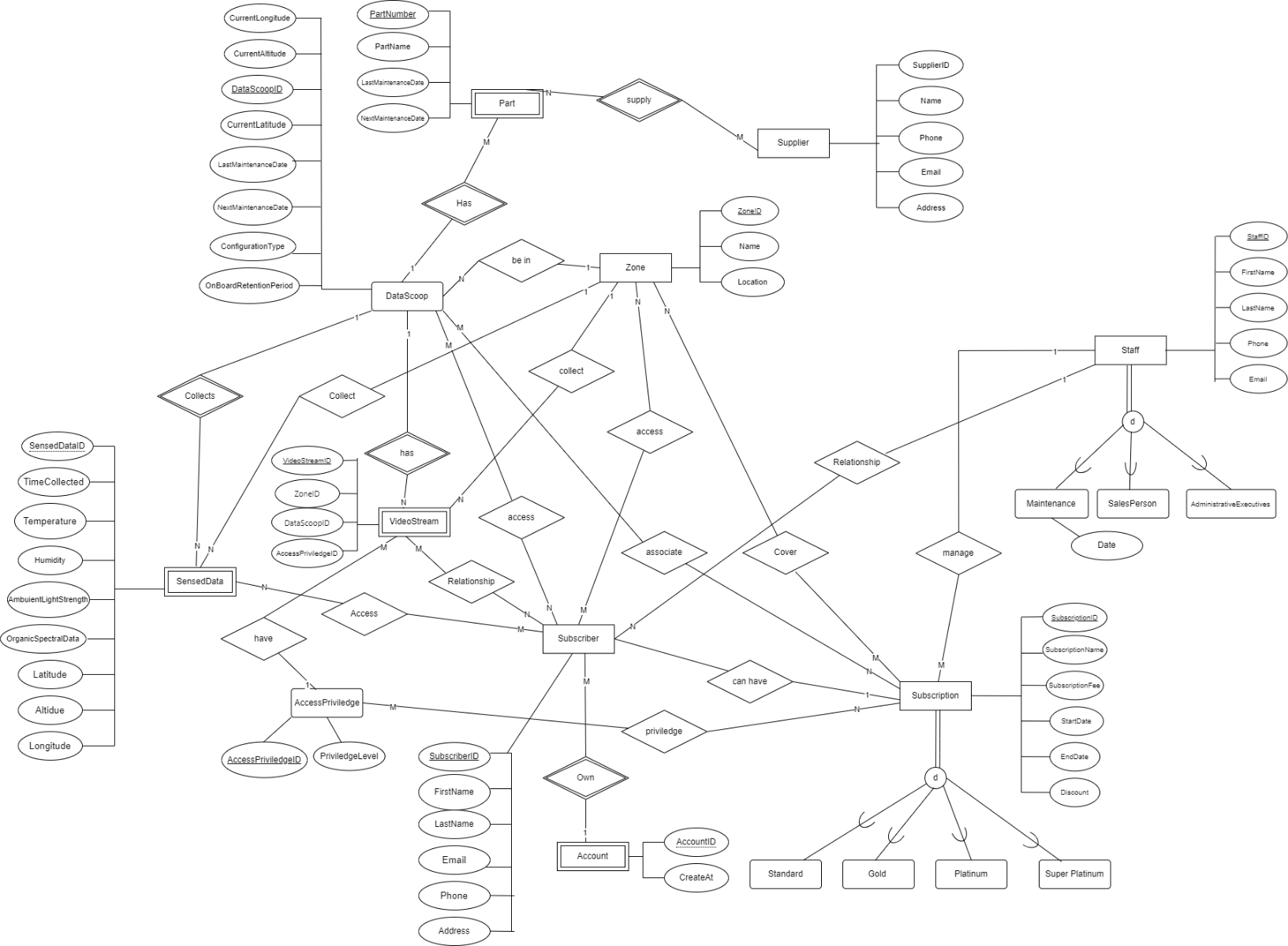
A diagram of a device

Description automatically generated

Figure 12 Categorization (UNION TYPES)

* **Attribute and Relationship Inheritance**
  + Enhanced entity-relationship diagram (EER) allows sub-classes to inherited attributes and its values from the superclass that it related to. *(Wikipedia, 2024).*
  + Relationship inheritance refers to a subclass entity inheriting the relationships that superclass participates in *(Wikipedia, 2024).*

## A Conceptual Entity-Relationship Diagram



* The purpose of this conceptual model is to represent the identified entities, attributes and the relationships involved from the data dictionary into a clearer and a better formatted version. This conceptual model uses Chen entity-relationship diagram components to shows the entities, attributes, and the relationship.
* This conceptual model used a rectangle-shape to represent what each entity are in terms of whether the entity is a weak or strong entity.   
  The different types of attributes are also shown in this model such as the primary-key attribute of the entity by underlying the attributes.
* The relationship or cardinality between the entities in this conceptual model are represent using the diamond-shape and straight line which are connected to the specific entities.
* This steps of creating a conceptual model allows for further discovering of any missed errors and anomalies data for the database.  
  It also helps deepen the understanding of what each data are and other important information.
* The conceptual model further ensures that the data are refined and are high quality as well as being consistent across for the whole database. These apply the principle of Integrity, Transparency and Protection of the fundamental principles of Data Management.

## Data Dictionary

### Table 1: Document Entities

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Name** | **Description** | **Aliases** | **Occurrence** |
| DataScoop | Drones deploys by Flight Stream |  | Is deploy as needed by the geographic coverage requirements of the contracts. |
| Subscription | Subscription in the form of contract | Contract | Is created when a subscriber purchases a subscription for services. |
| Part | Parts of the drones |  | Data scoop’s physical parts |
| Zone (region) | Region where the datascoop can be found |  | Is define when a new geographic region is added to the lists for datascoop to collect data. |
| Subscriber | Who subscribe to DS subscription |  | Is created when a person or an organisation brought a subscription from FlightStream company. |
| Staff | The employee of the Fight Stream |  | Is created for existing and new member of FlightStream organisation’s employee/staff. |
| Supplier | From which the drone’s parts is supply |  | Is created for existing and new supplier that supply parts for Datascoop. |
| SensedData | Data senses by each DataScoop |  | Is created for every data that is collected by each Datascoop.   * Is a weak entity dependent on Datascoop. |
| VideoStream | Viewing of live 3D video from datascoop. data from the video stream. |  | created when datascoop enable live 3d video. |
| Account | Account owns by subscriber. |  | Created every time there is a new subscriber. |

### Table 2: Document Relationship

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Entity Name** | **Cardinality** | **Participation** | **Relationship** | **Participation** | **Cardinality** | **Entity Name** |
| DataScoop | 1 | Partial | collects | Full | n | SensedData |
| DataScoop | m | Partial | allocate | Partial | n | Contract |
| DataScoop | 1 | Partial | cover | Partial | n | Zone |
| DataScoop | 1 | Full | Made up | Full | n | Part |
| Subscription | n | Full | have | Full | m | Subscriber |
| Subscription | n | Full | Cover zone | Full | m | Zone |
| Staff | 1 | Partial | Sell many | Partial | n | Contract |
| Zone | n | Full | Can have | Full | m | Subscriber |
| Supplier | n | Full | Supply many | Full | m | Part |
| Staff | 1 | Partial | manages | Partial | n | Subscriber |
| DataScoop | n | full | access | full | m | Subscriber |
| SensedData | N | Full | own | Full | M | Subscriber |
| Subscriber | 1 | Full | have | Full | 1 | Account |
| Subscriber | N | Full | Can own | partial | M | VideoStream |
| Zone | 1 | Full | Associate with | Partial | N | VideoStream |
| Datascoop | 1 | Full | provide | Partial | N | VideoStream |
| SensedData | N | Partial | Be collect in | Full | 1 | Zone |

