Design conceptual and logical database models for Spaces©. Implement the database in SQL Server.

DAT601 Assessment Report

Full Report

23/6/23

Jayden Houghton – 13524050

Contents

[Introduction 2](#_Toc137665982)

[Part One – Conceptual Database Design 3](#_Toc137665983)

[Introduction to Data Modelling 3](#_Toc137665984)

[Conceptual Model 3](#_Toc137665985)

[Logical Model 3](#_Toc137665986)

[Physical Model 4](#_Toc137665987)

[Diagram Example 4](#_Toc137665988)

[Conceptual Modelling 4](#_Toc137665989)

[Description 4](#_Toc137665990)

[Components of Chen’s Notation 5](#_Toc137665991)

[Extended Chen’s Notation 8](#_Toc137665992)

[Conceptual ERD 9](#_Toc137665993)

[Data Dictionary 10](#_Toc137665994)

[Entities 10](#_Toc137665995)

[Relationships 12](#_Toc137665996)

[Attributes 13](#_Toc137665997)

[Business Rules 14](#_Toc137665998)

[Part Two – TSQL Queries 17](#_Toc137665999)

[Basic Queries 17](#_Toc137666000)

[Sub Queries 20](#_Toc137666001)

[Combined Query 21](#_Toc137666002)

[Views 22](#_Toc137666003)

[Model Conversion 23](#_Toc137666004)

[Logical ERD 24](#_Toc137666005)

[Normalisation 26](#_Toc137666006)

[Forms and Steps 26](#_Toc137666007)

[Unnormalised (0NF) 26](#_Toc137666008)

[First Normal Form (1NF) 26](#_Toc137666009)

[Second Normal Form (2NF) 27](#_Toc137666010)

[Third Normal Form (3NF) 28](#_Toc137666011)

[Boyce-Codd Normal Form (BCNF) 29](#_Toc137666012)

[Fourth Normal Form (4NF) 29](#_Toc137666013)

[Normalised ERD 29](#_Toc137666014)

[Data Dictionary 30](#_Toc137666015)

[Document relations 31](#_Toc137666016)

[Document Attributes 33](#_Toc137666017)

[NaLER Analysis 37](#_Toc137666018)

[Process 37](#_Toc137666019)

[Analysis 37](#_Toc137666020)

[Entities 37](#_Toc137666021)

[Relationships 41](#_Toc137666022)

[SQL DDL 45](#_Toc137666023)

[Tables 45](#_Toc137666024)

[Check Constraints 45](#_Toc137666025)

[Roles and Users 46](#_Toc137666026)

[Views 46](#_Toc137666027)

[Test Data 47](#_Toc137666028)

[SQL Transactions 47](#_Toc137666029)

[Analysis 47](#_Toc137666030)

[Creation 52](#_Toc137666031)

[Indexing 52](#_Toc137666032)

[File Storage 52](#_Toc137666033)

[Controlled Redundancy 52](#_Toc137666034)

[Estimated Size 53](#_Toc137666035)

[Significant Issues 53](#_Toc137666036)

[Data 54](#_Toc137666037)

[Transactions 54](#_Toc137666038)

[Reflection 54](#_Toc137666039)

[Conclusion 54](#_Toc137666040)

[References 55](#_Toc137666041)

# Introduction

This document covers all parts in assignments one and two.

This report covers the design and implementation of the Spaces© database, starting with a conceptual Chen’s Notation model. It progresses through the logical phase with NaLER analysis and data dictionaries, then it is physically implement in Microsoft SQL Server.

# Conceptual Database Design

## Introduction to Data Modelling

*An introduction to data modelling in information systems. Include an overview of Conceptual, Logical and Physical data modelling.*

Data modelling is the process of creating a visual representation of an information system’s database (What is data modeling?, Retireved 2023). The goal is to highlight the data points and structures, the data used and stored within the system, and the relationships between these, so data can be grouped and organised in an efficient and effective manner. This results in a more cohesive database structure plan that can be applied to various database management systems (DBMS). Business needs can also be easily considered and checked that they can be applied within the system being designed.

In data modelling, there are three main models that are used: conceptual, logical and physical. Each cover different levels of detail in relation how the database will work and the physical implementation.

### Conceptual Model

Conceptual models offer an overview of what the database will contain, how it will be organised, and which business rules are involved, independent of all physical considerations. This is to show the overall structure of how the database will be designed (Sherman, 2015). Chen’s notation is one of the most commonly used notation system for creating conceptual models.

### Logical Model

Logical models cover the same database information as the conceptual model, however they represent it in the specific data model that has been selected for the database. These models are used to establish the structure of data elements and the relationships between them to show how the system will be implemented (What is a Logical Data Model?, Retrieved 2023). The main focus of a logical model is how the system has to be implemented.

### Physical Model

A physical diagram is the third step in the process, with the diagram describing how the system will be implemented using the specific DBMS selected. This will include the final selected keys and the exact data types (with lengths assigned, default values given and allow null chosen) selected in the DBMS. This will be used to help with the actual implementation of the database.

### Diagram Example

Diagram

Description automatically generated

(Data Modeling, Retrieved 2023)

## Conceptual Modelling

*A description of conceptual modelling. An introduction and description of the components of a Chen Entity-Relationship Diagram ERD, include extended components.*

### Description

Conceptual modelling is the process of creating an abstract and high-level (but not necessarily) summary of the database’s content, independent of all physical considerations. This creates a clear and concise overall model to assist with understanding how the database is designed. This is essential for ensuring that designers, developers and stakeholders are on the same page and can communicate effectively about the design.

There are many different ways to conceptually model a database. Chen’s notation is one of the most used notation method, that provides a more in-depth look at the database then many other forms.

Chen’s notation was developed by Peter Chen in 1976, who was one of the pioneers of using entity relationship concepts in information system modelling (Chen Notation, 2014). Chen’s notation falls under the category of an Entity Relationship Diagram (ERD), as it uses entities and their relationships to represent the database. Due to this nature, it is primarily used for relational databases but can be adapted for non-relational databases (Shin, Hwang, & Jung, 2017).

### Components of Chen’s Notation

Component summary information (Chen Notation, 2014).

**Entities**

* These are represented by rectangles.
* Entities represent real-world things or objects that can be distinguished from each other.
* These will become the tables in relational logical or physical models.
* Example: a book or an author.
* There are different types of entities:
  + Strong – can be uniquely identified by its attributes (e.g. a book)
  + Weak – depends on an owner entity (e.g. a book chapter)
  + Associative – represents a many-to-many relationship table(e.g. books to authors relationship, as authors can write multiple books and books can be written by multiple authors)

Graphical user interface, application

Description automatically generated

**Attributes**

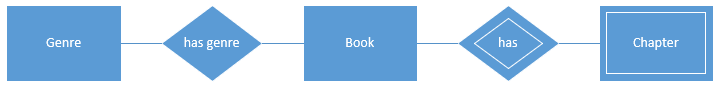
* These are represented by ovals.
* Attributes are linked to its parent entity with a line and represent a property of the entity.
* There are different types of attributes:
  + Simple – a basic property of the entity (e.g. book title)
  + Key – uniquely identifies an entity (e.g. International Standard Book Number (ISBN) code)
  + Partial key (discriminator) – combined with the owner entity key to make a key for the weak entity (e.g. book ID + chapter ID, if you don’t know the book, you can’t find the chapter)
  + Multivalued – can contain multiple values (e.g. book subject, can fit into multiple categories)
  + Derived – value calculated from other attributes (e.g. age calculated from date of birth)
  + Composite – contains smaller attributes (e.g. author name could be split into first name and last name)

Diagram

Description automatically generated

**Relationships**

* Represented by diamonds.
* Relationships connect two entities together to show how they relate and interact.
* There are different types of attributes:
  + Strong – entity is independent of other entities (e.g. book)
  + Weak – child entity depends on parent entity (e.g. a book’s chapter)



**Cardinality**

* Represented by the characters “1”, “N” and “M”.
* Cardinality shows how many records are related between entities in a relationship.
* There are four types of cardinality:
  + One-to-one (1:1) – One record is related to one record (e.g. a book only has one ISBN number and each number can only relate to one book)
  + One-to-many (1:N) – One record can relate to many records (e.g. one author can have many books but each is only written by one author)
  + Many-to-one (N:1) – Many records can relate to one record (e.g. many books belong to one library but that book cannot be in other libraries at the same time)
  + Many-to-many (N:M) – Many records relate to many records (e.g. many books can have many genres and each genre can relate to many books)

A picture containing chart

Description automatically generated

**Optionality**

* Represented by a solid or dashed line connecting entities to the relationship.
* Optionality refers to whether an entity can exist independantely of the relationship.
* There are two types:
  + Mandatory – Shown by a solid line (e.g. a book must have an author)
  + Optional – Shown by a dashed line (e.g. an author doesn’t have to have a book)

A blue diamond with white text

Description automatically generated with medium confidence

**Participation**

* Represented by a single or double line connecting entities to the relationship.
* Participation refers to whether all records must be involved in the relationship or only some.
* There are two types:
  + Total – Shown by two lines (e.g. all chapters must be related to a book)
  + Partial – Shown by a single line (e.g. each book doesn’t need to be related to a chapter)

A blue diamond with white text

Description automatically generated with medium confidence

### Extended Chen’s Notation

Basic Chen’s Notation can be too simple to model more complex databases. In these cases, it can be extended to include extra notation concepts. The most common is generalisation and specialisation, which is based off the concept of Object Orientation (Yue, 2018). This extension uses the idea of a general superclass which has multiple specialised subclasses. This eliminates repeating similar (but slightly different) entites by combining them into a class system.

Components:

* Superclass – The generalisaed entity that contains the shared attributes (e.g. borrower)
* Subclass – The specialised entity that contains the attributes only for that specific type (e.g. youth borrowser)
* These are connected via a circle. This circle will contain either the letter “D” or “O”.
  + D – Means that each subclass is disjoint. Each instance can only have one subclass.
  + O – Means that the subclasses are overlapping. Each instance can have multiple subclasses.
* Each line connecting the subclass to the circle has a “C” shape on it to show that the entity includes all of the attributes from the superclass.

