



JULY 12, 2023

SPACES DATABASE CREATION

DAT601 PROJECT

KIRA BYRNE

ID# 13509995

NMIT

Table of Contents

Data Modelling in Information Systems	2
Conceptual Modelling	3
SPACES Conceptual Model.....	8
Database Dictionary.....	11
BUSINESS RULES	24
From Conceptual to Logical	26
Normalisation	27
Logical ERD of SPACES	30
NaLER Analysis.....	33
Database Dictionary.....	34
Transitioning the Model	35
Creating the Database.....	35
Physical Model Diagram	36
Queries & Transactions	37
References.....	43

Data Modelling in Information Systems

Introduction

Data modelling is used to create a visual and logical representation of all components being recorded and stored in a proposed or existing database (Corbo, 2023). The main components seen in a data model are:

- **Entities** – A category that holds a set of data (e.g. Student, Order, Train)
- **Attributes** – A definitive labelling of the data that is being stored within an entity (e.g. A STUDENT has a *student id*, *first name*, *last name*, *email*, etc).
- **Relationships** – The connection two entities have with each other. A relationship should clearly show the cardinality and optionality each entity has within a connection.

Each entity and their attributes are the essence of a database. Between them are the relationships that form the structure and narrative. Relationships define not only the connections but the optionality and cardinality between entities.

Take the example below, a database for the New Zealand Land Transport Agency (NZTA). A road worthy car that is being sold in a legal manor must be registered with the (NZTA), who have many (N) cars within their database. A car does not always have a specific owner, such as if the car was in a dealership, it is not registered to an owner, rather a business. However, an owner of a car must register themselves with NZTA to legally own a car. An owner can have many cars, but only one (1) owner can be registered to a car within NZTA's database, in which they have many (N) members.

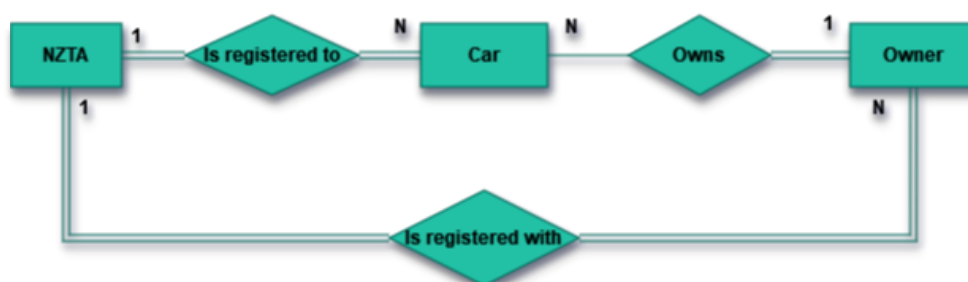


Figure 1: Car registration data model

Through data modelling, anyone within the project team should be able to understand the purpose and goals of a database. Typically, a proposed database will be modelled in stages as the project advances, beginning with highly visual and abstract drawings and ending with technical and physical representations as they would be seen from the perspective of software.

Conceptual Modelling

When a database is proposed, in need of major changes, or lacks documentation, conceptual modelling is typically used as a visual guide to the architecture of a database in a “big picture” view. In the context of a proposed database (such as SPACES), this type of model is generated early in the project design in a way that can be easily understood by most viewers. Most importantly, anyone on the project team. It can be created on any drawing medium like a whiteboard, paper, or modelling software and have the ability to be quickly changed as it lacks complexity (DataAcademy.in, 2020). The goal of this early modelling is to showcase the conceptual structure of all entities and simple connective relationships with little technicality outside of the chosen style (ERD, CHEN, etc). It is not yet concerned with the data that is being stored, making it highly abstract and modular.

Logical Modelling

Once entities and their connections have been established in the conceptual design, the project team can begin to model the fields of data that are to be stored. Logical models are typically developed as a platform-independent representations of the values an entity will have, the data that will be stored, and the relationships between them all (Rithka, 2022). The addition of keys is the logical focus of how the entities in a database are connected through primary and foreign keys. At this stage, only candidate keys are highlighted. These are values that can uniquely identify an individual set of data within a table, such as personal email, order number, or any other field that is entirely unique and traceable per dataset.

The creation of a logical model is often produced by a database architect and used by a business analyst to provide an organisation with insight pertaining to the limitation of the current technologies. (Agar, 2021), Logical models are made without foresight into software of coding language. Their purpose is to prepare a database architect for defining further detail to the modelling prior to implementation.

Physical Modelling

The final stage of data modelling prior to the coded construction of a database is the physical model, which focuses on datatypes, their respective lengths, tables, and columns that provide the structure of the data an entity will hold. The design of the physical modelling considers the systems requirements and constraints (Keys, size, scalability, etc) and analyses the feasibility of the data systems proposed in the conceptual and logical models. This stage is the most crucial in the design of a proposed database, therefore, documenting the format in which the data will be physically stored is vital (cloud-base, SQL, NoSQL, hard drive, etc). Following from this, the database architect can tweak the concept of the database to meet the requirements of particular database management systems (DBMS) (Nalimov, 2021). The end goal of the physical model is to provide a finalised design that can be engineered into an implementable relational database.

Conceptual Modelling

Chen's Extended Notation

An entity relationship diagram (ERD) is a method of modelling data that showcases all aspects of a database, primarily the entities, their attributes, and how they relate to one and other. An extension of the ERD is the Chen Notation. Made by Dr. Peter Chen in 1976, the purpose of the Chen Notation is to extend the logical model of an ERD through making changes to the:

- Associative shapes
- The complexity of entities
- How attributes are attached to an entity
- How relationships are shown
- Cardinality




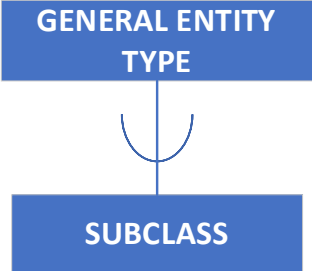
Chen Notation diagrams maintain the abstract look of a conceptual model while containing more detail similar to that of a logical model. In this style, entities have a level of complexity which allows a data analyst to further understand the context behind an entity, such as it's reliance or association with other entities.

As mentioned in the introduction, relationships make up the structure of a data model, providing connections and context between all entities. A standard ERD would use crow's foot methodology to

represent a relationship and its cardinality. In Chen Notation, a relationship is a diamond shaped textbox that connect to entities, with the cardinality shown through the connective lines (Dybka, 2014). This addition of text provides context for a relationship and forms a sentence to define it (e.g., “One OWNER has many DOGS”). The aim of Chen’s Notation is to construct a natural view of how the real world comprises entities and the relationships that exists between them (CS, Odessa, 2023).