### Table 3: Document Attributes

| **Entity / Relationship Name** | **Attributes** | **Description** | **Domain** | **Aliases** | **Composite** | **Derived** | **Nulls** | **Key?** | **Default Value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DataScoop | DataScoopID | A unique id for every datascoop | Integer generated for every datascoop |  |  |  | NOT NULL | YES | Auto-generated number - 1 |
|  | CurrentAltitude | Current location altitude of the datascoop | Decimal: to represent the current altitude. |  |  |  | NOT NULL |  | 0 |
|  | CurrentLongitude | Current location longitude of the datascoop | Decimal: to represent the current latitude |  |  |  | NOT NULL |  | 0 |
|  | CurrentLatitude | Current location latitude of the datascoop | Decimal : to represent the current latitude |  |  |  | NOT NULL |  | 0 |
|  | LastMaintenanceDate | The date the datascoop was last maintained | DATE |  |  |  | NOT NULL |  | --/--/---- |
|  | NextMaintenanceDate | The next date for the datascoop to be check in | DATE |  |  |  | NOT NULL |  | --/--/---- |
|  | ConfigurationType | The type of configuration that the datascoop can be configured. | One of Enum type of configuration. |  |  |  | NOT NULL |  | First Enum value |
|  | OnBoardRetentionPeriod | Number of days data is stored on board in Datascoop | Integer: days |  |  |  | NOT NULL |  | 0 |
|  |  |  |  |  |  |  |  |  |  |
| SensedData | SensedDataID | A unique id to refer to the data sensed by datascoop. | Auto generated. |  |  |  | NOT NULL | YES 1 | |
|  | DataScoopID | Refer to the Datascoop entity. | Auto generated. |  |  |  | NOT NULL | 1 | |
|  | TimeCollected | The exact time of when the data was collected. | Timestamp |  |  |  | NOT NULL | YYYY-MM-DD HH:MI:SS | |
|  | Temperature | The temperature data that datascoop collects. | Decimal Data type |  |  |  | NOT NULL | 0 | |
|  | Humidity | The humidity data that datascoop collect. | Float datatype |  |  |  | NOT NULL | 0 | |
|  | AmbuientLightStrength | The data sensed by datascoop. | Decimal datatype |  |  |  | NOT NULL | 0 | |
|  | OrganicSpectralData | The data sensed by datascoop. | Decimal datatype |  |  |  | NOT NULL | 0 | |
|  | Latitude | Refers to the latitude of when the data was collected | Decimal |  |  |  | NOT NULL | 0 | |
|  | Longitude | Refers to the longitude of when the data was collected. | Decimal |  |  |  | NOT NULL | 0 | |
|  | Altitude | Refers to the longitude of when the data was collected. | Decimal |  |  |  | NOT NULL | 0 | |
| Subscription  (is a base class for subsclassess of subscription type) | SubscriptionID | A unique id for every contract. | Integer |  |  |  | NOT NULL | YES 1 | |
|  | SubscriptionName | The name of the subscription. | Enum String |  |  |  | NOT NULL |  | |
|  | StartDate | The date the contract begins. | Date |  |  |  | NOT NULL |  | |
|  | EndDate | The date the contract end. | Date |  |  |  | NOT NULL |  | |
|  | SubscriptionFee | The price value of the specific subscription . | Decimal |  |  |  | NOT NULL |  | |
|  | Discount | Percentage given up to 3% | decimal |  |  |  | NOT NULL |  | |
|  |  |  |  |  |  |  |  |  | |
| SubscriptionZone  (is a base class for subclasses of type of subscription) | SubscriptionID | Foreign key in ContractZone entity | Integer |  | yes |  | NOT NULL | Auto-generated | |
|  | ZoneID | Foreign key in ContractZone entity | Integer |  | yes |  | NOT NULL | Auto-generated | |
| SubscriptionSubscriber | SubscriptionID | Foreign key reference | Integer |  | yes |  | NOT NULL | Auto-generated | |
|  | SubscriberID | Foreign key reference | Integer |  | yes |  | NOT NULL | Auto-generated | |
|  |  |  |  |  |  |  |  |  | |
| StaffSubscription | SubscriptionID | Foreign key reference | Integer |  | yes |  | NOT NULL | Auto-generated | |
|  | StaffID | Foreign key reference | Integer |  | yes |  | NOT NULL | Auto-generated | |
|  |  |  |  |  |  |  |  |  | |
| Subscriber | SubscriberID | Unique id for every subscriber | Integer |  |  |  | NOT NULL | YES Auto-generate | |
|  | FirstName | First name of subscriber | String |  |  |  | NOT NULL |  | |
|  | LastName | Last name of subscriber | String |  |  |  | NOT NULL |  | |
|  | Email | Email address of subscriber | String |  |  |  | NOT NULL |  | |
|  | Phone | Phone number of subscriber. | Integer |  |  |  |  |  | |
|  | Address | Current residency address of subscriber | String |  |  |  | NOT NULL |  | |
|  |  |  |  |  |  |  |  |  | |
| Staff  (this one is a base class and has subclasses) | StaffID | Unique id for every staff | integer |  |  |  | NOT NULL | YES auto-generate | |
|  | FirstName | First name of staff | String |  |  |  | NOT NULL |  | |
|  | LastName | Last name of staff | String |  |  |  | NOT NULL |  | |
|  | Email | Email address of the staff | String character |  |  |  | NOT NULL |  | |
|  | Phone | Phone number of the staff | integer |  |  |  |  |  | |
|  | Role | Type of role | String |  |  |  | NOT NULL |  | |
|  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  | |
| Zone | ZoneID | Unique id for each zone | Integer |  |  |  | NOT NULL | YES auto-generate | |
|  | Name | Name of the zone | String |  |  |  | NOT NULL |  | |
|  | Location | The location or coordinates of the zone | String |  |  |  | NOT NULL |  | |
|  |  |  |  |  |  |  |  |  | |
| Supplier | SupplierID | Unique id for each supplier | Integer |  |  |  | NOT NULL | YES auto generate | |
|  | SupplierName | The name of the supplier | String |  |  |  | NOT NULL |  | |
|  | Phone | The phone number of the supplier | Integer |  |  |  | NOT NULL |  | |
|  | Email | The email address of the supplier | String |  |  |  | NOT NULL |  | |
|  | Address | The location of the supplier |  |  |  |  | NOT NULL |  | |
|  |  |  |  |  |  |  |  |  | |
| Part | PartNumber | Unique number for each part by manufacturer. | String |  |  |  | Not Null |  | |
|  | PartName | The name of the part. | String |  |  |  | Not null |  | |
|  | LastMaintenanceDate | The last date time it was maintenance | Date |  |  |  | Not null |  | |
|  | NextMaintenanceDate | The next date due for maintenance | Date |  |  |  | Not null |  | |
|  |  |  |  |  |  |  |  |  | |
| PartSupplier | PartID | Foreign key to part table | Integer |  | yes |  | Not null | Auto generate | |
|  | SupplierID | Foreign key to supplier table | Integer |  | yes |  | Not null | Auto generate | |
|  |  |  |  |  |  |  |  |  | |
| SubscriptionDataScoop | SubscriptionID | Foreign key reference | Integer |  | Yes |  | Not null |  | |
|  | DataScoopID | Foreign key reference | Integer |  | Yes |  | Not null |  | |
| Account | AccountID | Unique number for each account number. is weak-entity, dependent on ‘Subscriber’ | Integer |  |  |  | Not null | Yes auto-generate | |
|  | CreatedAt | The date the account was created | Date |  |  |  | Not null |  | |
| VideoStream | VideoStreamID | Unique identifier for every video stream. |  |  |  |  | Not null | Yes auto-generate | |
|  | AccessPriviledge | Define the level of access each level of contract allow. | Enum |  |  |  | NOT NULL | One of the first value of enum. | |
|  | ZoneID | Foreign key reference |  |  |  |  | NOT NULL |  | |
|  | DatascoopID | Foreign key reference |  |  |  |  | NOT NULL |  | |
| VideostreamSubscriber | videostreamID | Reference to the foreign table ‘VideoStream’  Junction table | Integer |  |  |  | NOT NULL |  | |
|  | SubscriberID | Reference to the foreign table ‘Subscriber’  Define junction table | Integer |  |  |  | NOT NULL |  | |

Purpose of all parts of the model

* In this Conceptual ER model, there are three main components -Entity, Attribute and Cardinality (or relationship).
* This conceptual model has 10 total identified entities – Datascoop, Contract, Part, Zone, Subscriber, Staff, Supplier, SensedData and VideoStream and Account.
* **DataScoop entity** represent each drone that is used for data collecting and to provide live 3D video stream. It has attributes like ‘DataScoopID’, ‘CurrentAltitude’, ‘CurrentLongitude’, ‘CurrentLatitude’, ‘LastMaintenanceDate’ , ‘NextMaintenanceDate’ and ‘OnBoardRetentionPeriod’. These attributes provide important information that is need for managing this entity.

DataScoop entity has relationship with ‘SensedData’, ‘Contract’, ‘Zone’, ‘Part’, ‘Subscriber’ and ‘VideoStream’ entities.

* **Subscription entity** represent a subscription contract between the subscriber and the FlightStream organization. It has attributes like ‘ SubscriptionID’, ‘SubscriptionName, ‘StartData’, ‘EndDate’, ‘SubscriptionFee’, ‘Discount’ .These attributes are important and crucial to for Contract entity to have.  
  This entity has relationship with ‘Zone’, ‘Subscriber’, ‘Staff’ and ‘DataScoop’ entities.

**(Subscription is a parent class for every type of subscription available)**

* **Part entity** represent each physical component that made up the drone/DataScoop. It has several attributes – ‘PartNumber’, ‘PartName’, ‘LastMaintenanceDate’ , and ‘NextMaintenanceDate’.  
  These attributes contain values that contains important information relating to this ‘Part’ entity.  
  Part entity has relationship with ‘DataScoop’ and ‘Supplier’ entities.
* **Zone entity** represents a geographic region where the FlightStream company released DataScoop to collect data.  
  It has attributes like – ‘ZoneID’, ‘Name’ and ‘Location’ that provides information regarding to this entity.  
  Zone entity has relationship with ‘DataScoop’, ‘Contract’, ‘Subscriber’, ‘VideoStream’ and ‘SensedData’.
* **Subscriber entity** represent subscriber who brought subscription/contract from FlightStream company. It has many attributes - ‘SubscriberID’, ‘FirstName’, ‘LastName’, ‘Email’, ‘Phone’, ‘Address’ and SubscriptionType’.  
  These attributes provide crucial information about Subscriber entity.  
  Subscriber entity has relationship with ‘Contract’, ‘Zone’, ‘Staff’, ‘DataScoop’, ‘SensedData’, ‘Account’ and ‘VideoStream’ entities.
* **Staff entity** represent employee members of the FlightStream organization. It has attributes like – ‘StaffID’, ‘FirstName’, ‘LastName’, ‘Email’, ‘Phone’.  
  These attributes provide information about the staff entity.

Staff entity has relationships with ‘Contract’ and ‘Subscriber’ entities.

**(Staff is a parent/base class for every role available in the company subclasses)**

* **Supplier entity** represents supplier where the FlightStream organization get their drone part supply from. This entity has attributes including – ‘SupplierID’, ‘SupplierName’, ‘Phone’, ‘Email’ and ‘Address’. These attributes provide information regarding the Supplier entity.

Supplier entity has relationship with ‘Part’ entity.

* **SensedData entity** represents data that are sensed and collected by the DataScoop. It has several attributes – ‘SensedDataID’, ‘DataScoopID’, ‘TimeCollected’, ‘Temperature’, ‘Humidity’, ‘AmbuientLightStrength’, ‘OrganicSpectralData’, ‘Latitude’, ‘Longitude’ and ‘Altitude’.

These attributes include the types of data the DataScoop can sensed and collected as well as other crucial information.

SensedData entity has relationships with ‘DataScoop’, ‘Subscriber’ and ‘Zone’ entities.

* **VideoStream entity** represents the data of viewing of live 3D video from DataScoop.  
  It has attributes like –‘VideoStreamID’ and ‘AccessPriviledge’. These attributes contain important information regarding the VideStream entity.  
  VideoStream entity has relationships with ‘Subscriber’, ‘Zone’ and ‘DataScoop’ entities.
* **Account entity** represents an account owns by subscriber. It has ‘AccountID’ and ‘CreatedAt’ attributes. These two attributes are crucial to Account entity.

Account entity has relationships with ‘Subscriber’ entity.

* Data dictionaries provide a collection of data entities, attributes and relationships which are analysed and structured in a way that can be effectively used in a database.  
  It helps to understand the overall design and structure of a database before attempting to create an actual database.  
  This allows errors and anomalies in data to be discovered earlier which makes the whole process of creating a database much faster and efficient. It also assists with data consistency and security as well as managing quality of data *(Atlan, 2023).*

## Business Assumptions

**Assumptions:** Each DataScoop can be associate with one or more Contract – this would result in a one-to-many relationship.  
However, the project brief states that *“For contracts that have overlapping regions a DataScoop may be allocated to more than one contract.”*  
This implies that the relationship between DataScoop and Contract is a many-to-many relationship instead.  
Identifying the correct relationship between the entities ensure data integrity.