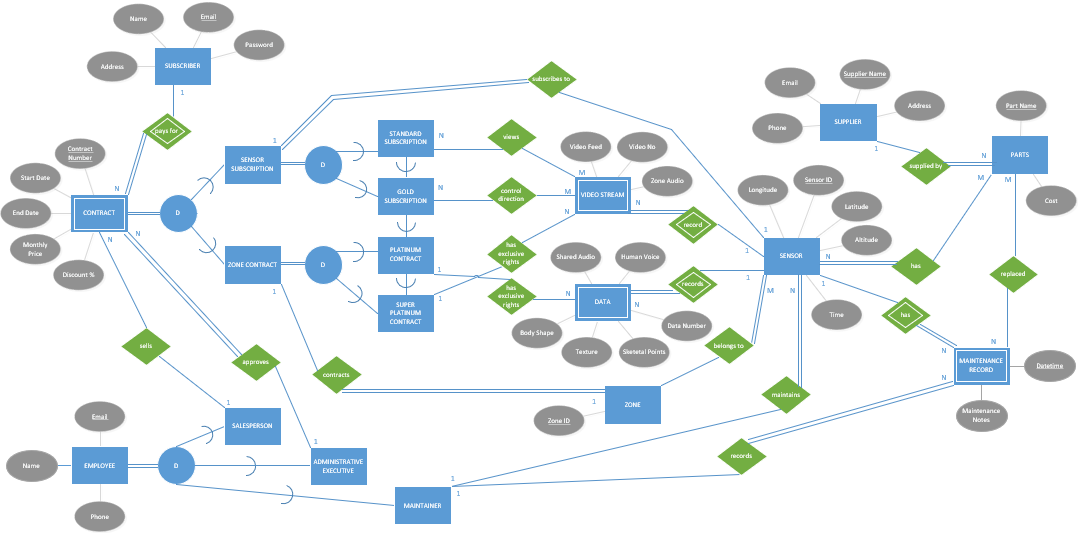
A picture containing text, screenshot, diagram, font

Description automatically generated

## Conceptual ERD

*A conceptual ER model using Chen Extended ERD notation of the database as derived from the case study. Accompany this with your rationale; describe and explain the reasoning and purpose of all parts of the model.*

Please refer to the conceptual model PDF file.



I have decided to have subscription as a separate entity to subscriber, so a subscriber can have multiple subscriptions, allowing access to multiple sensors and/or hosting zones.

I decided to have all contracts and subscriptions under a superclass called “Contract”. This is because they have the same attributes, so this avoids repetition. This class is broken into relevant subclasses to categorise and hold the different subscription levels.

I decided to have another subclass for my employees, again because of duplicated attributes.

The sensor subscription entity has a relationship to a sensor directly to avoid any traps going through the subscription level and video stream. This is the same for zone contract to zone.

The majority of my rationale are contained in the business rules section.

## Data Dictionary

*A data dictionary using the template provided on the course website, that covers the following:*

* *Entities (e.g. name description, aliases, occurrences*
* *Relationships (e.g. name, multiplicity)*
* *Attributes (e.g. name, description, domain, aliases, composite, derived, nulls, key, default value).*

*Accompany this with your rationale; describe and explain the reasoning and purpose of all parts of the model.*

### Entities

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name | Description | Aliases | Occurrence |
| Subscriber | Someone who has, is about to, or is currently in a subscription. | User | Created by website or salesperson. Similar amount as sensors. |
| Contract | An agreement between a subscriber (individual or organisation) and Spaces that the subscriber will gain access to one of more of the following depending on the subscription fee:   * View a sensor’s video streams * Control the direction of the sensor * Gain exclusive rights to the video stream * Gain Exclusive rights to the data stream * Set up a zone with sensors | Contract | Created when a subscriber signs up or when a current subscriber wants another contract. Estimated at least as many contracts as subscribers. |
| Sensor Subscription | A standard or gold tier subscription from an individual to a 3d sensor. | Video Contract | Created when a subscriber signs a contract for a standard or gold level subscription. This will be the most popular type of contract. Estimated 98% of contracts |
| Zone Contract | A platinum or super platinum tier contract (generally with an organisation) to a zone. This involves setting up a zone. | Data Contract | Zone contracts are created when a business signs a contract to set up a zone with a platinum or super platinum contract. Estimated 2% of contracts. |
| Standard Subscription | An agreement between a subscriber and Spaces where the subscriber can view the video streams of a specified 3d sensor. | Standard, Standard Contract | Created when a subscriber signs a contract for a standard level subscription. This will be the most popular type of subscription. Estimated 90% of contracts. |
| Gold Subscription | An agreement between a subscriber and Spaces where the subscriber can view the video streams of a specified 3d sensor and change its direction. | Gold, Gold Contract | Created when a subscriber signs a contract for a gold level subscription. Estimated 8% of contracts. |
| Platinum Contract | An agreement between a subscriber and Spaces where the subscriber can view the video streams of a specified 3d sensor, change its direction and have exclusive rights to the data stream. A new zone and sensors can be set up. | Platinum, Platinum Subscription | Created when a business signs a contract to set up a zone with a platinum level contract. Estimated 1.9% of contracts. |
| Super Platinum Contract | An agreement between a subscriber and Spaces where the subscriber has exclusive rights to view the video streams of a specified 3d sensor, change its direction and have exclusive rights to the data stream. A new zone and sensors can be set up. | Super Platinum, Super Platinum Subscription | Created when a business signs a contract to set up a zone with a super platinum level contract. Estimated 0.1% of contracts. |
| Video Stream | The constant video feed coming from a sensor. | Video | Created by a sensor when it is live. Up to 100 video feeds per sensor. |
| Data Stream | The constant data being transmitted from a sensor. Includes 3d human imagery, shared audio, human voice | Data | Created by a sensor when it is live. One stream of data per data focus (imagery, audio, voice). |
| 3d Sensor | A 3d sensor device that can capture video, audio and estimate human positioning data. | Sensor | Created when a sensor is added to a zone. Estimated 10 sensors per zone. |
| Zone | A zone is a network of one or more sensors put in place for the same purpose and from the same contracting organisation. |  | A zone is created with each zone contract. There should be the same number of zones as zone contracts. |
| Parts | Mechanical and electronic components of the sensors that must be replaced over time. |  | Parts are created when a new part is required by a sensor. Estimated few hundred parts. |
| Supplier | A business that supplies parts for the sensors. |  | A supplier is created when a part is sourced from a different company or a new part from a new company is added. Estimated less than the number of parts. |
| Maintenance Record | Details of any maintenance that has occurred on a sensor. |  | Created when a maintainer performs a check-up or maintenance on a sensor. |
| Employee | An employee of Spaces©. | Staff | Created when a new employee joins the Spaces©. |
| Maintainer | An employee of Spaces who maintains and checks the sensors. |  | A maintainer is created when a new employee is assigned that role. Estimated 60% of employees. |
| Administrative Executive | An employee of Spaces who approves contracts, gives discounts and performs other administrative work. | Admin | An administrative executive is created when a new employee is assigned that role. Estimated 2% of staff. |
| Salesperson | An employee of Spaces who represents the company and sells subscriptions to potential customers. |  | A salesperson is created when a new employee is assigned that role. Estimate 28% of employees. |

### Relationships

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Entity Name | Cardinality | Participation | Relationship | Participation | Cardinality | Entity Name |
| Subscriber | One | Partial | Pays for | Total | Many | Contract |
| Sensor Subscription | One | Total | Subscribes to | Partial | One | Sensor |
| Zone Contract | One | Partial | Contracts | Total | One | Zone |
| Standard | Many | Partial | Views | Partial | Many | Video Stream |
| Gold | Many | Partial | Controls direction | Partial | Many | Video Stream |
| Platinum | One | Partial | Has exclusive rights | Partial | Many | Data |
| Super Platinum | One | Partial | Has exclusive rights | Partial | Many | Video stream |
| Sensor | One | Partial | Records | Total | Many | Video Stream |
| Sensor | One | Partial | Records | Total | Many | Data |
| Sensor | Many | Total | Belongs to | Partial | One | Zone |
| Sensor | Many | Total | Has | Partial | Many | Parts |
| Sensor | One | Partial | Has | Total | Many | Maintenance Record |
| Sensor | Many | Total | Maintains | Partial | One | Maintainer |
| Parts | Many | Total | Supplied by | Partial | Many | Supplier |
| Parts | Many | Partial | Replaced | Partial | Many | Maintenance Record |
| Maintainer | One | Partial | Records | Total | Many | Maintenance Record |
| Admin Executive | One | Partial | Approves | Total | Many | Contract |
| Salesperson | One | Partial | Sells | Partial | Many | Contract |

### Attributes

Note: the data types for the data being recorded by sensors (audio, imagery) is unknown in what formats will be used and how the data will be packaged so I have put those as “data file”.