Chen Notation Glossary

Entities

	<p>An ENTITY represents a thing which has attributes and relationships with a database.</p> <p>E.g. A student, teacher, class room</p>
	<p>A WEAK ENTITY represents a thing or instance in which its existence is dependant on a relationship with another entity</p> <p>E.g. A subscription cannot exist without a subscriber</p>
	<p>An ASSOCIATIVE ENTITY represents an instance that occurs in a many-to-many relationship.</p> <p>E.g. Many owners can own many houses, so the associative entity would be the ownership of a particular house.</p>
	<p>A SUBCLASS is a an entity that takes on attributes of a general entity type. This is known as generalisation and specialisation and helps to minimise redundant data (Yue, 2017). SUBCLASSES can be used to create a tier group of subclasses that extend from one after the other.</p> <p>E.g A CAR SERVICE has a GOLD service that includes everything from the SILVER and BRONZE services with extra values that would be outlined in the GOLD subclass entitiy.</p>

Attributes

ATTRIBUTE

An ATTRIBUTE is a field detailing stored information about an entity.

E.g. A DOG has a BREED.

KEY ATTRIBUTE

A KEY ATTRIBUTE is a primary attribute that uniquely identifies a singular entity.

E.g. An ACCOUNT HOLDER is identified by an ACCESS NUMBER.

PARTIAL KEY ATTRIBUTE

A PARTIAL ATTRIBUTE is used to uniquely identify a weak entity through combination with the main entity's key attribute.

E.g. A SUPPLIER (e) with a SUPPLY CODE (a) processes an ORDER (w/e) that has an ORDER CODE (p/k/a).

MULTIVALUED ATTRIBUTE

A MULTIVALUED ATTRIBUTE is made up of multiple values, each being distinct from others within a single table.

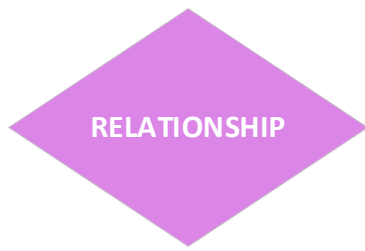
E.g. An ALBUM can fit many GENRE/s.

DERIVED ATTRIBUTE

A DERIVED ATTRIBUTE represents an attribute that is calculated from the values derived from other attributes.

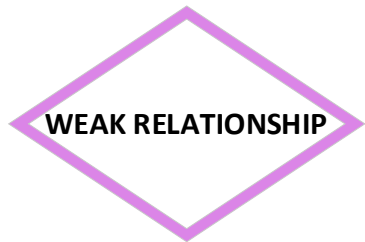
E.g. the total PRICE of an ITEM can be found when calculated with the GST and the DISCOUNT.

Relationships



A RELATIONSHIP represents the connection between two entities. Usually, the text within the diamond shape provides context.

E.g. DEFENDANT *is registered to* TERRITORY



A WEAK RELATIONSHIP is used to identify a relationship between parent (strong) and child (weak) entities, where one entity's existence is dependant on another.

E.g. A CLIENT (s/e) *has a* SUBSCRIPTION (w/e)

Optionality



A MANDATORY relationship connection represents a relationship that is solidified without optionality. The two entities must be related to each other but not all entries in an entity have total participation, only partial participation (Dybka, 2014).

E.g. A STUDENT **must** be enrolled in a COURSE, but a COURSE **does not** have to have any enrolled students.



An OPTIONAL relationship connection is represented with a dotted line. Entities with this connection are not required to have a relationship.

E.g. A PATIENT can *purchase* MEDICINE and can **optionally have** a PRESCRIPTION.



A TOTAL PARTICIPATION connection represents a relationship where all subjects in an entity must participate in a relationship with another entity.

E.g. An INPATIENT **must** be admitted to a WARD.

Cardinality

One to One 1:1

A relationship where only one entity is in participation with only one other entity.

E.g. 1 PRINCIPAL directs 1 SCHOOL.

One to Many 1:N

A relationship where only one entity has multiple instances of participation with another entity.

E.g. 1 SUPPLIER makes many ORDERS

Many to One N:1

A relationship where an entity has multiple instances of participation with another entity, however that other entity on has one instance or participation.

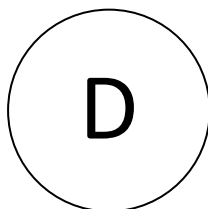
E.g. Many ACCOUNTS are registered to 1 OWNER.

Many to Many M:N

A relationship where both entities have multiple instances of participation. This relationship is usually represented as a it's own entity with attributes.

E.g. Many INGREDIENTS are used in MANY RECEPIES.

Extended



A DISTINCT icon is used to separate a set of subclasses from each other. It exists between the connect of a superclass and a subclass to show that the subclasses receive the given attributes but do not extend from each other. They are distinct.

E.g All EMPLOYEES have an ID, a name, and an email but their roles are distinct.