**Assumptions:** A Staff member can manage many Contracts. However, as the project brief states “*FlightStream employs staff who can take an Administrative Executive role. Only staff who have this role may change the price of subscriptions. Only an Administrative Executive may enter contract details.”*   
While the assumption is correct, this only apply to a staff member whose role is Administrative Executive.  
This adds constraints of who has access to what specific information in the database, ensuring the Principle of Accountability, Protection and Compliance.

**Assumptions:** Each DataScoop having to keep record of maintenance. The project brief states *“FlightStream is required to keep maintenance records for each DataScoop. Over time there is a record of the maintenance of each DataScoop.”*The DataScoop entity has attributes ‘LastMaintenanceDate’ and ‘NextMaintenanceDate’ which stored the date the maintenance checking was done and need to be done.   
This information allows for easier to track and manage the Datascoop and applying the Principle of Retention, Disposition and Availability of data management.

# Conceptual to Relational Logical model

## Transform to logical model

* To transform a conceptual model based on CHEN ERDs to a logical model, it involves understanding and reviewing the conceptual model as logical model stage is taking the conceptual model into the more details.
* Logical model is the blueprint of the database and represents what data will contains and how it will be structured by defining the entity types, attributes, data types, relationships between the entities and the constraints – data model Entity-Relationship Diagram (ERD) – more specifically ‘Crow’s Foot notation’ is used to visualize, which helps understand the data much clearer.

**Mapping Rules**

* Entities to Table
  + The entity name from the conceptual model should be converted to the name of a table in the logical model.

A yellow box with black text

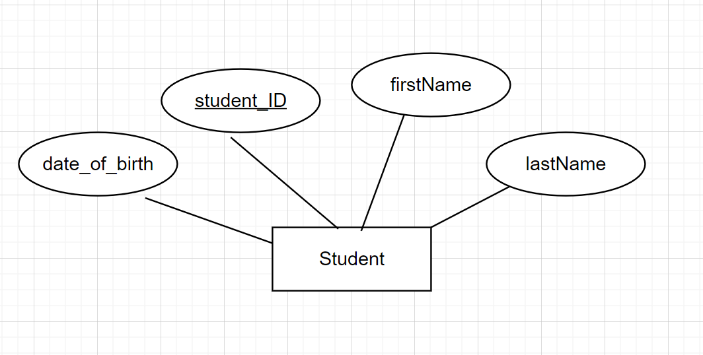
Description automatically generatedA diagram of a student

Description automatically generated

Figure 1 conceptual model

Figure 2 logical model

Figure 3logical model

* Attributes to columns
  + Each attribute of an entity should be listed out in the column of the table with the types of data define along with it.
  + A yellow box with black text

    Description automatically generatedExample: the attributes are - student\_ID, firstName, lastName and data\_of\_birth with the data types defined and the maximum storage data space required of the specific data type.

Figure 4 The logical model defines the data types as well as the maximum storage space the data can use.

Figure 5 The conceptual model does not define the data types nor the storage space requirement.

* Primary Key
  + An attribute that is define as the key attribute from the conceptual model should acts as the primary key in the logical model to uniquely identified each table.

A diagram of a student

Description automatically generated

Figure 6 The primary key or knows as the key attribute in conceptual model is circle in red.

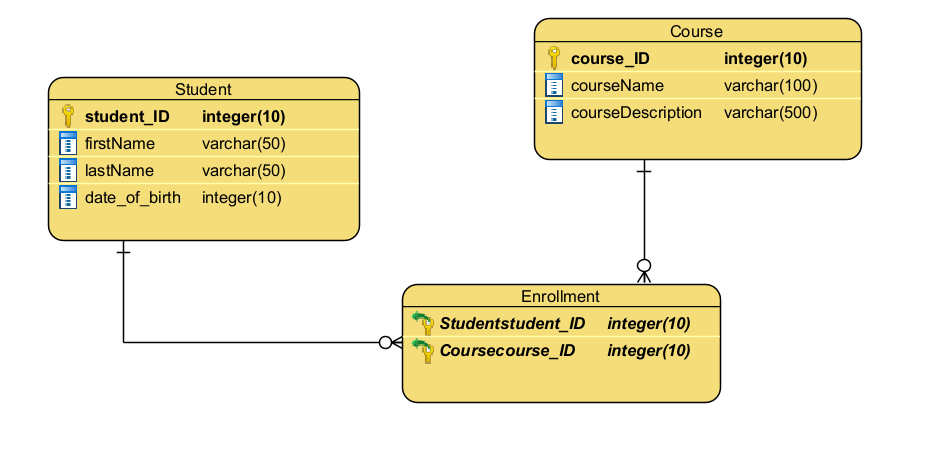
A yellow box with black text

Description automatically generated



Figure 7 The ‘Primary Key’ in logical model has a key icon next to it and the text is bold.

* Relationship between entities
  + Relationships in conceptual model are represented as one-to-one, many-to-many, many-to-one relationships.
  + The representation of relationships between entities works by referencing a primary key of a table in another table -which is now known as a foreign key and therefore establishing a relationship between the two entities.

A diagram of a network

Description automatically generatedExample: This conceptual model here represents the relationship between ‘Course’ and ‘Student’ entity in a many-to-many relationship.

Figure 8 The logical model shows the many-to-many relationship between the two entities ‘Student’ and ‘Course’ by implementing a junction table and used it to represent the many-to-many relationship.

Figure 9 The conceptual model shows the relationship between the two entities with the double-lined diamond in which it is an identifying relationship.  
The Student and Course have many-to-many relationship.

Constraints

* Transforming from conceptual model to logical model require in making sure that whether a specific attribute is ‘NULLABLE’ or ‘NOT NULLABLE’ (mandatory or optional) - this make sure that certain attributes are required to be present and are of essential data for the database.
* In logical model, we can ensure data integrity and performance by enforcing ‘Uniqueness’ and ‘Indexes’ in the attributes.
  + An ‘Index’ in a database speed up the process of querying. It is a method used to quickly look up a specific data *(codecademy, n.d.)*
  + A ‘unique’ key is a way to make sure that a specific column or an attribute does not have a repeated or duplicate value *(Fernigrini, 2022).*

## Logical Model

A black background with white rectangles

Description automatically generated

## Retionale for ERD

* Begin with reviewing the conceptual model that was done in Milestone One and recreating the diagram using Crow’s Foot notation with Visual Paradigm application tool.
* The next step involves identifying all the entities in the conceptual model and making sure that every single one of them is included in the logical model and each individual entity is noted as the name of the table.
* After putting down all the table’s name, the next steps was identifying the attributes of the ‘table/entity’.
  + Making sure that the attributes are correctly placed in the correct table as the column.
  + Ensuring that each table have a primary key identified and noted down.
  + The data types are correctly identified and placed next to the column.
  + Ensuring that the ‘Enum’ data types are represented as a reference table and link to the correct table.
* The relationship between the entities are reviewed and checked to ensure that the data are represented accurately as possible according to the needs of the organization.
  + Making sure that the foreign key are identified and are link to the correct table to establish the relationship correctly to ensure referential integrity.
  + Identifying the types of relationships between entities – one-to-many, many-to-many and many-to-many relationships.
    - Ensure that the many-to-many relationships are represented by using junction table -which contains the primary keys of the two involve tables.

**Example: “Subscriber” and “VideoStream” entity**

* Many “Subscriber” can access many “VideoStream”.
* Many “VideoStream” can be accessed by many “Subscribers”.

The relationship between the two table is many-to-many relationship and in logical model phase, this relationship is represented using a junction table which can be named “VideoStreamSubscriber” -that’ll reference back to the Subscriber and VideoStream by both primary keys.

## Normalization

### Description of Normalization

* Normalization is the process of ensuring and minimizing data redundancy for data relation. It defines a set of guidelines to follow in order to create the best design for database (geeksforgeeks, 2024b)*.*
* Normalization ensures that the three anomalies -insertion anomaly, update anomaly and deletion anomaly in database is reduce as much as possible.

### First Normal Form (1NF)

* The first normal form defines a set of three violations, which are

1. using the row order to convey information
2. mixing data types within the same column
3. not defining a primary key in a table

* Ensuring that the database design does not violate the three requirements would results in achieving the first safety assessment level of normalization.  
  The first normal form (1NF) eliminates duplicate data and simplifies query (geeksforgeeks, 2024b)*.*

### Second Normal Form (2NF)

* The second normal form defines a single violation:
  + each non-key attribute must depend on the entire primary key.
* The second normal form eliminates redundant data by having all non-key attributes functionally dependent on the entire primary key. To achieve the assessment level of 2NF, the database design must fully meet the First Normal Form (1NF) as well.

### Third Normal Form (3NF)

* The third normal form states that :
  + every non-key attribute in a table should depend on nothing else but the key itself and that there is not have any transitive dependency for non-prime attributes.
* The requirement to achieve the third normal form is by having achieve the goal of the second normal form(2NF). This level ensures a higher data integrity as well as data duplication *(A S, 2024).*

### Boyce-Codd Normal Form

* The Boyce Codd Normal Form is a more advanced version of the Third Normal Form (3NF) and is use solves anomalies which were not resolve with 3NF.
* Boyce Codd Normal Form (BCNF) is based on functional dependencies.
* The requirement to achieve BCNF is by first ensuring that till the Third Normal Form (3NF) is already achieved.
* Identifying all the determinants and make sure that they are candidate keys -meaning, each attribute (or set of attributes) that can determine another attribute must be able to uniquely identify a row in the table(geeksforgeeks, 2023a)*.*

To fully achieve Boyce-Codd Normal form, the identified table much be further broken down into more tables.