\* Spaces© will likely have standard base prices for standard and gold subscriptions which would be the default.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Entity Name | Attributes | Description | Domain | Aliases | Composite | Derived | Nulls | Key? | Default Value |
| Subscriber | Name | Subscriber name | Char(50) |  | First name + last name |  |  |  |  |
| Email | Contact email | Char(50) |  |  |  |  | y |  |
| Password | Account access | Char(10) |  |  |  |  |  |  |
| Address | Subscriber location | Char(100) |  |  |  | n |  |  |
| Contract | Contract No | ID | Int |  |  |  |  | Y |  |
| Start Date | Contract start | Date |  |  |  |  |  |  |
| End Date | Contract end | Date |  |  |  | n |  |  |
| Discount % | Any discount given | Int |  |  |  |  |  | 0 |
| Monthly Price | Cost each month | Int | Cost |  |  |  |  | \* |
| Sensor | Sensor ID | ID | Int |  |  |  |  | Y |  |
| Longitude | Location | Char(10) |  |  |  |  |  |  |
| Latitude | Location | Char(10) |  |  |  |  |  |  |
| Time | Current time | Time |  |  |  |  |  |  |
| Zone | Zone ID | ID | Int |  |  |  |  | y |  |
| Maintenance Record | Datetime | Datetime for history | Datetime |  |  |  |  | Y |  |
| Maintenance Notes | Log of maintenance changes/info | Char(254) |  |  |  |  |  | “Check-up, no issues.” |
| Parts | Part Name | Unique name | Char(254) |  |  |  |  | Y |  |
| Cost | Price, NZD | Decimal | Price |  |  |  |  |  |
| Supplier | Supplier Name | Company name | Char(254) |  |  |  |  | Y |  |
| Address | Location | Char(50) |  | Street address + suburb + city + country |  |  |  |  |
| Email | Contact email | Char(50) |  |  |  |  |  |  |
| Phone | Contact number | Int |  |  |  | n |  |  |
| Employee | Name | Employee name | Char(50) |  | First name + last name |  |  |  |  |
| Email | Contact email | Char(50) |  |  |  |  | Y |  |
| Phone | Contact phone | Int |  |  |  |  |  |  |
| Video Stream | Video Feed | Video data | Data file | Video, Video data |  |  |  |  |  |
| Video No | ID | Int |  |  |  |  | Y |  |
| Zone Audio | Audio in the zone picked up by sensor | Data file |  |  |  |  |  |  |
| Data Stream | Data No | ID | Int |  |  |  |  | Y |  |
| Human Voice | Isolated human voice audio | Data file |  |  |  |  |  |  |
| Shared Audio | Shared audio from other sources | Data file |  |  |  |  |  |  |
| Skeletal Points | Estimated human skeletal point data | Data file |  |  |  |  |  |  |
| Texture | Human texture data | Data file |  |  |  |  |  |  |
| Body Shape | Human body shape data | Data file |  |  |  |  |  |  |

## Business Rules

*Discuss the assumptions made about the business rules and the reasons for the choices you made.*

**Subscriptions vs Contracts**

A contract is an agreement between Spaces© and a subscriber (individual or company) to provide specified access to their services, both subscribing to a sensor or hosting a zone. Contracts are the top-level agreement. Essential, contracts are subscriptions are subcategories of the main contract entity.

Within contracts there are two subcategories:

* Contracts refer to large agreements, often made with a company. These are Zone contracts (platinum or super platinum contracts) which involve purchasing sensors and setting up a zone. These are more likely to have discounts.
* Subscriptions usually refer to smaller contracts with individuals to gain access to video feed. These fall into the category of sensor subscription (standard and gold subscriptions).

While these do have different conations, they can be used interchangeably to some degree. It is best not to do so, however to reduce confusion.

“A subscription is a contract.” – Project Brief, page 2, Subscriptions and contracts

**Subscribers and Subscriptions**

I have decided to have subscribers and contracts as separate entities. This is because a subscriber could have previously had a contract but does not currently (but their account still exists), or the subscriber could have subscriptions to multiple sensors (see next assumption). A business may also have a zone contract to host a zone and sensor subscriptions simultaneously.

**Multiple Subscriptions**

I have assumed that since each subscription is only to a single sensor, then, subscribers must be able to have multiple subscriptions in order to access other sensors, both in and outside the zone. This allows for easier management of contracts to different sensors and zones, and allows for individual prices and payment dates.

**Subscription Generalisation and Specialisation**

Each contract is represented by a base contract entity in the database (generalisation), which will then be connected to the relevant contract type entity (sensor or zone). I have chosen to do this as while they are both contracts, sensor subscriptions and zone contracts function differently so it is clearer to separate them into categories. Under the contract type entity is the subscription/contract level. These contract levels entities are connected as a subscription will include its relevant level access plus any lower-level subscription access. E.g., a gold level subscription will also have access to the same access that a standard subscription has.

**Direction Control**

A gold level subscription allows subscribers to change the direction of the video stream. Spaces© has not specified how this would function in terms of having multiple gold watching a single video stream. I have decided to use a “first-in first served” based. What this means is that the first gold subscriber to start watching a video stream is grated the ability to control the direction. Any other golds subscribers watching at the same time will not be able to control the direction, as this could result in “fights” constantly changing the direction and disrupting an enjoyable video stream. When the gold subscriber in charge leaves the video stream, the gold subscriber who joined next will be granted the permission. If other gold subscribers are watching when one joins, they will gain access straight away, and if there are none at all, the direction cannot be changed. This could be thought about like a queue of gold subscribers, and whoever is at the top has control. This would eliminate any issues of conflict between gold subscribers.

**Sensor Video Streams**

Sensors can have any number of video streams, however this number is limited to 100 maximum. Having an unlimited amount could result in overloading the capacity of the sensor and the network connection to the servers and database.

**Considerations**

There were a couple of changes I considered but decided against.

* Subscriber and employee could be combined as a “person” superclass with employee and subscriber as subclasses, as there is some duplication between attributes. However, I think it is best to keep separate as they are quite different in how they interact with the rest of the database and from a business perspective (customer vs worker).
* Subscriber could include date of birth which could have the derived attribute age. This would allow Spaces© to have age restricted zones (perhaps for some concerts or various events). Spaces© has not mentioned any interest in this and I do not see any other reason to store subscribers’ age, so I have not included it.
* Subscriber could include payment details, but I think it is best for payments to go through an external payment system for better security.

**Audio**

Originally all audio was categorised as sensor data by Spaces©, which required a platinum subscription or higher. This implies that the video streams would have no audio. I think that this is a bad and unusual business decision, as subscribers will want to hear audio from the zone. I have decided to add “zone audio” as an attribute to the video stream entity. This will include all zone audio picked up by the sensor. Any other audio, such as shared audio or isolated human voice data will still be part of the data entity.

**IDs**

I decided to create a couple of ID attributes in the case of no appropriate candidate keys.

For example, a contract could have the same start and end dates as other contracts. Monthly price will be similar between lots of the contracts, especially for standard and gold subscriptions. Discount will also have many duplicate values between contracts. None of these are good candidate keys, so I have decided to create Contract No. This is the same for all other IDs created.

**Weak/Strong Entities**

I have included four weak entities:

1. Contract – By definition a contract is an agreement between the subscriber and Spaces©, so a contract cannot exist without a subscriber.
2. Video Stream – A video stream cannot exist without being recorded by a sensor.
3. Data - A data stream cannot exist without being recorded by a sensor.
4. Maintenance Record – A maintenance record cannot exist without a sensor, as the record is for maintenance on a sensor.

I considered having parts as a weak entity owned by a supplier, but since a part can be supplied by multiple suppliers, it is independent of supplier and can exist on its own, so it must be a strong entity.

The same situation happened with sensors and zones. Since sensors can be in multiple zones, they are independent and can’t be weak entities.

# T-SQL Queries

Note 1: all SQL queries are in a separate SQL file from GitHub: <https://github.com/Jayden-Htn/DAT601_TSQL_Practice.git>

Note 2: the queries script is set to use small business database. Please make sure that this name and the populated database have the same name.

## Basic Queries

**Query 1**

|  |  |
| --- | --- |
| A picture containing text, font, screenshot, line  Description automatically generated | Graphical user interface, text, application, table  Description automatically generated |

**Query 2**

|  |  |
| --- | --- |
| **A group of words on a white background  Description automatically generated with low confidence** |  |

**Query 3**

|  |  |
| --- | --- |
| Graphical user interface, text, website  Description automatically generated | **A picture containing text, screenshot, font, number  Description automatically generated** |

**Query 4**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, line  Description automatically generated** | **A picture containing text, font, screenshot, line  Description automatically generated** |

**Query 5**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, line  Description automatically generated** | **A screenshot of a product list  Description automatically generated with low confidence** |

**Query 6**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, white  Description automatically generated** | **A picture containing text, font, screenshot, number  Description automatically generated** |

**Query 7**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, line  Description automatically generated** | **A picture containing text, font, screenshot, line  Description automatically generated** |

**Query 8**

|  |  |
| --- | --- |
| **A picture containing text, screenshot, font, line  Description automatically generated** |  |

**Query 9**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, white  Description automatically generated** |  |

Note: this answer could be rounded to a specified number of decimal points using the ROUND() function.

**Query 10**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, design  Description automatically generated** | **A picture containing text, screenshot, font, line  Description automatically generated** |

**Query 11**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, white  Description automatically generated** |  |

**Query 12**

|  |  |
| --- | --- |
|  |  |

****

**Query 13**

|  |  |
| --- | --- |
| **A picture containing text, font, line, screenshot  Description automatically generated** | **A picture containing text, screenshot, font, number  Description automatically generated** |

**Query 14**

|  |  |
| --- | --- |
|  | **A picture containing text, screenshot, font, number  Description automatically generated** |

Note: change population script to have correct value (5-> 50 for Raggedy Ann).

## Sub Queries

**Query 15**

|  |  |
| --- | --- |
| **A picture containing text, font, screenshot, line  Description automatically generated** | **A close-up of numbers  Description automatically generated with low confidence** |
| **A picture containing text, font, screenshot, number  Description automatically generated** |  |
|  | **A picture containing text, font, screenshot, line  Description automatically generated** |

**Query 16**

|  |  |
| --- | --- |
| **A picture containing text, screenshot, font, line  Description automatically generated** | **A screenshot of a computer  Description automatically generated with low confidence** |

## Combined Query

**Query 17**

|  |  |
| --- | --- |
| **A screenshot of a computer code  Description automatically generated with low confidence** | **A picture containing text, screenshot, font, line  Description automatically generated** |

Note: this query could be done with a union, but a subquery approach feels much more logical to me.

## Views

**Query 18**

|  |  |
| --- | --- |
|  | **A picture containing text, font, screenshot, line  Description automatically generated** |

**Query 19**

|  |  |
| --- | --- |
|  |  |
| **A screenshot of a phone number  Description automatically generated with low confidence** |  |
|  |  |
| **A screenshot of a computer  Description automatically generated with low confidence** |  |
|  |  |
| **A screenshot of a computer  Description automatically generated with low confidence** |  |

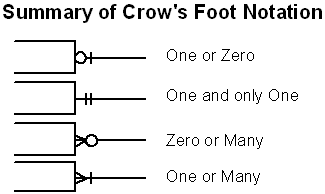
# Model Conversion

*Describe how to transform a conceptual model based on CHEN ERDs to a logical model. Create a list of “mapping rules”.*

Once a conceptual model has been created, it is a fairly simple process to convert this into a logical diagram. This guide will be focussed on converting a Chen’s Notation conceptual model to a logical ERD.