SPACES Conceptual Model

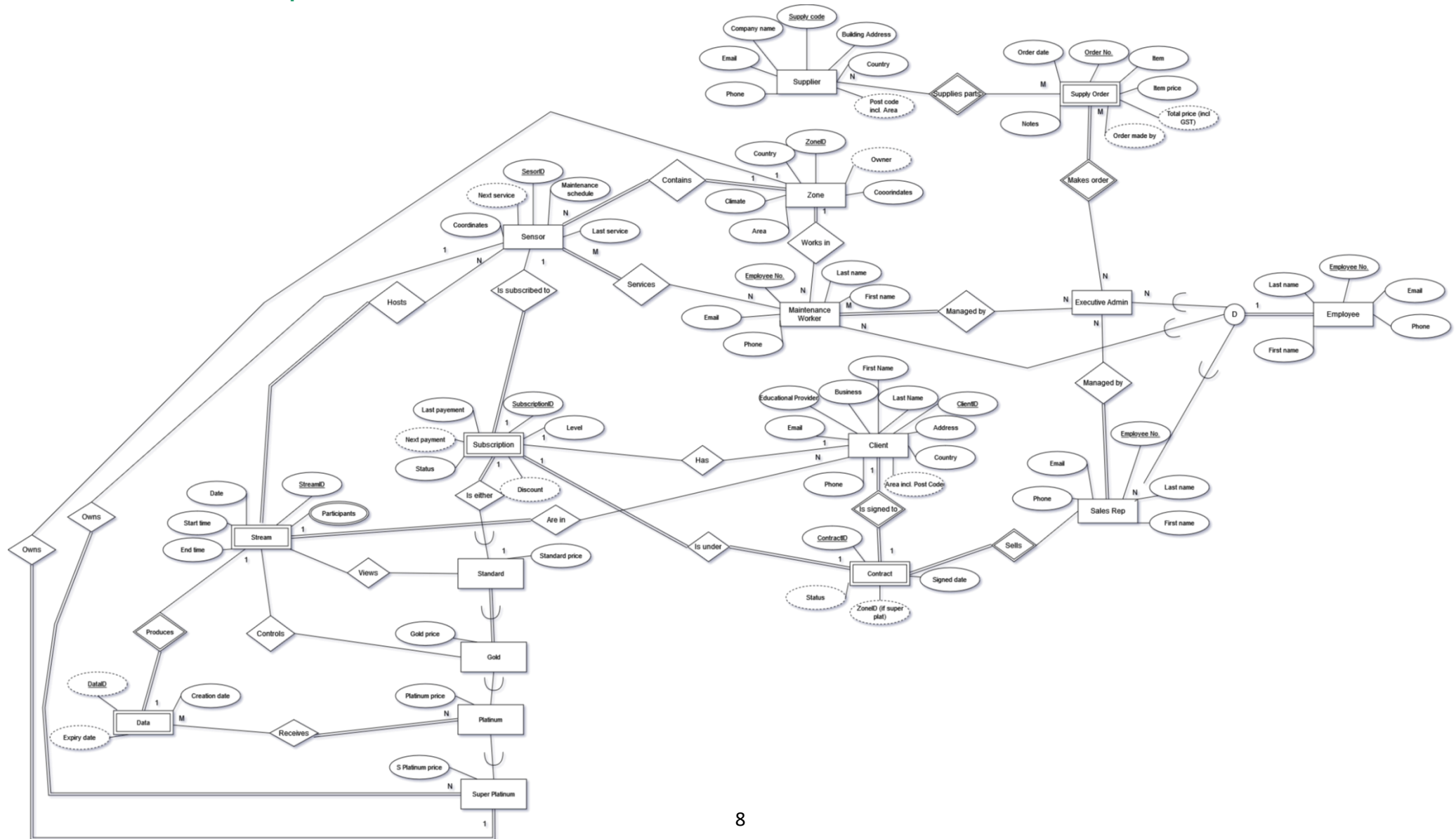


Diagram Rationale

All EMPLOYEEs of SPACES contain essential information about each individual including their unique id, their first and last names, and work-related contact details. Extending from the EMPLOYEE attributes are the three distinct roles.

1. An EXECUTIVE ADMIN is responsible for managing other employees and making SUPPLY ORDERS for the sensors which are fulfilled by SUPPLIERS. They have full view, update, and delete abilities within the database. Not every EXECUTIVE ADMIN manages all employees by all employees must be managed by an EXECUTIVE ADMIN.
2. A MAINTENANCE WORKER is responsible for maintaining sensors. They are permanently allocated to a zone where they service many sensors. With sensor data, they have view and update abilities.
3. A SALES REP is responsible for forming and maintaining client relationships. They sell and sign contracts, attend to client needs, organise subscriptions, discounts, and contact clients when contracts are nearing void status and/or deletion.

A CLIENT is an individual who has a CONTRACT with SPACES. Similar to the EMPLOYEE, the CLIENT table stores essential information about the individual that is relevant to the business structure. This includes, their full name, contact details, physical address, and if they are using SPACES for educational or business-related purposes. If they are, then they are entitled to a 3% discount on the price of their subscription.

A CLIENT discusses and signs a CONTRACT with a sales rep. The importance of the CONTRACT is that it stores long-term, stable information about the CLIENT and their SUBSCRIPTION. This information is only subject to change under certain circumstances such as a contract becoming void (status), CLIENT details changing, or the SUBSCRIPTION level changing. The CONTRACT is a weak entity as its existence is reliant on both the CLIENT and the SALES REP who are involved.

A SUBSCRIPTION stores both stable and transient information about a CLIENTs use of a SENSOR. This includes the level of SUBSCRIPTION, the ID of the sensor they are connected to, the status of the SUBSCRIPTION (subscribed/unsubscribed), if a discount applies, and the dates of the last subscription payment and the next due payment. Each SUBSCRIPTION is reliant on the validity of a CONTRACT (whether it is current or void), therefore making it a weak entity. Each SUBSCRIPTION has a level which defines the CLIENTs abilities with a sensor. The level exists in a tier formation, meaning the abilities of the level prior will continue to greater tiers.

- STANDARD, being the first level, is the least expensive and allows a CLIENT to view a stream.
- GOLD will allow the CLIENT to control the view of the stream, changing camera angles and being able to “look around” a ZONE.
- PLATINUM allows the CLIENT to receive recorded DATA after the STREAM is over.
- SUPER PLATINUM is negotiated when a CLIENT is forming a CONTRACT. This level of subscription allows the CLIENT to secure a private ZONE and SENSOR. The ownership of a ZONE is negotiated and recorded in the CONTRACT.

A STREAM is a live presentation, similar to a zoom call. When in the STREAM, it is a constant live video that cannot be rewound or fast forwarded. This can only happen once the STREAM has ended, and the SENSOR has exported a recording of the STREAM as DATA, receivable by PLATINUM and SUPER PLATINUM subscribers. A STREAM is reliant on at least 1 participating

client to exists, making it a weak entity. This extends to the DATA. If a STREAM does not exists, there is no DATA to output.

SENSORS are the physical hardware that are placed in ZONEs to host a STREAM and project the CLIENTs into a ZONE. Each SENSOR has a maintenance schedule that is adhered to by MAINTENACE WORKERs. When a SUPPLY ORDER is made by an EXECUTIVE ADMIN, it contains information about the ID, contents, price, and reason for order including the ID of the staff member who made it. The SUPPLY ORDER is a weak entity as its existence only occurs when it is being made by an EXECUTIVE ADMIN.

A SENSOR is located by coordinates which are ascertained at the time of the SENSOR being installed in a ZONE. A ZONE is a specific area in which a SENSOR will be installed and used to host a STREAM. A ZONE is defined by its climates (urban, snowy, forest, etc). It identified with a unique ID, given general location data (county, area) as well as coordinates for specific locating. If a ZONE is owned by a SUPER PLATINUM subscriber, their CLIENT ID will be recorded as well.