### Fourth Normal Form (4NF)

* The fourth normal form is the stage where there are no non-trivial multivalued dependencies other than a candidate key.
* To achieve the 4NF, the table must be in Boyce-Codd Normal Form and have already achieved all normal form from the first to the third.
* Fourth Normal Form eliminates unnecessary redundancies and contribution of inconsistent data *(geeksforgeeks, 2023b).*

### Normalization for FlightStream Organization

* **Subscription** (SubscriptionID, SubscriptionName, StartDate, EndDate, SubscriptionPrice, Discount, StaffStaffId\* )
* **Staff** (StaffID, FirstName, LastName, Email, Phone, Role)
* **Subscriber** ( SubscriberID, FirstName, LastName, Email, Phone, Address, StaffStaffID\*)
* **DataScoopSubscriber** ( DataScoopDataScoopID\* , SubscriberSubscriberID\* )
* **SubscriptionSubscriber**( SubscriptionSubscriptionID\* , SubscriberSubscriberID\* )
* **Account** (AccountID, CreatedAt, SubscriberSubscriberID\*)
* **DataScoop** (DataScoopID, CurrentAltitude, CurrentLongitude, CurrentLatitude, LastMaintenanceDate, NextMaintenanceDate, OnBoardRetentionPeriod, Configuration\_TypeTypeID\* )
* **Configuration\_Type** (TypeID, ConfigurationName, ParentConfigurationTypeID\* )
* **SubscriptionDataScoop** ( SubscriptionSubscriptionID \* , DataScoopDataScoopID\* )
* **Zone** ( ZoneID, Name, Location, DataScoopDataScoopID\* )
* **SubscriptionZone** (SubscriptionSubscriptionID \* , ZoneZoneID\* )
* **ZoneSubscriber** ( ZoneZoneID\* , SubscriberSubscriberID\* )
* **VideoStream** ( VideoStreamID, ZoneZoneID\* , DataScoopDataScoopID\* , AccessPriviledgeAccessPriviledgeID\* )
* **AccessPriviledge** ( AccessPriviledgeID, PriviledgeLevel)
* **VideoStreamSubscriber** (VideoStreamVideoStreamID\* , SubscriberSubscriberID\* )
* **Part** (PartNumber, PartName, LastMaintenanceDate, NextMaintenanceDate, DataScoopDataScoopID\* )
* **PartSupplier** ( PartPartNumber\* , SupplierSupplierID\* )
* **Supplier (**SupplierID, SupplierName, Phone, Email, Address)
* **SensedData** ( SensedDataID, TimeCollected, Temperature, Humidity, AmbuientLightStrength, OrganicSpectralData, Latitude, Longitude, Altitude, DataScoopDataScoopID \*)
* **ZoneSensedData** ( ZoneZoneID\* , SensedDataSensedDataID\* )
* **SensedDataSubscriber** ( SensedDataSensedDataID\* , SubscriberSubscriberID\* )
* The tables and columns are reviewed and taken from the conceptual model which was done back in Milestone One.  
  In this phase of logical model, it involves ensuring that the data is check for normalization and the relationships are correctly identified.
* Each identified tables have a primary key to ensure unique identification of the data which follow the guidelines to achieving the First Normal Form (1NF).  
  After achieving the 1NF, identified that all non-key attributes are functionally dependent on the entire primary key.
* Making sure that the database tables are in Second Normal Form (2NF) and all attributes are fully functionally dependent on the primary key to ensure the Third Normal Form rule is followed and as per as the guideline to achieve it. This involves following the rules for the remaining Boyce-Codd Normal Form and the Fourth Normal Form to create the best database design.

## Data Dictionary

### Document Relations

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation Name** | **Start Volume** No. of rows loaded at the beginning | **Growth** e.g. no growth / 10% per year | **Comments** |
| Subscription | 2000 | 10% | Based on the starting contract/subscription of existing 2000 |
| Staff | 150 | 5% | Estimation of staff employ by the organization and expect grow |
| Subscriber | 500,000 | 20% | Existing 500,000 subscriber initially with growth expect |
| DataScoopSubscriber | 500,000 | 20% | Match subscriber count and growth along with it |
| SubscriptionSubscriber | 2000 | 10% | match with Contract count initial and grow along |
| Account | 500,000 | 20% | Match with the number of subscribers and grow along with subscriber as increase |
| DataScoop | 50000 | 5% | Based on over 50000 DataScoop existing and expect growth as time |
| Configuration\_Type | 4 (+ 5) | No growth | Fixed of the existing types of configurations |
| SubscriptionDataScoop | 50000 | 5% | Match with the number count of the DataScoop and expect grows as DataScoop deploy increase |
| Zone | 100 | 10% | Based on over 100 existing zones and growth is expect in over time |
| SubscriptionZone | 2000 | 10% | Match with contract count and expect growth as the contracts sign increase and more subscriber |
| ZoneSubscriber | 500,000 | 20% | Initial is set to the count of subscriber and increase as the subscriber increase |
| VideoStream | 100 | 5% | Based on existing VideoStream of 100 and expect grow in over time. |
| AccessPriviledge | 4 | No growth | Based on the initial levels of subscription level set |
| VideoStreamSubscriber | 500,000 | 20% | Match with number of subscriber and is expected to increase as subscriber increase |
| Part | 1000 | 10% | Estimation of part with expect growth in time |
| PartSupplier | 2000 | 5% | Estimating initial supplier relationships with moderate growth |
| Supplier | 50 | 5% | initial supplier count set in 50 with moderate growth |
| SensedData | 500,000 \* 2 | 50% | Match with the double amount of DataScoop count initial as it is data sensed collected by the DataScoop, has high growth |
| ZoneSensedData | 100 | 30% | Match with zone initial count and is dependent on the contracts |
| SensedDataSubscriber | 500,000 | 20% | Initial is set to number of subscriber and growth is expected to increase as the number of subscriber increase |

### Document Attributes

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Subscription**  **(is a base class for subclasses of subscription types)** | SubscriptionID | Unique number for each contract | integer | 10 |  | unique, auto-increment | From 1 | NO | Yes | no | uniqueness |
|  | StartDate | The start date of the contract | date |  |  |  |  | NO |  |  | N/A |
|  | EndDate | End date of the contract | date |  |  |  |  | NO |  |  | N/A |
|  | SubscriptionPrice | The price for the subscription | decimal | (8, 2) |  |  |  | NO |  |  | N/A |
|  | Discount | Boolean value of whether the discount is allow or not | bit |  |  |  |  | NO |  |  |  |
|  | SubscriptionName | The percentage of the discount – allowed up to 3% | decimal | (4, 2) |  |  |  | NO |  |  |  |
|  | Staff.StaffID | Foreign key reference to the Staff table | interger | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Staff  (is a base class for subclasses of staff types) | StaffID | Unique identifier for each staff | integer | 10 |  | Unique, auto-increment |  | NO | YES |  | uniqueness |
|  | FirstName | The first name of the staff | varchar | 50 |  |  |  | NO |  |  |  |
|  | LastName | Last name of the staff | varchar | 50 |  |  |  | NO |  |  |  |
|  | Email | Email address of the staff member | varchar | 100 |  |  |  | NO |  |  | uniqueness |
|  | Phone | Phone number of the staff | varchar | 15 |  |  |  | NO |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Subscriber | SubscriberID | Unique identifier for each subscriber | integer | 10 |  | Auto-increment, unique |  | NO | yes |  | UNIQUE |
|  | FirstName | The first name of the subscriber | varchar | 50 |  |  |  | NO |  |  |  |
|  | LastName | The last name of the subscriber | varchar | 50 |  |  |  | NO |  |  |  |
|  | Email | Email address of the subscriber | varchar | 100 |  |  |  | NO |  |  | UNIQUE |
|  | Phone | Phone number of the subscriber | varchar | 15 |  |  |  | YES |  |  |  |
|  | Address | The physical address of the subscriber | varchar | 255 |  |  |  | NO |  |  |  |
|  | SubscriptionType | The current type of subscription the subscriber subscribed to | varchar | 50 |  |  |  | NO |  |  |  |
|  | Staff.StaffID | Foreign key reference link to the Staff table | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| DataScoopSubscriber | DataScoop.DataScoopID | Foreign key reference to DataScoop table  junction | integer | 10 |  |  |  | NO | YES | YES | UNIQUE |
|  | Subscriber.SubscriberID | Foreign key reference to Subscriber table.  Junction table | integer | 10 |  |  |  | NO | YES | YES | UNIQUE |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Account | AccountID | Unique identifier for each account created | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | uniqu |
|  | CreatedAt | The datatime the account was created | timestamp |  |  |  |  | NO |  |  |  |
|  | Subscriber.SubscriberID | Foreign key reference to Subscriber table | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| DataScoop | DataScoopID | Number to identify each DataScoop uniquely | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | unique |
|  | CurrentAltitude | The current altitude of the datascoop | decimal | (10, 2) |  |  |  | NO |  |  |  |
|  | CurrentLongitude | Current longitude f the datascoop | decimal | (10, 6) |  |  |  | NO |  |  |  |
|  | CurrentLatitude | Current latitude of the datascoop | decimal | (10, 6) |  |  |  | NO |  |  |  |
|  | LastMaintenanceDate | The date of last maintenanc | date |  |  |  |  | NO |  |  |  |
|  | NextMaintenanceDate | The next date for datascoop maintenance | date |  |  |  |  | NO |  |  |  |
|  | OnBoardRetentionPeriod | The day period in which the datascoop can retain data | integer | 10 |  |  |  | NO |  |  |  |
|  | Configuration\_Type.TypeID | Foreign key reference to Configuration\_Type table. It is actually an Enum but is represent this way | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Configuration\_Type | TypeID | Unique identifier for each configuration type | integer | 10 |  |  |  | NO | YES |  | unique |
|  | ConfigurationName | The name of the configuration | varchar | 30 |  |  |  | NO |  |  |  |
|  | ParentConfigurationTypeID | It is a recursive attribute. | integer | 10 |  |  |  | NO |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| SubscriptionDataScoop | Subscription.SubscriptionID | Reference to Contract table.  Foreign key reference | integer | 10 |  |  |  | NO | YES | YES | UNIQUE |
|  | DataScoop.DataScoopID | DataScoop table foreign key reference | integer | 10 |  |  |  | NO | YES | YES | UNIQUE |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Zone | ZoneID | Unique identifier for each zone | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | unique |
|  | Name | The name of the zone | varchar | 100 |  |  |  | NO |  |  |  |
|  | Location | The location of the zone | varchar | 255 |  |  |  | NO |  |  |  |
|  | DataScoop.DataScoopID | Foreign key reference to DataScoop table | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| SubscriptionZone | Subscription.SubscriptionID | Foreign key reference to Contract table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  | Zone.ZoneID | Foreign key reference to Zone table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  |  |  |  |  |  |  |  |  |  |  |  |
| ZoneSubscriber | Zone.ZoneID | Foreign key reference to zone table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  | Subscriber.SubscriberID | Foreign key reference to Subscriber table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  |  |  |  |  |  |  |  |  |  |  |  |
| VideoStream | VideoStreamID | Unique identifier for each VideoStream | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | UNIQUE |
|  | Zone.ZoneID | Foreign key reference to Zone table | integer | 10 |  |  |  | NO |  | YES |  |
|  | DataScoop.DataScoopID | Foreign key reference to DataScoop table | integer | 10 |  |  |  | NO |  | YES |  |
|  | AccessPriviledge.AccessPriviledgeID | Foreign key reference to AccessPriviledge table | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| AccessPriviledge | AccessPriviledgeID | Unique identifier for each level access of the priviledge | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | unique |
|  | PriviledgeLevel | The level of priviledge access | varchar | 80 |  |  |  | NO |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| VideoStreamSubscriber | VideoStream.VideoStreamID | Foreign key reference to VideoStream table | integer | 10 |  | unique |  | NO | YES | YES | unique |
|  | Subscriber.SubscriberID | Foreign key reference to Subscriber table | integer | 10 |  | unique |  | NO | YES | YES | unique |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Part | PartNumber | Unique identifier to uniquely identified each part | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | unique |
|  | PartName | The name of the part | varchar | 50 |  |  |  | NO |  |  |  |
|  | LastMaintenanceDate | The date of the last maintenance the part was check | date |  |  |  |  | NO |  |  |  |
|  | NextMaintenanceDate | The next due date of maintenance to be done | date |  |  |  |  | NO |  |  |  |
|  | DataScoop.DataScoopID | Foreign key reference to DataScoop table | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| PartSupplier | Part.PartNumber | Foreign key reference to Part table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  | Supplier.SupplierID | Foreign key reference to Supplier table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Supplier | SupplierID | To be able to uniquely identify each supplier | integer | 10 |  | Auto-increment, unique |  | NO | YES |  | unique |
|  | SupplierName | The name of the suppliername | varchar | 100 |  |  |  | NO |  |  |  |
|  | Phone | Contact phone of the supplier | varchar | 50 |  |  |  | NO |  |  |  |
|  | Email | The email address of the supplier | varchar | 100 |  |  |  | NO |  |  |  |
|  | Address | The address of the supplier | varchar | 255 |  |  |  | NO |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| SensedData | SensedDataID | Unique identifier to keep track of each sensed data by the DataScoop | integer | 10 |  | Auto-increment, |  | NO | YES |  | unique |
|  | TimeCollected, | The time the sensedData was collect | timestamp |  |  |  |  | NO |  |  |  |
|  | Temperature | The temperature recorded at the time of data collection | decimal | (6, 2) |  |  |  | NO |  |  |  |
|  | Humidity | The humidity level recorded at the time of data collection. | decimal | (5, 2) |  |  |  | NO |  |  |  |
|  | AmbuientLightStrength | The strength of the ambient light recorded at the time of data collection. | decimal | (15, 5) |  |  |  | NO |  |  |  |
|  | OrganicSpectralData | Data related to the organic spectral measurements recorded | Decimal | (15, 5) |  |  |  | NO |  |  |  |
|  | Latitude | The latitude coordinate where the data was collected | Decimal | (10, 6) |  |  |  | NO |  |  |  |
|  | Longitude | The longitude coordinate where the data was collected | Decimal | (10, 6) |  |  |  | NO |  |  |  |
|  | Altitude | The altitude at which the data was collected. | Decimal | (10, 2) |  |  |  | NO |  |  |  |
|  | DataScoop.DataScoopID | Foreign key referencing the DataScoop that collected the data | integer | 10 |  |  |  | NO |  | YES |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| SensedDataSubscriber | SensedData.SensedDataID | Foreign key reference to SensedData table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  | Subscriber.SubscriberID | Foreign key reference to Subscriber table | integer | 10 |  |  |  | NO | YES | YES | unique |
|  |  |  |  |  |  |  |  |  |  |  |  |