The 6 steps will help guide this process:

1. Create Tables
   1. Entities will become tables.
   2. Create new tables for any relationship attributes and connect with an appropriate key.
2. Update Attributes
   1. Attributes will be added as columns in their respective tables.
   2. Convert composite attributes to individual attributes.
   3. Multi-valued attributes become another weak table with a composite key of parent entity ID and item ID.
   4. Add data types and lengths if necessary.
3. Select Primary Keys
   1. Convert appropriate candidate key into primary keys.
   2. Create ID columns if no appropriate candidate keys are present for a table.
4. Create Relationships
   1. 1:1 - Add the foreign key to either table.
   2. 1:M – Add the foreign key to the many table.
   3. Create join tables for many to many relationships.
   4. Add the foreign key to the join table from both tables in the relationship.
5. Super/Subclass
   1. There are multiple options for converting a conceptual Chen Notation generalisation/specialisation structure. Select the method that is most appropriate for the database design and most logical to the designer /developer.
   2. Option 1 – Make one big table that includes all super and subclass attributes. Irrelevant fields will be null.
   3. Option 2 – Create a superclass table with the shared attributes and subclass tables. The subclass tables will have foreign keys to the superclass.
   4. Option 3 – Create a superclass table with the shared attributes and subclass table. The super and subclasses will share primary keys.
6. Check Crow’s Foot Notation
   1. Make sure all key connections have the correct Crow’s Foot notation for the relationship.
      1. Single goes to just a line.
      2. Many goes to crows foot.
      3. Optional adds a circle.
      4. Mandatory/one adds a perpendicular line.



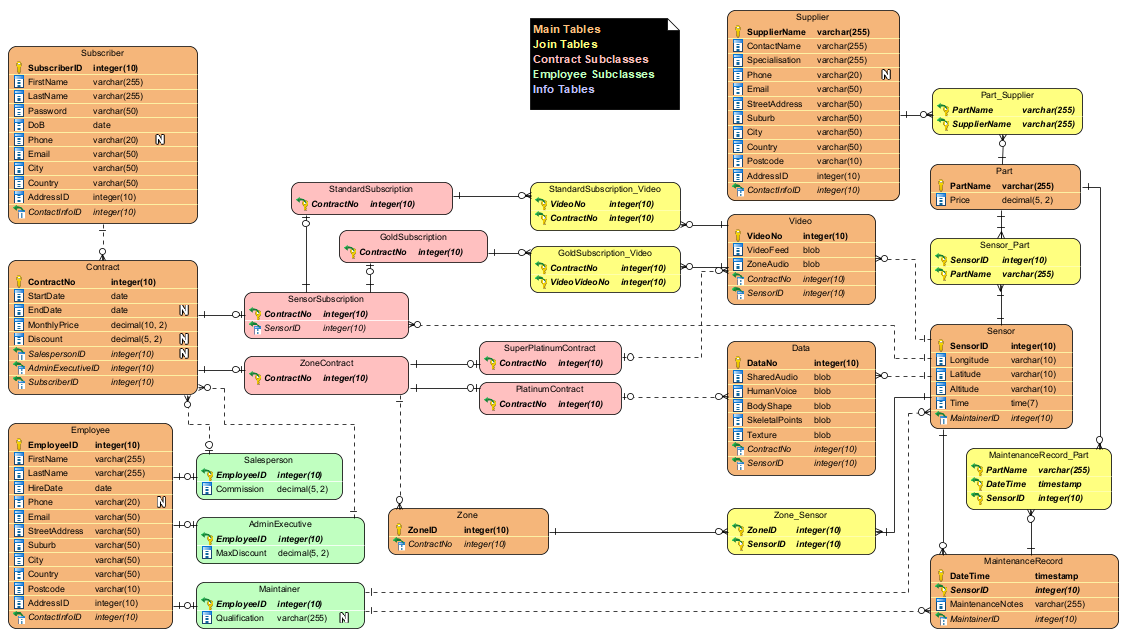
(Stewart, 2008)

# Logical ERD

*Create an ERD using an appropriate notation. Accompany this with your rationale; describe and explain the reasoning and purpose of all parts of the model. This is to demonstrate how you applied the “mapping rules” from the Conceptual to the Logical Model.*

Both diagrams can be viewed as separate attached images.

**Unnormalised ERD:**



The logical ERD was fairly simple to build due to my high-quality conceptual model. The mapping rules (view previous section) created a straightforward and simple guide to follow. I followed the process in the same step-by-step order as in the guide, starting with the tables and attributes, then adding keys and relationship details.

Most data types were very self-explanatory as to what to use, however I have selected the blob type for audio, video and 3d imagery data. I selected this as Spaces© has not specified what data formats they will be using to store data. Blob gives them the most flexibility in what is stored, and this can always be changed in the physical implementation if they require it.

IT may be beneficial for Spaces© to add in a manager attribute and relationship in the employee table, so they can store and see the company hierarchy. However, they have not stated that they wish to do this, so it has been excluded.

In the conceptual model, contract was a weak entity of subscriber, as a contract cannot exist without a subscriber. As contracts can identify themselves just with the contract number, it is not necessary to have the subscriber ID as well. I have decided to convert contract into a strong entity.

Both video and data entities are also now strong entities as with their own primary key identifiers, the parent sensor ID is not required to identify them. I think this makes more logical sense in the current ERD.

One major business change that has been made is allowing zone contracts to have multiple zones. This is a large business change but has a very minor effect on the ERD (1:1 -> 1:M).

I have also added a couple of minor attributes to the ERD as well, such as a hire date to employee and more contact information to subscribers, as I believe that this is useful information for Spaces©.

I considered storing both the date of birth and current age, where current age is calculated from the DoB, however I decided that it was simpler to just calculate age when needed from DOB instead of having to store and update it daily.

# Normalisation

## Forms and Steps

*Include a description of normalisation to 4th Normal form.*

Normalisation is the process of reworking a database to remove errors, anomalies and duplications within the data, based upon specified rules (Muñoz, 2022). Removing duplicate data will reduce the risk of data conflicts and anomalies from occurring, reducing future errors and other issues that may occur. Normalisation can also help improve efficiency and performance in some cases (Romani, 2023). There are many different normal forms, however this assignment will only discuss up to Boyce-Codd and 4th normal form.

As described by Romani (Romani, 2023), the following describes normalisation up to the fourth form.

### Unnormalised (0NF)

* A database with no normalisation.

*Book Table*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Book | Title | Author 1 | Author 2 | Author Name | Genre | Genre Description |
| 1 | Todd’s Terrific Databases | 1 | 2 | Todd Cochrane, Jayden Houghton | Educational | Educational IT non-fiction. |

### First Normal Form (1NF)

* Remove duplicate columns (repeated data).
* No multi-value attributes (atomic data).
* Select primary keys.

*Book Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Book ID | Title | Genre | Genre Description |
| 1 | Todd’s Terrific Databases | Educational | Educational IT non-fiction. |

*Book-Author Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Book ID | Author ID | First Name | Last Name |
| 1 | 1 | Todd | Cochrane |
| 1 | 2 | Jayden | Houghton |

### Second Normal Form (2NF)

* Requires 1NF.
* Full functional dependency on keys.

*Book Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Book ID | Title | Genre | Genre Description |
| 1 | Todd’s Terrific Databases | Educational | Educational IT non-fiction. |

*Book- Author Table*

|  |  |
| --- | --- |
| Book | Author ID |
| 1 | 1 |
| 1 | 2 |

*Book-Author Table*

|  |  |  |
| --- | --- | --- |
| Author ID | First Name | Last Name |
| 1 | Todd | Cochrane |
| 1 | Jayden | Houghton |

### Third Normal Form (3NF)

* Requires 2NF.
* Remove transitive dependencies (non-key attributes dependant on other non-key attributes).

Often the first 3 forms are done together as they work hand in hand.

*Book Table*

|  |  |  |  |
| --- | --- | --- | --- |
| Book ID | Title | Genre | Genre Description |
| 1 | Todd’s Terrific Databases | Educational | Educational IT non-fiction. |

*Book-Genre Table*

|  |  |
| --- | --- |
| Book ID | Genre ID |
| 1 | 1 |

*Book Table*

|  |  |  |
| --- | --- | --- |
| Genre ID | Genre Name | Genre Description |
| 1 | Educational | Educational IT non-fiction. |

*Book- Author Table*

|  |  |
| --- | --- |
| Book | Author ID |
| 1 | 1 |
| 1 | 2 |

*Book-Author Table*

|  |  |  |
| --- | --- | --- |
| Author ID | First Name | Last Name |
| 1 | Todd | Cochrane |
| 1 | Jayden | Houghton |

### Boyce-Codd Normal Form (BCNF)

* Requires 3NF.
* 3NF focusses on removing functional dependencies on non-key attributes, where as BCNF does this for potential key attributes.

Sometimes BCNF is referred to as 3.5 normal form, as it is an extension on the third form. It is less commonly used as it doesn’t always apply to a database and can cause over-normalisation, potentially affecting database performance.

### Fourth Normal Form (4NF)

* Requires BCNF.
* Remove any multi-valued dependency for tables with at least 3 columns.

4NF is very rarely used and is mostly a theoretical practice to further reduce database anomalies.

## Normalised ERD

*Make sure your relations are fully normalised (to 3NF, consider Boyce Codd and 4th Normal forms). Document your relations. Describe how you undertook normalisation of your database design.*

Both ERDs can be viewed as separate attached images.

**Normalised ERD:**A picture containing text, screenshot, font, number

Description automatically generated

All tables are normalised, view the data dictionary or NaLER analysis for a list of relations.

Because of my high-quality conceptual model, after following the mapping rules as stated previously, the ERD fulfilled most normalisation requirements already. The only changes I implemented were separating out the address fields (street address, suburb, city, country, postcode) and contact details (phone number, email address, backup email address) into their own tables. I did this to remove the duplication of those fields between the employee, supplier and subscriber tables. After this change, the ERD meets 3NF.