Database Dictionary

Entities

Entity Name	Description	Aliases	Occurrence
Employee	An employee of spaces.	E.Employee	Many
Executive Admin	Members of the executive administration team at SPACE. Responsible for managing sales reps, maintenance workers, and making supply orders.	EA.ExecutiveAdmin	Many
Sales Rep	Employee of SPACE that work with clients to procure contracts. Manage client contact.	SR.SalesRep	Many
Maintenance Worker	Performs maintenance work on the sensors for a particular zone.	M.MaintenanceWorker	Many
Supplier	Supplies the items for a supply order for SPACES.	SU.Supplier	Many
Supply Order	The order made by executive admins and fulfilled by a supplier.	SO.SupplyOrder	Many
Zone	An area where sensors are placed to create the environment for a stream.	Z.Zone	Many
Sensor	The device used to perform a stream. Subscribed to by clients.	SE.Sensor	Many
Stream	The live video performed by a sensor in a zone that clients partake in. A client's abilities vary within a stream depending on their subscription level.	ST.Stream	Many
Data	The recorded video after a stream is complete. Only accessible to platinum and super platinum subscribers.	D.Data	Many
Client	A customer with a contract with SPACES.	C.Client	Many
Contract	A contract between SPACES and a client that states their subscription details.	CO.Contract	Many
Subscription	A contracted subscription to a sensor. Prices vary depending on level of subscription. Each level will take on the abilities of the previous tier. Price can be discounted if client is using SPACES for business or educational purposes.	SB.Subsription	Many
Standard	The base level of a subscription. Allows a client to partake in a stream	S.Standard	Many

	on a public sensor.		
Gold	Allows a client to control the view of a sensor while partaking in a stream.	G.Gold	Many
Platinum	Client receives recorded data from a stream.	P.Plat	Many
Super Platinum	Client can negotiate the ownership of a private zone and sensor.	SP.Splat	Many

Entity Relationships

Entity Name	Cardinality	Participation	Relationship	Participation	Cardinality	Entity Name
Supplier	1	Strong	Provides items	Strong	1	Supply Order
Employee	1	Mandatory	Is either	Strong	1	Executive Admin
Employee	1	Mandatory	Is either	Strong	1	Sales Rep
Employee	1	Mandatory	Is either	Strong	1	Maintenance Worker
Executive Admin	1	Strong	Makes order	Total	1	Supply Order
Executive Admin	N	Strong	Manages	Total	M	Sales Rep
Executive Admin	N	Strong	Manages	Total	M	Maintenance Worker
Maintenance Worker	N	Strong	Works in	Total	M	Zone
Maintenance Worker	N	Strong	Services	Total	M	Sensor
Zone	N	Total	Contains	Total	M	Sensor
Sensor	1	Strong	Hosts	Total	M	Stream
Stream	1	Strong	Creates	Total	1	Data
Sales Rep	N	Strong	Sells	Total	M	Contract
Client	1	Total	Signs	Total	1	Contract
Client	1	Strong	Is contracted to	Strong	1	Subscription
Client	N	Strong	Are in	Total	1	Stream
Subscription	1	Total	Is (either)	Strong	1	Standard
Subscription	1	Strong	Is (either)	Strong	1	Gold
Subscription	1	Strong	Is (either)	Strong	1	Platinum
Subscription	1	Strong	Is (either)	Strong	1	Super Platinum

Standard	1	Strong	Can view	Strong	1	Stream
Gold	1	Strong	Can control	Strong	1	Stream
Platinum	1	Total	Receives	Strong	N	Data
Super Platinum	1	Total	Owns	Strong	1	Sensor
Super Platinum	1	Total	Owns	Strong	1	Zone

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
Employee	Employee No	The unique employee number used to identify individual employees of SPACES.	Int	E.EmpNo	N	N	Not null	Primary Key	Increment by 1
Employee	First Name	The first name of the employee	Char (50)	E.firstname	N	N	Not null	N	None/Not Null
Employee	Last Name	The surname of the employee	Char (50)	E.lastname	N	N	Not null	N	None/Not Null
Employee	Email	The work email of the employee	Char (100)	E.email	N	EA.Firstname EA.lastname	Not null	Candidate key	None/Not null
Employee	Phone	The work phone of the employee	Char (20)	E.Phone	N	N	Not null	Candidate key	None/Not null
Executive Admin	Employee No	The unique employee number used to identify individual employees of SPACES.	Int	EA.EmpNo	N	N	Not null	Primary Key	Increment by 1
Executive Admin	First Name	The first name of the executive admin	Char (50)	EA.firstname	N	N	Not null	N	None/Not Null
Executive Admin	Last Name	The surname of the executive admin	Char (50)	EA.lastname	N	N	Not null	N	None/Not Null

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
Executive Admin	Email	The work email of the executive admin	Char (100)	EA.email	N	EA.Firstname EA.lastname	Not null	Candidate key	None/Not null
Executive Admin	Phone	The work phone of the executive admin	Char (20)	EA.Phone	N	N	Not null	Candidate key	None/Not null
Supplier	Supply Code	Code used to identify a supply company on their order receipts	Int	SU.SupplyCode	N	N	Not null	Primary Key	None/Not null
Supplier	Company Name	Name of the supply company	Char (100)	SU.CompanyName	N	N	Not null		None/Not null
Supplier	Email	General email address of the supply company	Char (100)	SU.Email	N	N	Y	Candidate Key	None
Supplier	Phone	Phone number of the supply company	Char (20)	SU.Phone	N	N	Not null	Candidate Key	None
Supplier	Address	Address of the Supply building or office	Char (100)	SU.Address	N	N	Not null	Candidate Key	None
Supplier	Country	Country the supplier is located in	Char (50)	SU.Country	N	N	Not null	N	None
Supplier	Area (including post code)	Area and postal code of the	Char (50)	SU.Area	Y	From Country +	Not null	N	None

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
		supplier				Area			
Supply Order	Order Number	Unique identifier of a supply order	Int	SO.OrderNo	N	N	Not null	Primary Key	None
Supply Order	Order Date	The date the order was made (DD-MM-YYYY)	Date	SO.Date	N		Not null	N	The date the order was finalised
Supply Order	Order made by	The first and last name of the executive admin that made the order	Char (100)	SO.MadeBy	Y	From EA.FirstName + EA.LastName	Not null	N	The name of the person writing the order.
Supply Order	Item	Names or descriptions of an item	Char (100)	SO.Item	N	N	Not null	N	None
Supply Order	Item price	Price of an individual item	Decimal	SO.ItemPrice	N	N		N	\$0.00
Supply Order	Total Price (including GST)	Total price of the order including GST	Decimal	SO.TotalPrice	N	GST derived from total price excluding GST.	Not null	N	\$0.00
Supply Order	Notes	Descriptive notes for the executive admin to use (e.g. order for sensorID)	Text	SO.Notes	N	N	Y	N	None
Sales Rep	Employee ID	The unique identifier of each	Int	SR.EmpID	N	N	Not	Primary	Increment from 1