### Derived Attributes

| **Relation Name** | **Attributes** | **Derived from / calculation** |
| --- | --- | --- |
| **n/a** | n/a | n/a |
|  |  |  |
|  |  |  |

## NaLER Analysis

### Introduction to NaLER Analysis

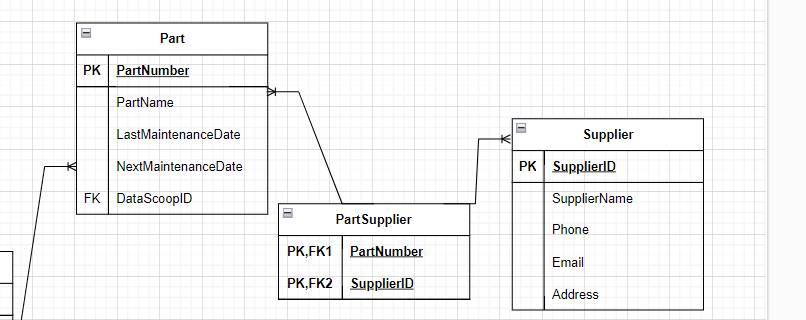
* NaLER or Natural Language for Entity-Relationship is a natural language model for interpreting Entity-relationship model (ERD).
* It helps the user evaluate the ERD model using the common English language *(lukewarren1, 2017)* and makes it easier tocomprehend about the relationship between the data and their used.

### NaLER Analysis – for my ERD

* The entity is display on the topmost of the square box.
* Primary key is shown by displaying a letter ‘PK’ beside the chosen attribute which is underline and in bold.
* The other attributes of the tables are all lists after the primary key.
* The Foreign key attribute or the (FK) is indicate with a letter ‘FK’

A screenshot of a computer program

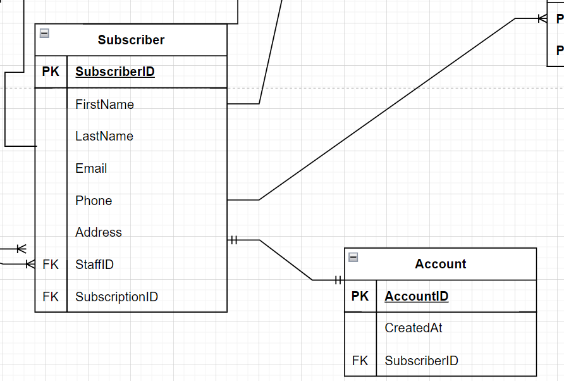
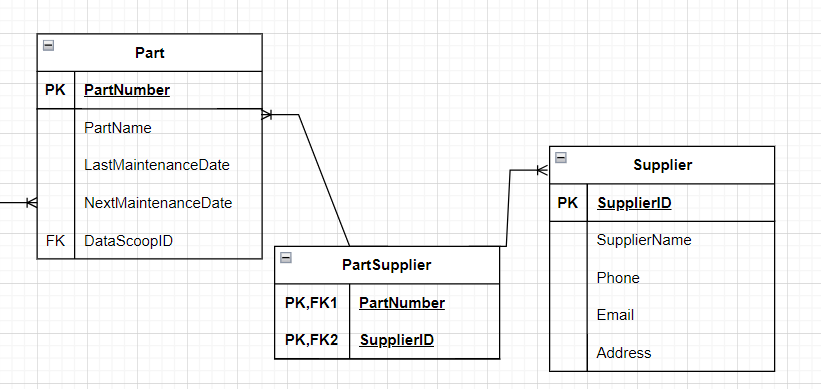
Description automatically generated

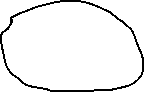


### NaLER - Cardinality

* The cardinality is shown using the Crow’s notation.

One-to-One relationship Many-to-Many relationship





One -to-Many relationship

A screenshot of a computer

Description automatically generated



### NaLER - Attribute Sentences:

C1. Each Subscription(SubscriptionID) is uniquely identified by one ContractID.  
C2. One Subscription (SubscriptionID) must have one StartDate.  
C3. One Subscription (SubscriptionID) must have one EndDate.  
C4. One Subscription (SubscriptionID) may have one SubscriptionPrice.  
C5. One Subscription (SubscriptionID) may have one Discount.  
C6. One Subscription (SubscriptionID) may have one SubscriptionName.  
C8. Each Subscription (SubscriptionID) is managed by one Staff member through StaffStaffId.

S1. Each Staff (StaffID) is uniquely identified by one StaffID.  
S2. One Staff (StaffID) must have one FirstName.  
S3. One Staff (StaffID) must have one LastName.  
S4. One Staff (StaffID) may have one Email.  
S5. One Staff (StaffID) may have one Phone.  
S6. One Staff (StaffID) must have one Role.

SU1. Each Subscriber (SubscriberID) is uniquely identified by one SubscriberID.  
SU2. One Subscriber (SubscriberID) must have one FirstName.  
SU3. One Subscriber (SubscriberID) must have one LastName.  
SU4. One Subscriber (SubscriberID) may have one Email.  
SU5. One Subscriber (SubscriberID) may have one Phone.  
SU6. One Subscriber (SubscriberID) must have one Address.  
SU7. Each Subscriber (SubscriberID) is associated with one SubscriptionType.  
SU8. Each Subscriber (SubscriberID) is managed by one Staff member through StaffStaffID.

A1. Each Account (AccountID) is uniquely identified by one AccountID.  
A2. One Account (AccountID) must have one CreatedAt timestamp.  
A3. Each Account (AccountID) is associated with one Subscriber through SubscriberSubscriberID

DS1. Each DataScoop (DataScoopID) is uniquely identified by one DataScoopID.  
DS2. One DataScoop (DataScoopID) must record one CurrentAltitude.  
DS3. One DataScoop (DataScoopID) must record one CurrentLongitude.  
DS4. One DataScoop (DataScoopID) must record one CurrentLatitude.  
DS5. One DataScoop (DataScoopID) must have one LastMaintenanceDate.  
DS6. One DataScoop (DataScoopID) must have one NextMaintenanceDate.  
DS7. One DataScoop (DataScoopID) must have one OnBoardRetentionPeriod.  
DS8. Each DataScoop (DataScoopID) is configured with one Configuration\_Type through Configuration\_TypeTypeID.

CT1. Each Configuration\_Type (TypeID) is uniquely identified by one TypeID.  
CT2. One Configuration\_Type (TypeID) must have one ConfigurationName.  
CT3. Each Configuration\_Type (TypeID) may be a subtype of another Configuration\_Type through ParentConfigurationTypeID

SP1. Each Supplier (SupplierID) is uniquely identified by one SupplierID.  
SP2. One Supplier (SupplierID) must have one SupplierName.  
SP3. One Supplier (SupplierID) may have one Phone.  
SP4. One Supplier (SupplierID) may have one Email.  
SP5. One Supplier (SupplierID) must have one Address.