I also considered moving *FirstName* and *LastName* into the contact table as the fields are duplicated, however this feels like over-normalisation and bad practice.

After normalising to 3NF, I considered any possible BCNF or 4NF issues that would require these normalisation forms to be implemented. I did not identify any violations of these forms and I believe that the database ERD already fulfils these requirements.

# Data Dictionary

*Documentation of your relations. Include a data dictionary – this must be in tabular form as given in class. Document your logical model in a data dictionary.*

* *Entities (e.g. name, start volume, growth)*
* *Attributes (e.g. name, description, data type, length, value range, validation rules, default value, nulls, keys, entity references, integrity constraints)*

## Document relations

|  |  |  |  |
| --- | --- | --- | --- |
| Relation Name | Start Volume No. of rows loaded at the beginning | Growth e.g. no growth / 10% per year | Comments |
| Subscriber | 1,000,000 | 10% |  |
| Employee | 6,000 | 5% | Staff required to grow to match demand |
| Supplier | 100 | 2 | Suppliers should be quite consistent as the required parts won’t change much |
| ContactInfo | 1,006,100 | 15% | All staff, suppliers and employees will have a contact |
| Address | 7,000 | 8,200 | Supplier, staff and zone contracts |
| Salesperson | 900 | 30 |  |
| Admin Executive | 100 | 5 |  |
| Maintainer | 5,000 | 100 | Maintain ~100 sensors each, lots are part-time contractors |
| Contract | 2,000,000 | 10% | Subscribers can have multiple contracts |
| Sensor Subscription | 1,997,900 | 10% | Majority of contracts |
| Standard Subscription | 1,700,000 | 10% | Majority of sensor subscriptions |
| Gold Subscription | 247,900 | 5% |  |
| Standard Subscription \_Video | 1,500,000,000 | 20% | Assume max 100 video streams |
| Gold Subscription \_Video | 447,900,000 | 20% | Assume max 100 video streams |
| Zone Contract | 2,100 | 110 |  |
| Platinum Contract | 2,000 | 100 |  |
| Super Platinum Contract | 100 | 10 |  |
| Zone | 50,000 | 5% | Similar rate as sensors |
| Zone\_Sensor | 600,000 | 10% | Sensors can be in multiple zones, so the amount will be greater than the sensor number |
| Data | 3,000,000,000 | 10% | Sensor\*100 video streams\*60fps month (note) |
| Video | 3,000,000,000 | 10% | Sensor\*100 video streams\*60fps |
| Sensor | 500,000 | 5% |  |
| Maintenance Record | 1,000,000,000 | 5% | Assume 2 per sensor (unknow how long the business has been operating, under 10y operations) |
| Maintenance Record\_Part | 1,000,000,000 | 5% | Some wont need parts, some will need multiple |
| Part | 400 | 5 | Parts likely won’t change much unless a new sensor is implemented |
| Sensor\_Part | 200,000,000 | 8% | parts\*sensors |
| Part\_Supplier | 500 | 4 | Some suppliers produce multiple |

Note: 1 month of data is stored as a backup on the sensor, but there is no backup in the database system itself.

## Document Attributes

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Relation Name | Attribute | Description | Data type | Length | Value range | Validation Rules | Default Value | Null | Key | References Entity | Integrity Constraints |
| Contract | ContractNo | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| StartDate | When the contract started | date |  |  |  |  |  |  |  |  |
| EndDate | Agreed end date if applicable | date |  |  |  |  | N |  |  |  |
| MonthlyPrice | Cost of subscription/contract | decimal | 10,2 | 0< |  |  |  |  |  |  |
| Discount | The discount % applied | decimal | 5,2 | 0<d<100 |  |  | N |  |  |  |
| SalespersonID | Salesperson who sold the contract | int | 10 | 0< |  |  | N |  | Salesperson | FK |
| AdminExecutiveID | Admin Exec who approved the contract | int | 10 | 0< |  |  |  |  | Admin  Executive | FK |
| SubscriberID | Subscriber who made the contract | int | 10 | 0< |  |  |  |  | Maintainer | FK |
| Subscriber | SubscriberID | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| FirstName | Subscriber first name | varchar | 255 |  |  |  |  |  |  |  |
| LastName | Subscriber last name | varchar | 255 |  |  |  |  |  |  |  |
| Password | Account password | varchar | 50 |  | Inc. num and symbol |  |  |  |  |  |
| DoB | Date of birth | date |  |  | 1900-(today-min age) |  |  |  |  |  |
| AddressID | Address link | int | 10 | 0< |  |  |  |  | Address | FK |
| ContactInfoID | Contact link | int | 10 | 0< |  |  |  |  | ContactInfo | FK |
| Employee | EmployeeID | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| FirstName | Employee first name | varchar | 255 |  |  |  |  |  |  |  |
| LastName | Employee last name | varchar | 255 |  |  |  |  |  |  |  |
| HireDate | Date hired | date |  |  |  | Today’s date |  |  |  |  |
| AddressID | Address link | int | 10 | 0< |  |  |  |  | Address | FK |
| ContactInfoID | Contact link | int | 10 | 0< |  |  |  |  | ContactInfo | FK |
| Supplier | SupplierName | Identifier | varchar | 255 |  |  |  |  | PK |  |  |
| ContactName | First name | varchar | 255 |  |  |  |  |  |  |  |
| Specialisation | Area of manufacturing | varchar | 255 |  |  |  |  |  |  |  |
| AddressID | Address link | int | 10 | 0< |  |  |  |  | Address | FK |
| ContactInfoID | Contact link | int | 10 | 0< |  |  |  |  | ContactInfo | FK |
| ContactInfo | ContactID | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| Phone | Phone number (mobile or landline) | varchar | 20 |  |  |  | N |  |  |  |
| Email | Email address | varchar | 50 |  |  |  |  |  |  |  |
| SecondaryEmail | Back up email | varchar | 50 |  |  |  | N |  |  |  |
| Address | AddressID | Identifier | int | 10 | 0< |  | Auto | N | PK |  |  |
| StreetAddress | Street and number | varchar | 50 |  |  |  | N |  |  |  |
| Suburb | Suburb name | varchar | 50 |  |  |  | N |  |  |  |
| City | City name | varchar | 50 |  |  |  |  |  |  |  |
| Country | Country | varchar | 50 |  |  |  |  |  |  |  |
| Postcode | Postcode | varchar | 10 |  |  |  | N |  |  |  |
| Sensor\_  Subscription | ContractNo | Superclass key | int | 10 |  | =superclass |  |  | PK | Contract | FK |
| SensorID | Identifier | int | 10 |  |  |  |  |  | Sensor | FK |
| Zone\_ Contract | ContractNo | Superclass key | int | 10 |  | =superclass |  |  | PK | Contract | FK |
| Standard  Subscription | ContractNo | Superclass key | int | 10 |  | =superclass |  |  | PK | Sensor Subscription | FK |
| Gold Subscription | ContractNo | Superclass key | int | 10 |  | =superclass |  |  | PK | Sensor Subscription | FK |
| Super Platinum  Contract | ContractNo | Superclass key | int | 10 |  | =superclass |  |  | PK | Zone Contract | FK |
| Platinum Contract | ContractNo | Superclass key | int | 10 |  | =superclass |  |  | PK | Zone Contract | FK |
| Standard  Subscription  \_Video | VideoNo | Identifier | int | 10 |  |  |  |  | PK | Video | FK |
| ContractNo | Superclass key | int | 10 |  | <100 per contract |  |  | PK | Standard  Subscription | FK |
| Gold Subscription  \_Video | VideoNo | Identifier | int | 10 |  |  |  |  | PK | Video | FK |
| ContractNo | Superclass key | int | 10 |  |  |  |  | PK | Gold  Subscription | FK |
| Salesperson | EmployeeID | Identifier | int | 10 |  | =superclass |  |  | PK | Employee | FK |
| Commission | Commission per sale | decimal | 5,2 |  |  | 0 |  |  |  |  |
| Admin Executive | EmployeeID | Identifier | int | 10 |  | =superclass |  |  | PK | Employee | FK |
| MaxDiscount | Max discount allowed | decimal | 5,2 |  |  | 0 |  |  |  |  |
| Maintainer | EmployeeID | Identifier | int | 10 |  | =superclass |  |  | PK | Employee | FK |
| Qualification | Qualification/certifications |  |  |  |  |  | N |  |  |  |
| Zone | ZoneID | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| ContractNo | Superclass key |  |  |  | =superclass |  |  |  | Zone Contract | FK |
| Zone\_ Sensor | ZoneID | Identifier | int | 10 |  |  |  |  | PK | Zone | FK |
| SensorID | Identifier | int | 10 |  |  |  |  | PK | Sensor | FK |
| Data | DataNo | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| SharedAudio | Shared audio from other sources | blob |  |  |  |  |  |  |  |  |
| HumanVoice | Recorded human voice | blob |  |  |  |  |  |  |  |  |
| BodyShape | 3d human body shape | blob |  |  |  |  |  |  |  |  |
| SkeletalPoints | Estimated 3d human skeletal points | blob |  |  |  |  |  |  |  |  |
| Texture | Human texture | blob |  |  |  |  |  |  |  |  |
| ContractNo | Superclass key | int | 10 |  |  |  |  |  | Platinum Contract | FK |
| SensorID | Identifier | int | 10 |  |  |  |  |  | Sensor | FK |
| Video | VideoNo | Identifier | int | 10 | 0< |  | Auto |  |  |  |  |
| VideoFeed | Recorded video | blob |  |  |  |  |  |  |  |  |
| ZoneAudio | Zone audio | blob |  |  |  |  |  |  |  |  |
| ContractNo | Superclass key | int | 10 |  |  |  |  |  | Standard Subscription  \_Video | FK |
| SensorID | Identifier | int | 10 |  |  |  |  |  | Sensor | FK |
| Sensor | SensorID | Identifier | int | 10 | 0< |  | Auto |  | PK |  |  |
| Longitude | Longitude position | varchar | 10 |  |  |  |  |  |  |  |
| Latitude | Latitude position | varchar | 10 |  |  |  |  |  |  |  |
| Altitude | Altitude | varchar | 10 |  |  |  |  |  |  |  |
| Time | Current time | time | 7 |  |  |  |  |  |  |  |
| MaintainerID | Identifier | int | 10 |  |  |  |  |  | Maintainer | FK |
| Maint-enance  Record | DateTime | Date and time | timestamp |  |  |  |  |  | PK |  |  |
| SensorID | Identifier | int | 10 |  |  |  |  | PK | Sensor | FK |
| Maintenance  Notes | Observations, part changes, issues, etc. | varchar | 255 |  |  | Routine check. |  |  |  |  |
| MaintainerID | Identifier | int | 10 |  |  |  |  |  | Maintainer | FK |
| Maintenance  Record\_Part | PartName | Part name | varchar | 255 |  | <today |  |  | PK | Part | FK |
| DateTime | Maintenance date+time | timestamp |  |  |  |  |  | PK | Maintenance Record | FK |
| SensorID | Identifier | int | 10 | 0< |  |  |  | PK | Maintenance Record | FK |
| Part | PartName | Identifier | varchar | 255 |  |  |  |  | PK |  |  |
| Price | Item price | decimal | 5,2 | 0< |  |  |  |  |  |  |
| Sensor\_Part | SensorID | Identifier | int | 10 |  |  |  |  | PK | Sensor | FK |
| PartName | Identifier | varchar | 255 |  |  |  |  | PK | Part | FK |
| Part\_Supplier | PartName | Identifier | varchar | 255 |  |  |  |  | PK | Part | FK |
| SupplierName | Identifier | varchar | 255 |  |  |  |  | PK | Supplier | FK |