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
		sales rep					null	Key	
Sales Rep	First Name	The first name of the sales rep	Char (50)	SR.FirstName	N	N	Not null	N	None
Sales Rep	Last Name	The surname of the sales rep	Char (50)	SR.LastName	N	N	Not null	N	None
Sales Rep	Email	The employee email given to the sales rep	Char (100)	SR.Email	N	N	Not null	Candidate key	None
Sales Rep	Phone	The employee phone number given to the sales rep	Char (20)	SR.Phone	N	N	Not null	Candidate key	None
Maintenance Worker	Employee ID	The unique identifier of each maintenance worker	Int	MW.EmpID	N	N	Not null	Primary Key	Increment from 1
Maintenance Worker	First Name	The first name of the sales rep	Char (50)	SR.FirstName	N	N	Not null	N	None
Maintenance Worker	Last Name	The surname of the maintenance worker	Char (50)	SR.LastName	N	N	Not null	N	None
Maintenance Worker	Email	The employee email given to the maintenance worker	Char (100)	SR.Email	N	N	Not null	Candidate key	None
Maintenance Worker	Phone	The employee phone number given to the	Char (20)	SR.Phone	N	N	Not null	Candidate key	None

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
		maintenance worker							
Client	Client ID	The unique identifier of each client	Int	CL.ClientID	N	N	Not null	Primary Key	Increment from 1
Client	First Name	The first name of the client	Char (50)	CL.FirstName	N	N	Not null	N	None
Client	Last Name	The surname of the client	Char (50)	CL.LastName	N	N	Not null	N	None
Client	Email	Email address of the client	Char (100)	CL.Email	N	N	Not null	Candidate key	None
Client	Phone	Contact phone number of the client	Char (20)	CL.Phone	N	N	Not null	Candidate key	None
Client	Address	Mailing address of the client	Char (100)	CL.Address	N	N	Not null	N	None
Client	Country	Country the client resides in	Char (100)	CL.Country	N	N	Not null	N	None
Client	Area (incl. Post Code)	Area the client lives in include postal code	Char (100)	CL.Area	N	Postal code derived from address + area + country	Not null	N	None
Client	Educational Provider	School/institution a client may be	Char (100)	CL.Education	N	N	Y	N	None

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
		part of							
Client	Business	Name of the business/company a client may be part of	Char (100)	CL.Business	N	N	Y	N	None
Contract	Contract ID	The unique identifier of each contract	Int	CO.ContractID	N	N	Not null	Primary Key	Increment from 1
Contract	Signed Date	Date the contract was made (signed off by both parties)	Date	CO.SignedDate	N	N	Not null	N	DD-MM-YYYY
Contract	Status	Status of a contract (current or void)	Char (15)	CO.Status	N	From Subscription Status and Subscription Last payment date	Not null	N	None
Contract	Negotiated Zone	ID of the negotiated zone owned by a super platinum subscriber. Must be manually updated as contract does not connect to zone.	Int	CO.Zone	N	ZoneID	Nullable	N	None

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
Subscription	Subscription ID	The unique identifier of each subscription	Int	SB.SubID	N	N	Not null	Primary Key	Increment from 1
Subscription	Level	Level of the subscription (standard/gold/platinum/super platinum)	Char (20)	SB.Level	N	N	Not null	N	Standard
Subscription	Discount	A discounted percentage from the price if client is using SPACES for education or business	Percent	SB.Discount	N	From Client's educational provider or business	Y	N	3% of subscription price
Subscription	Status	Current status of the subscription (Subscribed/Unsubscribed)	Char (15)	SB.Status	N	N	Not null	N	None
Subscription	Last Payment	The last payment made by the client	Date	SB.LastPay	N	N	Not null	N	None
Subscription	Next Payment	The next due payment of a subscription	Date	SB.NextPay	N	Calculated as one month from last payment	Not null	N	One month from last payment (DD-MM-YYYY)
Sensor	Sensor ID	The unique identifier of each sensor	Int	SE.SubID	N	N	Not null	Primary Key	Increment from 1

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
Sensor	Coordinates	The coordinates of a sensor	GPS	SE.Coordinates	N	N	Not null	N	None
Sensor	Maintenance schedule	The schedule for a sensor's maintenance (weekly, monthly, yearly depending on zone condition)	Char (15)	SE.Maintenance	N	N	Not null	N	None
Sensor	Last service	Date of the last service a sensor received	Date	SE.LastService	N	N	Y	N	DD-MM-YYYY
Sensor	Next service	Date of the next service a sensor receives	Date	SE.NextService		Calculated from Maintenance schedule + Last service	Y	N	DD-MM-YYYY
Zone	Zone ID	The unique identifier of each zone	Int	Z.ZoneID	N	N	Not null	Primary Key	Increment from 1
Zone	Coordinates	Area coordinates of a zone	GPS	Z.Coordinates	N	N	Not null	N	None
Zone	Country	Country a zone is located in	Char (100)	Z.Country	N	N	Not null	N	None
Zone	Area	Area of a country a zone is located in	Char (100)	Z.Area	N	N	Not null	N	None

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
Zone	Climate	General climate of the zone (Forest, plains, beach, mountain, urban)	Char (30)	Z.Climate	N	N	Not null	N	None
Zone	Owner	Full name of the client which may own a zone with a super platinum subscription	Char (100)	Z.Owner	Client first name + Client last name	Client	Y	N	SPACES (if not owned by client)
Stream	Stream ID	The unique identifier of a stream	Int	ST.StreamID	N	N	Not null	Primary Key	Increment from 1
Stream	Date	Date of a stream	Date	ST.Date	N	N	Not null	N	DD-MM-YYYY
Stream	Start Time	Start time of the stream	Time	ST.Start	N	N	Not null	N	00:00 (24 hour)
Stream	End Time	End time of the stream	Time	ST.End	N	N	Y	N	00:00 (24 hour)
Stream	Participants	Full names of each participant in the stream	Text	ST.Participants	Client first name + Client last name	Client	Not null	N	None
Data	Data ID	The unique identifier of a recorded stream	Int	SE.SubID	Stream ID + Stream Date	Stream	Not null	Primary Key	Increment from 1
Data	Creation Date	Creation date of the data	Date	D.Creation	N	N	Not null	N	DD-MM-YYYY

Entity Name	Attributes	Description	Domain	Aliases	Composite	Derived	Nulls	Key?	Default Value
Data	Expiry Date	Expiry date of the data	Date	D.Expiry	N	Calculated as 1 month from the creation date	Y	N	One month from the creation date (DD-MM-YYYY)
Standard	Standard price	Price of a standard subscription	Decimal	S.Price	N		Y	N	Set price of a standard subscription (bar discount)
Gold	Gold Price	Price of a gold subscription	Decimal	G.Price	N		Y	N	Set price of a gold subscription (bar discount)
Platinum	Platinum Price	Price of a platinum subscription	Decimal	PL.Price	N		Y	N	Set price of a platinum subscription (bar discount)
Super Platinum	S Platinum Price	Price of a super platinum subscription	Decimal	SP.Price	N		Y	N	Set price of a super platinum subscription (bar discount)