P1. Each Part (PartNumber) is uniquely identified by one PartNumber.  
P2. One Part (PartNumber) must have one PartName.  
P3. One Part (PartNumber) must have one LastMaintenanceDate.  
P4. One Part (PartNumber) must have one NextMaintenanceDate.

### NaLER – Relationship Sentences:

1. One Subscriber (SubscriberID) owns exactly one Account (AccountID).
2. One Account (AccountID) is owned by exactly one Subscriber (SubscriberID).
3. One Subscriber (SubscriberID) can have multiple Subscriptions (SubscriptionID).
4. One Subscription (SubscriptionID) is purchased by one Subscriber (SubscriberID).
5. One Subscription (SubscriptionID) must be one of the types: Standard, Gold, Platinum, or Super Platinum.
6. One type (Standard, Gold, Platinum, Super Platinum) can categorize multiple Subscriptions (SubscriptionID).
7. All Subscription types (Standard, Gold, Platinum, Super Platinum) are specializations of the general Subscription class.
8. All Staff roles are instances of the Staff class. There are three types of staff -Maintenance, SalesPerson and AdministrativeExecutives.
9. One zone can contain many VideoStream.

### Conclusion

* Working on the Logical Model Milestone 2 phase allows a clearer understanding of the whole overall project for this assessment.   
  It clears up so many things that I couldn’t grasp before while working on the conceptual model phase in Milestone One due to normalization and it allows me to see the mistakes that I’ve make during working on Milestone One.
* I feel that I may have missed some important part in Milestone two but hopefully it would all clear up on Milestone three.

# Physical database design and implementation

## Physical Database Design

* Physical Database Design is the final phase of Database Design methodology.   
  This phase involves the process of choosing the storage structure and access paths to achieving efficient performance.
* Physical Database design describe indexes used to achieve efficient access to the data as well as any other associated security measures and constraints.
* It focuses on translating the logical database design into a physical design structures that can be implemented in a specific database management system – like SQL Server Management Studio (SSMS) for this project *(sharma, 2024).*

## Translate global logical data model for target DBMS

### Design Base relation for target DBMS

CREATE TABLE Part (  
PartNumber INT IDENTITY(1,1) NOT NULL,  
PartName VARCHAR(50) NOT NULL,  
LastMaintenanceDate DATE NOT NULL,  
NextMaintenanceDate DATE NOT NULL ,  
PRIMARY KEY (PartNumber),  
DataScoopID INT NOT NULL,  
FOREIGN KEY (DataScoopID) REFERENCES DataScoop(DataScoopID)  
);

CREATE TABLE Supplier (  
SupplierID INT IDENTITY(1,1) NOT NULL,  
SupplierName VARCHAR(100) NOT NULL,  
Phone VARCHAR(15) NOT NULL,  
Email VARCHAR(100) UNIQUE NOT NULL ,  
Address VARCHAR(255) NOT NULL,  
PRIMARY KEY (SupplierID)  
);

CREATE TABLE PartSupplier (   
PartNumber INT NOT NULL,   
SupplierID INT NOT NULL,   
PRIMARY KEY (PartNumber, SupplierID),   
FOREIGN KEY (PartNumber) REFERENCES Part(PartNumber),   
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)   
);

CREATE TABLE Configuration\_Type (   
TypeID INT IDENTITY(1,1) NOT NULL,   
ConfigurationName VARCHAR(30) NOT NULL,   
ParentConfigurationTypeID INT,   
PRIMARY KEY (TypeID),   
FOREIGN KEY (ParentConfigurationTypeID) REFERENCES Configuration\_Type(TypeID)   
);

CREATE TABLE DataScoop (   
DataScoopID INT IDENTITY(1,1) NOT NULL PRIMARY KEY,   
CurrentAltitude DECIMAL(10, 2) NOT NULL ,   
CurrentLongitude DECIMAL(10, 6) NOT NULL,   
CurrentLatitude DECIMAL(10, 6) NOT NULL,   
LastMaintenanceDate DATE NOT NULL,   
NextMaintenanceDate DATE NOT NULL,   
OnBoardRetentionPeriod INT,   
Configuration\_TypeID INT,   
FOREIGN KEY (Configuration\_TypeID) REFERENCES Configuration\_Type(TypeID)   
);

CREATE TABLE Subscription(  
SubscriptionID INT DECIMAL(1,1) NOT NULL PRIMARY KEY,  
StartDate DATE NOT NULL,  
EndDate DATE NOT NULL,  
SubscriptionFee DECIMAL(8,2) NOT NULL,  
Discount DECIMAL(4,2),  
SubscriptionName VARCHAR(30) NOT NULL,  
StaffID INT,  
FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)  
);

-- Derived table for Standard subscriptions

CREATE TABLE Standard(  
SubscriptionID INT NOT NULL,  
PRIMARY KEY (SubscriptionID),  
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID)  
);

-- Derived table for Gold subscriptions

CREATE TABLE Gold(  
SubscriptionID INT NOT NULL,  
PRIMARY KEY (SubscriptionID),  
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID)  
);

-- Derived table for Platinum subscriptions

CREATE TABLE Platinum (  
SubscriptionID INT NOT NULL,  
PRIMARY KEY (SubscriptionID),  
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID)  
);

-- Derived table for Super Platinum subscriptions

CREATE TABLE SuperPlatinum (  
SubscriptionID INT NOT NULL,  
PRIMARY KEY (SubscriptionID),  
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID)

);

CREATE TABLE Zone (   
ZoneID INT IDENTITY(1,1) NOT NULL,   
Name VARCHAR(100) NOT NULL,   
Location VARCHAR(255) NOT NULL,   
PRIMARY KEY (ZoneID)   
DataScoopID INT,  
FOREIGN KEY (DataScoopID) REFERENCES DataScoop (DataScoopID)  
);

--Staff table is a base table for type of staff -Maintenance, SalesPerson, AdministrativeExecutives

CREATE TABLE Staff (   
StaffID INT IDENTITY(1,1) NOT NULL,   
FirstName VARCHAR(50) NOT NULL,   
LastName VARCHAR(50) NOT NULL,   
Phone VARCHAR(15) NOT NULL,   
Email VARCHAR(100) UNIQUE NOT NULL ,   
PRIMARY KEY (StaffID)   
);

CREATE TABLE Subscriber (   
SubscriberID INT IDENTITY(1,1) NOT NULL,   
FirstName VARCHAR(50) NOT NULL,   
LastName VARCHAR(50) NOT NULL,   
Email VARCHAR(100) UNIQUE NOT NULL,   
Phone VARCHAR(15),   
Address VARCHAR(255 NOT NULL),   
PRIMARY KEY (SubscriberID),  
StaffID INT,  
SubscriptionID INT,  
FOREIGN KEY(StaffID) REFERENCES Staff(StaffID),  
FOREIGN KEY(SubscriptionID) REFERENCES Subscription(SubscriptionID)  
);

CREATE TABLE ZoneSubscriber(  
ZoneID INT NOT NULL,  
SubscriberID INT NOT NULL,  
PRIMARY KEY(ZoneID, SubscriberID),  
FOREIGN KEY(ZoneID) REFERENCES Zone(ZoneID),  
FOERIGN KEY(SubscriberID) REFERENCES Subscriber(SubscriberID),  
);

CREATE TABLE DataScoopSubscriber(  
DataScoopID INT NOT NULL,  
SubscriberID INT NOT NULL,  
PRIMARY KEY(DataScoopID, SubscriberID),  
FOREIGN KEY(DataScoopID) REFERENCES DataScoop(DataScoopID),  
FOREIGN KEY(SubscriberID) REFERENCES Subscriber(SubscriberID)  
);

CREATE TABLE Account(  
AccountID INT IDENTITY(1,1) NOT NULL,  
CreateAt DATETIME,   
PRIMARY KEY (AccountID),  
SubscriberID INT,  
FOREIGN KEY(SubscriberID) REFERENCES Subscriber(SubscriberID)  
);

CREATE TABLE SensedData (   
SensedDataID INT IDENTITY(1,1) NOT NULL,   
TimeCollected DATETIME NOT NULL,   
Temperature DECIMAL(6, 2) NOT NULL,  
Humidity DECIMAL(5, 2) NOT NULL,   
AmbientLightStrength DECIMAL(15, 5) NOT NULL,   
OrganicSpectralData DECIMAL(15,5) NOT NULL,  
Latitude DECIMAL(10,6) NOT NULL,  
Longitude DECIMAL(10,6) NOT NULL,   
Altitude DECIMAL(10, 2) NOT NULL,   
PRIMARY KEY(SensedData),  
ZoneID INT,  
DataScoopID INT,  
FOREIGN KEY(ZoneID) REFERENCES Zone(ZoneID),  
FOREIGN KEY(DataScoopID) REFERENCES DataScoop(DataScoopID)  
);

CREATE TABLE SensedDataSubscriber(  
SensedDataID INT NOT NULL,   
SubscriberID INT NOT NULL,  
PRIMARY KEY (SensedDataID, SubscriberID),  
FOREIGN KEY (SensedDataID) REFERENCES SensedData(SensedDataID),  
FOREIGN KEY(SubscriberID) REFERENCES Subscriber(SubscriberID)  
);

CREATE TABLE AccessPrivilege (   
AccessPrivilegeID INT IDENTITY(1,1) NOT NULL,   
PrivilegeLevel VARCHAR(80),   
PRIMARY KEY (AccessPrivilegeID)   
);

CREATE TABLE VideoStream (   
VideoStreamID INT IDENTITY(1,1) NOT NULL,   
ZoneID INT,   
DataScoopID INT,   
AccessPrivilegeID INT,   
PRIMARY KEY (VideoStreamID),   
FOREIGN KEY (ZoneID) REFERENCES Zone(ZoneID),   
FOREIGN KEY (DataScoopID) REFERENCES DataScoop(DataScoopID),   
FOREIGN KEY (AccessPrivilegeID) REFERENCES AccessPrivilege(AccessPrivilegeID)   
);

CREATE TABLE VideoStreamSubscriber (   
VideoStreamID INT NOT NULL,   
SubscriberID INT NOT NULL,   
PRIMARY KEY (VideoStreamID, SubscriberID),  
FOREIGN KEY (VideoStreamID) REFERENCES VideoStream(VideoStreamID),   
FOREIGN KEY (SubscriberID) REFERENCES Subscriber(SubscriberID)   
);