# NaLER Analysis

*Carry out a NaLER analysis on the ERD. Make any changes to your logical model.*

## Process

NaLER is a natural language method for interpreting ERDs. It uses present sentence formats to identify any errors or holes in an ERD.

A NaLER analysis can be done in 6 steps:

1. Identify and document diagram conventions.
2. Perform a syntax check of the model.
3. Construct sentences attribute and binary relationships
   1. Construct a sentence for the primary key attributes.
   2. Construct a sentence for each non-key attribute.
   3. Construct a sentence for each binary relationship.
4. Populate the sentences from 3.2 ad 3.3. with valid examples.
5. Construct m:m relationship sentences
   1. Construct a sentence for any ternary or higher m:m relationships with a composite name.
   2. Construct a sentence for any ternary or higher m:m relationships with clear identifying name.
6. Produce a NaLER description by listing all constructed sentences with examples.

## Analysis

**Conventions:**

* Pascal case is used for all naming (e.g. FirstName).
* Table names are singular (e.g. Customer not Customers).
* Join tables join the table names with an underscore (e.g. Zone\_Sensor).
* Crow’s foot notation is used to show relationship nature.

### Entities

Each Subscriber is uniquely identified by a SubscriberID.

*One* Subscriber *identified by* SubscriberID *must have one* FirstName.

*One* Subscriber *identified by* SubscriberID *must have one* LastName.

*One* Subscriber *identified by* SubscriberID *must have one* Password.

*One* Subscriber *identified by* SubscriberID *must have one* DoB.

*One* Subscriber *identified by* SubscriberID *must have one* AddressID.

*One* Subscriber *identified by* SubscriberID *must have one* ContactInfoID.

Each Employee is uniquely identified by a EmployeeID.

*One* Employee *identified by* EmployeeID *must have one* FirstName.

*One* Employee *identified by* EmployeeID *must have one* LastName.

*One* Employee *identified by* EmployeeID *must have one* HireDate.

*One* Employee *identified by* EmployeeID *must have one* AddressID.

*One* Employee *identified by* EmployeeID *must have one* ContactInfoID.

Each Supplier is uniquely identified by a SupplierName.

*One* Supplier *identified by* SupplierName *must have one* ContactName.

*One* Supplier *identified by* SupplierName *must have one* Specialisation.

*One* Supplier *identified by* SupplierName *must have one* AddressID.

*One* Supplier *identified by* SupplierName *must have one* ContactInfoID.

Each ContactInfo is uniquely identified by a ContactInfoID.

*One* ContactInfo *identified by* ContactInfoID *may have one* Phone.

*One* ContactInfo *identified by* ContactInfoID *must have one* Email.

*One* ContactInfo *identified by* ContactInfoID *may have one* SecondaryEmail.

Each Address is uniquely identified by a AddressID.

*One* Address *identified by* AddressID *may have one* StreetAddress.

*One* Address *identified by* AddressID *may have one* Suburb.

*One* Address *identified by* AddressID *must have one* City.

*One* Address *identified by* AddressID *must have one* Country.

*One* Address *identified by* AddressID *may have one* Postcode.

Each Salesperson is uniquely identified by a EmployeeID.

*One* Salesperson *identified by* EmployeeID *must have one* Commission.

Each AdminExecutive is uniquely identified by a EmployeeID.

*One* AdminExecutive *identified by* EmployeeID *must have one* MaxDiscount.

Each Maintainer is uniquely identified by a EmployeeID.

*One* Maintainer *identified by* EmployeeID *may have one* Certification.

Each Contract is uniquely identified by a ContractNo.

*One* Contract *identified by* ContractNo *must have one* StartDate.

*One* Contract *identified by* ContractNo *may have one* EndDate.

*One* Contract *identified by* ContractNo *must have one* MonthlyPrice.

*One* Contract *identified by* ContractNo *may have one* Discount.

*One* Contract *identified by* ContractNo *must have one* Discount.

*One* Contract *identified by* ContractNo *may have one* SalespersonID.

*One* Contract *identified by* ContractNo *must have one* AdminExecutiveID.

*One* Contract *identified by* ContractNo *must have one* SubscriberID.

Each SensorSubscription is uniquely identified by a ContractNo.

*One* SensorSubscription *identified by* ContractNo *must have one* SensorID.

Each StandardSubscription is uniquely identified by a ContractNo.

Each GoldSubscription is uniquely identified by a ContractNo.

Each StandardSubscription\_Video is uniquely identified by a ContractNo and a VideoNo.

Each GoldSubscription\_Video is uniquely identified by a ContractNo and a VideoNo.

Each ZoneContract is uniquely identified by a ContractNo.

Each PlatinumContract is uniquely identified by a ContractNo.

Each SuperPlatinumContract is uniquely identified by a ContractNo.

Each Zone is uniquely identified by a ZoneID.

*One* Zone *identified by* ZoneID *must have one* ContractNo.

Each Zone\_Sensor is uniquely identified by a ZoneID and a SensorID.

Each Data is uniquely identified by a DataNo.

*One* Data *identified by* DataNo *must have one* SharedAudio.

*One* Data *identified by* DataNo *must have one* HumanVoice.

*One* Data *identified by* DataNo *must have one* BodyShape.

*One* Data *identified by* DataNo *must have one* SkeletalPoints.

*One* Data *identified by* DataNo *must have one* Texture.

*One* Data *identified by* DataNo *must have one* ContractNo.

*One* Data *identified by* DataNo *must have one* SensorID.

Each Video is uniquely identified by a VideoNo.

*One* Video *identified by* VideoNo *must have one* VideoFeed.

*One* Video *identified by* VideoNo *must have one* ZoneAudio.

*One* Video *identified by* VideoNo *must have one* ContractNo.

*One* Video *identified by* VideoNo *must have one* SensorID.

Each Sensor is uniquely identified by a SensorID.

*One* Sensor *identified by* SensorID *must have one* Longitude.

*One* Sensor *identified by* SensorID *must have one* Latitude.

*One* Sensor *identified by* SensorID *must have one* Altitude.

*One* Sensor *identified by* SensorID *must have one* Time.

*One* Sensor *identified by* SensorID *must have one* MaintainerID.

Each MaintenanceRecord is uniquely identified by a DateTime and a SensorID

*One* MaintenanceRecord *identified by* DateTime and SensorID *must have one* MaintenanceNotes.

*One* MaintenanceRecord *identified by* DateTime and SensorID *must have one* MaintainerID.

Each MaintenanceRecord\_Part is uniquely identified by a PartName, DateTime and a SensorID.

Each Part is uniquely identified by a PartName.

*One* Part *identified by* PartName *must have one* Price.

Each Sensor\_Part is uniquely identified by a SensorID and a PartName.

Each Part\_Supplier is uniquely identified by a PartName and a SupplierName.

### Relationships

The relationships are grouped by table. Each relationship has a constructed sentence and an example sentence.

*One* Subscriber *identified by* SubscriberID *must have one* ContactInfo *identified by* ContactInfoID.

*One* Subscriber *identified by* SubscriberID *must have one* ContactInfo *identified by* ContactInfoID.

*One* Subscriber *identified by* SubscriberID *must have one* Address *identified by* AddressID.

*One* Subscriber *identified by* SubscriberID *may pay for one or more* Contracts *identified by* ContractNo.

*One* Employee *identified by* EmployeeID *must have one* ContactInfo *identified by* ContactInfoID.

*One* Employee *identified by* EmployeeID *must have one* Address *identified by* AddressID.

*One* Employee *identified by* EmployeeID *may also be one* Salesperson *identified by* EmployeeID.

*One* Employee *identified by* EmployeeID *may also be one* AdminExecutive *identified by* EmployeeID.

*One* Employee *identified by* EmployeeID *may also be one* Maintainer *identified by* EmployeeID.

*One* Salesperson *identified by* EmployeeID may have sold one or more Contracts *identified by* ContractNo.

*One* Salesperson *identified by* EmployeeID belongs to one Employee *identified by* EmployeeID.

*One* AdminExecutive *identified by* EmployeeID may have approved (and discounted) one or more Contracts *identified by* ContractNo.

*One* AdminExecutive *identified by* EmployeeID belongs to one Employee *identified by* EmployeeID.

*One* Maintainer *identified by* EmployeeID may be maintaining one or more Sensors *identified by* SensorID.

*One* Maintainer *identified by* EmployeeID may have written one or more maintenanceRecords *identified by* DateTime and SensorID.

*One* Maintainer *identified by* EmployeeID belongs to one Employee *identified by* EmployeeID.

*One* Supplier *identified by* SupplierID *must have one* ContactInfo *identified by* ContactInfoID.