BUSINESS RULES	
Title	Client – SPACES relationships
Description	<p>Client relationships are established through a contract, not a subscription. A client must have a contract but does not have to have a subscription to a sensor. This way, if a client only needs access to a sensor on an ad-hoc basis then they will be able to maintain their registry with SPACES without having to pay a constant subscription fee.</p> <p>For example, an account owner of <i>Netflix</i> does not need to make a new account every time they opt out of subscribing for a period of time. Their access to the streaming content is merely withheld until a subscription payment can be made.</p>
Tables affected	Client, Contract, Subscription
Fields affected	N/A
Title	Void contracts
Description	<p>Extending from business rule 1, a client may have their contract made void if they have not paid for a subscription within a year of the last payment. A client should be warned 3 months in advance prior to the contract becoming void, giving them 3 monthly opportunities to pay for a subscription. A client can contact a sales representative to extend the void with reason. If not, and the client fails to pay, the contract will be terminated.</p> <p>Keeping data on clients that have potentially abandoned their relationship with SPACES is unnecessary data that is taking up storage in the database.</p>
Tables affected	Client, Contract, Subscription
Fields affected	SB.LastPay, SB.Status, CO.Status
Title	Discounts
Description	A discount of 3% can be allotted to a customer that is using SPACES for education or work. This client must register the name of their business or educational institute to be validated by a sales representative for discounting. This promotes accessibility to larger groups of clients all subscribing for a shared purpose versus one-off clients who subscribe for personal interest.
Tables affected	Client, Standard, Gold, Platinum, Super Platinum
Fields affected	CL.Business, CL.Education, S.Discount, S.Price, G.Price, PL.Price,

	SP.Price
Title	Super Platinum Zones
Description	The purpose of the Super Platinum subscription level is so that the client may have a private zone with a private sensor in which they “own”. When a client opts for super platinum subscription, they must negotiate the procurement of a zone to place a private sensor in. This will be recorded in their contract. SPACES will need to provide a zone for them that fits their choice of climate and does not overlap existing zones to maintain privacy when used.
Tables affected	Zone, Super Platinum, Contract
Fields affected	Z.Owner, CO.Zone
Title	Employee views
Description	<p>The three main employees of SPACES have separate view rights to information within the database. Each role may only view or manipulate data that is relevant to their job.</p> <p>A Sales Rep can only view information about clients, contracts, subscriptions, sensors, and zones. They can update all data on clients, contracts, and subscriptions but have no delete abilities.</p> <p>Maintenance workers are able to access information about supply orders, sensors, and zones. They can update the dates and schedules for maintenance for sensors, but they have no delete abilities.</p> <p>Executive Admin can access all information, update, and delete where needed.</p>
Tables affected	All tables
Fields affected	All fields

From Conceptual to Logical

Converting Chen's Notation to Crows Foot

The Chen notation extends from the standard conceptual Entity Relationship model (ER) with its own standard of shapes used to display entities, attributes, and relationships. In Chen, the style in which a shape is displayed can also provide further information. The abstractness of this notation makes it best suited for the conceptual design stage (Gordon, 2007)

Crow's Foot notation, created in 1976 by Gordon Everest, is another style used for data modelling (Dybka, 2016). It displays database components in the standard ER format and can be used at any stage in the database modelling process.

Documented in this assessment is the process of converting a Chen notation conceptual database model to a Crow's foot notation logical model. Highlighting the key differences between the two notations will be summarised in a list of mapping rules. These rules will act as a reference point when designing the logical model in a way that improves upon the conceptual design while maintaining the requirements of the overall database. The final product of a quality logical design will be the next milestone towards designing implementation.

Attributes and Entities

The Chen notation displays attributes as attachments to entity. Each attribute is shown as an ellipse connected to the entity. The stylisation of the ellipse determines the type of attribute such as primary keys, derived attributes, and weak attributes.

The Crows Foot notation displays entity and attributes in a tabular format. Each attribute is assigned a data type and length and their primary keys exist as foreign keys in applicable relationships with other entity tables.

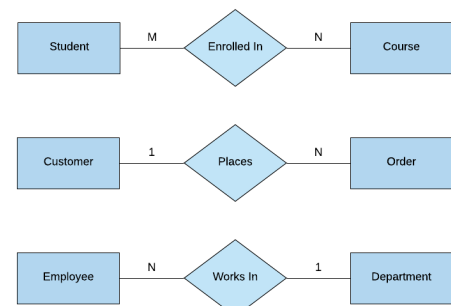
Relationships, Cardinality, and Optionality

Cardinality and optionality extend from two entities to meet at a diamond shape. Within this diamond are words that provide context to the relationship.

E.g. *Many STUDENTS are enrolled in 1 COURSE.*

In Chen's Notation, symbols are used to represent the cardinality between entities. These symbols sit on either end of the connected lines in a relationship closest to represented entity.

Crow's foot uses its own style of connectors to show the cardinality in entity relationships. Rather than labelling symbols, Crow's foot cardinality is seen at the ends of the connected lines.



Chen's Notation Cardinality (Brumm, 2022)

Summary of Crow's Foot Notation

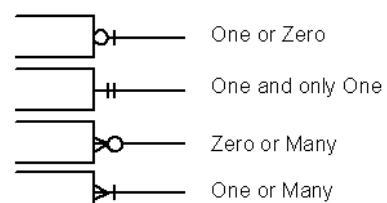


Figure 2 Crow's Foot Cardinality (Stewart, 2008)

Mapping Rules

1. The logical model must be normalised to third normal form (3NF).
2. Each attribute will be listed within the entity.
3. Attributes will be independent, i.e. not composites.
4. Each entity will be uniquely identified by a suitable primary key (has meaning, is appropriate in length, and is consistent).
5. Data types, specialities, and lengths will be added to each attribute.
6. Foreign keys will exist in related entities depending on their cardinality:
 - a. FK can exist in either entity in a 1-1 relationship
 - b. FK will exist in the “many” entity in a 1-Many relationship
 - c. The FKs of both entity in a many-many relationship will be stored in a new table as many-many relationships need to be represented as a separate table where the two keys become the primary key.
7. Similar to many-many relationship tables, a relationship with attributes will be displayed as its own entity. The entity being the instance of that relationship and its related data. The two primary entities that would have had this relationship will relate to this table only as there can be no relationship with attributes as seen in Chen’s Notation.
8. The relationships must maintain a natural language (NaLER) flow of the conceptual diagram so that meaning in a relationship is not lost.
9. Cardinality of relationships must be appropriately and accurately converted from Chen to Crows foot (*See above in Relationships, Cardinality, and Optionality*).
10. Super/subclass relationships will exist as their own entities that take consistent attributes from a super entity. Each following subclass can be displayed as inheriting from each other following the $X \rightarrow Y$ notation.
11. Logical model must abide by the business rules assumed and documented in the previous report (Byrne, 2023).