CREATE TABLE SubscriptionDataScoop (   
SubscriptionID INT NOT NULL,   
DataScoopID INT NOT NULL,   
PRIMARY KEY (SubscriptionID, DataScoopID),   
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID),   
FOREIGN KEY (DataScoopID) REFERENCES DataScoop(DataScoopID)   
);

CREATE TABLE SubscriptionZone (   
SubscriptionID INT NOT NULL,   
ZoneID INT NOT NULL,   
PRIMARY KEY (SubscriptionID, ZoneID),   
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID),   
FOREIGN KEY (ZoneID) REFERENCES Zone(ZoneID)   
);

--Staff class’s subclasses   
CREATE TABLE Maintenance (   
StaffID INT NOT NULL,   
Date DATE,   
PRIMARY KEY (StaffID),   
FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)   
);

CREATE TABLE AdministrativeExecutives (   
StaffID INT NOT NULL,   
PRIMARY KEY (StaffID),   
FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)   
);

CREATE TABLE SalesPerson (  
StaffID INT NOT NULL,  
PRIMARY KEY (StaffID),  
FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)  
);

CREATE TABLE SubscriptionAccessPrivilege(  
SubscriptionID INT NOT NULL,  
AccessPrivilege INT NOT NULL,  
PRIMARY KEY (SubscriptionID, AccessPrivilegeID),  
FOREIGN KEY (SubscriptionID) REFERENCES Subscription(SubscriptionID)  
FOREIGN KEY (AccessPrivilegeID) REFERENCES AccessPrivilege(AccessPrivilegeID)  
);

## Design Enterprise constraints for DBMS

Entity Integrity Constraints

* Implementing entity constraints will ensure that each row in a table is uniquely identifiable :
* Primary Keys: Having a primary key in all the tables to uniquely identify rows.
* Unique Constraints: Implement unique constraints where appropriate, such as on email fields to prevent duplicate entries.

Referential Integrity

* Referential integrity requires that a foreign key must have a matching primary key or it must be null. This constraint is specified between two tables (LibreTexts, 2022). It ensures that relationships between tables remain consistent.

Enterprise Constraints

* A subscription may be discounted by a subscription salesperson by up to 3%.
* Standard video subscribers can view any of the 100 live 3D video streams.
* Gold video subscribers can control one of the 100 live 3D video streams.
* Platinum and Super Platinum subscribers get exclusive rights to the data and video streams in their subscribed zones.
* Only staff in an Administrative Executive role may change the price of subscriptions and enter contract details.
* Use features supported by the target DBMS such as Stored procedures to handle subscriptions.
* **Define User Roles and Permissions** to restrict access to subscription pricing and contract details to Administrative Executives.

## Design physical Representation

* Use Stored procedure to handle the creation of Subscription as well as applying discount.

## Analyze Transactions

* Understand the functionality of the transactions that will run on the database and analyze the important transactions.  
  For each transaction, we should determine:
* The expected frequency at which the transaction will run.
* The relations and attributes accessed by the transaction and the type of access; that is, query, insert, update, or delete.
* The attributes used in any predicates (in SQL, the predicates are the conditions specified in the WHERE clause). Check whether the predicates involve pattern matching, range searches, or exact match key retrieval.
* For a query, the attributes that are involved in the join of two or more relations.
* The time constraints imposed on the transaction

## Choose File Organizations

* I think B+-Tree would be a suitable choice for the FlightStream database project as it is easy to find and manage data quickly and efficiently.   
  The indexes are good at handling lots of changes and queries, which is perfect for FlightStream’s needs. The B+-Tree works will with many database systems and are easy use.

## Choose Secondary Indexes

* I think not using Secondary Indexes will be more beneficial to the database and the cost as it may lower query performance.

## Consider the introduction of controlled redundance

- I think that this will not be required for the FlightStream database.

## Estimate disk space requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Relation Name | Attribute | Data type | Length | Total Per Row |
| Part | PartNumber | INT | 4 |  |
|  | PartName | VARCHAR | 50 |  |
|  | LastMaintenanceDate | DATE | 3 |  |
|  | NextMaintenanceDate | DATE | 3 |  |
|  | DataScoopID | INT | 4 |  |
|  |  |  |  | 64 |
| Supplier | SupplierID | INT | 4 |  |
|  | SupplierName | VARCHAR | 100 |  |
|  | Phone | VARCHAR | 15 |  |
|  | Email | VARCHAR | 100 |  |
|  | Address | VARCHAR | 255 |  |
|  |  |  |  | 474 |
| PartSupplier | PartNumber | INT | 4 |  |
|  | SupplierID | INT | 4 |  |
|  |  |  |  | 8 |
| Configuration\_Type | TypeID | INT | 4 |  |
|  | ConfigurationName | VARCHAR | 30 |  |
|  | ParentConfigurationTypeID (nullable) | INT | 4 (+1 for null) |  |
|  |  |  |  | 39 |
| DataScoop | DataScoopID | INT | 4 |  |
|  | CurrentAltitude | DECIMAL | 5 |  |
|  | CurrentLongitude | DECIMAL | 5 |  |
|  | CurrentLatitude | DECIMAL | 5 |  |
|  | LastMaintenanceDate | DATE | 3 |  |
|  | NextMaintenanceDate | DATE | 3 |  |
|  | OnBoardRetentionPeriod | INT | 4 |  |
|  | Configuration\_TypeID | INT | 4 |  |
|  |  |  |  | 33 |
| Subscription | SubscriptionID | INT | 4 |  |
|  | StartDate | DATE | 3 |  |
|  | EndDate | DATE | 3 |  |
|  | SubscriptionFee | DECIMAL | 5 |  |
|  | Discount | DECIMAL | 5 |  |
|  | SubscriptionName | VARCHAR | 30 |  |
|  | StaffID | INT | 4 |  |
|  |  |  |  | 54 |
| Standard, Gold, Platinum, Super Platinum Tables | SubscriptionID x 4 | INT | 4 \* 4 |  |
|  |  |  |  | 16 |
| Zone | ZoneID | INT | 4 |  |
|  | Name | VARCHAR | 100 |  |
|  | Location | VARCHAR | 255 |  |
|  |  |  |  | 359 |
| Staff | StaffID | INT | 4 |  |
|  | FirstName | VARCHAR | 50 |  |
|  | LastName | VARCHAR | 50 |  |
|  | Phone | VARCHAR | 15 |  |
|  | Email | VARCHAR | 100 |  |
|  |  |  |  | 219 |
| Maintenance | StaffID | INT | 4 |  |
|  | Date | DATE | 3 |  |
|  |  |  |  | 7 |
| SalesPerson, AdministrativeExecutives | StaffID x 2 | INT | 4 \* 2 |  |
|  |  |  |  | 8 |
| Subscriber | SubscriberID | INT | 4 |  |
|  | FirstName | VARCHAR | 50 |  |
|  | LastName | VARCHAR | 50 |  |
|  | Email | VARCHAR | 100 |  |
|  | Phone | VARCHAR | 15 |  |
|  | Address | VARCHAR | 255 |  |
|  | StaffID | INT | 4 |  |
|  | SubscriptionID | INT | 4 |  |
|  |  |  |  | 482 |
| ZoneSubscriber | ZoneID | INT | 4 |  |
|  | SubscriberID | INT | 4 |  |
|  |  |  |  | 8 |
| DataScoopSubscriber | DataScoopID | INT | 4 |  |
|  | SubscriberID | INT | 4 |  |
|  |  |  |  | 8 |
| SensedData | SensedDataID | INT | 3 |  |
|  | TimeCollected | DATETIME | 8 |  |
|  | Temperature | DECIMAL | 5 |  |
|  | Humidity | DECIMAL | 5 |  |
|  | AmbientLightStrength | DECIMAL | 9 |  |
|  | OrganicSpectralData | DECIMAL | 9 |  |
|  | Latitude | DECIMAL | 5 |  |
|  | Longitude | DECIMAL | 5 |  |
|  | Altitude | DECIMAL | 5 |  |
|  | ZoneID | INT | 4 |  |
|  | DataScoopID | INT | 4 |  |
|  |  |  |  | 63 |
| SensedDataSubscriber | SensedDataID | INT | 4 |  |
|  | SubscriberID | INT | 4 |  |
|  |  |  |  | 8 |
| AccessPrivilege | AccessPrivilegeID | INT | 4 |  |
|  | PrivilegeLevel | VARCHAR | 80 |  |
|  |  |  |  | 84 |
| VideoStream | VideoStreamID | INT | 4 |  |
|  | ZoneID | INT | 4 |  |
|  | DataScoopID | INT | 4 |  |
|  | AccessPrivilegeID | INT | 4 |  |
|  |  |  |  | 16 |
| VideoStreamSubscriber | VideoStreamID | INT | 4 |  |
|  | SubscriberID | INT | 4 |  |
|  |  |  |  | 16 |
| SubscriptionDataScoop | SubscriptionID | INT | 4 |  |
|  | DataScoopID | INT | 4 |  |
|  |  |  |  | 16 |
| SubscriptionZone | SubscriptionID | INT | 4 |  |
|  | ZoneID | INT | 4 |  |
|  |  |  |  | 16 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Relation Name | Total Per Row | Max Size | Total Relation Size | Pages Per Relation | New Relation Size |
| Part | 64 | 640,000 | 640,000 | 79 | 640,000 |
| DataScoop | 33 | 1,650,000 | 1,650,000 | 202 | 1,650,000 |
| Supplier | 474 | 948,000 | 948,000 | 116 | 948,000 |
| PartSupplier | 8 | 160,000 | 160,000 | 20 | 160,000 |
| Configuration\_Type | 38 | 19,000 | 19,000 | 3 | 19,000 |
| Subscription | 54 | 270,000 | 270,000 | 33 | 270,000 |
| Zone | 359 | 35,900 | 35,900 | 5 | 35,900 |
| Staff | 219 | 219,000 | 219,000 | 27 | 219,000 |
| Subscriber | 482 | 4,820,000 | 241,000,000 | 29,428 | 241,000,000 |
|  |  |  |  |  |  |

Total Database size (bytes): 245,042,900 bytes

Total Database size (kilobytes): 239,298.73 KB

Total Database size (megabytes): 233.59 MB

Total Database size (gigabytes): 0.228 GB

## Physical Design for FlightStream

A group of white rectangular objects

Description automatically generated

## Issues & Resolutions

* There were quite a lot of issue and I have do redo some part and where for other, I just left it since time is tight.  
  For example, I accidently put a ‘d’ in AccessPriviledge-where it suppose to be AccessPrivilege.  
  After realizing it, it was too late, I had already run the script and all the foreign keys and relating table were all create so I wasn’t able to delete it so I create a new column with the right spelling name.
* For creating stored procedures – they all quite follow the same step, so it was ok, but it took a long time to settle it since the query weren’t easy. But since I can drop procedure and recreate it again and again, I didn’t meet with any trouble over this.
* I also had to alter and add some new column since I didn’t include one when creating but when I had to do the required transaction -I realised I don’t have it so, I had to create one the Part Table required the Cost column to calculate the total Cost for maintaining the DataScoop.
* Making sure the inserted Values are all in correct range -for example I have 7 DataScoopID so when I refer to the DataScoopID – it needs to make sure that the DataScoopID is within in range.
* The hardest one was the “Delete the data collected for a given Contract” where in my case ‘Contract’ is ‘Subscription’. This required to include all the linking tables and the right one at that or else, the query won’t run due to error, so this particular one took so many try to get it right, but it works out.
* Other things were ensuring all the linking many-to-many tables were defined and included when creating it as there are so many of them and forgetting one would really mess by the whole script, so I had to check side by side for this.
* Another important thing was, creating the tables in the priorities order -in whether it depending on other tables or other tables are depending on it.