*One* Supplier *identified by* SupplierID *must have one* Address *identified by* AddressID.

*One* Supplier *identified by* SupplierID *may be the Part\_Supplier for one or more* Parts *identified by* PartName.

*One* ContactInfo *identified by* ContactInfoID *may have one* Employee *identified by* EmployeeID.

*One* ContactInfo *identified by* ContactInfoID *may have one* Subscriber *identified by* SubscriberID.

*One* ContactInfo *identified by* ContactInfoID *may have one* Supplier *identified by* SupplierName.

*One* Address *identified by* AddressID *may have one* Employee *identified by* EmployeeID.

*One* Address *identified by* AddressID *may have one* Subscriber *identified by* SubscriberID.

*One* Address *identified by* AddressID *may have one* Supplier *identified by* SupplierName.

*One* Contract *identified by* ContractNo may be sold by one or more Salespeople *identified by* EmployeeID.

*One* Contract *identified by* ContractNo must be approved (and discounted) by one AdminExecutive *identified by* EmployeeID.

*One* Contract *identified by* ContractNo must be paid for/owned by one Subscriber *identified by* SubscriberID.

*One* Contract *identified by* ContractNo may include one SensorSubscription *identified by* ContractNo.

*One* Contract *identified by* ContractNo may include one ZoneContract *identified by* ContractNo.

*One* SensorSubscription *identified by* ContractNo may include one StandardSubscription *identified by* ContractNo.

*One* SensorSubscription *identified by* ContractNo may include one GoldSubscription *identified by* ContractNo.

*One* SensorSubscription *identified by* ContractNo must have one Sensor *identified by* SensorID.

*One* SensorSubscription *identified by* ContractNo must belong to one Contract *identified by* ContractNo.

*One* StandardSubscription *identified by* ContractNo may be the StandardSubscription\_Video (viewed) for one or more Videos *identified by* VideoNo.

*One* StandardSubscription *identified by* ContractNo must belong one Contract *identified by* ContractNo.

*One* GoldSubscription *identified by* ContractNo may be the GoldSubscription\_Video (controlled) for one or more Videos *identified by* VideoNo.

*One* GoldSubscription *identified by* ContractNo must belong one Contract *identified by* ContractNo.

*One* ZoneContract *identified by* ContractNo may include one PlatinumContract *identified by* ContractNo.

*One* ZoneContract *identified by* ContractNo may include one SuperPlatinumContract *identified by* ContractNo.

*One* ZoneContract *identified by* ContractNo must have one Zone *identified by* ZoneID.

*One* ZoneContract *identified by* ContractNo must belong to one Contract *identified by* ContractNo.

*One* PlatinumContract *identified by* ContractNo may link to one or more pieces of Data *identified by* DataNo.

*One* PlatinumContract *identified by* ContractNo must belong one ZoneContract *identified by* ContractNo.

*One* SuperPlatinumContract *identified by* ContractNo may have exclusive access to one or more Videos *identified by* VideoNo.

*One* SuperPlatinumContract *identified by* ContractNo must belong one ZoneContract *identified by* ContractNo.

*One* Zone *identified by* ZoneID may be the zone (Zone\_Sensor) for one or more Sensors *identified by* SensorID.

*One* Zone *identified by* ZoneID must belong one ZoneContract *identified by* ContractNo.

*One* Video *identified by* VideoNo may have a StandardSubscription\_Video (viewed by) for one or more StandardSubscriptions *identified by* ContractNo.

*One* Video *identified by* VideoNo may have a GoldSubscription\_Video (controlled by) for one or more GoldSubscriptions *identified by* ContractNo.

*One* Video *identified by* VideoNo may have exclusive access by one SuperPlatinumContract *identified by* ContractNo.

*One* Video *identified by* VideoNo must be recorded by one Sensor *identified by* SensorNo.

*One* piece of Data *identified by* DataNo may be accessed by one PlatinumContract *identified by* ContractNo.

*One* piece of Data *identified by* DataNo must be recorded by one Sensor *identified by* SensorNo.

*One* Sensor *identified by* SensorID may record one or more Videos *identified by* VideoNo.

*One* Sensor *identified by* SensorID may record one or more pieces of Data *identified by* DataNo.

*One* Sensor *identified by* SensorID may be linked to one or more SensorSubscriptions *identified by* VideoNo.

*One* Sensor *identified by* SensorID must be maintained by one Maintainer *identified by* EmployeeID.

*One* Sensor *identified by* SensorID may record one or more Videos *identified by* VideoNo.

*One* Sensor *identified by* SensorID must be a sensor in a Zone\_Sensor for one or more Zones *identified by* ZoneID.

*One* Sensor *identified by* SensorID must have a sensor in a Sensor\_Part for one or more Parts *identified by* PartName.

*One* Part *identified by* PartName may have Part\_Suppliers for one or more Suppliers *identified by* SupplierName.

*One* Part *identified by* PartName must have Sensor\_Parts for one or more Sensors *identified by* SensorID.

*One* Part *identified by* PartName may have SensorMaintenanceRecord\_Parts for one or more MaintenanceRecords *identified by* DateTime and SensorID.

*One* MaintenanceRecord *identified by* DateTime and SensorID must belong to Sensor *identified by* SensorID.

*One* MaintenanceRecord *identified by* DateTime and SensorID may have MaintenanceRecord\_Parts for one or more Parts *identified by* PartName.

*One* MaintenanceRecord *identified by* DateTime and SensorID must be written by one Maintainer *identified by* EmployeeID.

# SQL DDL

*Build the database in SQL Server using DDL statements.*

## Tables

Please refer to the attached file *Spaces\_Create.sql*. This file contains all of the table creation and foreign key connection scripts.

I used Visual Paradigm (Visual Paradigm, Retrieved 2023) to logically model my database. I then used the generate database functionality to generate the DDL script. This automatically generated the table creation and foreign key alters. This worked well, although I did have to reformat them to be more readable. I also had to add square brackets to names like data or datetime, as these are keywords.

## Check Constraints

I have implemented three check constraints into my database.

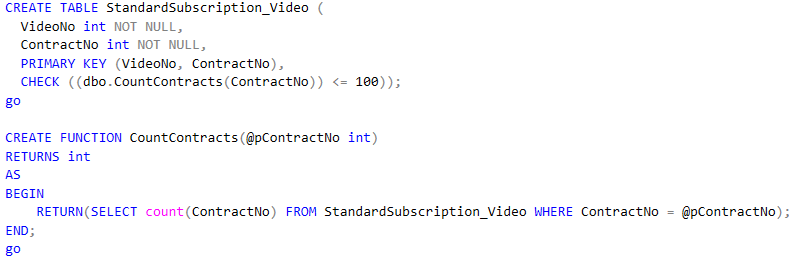
Two of them are for discounts:

1. AdminExecutive maxDiscount
2. Contract Discount

This is because Spaces© has a business rule that only allows discounts unto 3%. By putting in a check, this will ensure that larger values are not accidentally added.



The other discount is to limit the number of video streams a subscriber can view. Spaces© has a business rule that limits subscribers to viewing up to 100 video streams from each sensor. As this is a more complex check, I have implemented it using a function as shown below.



## Roles and Users

I have included five basic roles and added a example user to each role.

1. Administrator
2. Technician
3. Salesperson
4. AdminExecutive
5. Maintainer

These cover the basic business roles needed as well as the technical database roles. These have been granted various permissions depending on the access needed. For example, the maintainer only has access to select, insert and update maintenance records in the database, as that is the only data they will need.

In a professional situation, these permissions would likely be more in-depth to have tighter security of the data. They would also likely have other permissions, such as accessing various transactions, but for the purposes of this assignment, I believe that this demonstrates how this functions.

## Views

I have added a contract details view to the database, which shows contract and subscriber information. This is an easier way to view the data across multiple tables at the same time. It also gives more control over which columns can be accessed by who. Depending on how the different users and roles use the database, it may be beneficial to add other views in the future.

A picture containing text, font, screenshot

Description automatically generated

# Test Data

*Generate and load appropriate test data. There should be a minimum of 100 rows across all the tables and no table should contain less than 5 rows. Provide a list of data in your tables. Please provide the appropriate DML statements for this step in the form of INSERT statements.*

Please refer to the attached file *Spaces\_Populate.sql*. This file contains all of the test data insert scripts.

I used Mockaroo (Mockaroo, Retrieved 2023) to generate my test date. This is a free online tool that contains a lot of features and customisation.

There are approximately 850 records across all tables, with 5-100 per table.

# SQL Transactions

*Write and run a set of SQL queries required to provide the required information for the project case study. These will be resolved during class discussions, but should be determined from the case study documentation. Provide the SQL queries and their results. Include a brief description of the purpose of each query.*

## Analysis

I have analysed each of the transactions that will be implemented in the database using a tabular format.

Since Spaces© is an international company and the database will be interacted with from all around the world, it is hard to judge when peak time will be. I have decided just to include an average estimate.

Code

* Insert = I
* Read = R
* Delete = D
* Update = U
* Entry point = E

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction A | | Frequency: | 10-20/h | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Contact Info | Phone  Email  Secondary Email | I (E)  I  I | 1 |
| - | Address | Street Address  Suburb  City  Country  Postcode | I (E)  I  I  I  I | 1 |
| - | Subscriber | First Name  Last Name  Password  DoB  Address ID  Contact Info ID | I (E)  I  I  I  I  I | 1 |
| - | Contract | Start Date  End Date  Monthly Price  Discount  Salesperson ID  Admin Executive ID  Subscriber ID | I (E)  I  I  I  I  I  I | 1 |
| - | Sensor Subscription | Contract No  Sensor ID | I (E)  I | 1 |
| - | Standard Subscription | Contract No | I (E) | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction B | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Contract | Discount  Salesperson ID  Subscriber ID | R (E)  R  R | 2,000,000\* |
| Contract | Subscriber | First Name  Last Name  Subscriber ID  Address ID | R  R  R (E)\*  R | 2,000,000 |
| Subscriber | Address | Address ID  Street Address  Suburb  City  Country  Postcode | R (E)\*  R  R  R  R  R | 2,000,000 |
| Contract | Salesperson | Employee ID | R (E)\* | 1,000,000\*\* |

\* Estimated average 2 subscriptions per subscriber, at 1,000,000 subscribers.