Normalisation

Normalisation is the process of optimising data through a series of structural design. The significance of normalising data is to create an implementable model that data engineers/architects can work with to build quality Database Management Systems (Romani, 2023). To normalise data is to analyse each established table and its values in the pursuit of removing redundant or illogical information and minimising anomalies. Normalisation is broken down into a series of stages, each relying on the completion of the last to produce a database that is simple to understand, easier to enhance, and it protected by a foundation of integrity (Decomplexify, 2021).

Zero Normal Form & First Normal Form

Data that begins at zero normal form (ONF) has not been normalised at any rate. From ONF, first normal form (1NF) can be applied. The purpose of 1NF is to eliminate duplicate data, ensure that attributes are restricted to single value entries, and to provide unique identifiers to all entities (which can then be used to define functional dependencies).

An example of data moving from ONF to 1NF in a primary school database:

<ONF>

ChildID	DOB	Parent
FJD1455	07/02/1998	John Doe, Jane Doe

<1NF>

ChildID	DOB	Parent1ID	Parent2ID
FJD1455	07/02/1998	ASF53002	AAD4202

Second Normal Form

Second normal form (2NF) first relies on data being in 1NF before ensuring functional dependency on all key identifiers. A functional dependency occurs when the existence of non-key attributes relies on a primary key. For a table to violate 2NF, the non-key attributes would be partially dependant meaning they only rely on part of a primary key.

Violation of 2NF:

<OwnerPet>

OwnerID	PetID	AnimalType	Address
O7204	P173814	Lizard	57 Downing St. Kent
O14093	P59742	Rodent	845 Isbjorn Rd. Kepplavik
O129741	P91389	Horse	4 Evangelion Ave. Nazereth

The above is a table in a veterinarian database. The primary keys are {OwnerID and PetID}. Each non-key column is only dependant on half of the primary key. If a pet within this table were to pass away, their information would be removed along with the owner's details and the address. If that was the only instance of that particular owner, then their data would be completely deleted from the database, creating one of many possible anomalies created by tables lacking 2NF.

Owners and Pets should have their own respective tables where all non-key attributes are totally reliant the whole of the primary key.

Third Normal Form

Third normal form (3NF) first relies on a database being in 2NF. The primary focus of 3NF is to eliminate *transitive dependencies*. For example, see the following table that violates 3NF:

<Employee>

DeskID	Department	EmployeeName
DA503	Data Analysis.	Jane Doe

Jane Doe works in a department where each desk is numbered. Jane (Z) is dependent on her department (Y) that is identified by the ID of the desk (X). This is transient dependency ($X \rightarrow YZ$). If Jane were to change departments, her desk would also need to be updated. If not, there would be an inconsistency in data as the desk location and department would be conflicting.

3NF states that all non-key attributes should be entirely dependant on the primary key and nothing else. Therefore, a table would be made that lists all deskIDs and their respective departments. A desk ID would be inserted into an employee profile to locate them to desk and department.

Extending on 3NF is **Boyce-Codd normal form** which states that all attributes, key or non-key should depend entirely on the key.

Fourth Normal Form

Fourth Normal Form (4NF) first relies on normalisation to Boyce-Codd NF. Once all attributes are dependant entirely on the key and nothing but the key, 4NF accounts for the occasions where a non-key attribute can come in many repeating values, known as multi-values dependencies. An example would be the following table in a Honda dealership database:

<CarModel>

Model	Year	Colour
CRV	2019	Silver
CRV	2019	White
CRV	2020	Green
Civic	2019	Silver
Civic	2021	White
City	2018	White
City	2022	Silver
Ridgeline	2018	Blue

Each available model of Honda has both a year and colour, both of which are independent of each other but are both dependant on the model. This is a multivalued dependency where $X \twoheadrightarrow Y$ but there are multiple values of Y (Study Tonight, 2023). To satisfy 4NF, a table would be made for each value and the model they depend on. I.e. <ModelColour> and <ModelYear>.