## Conclusion

* It is a learning experience at the end of the day. I learn a lot of stuff doing the assessment and enjoy part of it.  
  It was a really long and complicated process but I’m glad that I managed to complete and get as much done as I can.
* I was able to references a lot of the script that I wrote using the class notes and exercises that were available, so it definitely was very helpful and save a lot of time.
* I can’t say I enjoy it but all the things that I was able to learn from this is really helpful and help me gain a new perspective.

# References

A S, R. (2024, May 31). *What is Normalization in SQL with Examples? 1NF, 2NF, 3NF and BCNF | Simplilearn*. Simplilearn.Com. https://www.simplilearn.com/tutorials/sql-tutorial/what-is-normalization-in-sql

Atlan. (2023, March 17). *Top 6 Benefits of a Data Dictionary (2024 Guide)*. https://atlan.com/benefits-of-a-data-dictionary/

Barnes, R. (2023, September). *Extended Entity-Relationship (EE-R) Model*. https://www.tutorialspoint.com/Extended-Entity-Relationship-EE-R-Model

*Chen notation & Crow’s Foot | Gleek diagram maker*. (n.d.). Retrieved March 9, 2024, from https://www.gleek.io/er-diagrams

codecademy. (n.d.). *What is a Database Index?* Codecademy. Retrieved June 4, 2024, from https://www.codecademy.com/article/sql-indexes

Fernigrini, L. (2022, January 27). *5 Examples of a UNIQUE Constraint in a Relational Database*. Vertabelo Data Modeler. https://vertabelo.com/blog/unique-constraint-examples/

fuchs, michael. (2021, 03). *Entity-Relationship-Diagram (ERD)—Michael Fuchs SQL*. https://michael-fuchs-sql.netlify.app/2021/03/03/entity-relationship-diagram-erd/#associative-entity

geeksforgeeks. (2023a, May 22). *Boyce-Codd Normal Form (BCNF)*. GeeksforGeeks. https://www.geeksforgeeks.org/boyce-codd-normal-form-bcnf/

geeksforgeeks. (2023b, November 6). *Introduction of 4th and 5th Normal Form in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-of-4th-and-5th-normal-form-in-dbms/

geeksforgeeks. (2024a, March 22). *Generalization, Specialization and Aggregation in ER Model*. GeeksforGeeks. https://www.geeksforgeeks.org/generalization-specialization-and-aggregation-in-er-model/

geeksforgeeks. (2024b, May 23). *Normal Forms in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/normal-forms-in-dbms/

GreeksforGreeks. (2023, February 22). *Difference between Strong and Weak Entity*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-strong-and-weak-entity/

IBM. (n.d.). *What is Data Modeling? | IBM*. Retrieved February 27, 2024, from https://www.ibm.com/topics/data-modeling

Jain, M. (2022, May 25). *Types of Attributes in DBMS*. Scaler Topics. https://www.scaler.com/topics/types-of-attributes-in-dbms/

javatpoint. (n.d.). *Enhanced ER Model—Javatpoint*. Www.Javatpoint.Com. Retrieved March 9, 2024, from https://www.javatpoint.com/enhanced-er-model

LibreTexts. (2022, December 30). *9.1: Constraints and Intergrity*. Engineering LibreTexts. https://eng.libretexts.org/Courses/Delta\_College/Database\_Design\_-\_NOT\_GOOD/09%3A\_Chapter\_9\_-\_Integrity\_Rules\_and\_Constraints/9.1%3A\_Constraints\_and\_Intergrity

Lucidchart. (n.d.). *Enhanced ER Diagram Tutorial*. Lucidchart. Retrieved May 1, 2024, from https://www.lucidchart.com/pages/enhanced-entity-relationship-diagram

lukewarren1. (2017, April 7). NaLER Analysis. *Dat601 - Luke W.* https://dat601lukewarren.wordpress.com/2017/04/07/naler-analysis/

Relationaldbdesign. (n.d.). *Purpose of Entity Identifiers (Attributes)*. Retrieved March 22, 2024, from https://www.relationaldbdesign.com/database-design/module5/store-entity-data-values.php

Salahuddin, M. (2021, December 19). *8 Key Principles of Data Management | LinkedIn*. https://www.linkedin.com/pulse/8-key-principles-data-management-muhammad-salahuddin/

sharma, sudhir. (2024, January 31). *Physical Database Design and Tuning*. https://www.tutorialspoint.com/physical-database-design-and-tuning

Visual Paradigm. (2023, September 18). Navigating the Three Levels of Database Design: Conceptual, Logical, and Physical. *Visual Paradigm Guides*. https://guides.visual-paradigm.com/navigating-the-three-levels-of-database-design-conceptual-logical-and-physical/

Wikipedia. (2024). Enhanced entity–relationship model. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Enhanced\_entity%E2%80%93relationship\_model&oldid=1212862365

A S, R. (2024, May 31). *What is Normalization in SQL with Examples? 1NF, 2NF, 3NF and BCNF | Simplilearn*. Simplilearn.Com. https://www.simplilearn.com/tutorials/sql-tutorial/what-is-normalization-in-sql

Atlan. (2023, March 17). *Top 6 Benefits of a Data Dictionary (2024 Guide)*. https://atlan.com/benefits-of-a-data-dictionary/

Barnes, R. (2023, September). *Extended Entity-Relationship (EE-R) Model*. https://www.tutorialspoint.com/Extended-Entity-Relationship-EE-R-Model

*Chen notation & Crow’s Foot | Gleek diagram maker*. (n.d.). Retrieved March 9, 2024, from https://www.gleek.io/er-diagrams

codecademy. (n.d.). *What is a Database Index?* Codecademy. Retrieved June 4, 2024, from https://www.codecademy.com/article/sql-indexes

Fernigrini, L. (2022, January 27). *5 Examples of a UNIQUE Constraint in a Relational Database*. Vertabelo Data Modeler. https://vertabelo.com/blog/unique-constraint-examples/

fuchs, michael. (2021, 03). *Entity-Relationship-Diagram (ERD)—Michael Fuchs SQL*. https://michael-fuchs-sql.netlify.app/2021/03/03/entity-relationship-diagram-erd/#associative-entity

geeksforgeeks. (2023a, May 22). *Boyce-Codd Normal Form (BCNF)*. GeeksforGeeks. https://www.geeksforgeeks.org/boyce-codd-normal-form-bcnf/

geeksforgeeks. (2023b, November 6). *Introduction of 4th and 5th Normal Form in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-of-4th-and-5th-normal-form-in-dbms/

geeksforgeeks. (2024a, March 22). *Generalization, Specialization and Aggregation in ER Model*. GeeksforGeeks. https://www.geeksforgeeks.org/generalization-specialization-and-aggregation-in-er-model/

geeksforgeeks. (2024b, May 23). *Normal Forms in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/normal-forms-in-dbms/

GreeksforGreeks. (2023, February 22). *Difference between Strong and Weak Entity*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-strong-and-weak-entity/

IBM. (n.d.). *What is Data Modeling? | IBM*. Retrieved February 27, 2024, from https://www.ibm.com/topics/data-modeling

Jain, M. (2022, May 25). *Types of Attributes in DBMS*. Scaler Topics. https://www.scaler.com/topics/types-of-attributes-in-dbms/

javatpoint. (n.d.). *Enhanced ER Model—Javatpoint*. Www.Javatpoint.Com. Retrieved March 9, 2024, from https://www.javatpoint.com/enhanced-er-model

LibreTexts. (2022, December 30). *9.1: Constraints and Intergrity*. Engineering LibreTexts. https://eng.libretexts.org/Courses/Delta\_College/Database\_Design\_-\_NOT\_GOOD/09%3A\_Chapter\_9\_-\_Integrity\_Rules\_and\_Constraints/9.1%3A\_Constraints\_and\_Intergrity

Lucidchart. (n.d.). *Enhanced ER Diagram Tutorial*. Lucidchart. Retrieved May 1, 2024, from https://www.lucidchart.com/pages/enhanced-entity-relationship-diagram

lukewarren1. (2017, April 7). NaLER Analysis. *Dat601 - Luke W.* https://dat601lukewarren.wordpress.com/2017/04/07/naler-analysis/

Relationaldbdesign. (n.d.). *Purpose of Entity Identifiers (Attributes)*. Retrieved March 22, 2024, from https://www.relationaldbdesign.com/database-design/module5/store-entity-data-values.php

Salahuddin, M. (2021, December 19). *8 Key Principles of Data Management | LinkedIn*. https://www.linkedin.com/pulse/8-key-principles-data-management-muhammad-salahuddin/

sharma, sudhir. (2024, January 31). *Physical Database Design and Tuning*. https://www.tutorialspoint.com/physical-database-design-and-tuning

Visual Paradigm. (2023, September 18). Navigating the Three Levels of Database Design: Conceptual, Logical, and Physical. *Visual Paradigm Guides*. https://guides.visual-paradigm.com/navigating-the-three-levels-of-database-design-conceptual-logical-and-physical/

Wikipedia. (2024). Enhanced entity–relationship model. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Enhanced\_entity%E2%80%93relationship\_model&oldid=1212862365