\*\* Estimated half will subscribe online and wont have a salesperson.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction C | | Frequency: | 15m/s\* | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Data | Shared Audio  Human Voice  Body Shape  Skeletal Points  Texture  Date Time  Longitude  Latitude  Altitude  PContract No  Sensor ID | I (E)  I  I  I  I  I  I  I  I  I  I | 1 |

\* All sensors record data (even if the contract company is not receiving it) as stated in the brief. 500,000 sensors x 30fps = 15,000,000 records/second if all sensors are constantly on.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction D | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Sensor | Sensor ID  Latitude  Longitude | R (E)  R  R | 500,000\* |
| Sensor | Sensor Subscription | Sensor ID  Contract No | R (E)\*  R | 2,000,000\*\* |
| Sensor Subscription | Contract | Contract No  Subscriber ID | R (E)\*  R | 2,000,000 |
| Contract | Subscriber | Subscriber ID  First Name  Last Name | R (E)\*  R  R | 2,000,000 |

\* Over 500,000 sensors are deployed. Assume all have at least one subscription.

\*\* Estimated 2 subscriptions per subscriber.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction E | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Contract | Contract No  Subscriber ID | R (E)  R | 1 |
| Contract | Subscriber | Subscriber ID  First Name  Last Name | R (E)\*  R  R | 1 |
| Contract | Data | Contract No  Human Voice  Shared Audio  Body Shape  Skeletal Points  Texture  Date Time  Longitude  Latitude  Altitude | R (E)\*  R  R  R  R  R  R  R  R  R | 77.76m\* |

\* Assume data is being recorded at 30fps. The sensor stores 1 month of data, but it is unknow how long data is stored on the Spaces© database. Assuming 1 month (30 days) as well -> 77,760,000 or 77.76 million records. This is a very large amount of data that could have a significant impact of the performance of their database. Spaces may with to reduce this or store historical data on a separate database much as a data warehouse for analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction F | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Standard Subscription \_Video | Video No  Contract No | R (E)  R (E) | 1 |
| Standard Subscription \_Video | Video | Video No  Contract No | R (E)\*  R | 100,000\* |
| Standard Subscription \_Video | Contract | Contract No  Subscriber ID | R (E)\*  R | 100,000 |
| Contract | Subscriber | Subscriber ID  First Name  Last Name | R (E)\*  R  R | 100,000 |

\* 1,000,000 subscribers, estimate watching an average of 10% of the time = 100,000.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction G | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Sensor\_Part | PartName  Sensor ID | R (E)  R | 1 |
| Sensor\_Part | Part\_Supplier | PartName  SupplierName | R (E)\*  R | 100\* |

\* This would depend on the number of parts that each sensor will have. Estimated 100 but less with the test data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction H | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Sensor | Sensor ID  Latitude  Longitude | R (E)  U  U | 1 |
| - | Zone \_Sensor | Sensor ID  Zone ID | R (E)  R + U | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction I | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Data | Data No  Shared Audio  Human Voice  Body Shape  Skeletal Points  Texture  Contract No  Sensor ID | D (E)  D  D  D  D  D  D  D | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transaction J | | Frequency: | Occasionally | |
| **From** | **To** | **Attributes** | **Access** | **Num Times** |
| - | Maintenance Record | Sensor ID  Date Time | R (E)  R (E) | 1,000,000\* |
| Maintenance Record | Maintenance Record\_Part | Part Name  Sensor ID  Date Time | R (E)  R  R | 200,000\*\* |
| Maintenance Record\_Part | Part | Part Name  Price | R (E)  R | 500,000\*\*\* |

\* Assume each sensor has had maintenance an average of two times.

\*\* Assuming 20% of maintenance records require new parts.

\*\*\* Assume an average of 1 new parts per two sensors.

## Creation

Please refer to the attached file *Spaces\_Transactions.sql*. This file contains all of the transactions.

Each written query includes comments to explain their purpose and how they work.

## Indexing

After performing transactional analysis to identify high-frequency tables and columns, I have identified four secondary indexes to add:

1. Subscriber: subscriber ID
2. Contract: contract no
3. Video: video no
4. Data: data no

Secondary indexes can be added by adding the following code:



After more extensive testing of database performance, Spaces© may with to add other index, such as to address, contact info and sensor.

## File Storage

File storage will be done using the default Microsoft SQL Server system. The default method is to uses balance trees with clustered indexes. The bulk of the data is stored at each leaf node.

## Controlled Redundancy

Redundancy is the practice of keeping a piece of data in multiple places at the same time. Generally, data redundancy is avoided as it can cause data discrepancies, corruption and increased database sizes, however it can be beneficial in some situations (Gillis, 2021).

Controlled redundancy is the practice of introducing redundant data to improve database read times. While this does use more space, copying stable data can considerably reduce the number of pages accessed and thus reduce the time required (Pattern: Controlled Redundancy, Retrieved 2023). This process require denormalising the database, as redundancy violates first normal form.

This only works or stable data, or data that is unlikely to change. The majority of Spaces© database will be changing frequently, so very little of the data could be implemented with controlled redundancy. Controlled redundancy would likely not be worthwhile implementing in the Spaces© database.

## Estimated Size

I have estimated the database size in the external Excel Spreadsheet. The is a page for each table and a total page at the end. I have estimated the database size to be 373 PetaBytes. This would vary drastically by the actually sizes of the video and data being recorded, as well as how long data is being stored for.

All row counts used for this calculation are the estimates from the tables data dictionary with the exception of the video and data amounts.

* Video: 500,000 sensors \* average streams 50 \* 30fps \* 60 seconds = 45,000,000,000 (45b)
* Data: 500,000 sensors \* 30fps \* 60 seconds = 900,000,000

I have estimated that video and data will be stored on the Spaces© database for 1 minute. This gives room to deal with latency, minor buffering and stream timing issues. If video or data is needed past this, it can be requested from the sensor which will store it for up to 1 month. I think that it is unrealistic for all video and data to be stored much longer than this (373 PB is huge), and this time may be able to be reduced.

# Significant Issues

*Document any significant problem(s) you encountered and the approach you took to solving it/them. This should show you made a genuine effort to solve these problem(s) and the learning you achieved from that process.*

I found that since my logical model was well designed, there were few significant problems during the database implementation. This shows that good planning can save a lot of time and effort during the database constructions.

### Data

While Mockaroo saved me lots of time, it wasn’t perfect. On a couple of occasions, I had to modify the data formatting it outputted or the foreign key values so it would work properly.

I also had to make a few other adjustments to the data over the course of the implementation and transaction building, where a datatype or values needed to be changed. These did slow my progress but were minor overall.

I later learned that Mockaroo is able to same generated data types, which can then be referenced when generating other primary keys. While this would have taken time to learn and set up, it likely would have saved me time in the long run, especially if I was working with a larger database.

### Transactions

I did not have any significant issues will writing or analysing the transactions. While it was not an issue, I did write the SQL transactions first an perform analysis after, which I later learnt was the wrong order. This actually worked well for me, as both steps were relatively easy for the most part. I did however have to make minor adjustments to the database to be able to perform all transactions correctly. For example, I added datetime and location to each data record so this could be displayed.

### Reflection

Overall I am quite happy with how this project has turned out. I believe that I have covered almost all of the requirements to a consistently high standard.

# Conclusion

This report has covered the design and implementation process for the Spaces database, further developing and practicing my knowledge of databases and SQL.

# References

*Chen Notation*. (2014, August 2). Retrieved from Vertabelo: https://vertabelo.com/blog/chen-erd-notation/

*Data Modeling*. (Retrieved 2023). Retrieved from 1 Key Data: https://www.1keydata.com/datawarehousing/data-modeling-levels.html

Gillis, A. S. (2021, December). *Redundant*. Retrieved from Tech Target: https://www.techtarget.com/searchstorage/definition/redundant

Muñoz, A. (2022, August 23). *Why is Database Normalization so Important?* Retrieved from Sales Layer: https://blog.saleslayer.com/why-is-database-normalization-so-important#:~:text=Normalization%20is%20an%20essential%20part,and%20making%20it%20more%20concise.

*Pattern: Controlled Redundancy*. (Retrieved 2023). Retrieved from Object Architects: http://www.objectarchitects.de/ObjectArchitects/orpatterns/Performance/ControlledRedundancy/#:~:text=Controlled%20Redundancy%20is%20a%20technique,speed%20up%20reading%20database%20access.

(Retrieved 2023). Retrieved from Mockaroo: https://www.mockaroo.com/

(Retrieved 2023). Retrieved from Visual Paradigm: https://www.visual-paradigm.com/

Romani, E. (2023, March 2). *What Is Database Normalization?* Retrieved from Built In: https://builtin.com/data-science/database-normalization

Sherman, R. (2015). *Business Intelligence Guidebook: From Data Integration to Analytics.* Retrieved from Science Direct: https://www.sciencedirect.com/topics/computer-science/conceptual-data-model#:~:text=The%20conceptual%20data%20model%20represents,data%20in%20a%20business%20context.

Shin, K., Hwang, C., & Jung, H. (2017). *NoSQL Database Design Using UML Conceptual Data Model Based on Peter Chen’s Framework.* Retrieved from DocPlayer: https://docplayer.net/49346596-Nosql-database-design-using-uml-conceptual-data-model-based-on-peter-chen-s-framework.html

Stewart, J. (2008, June 1). *Crow’s Feet Are Best*. Retrieved from The Data Administration Newsletter: https://tdan.com/crows-feet-are-best/7474#

*What is a Logical Data Model?* (Retrieved 2023, 9 March). Retrieved from TIBCO: https://www.tibco.com/reference-center/what-is-a-logical-data-model

*What is data modeling?* (Retireved 2023, March 3). Retrieved from IBM: https://www.ibm.com/topics/data-modeling

Yue, K. (2018). *The Extended ER Diagram.* Retrieved from University of Houston-Clear Lake: http://dcm.uhcl.edu/yue/courses/itec3335/Fall2018/notes/model/Extended\_ERDiagram.html