Logical ERD of SPACES

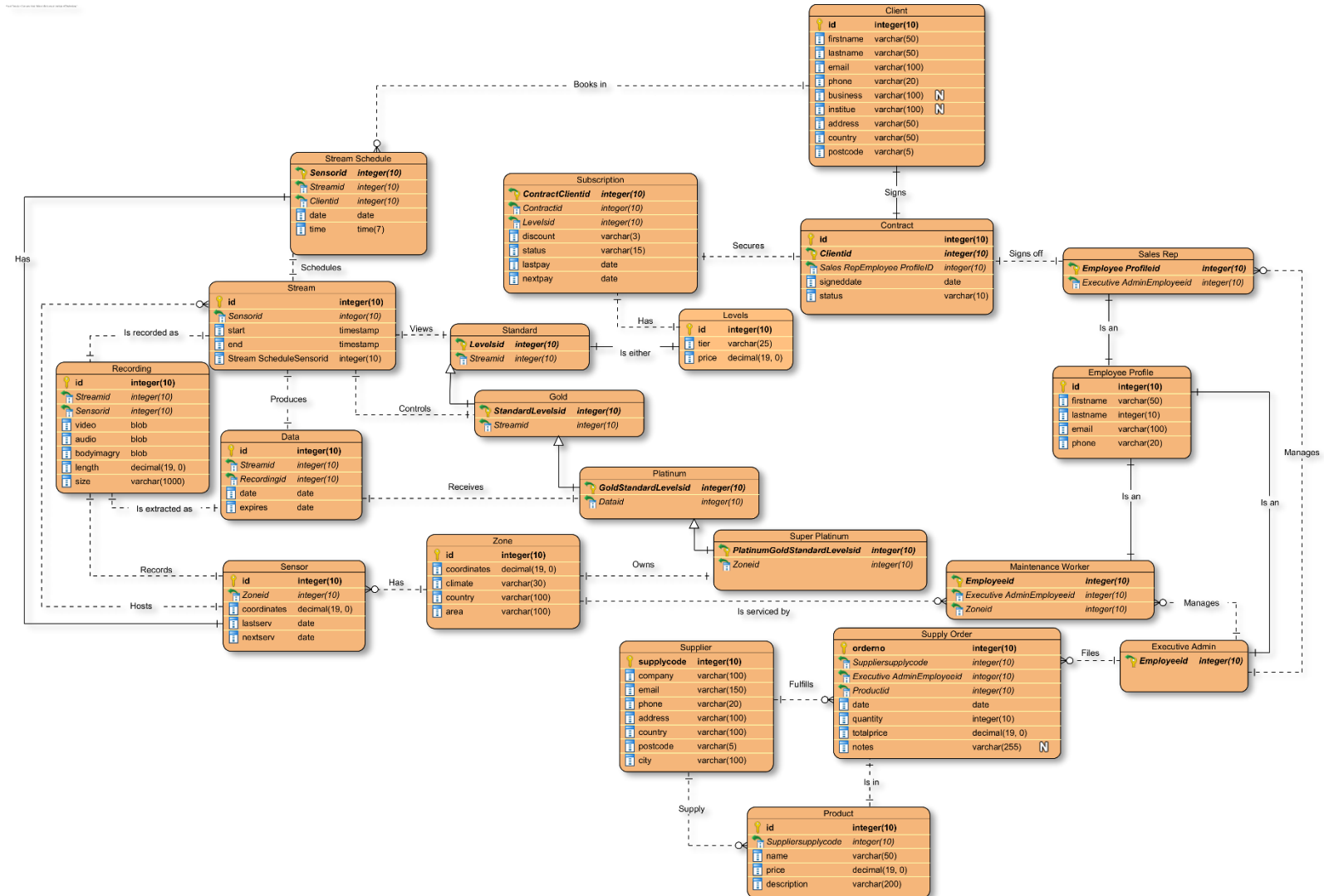


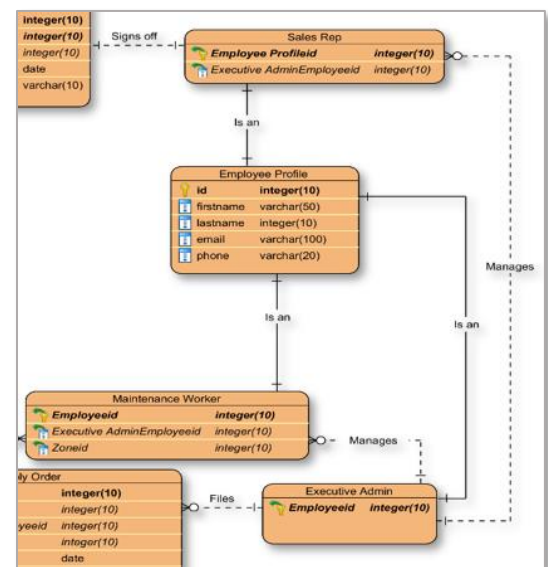
Diagram Rationale

Employees

Each **Employee** has their details recorded in a profile (<Employee>). This profile includes their unique ID, name, and work-provided contact details.

There are three **Employee** roles, each with their own responsibilities and involvement in other sections of the database. Separating these roles into their own table enables the model to display these different abilities.

- **Executive Admins** are charged with making supply orders and managing the other staff roles.
- **Sales Reps** are involved with client relations and signing off subscription contracts.
- **Maintenance Workers** live in or near zones where they service the deployed sensors of that area.
- Sales Reps and Maintenance Workers both report back to an **Executive Admin** who manages them.



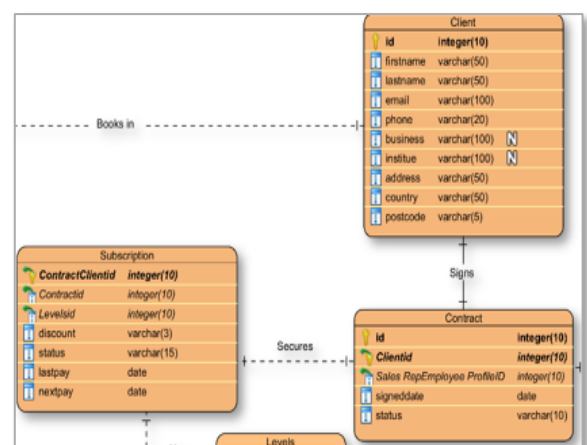
Clients, Contracts & Subscriptions

A subscriber of SPACES is referred to as a **Client**. Their details are recorded within a client profile table (<Client>). Details include a unique id, the client's name, contact details, and the name of their business or educational institute if there are using SPACE for business or education purposes. If this is the case, then they will receive a 3% against the subscription fee.

A **Client** is related indirectly to their subscription through a **Contract** which holds details about their involvement with SPACES. The reason for this is that if the client decides to unsubscribe for a time, their partnership with SPACES is not made void.

Within the **Subscription** table (<Subscription>), information about the subscription itself is stored including the applied discount (if so), the level of subscription, most recent and next due payments, and the status (subscribed/unsubscribed). Each subscription is identified by the combined unique keys of the **ClientID** and the **ContractID** as a client can only have one contract that secures one subscription at a time.

As a business rule, if a client fails to pay the subscription fee for a year from the last due payment date (making the status unsubscribed), their **Contract** will be made void and be up for deletion if a client fails to respond to the contact of their Sales Rep.

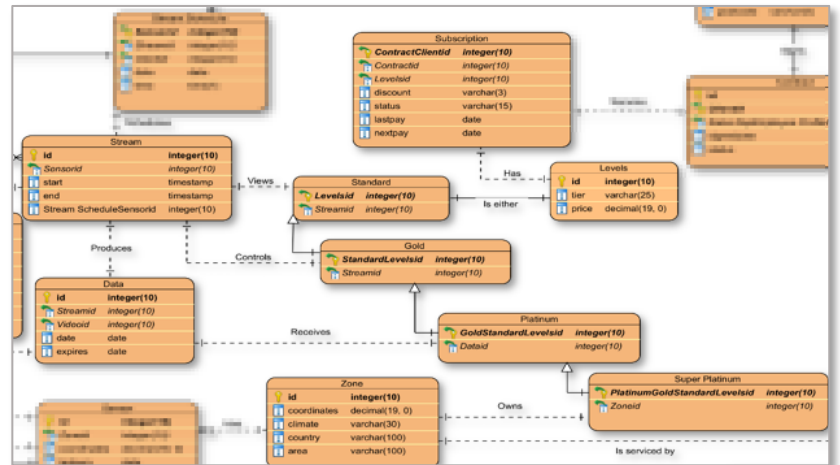


Subscription Levels

A **Client** must select a tier for their **Subscription**. Each tier is identified, tier-label, has their own price, and abilities as recorded in the **Levels** table (<Levels>).

Each tier has a different level ability with/within a SPACES **Stream**. Each increasing tier will inherit the abilities of the previous tier which is identified by their connecting links (W → X → Y → Z) and in their unique **LevelIDs**. Each tier is extended from <Levels> to display their ability.

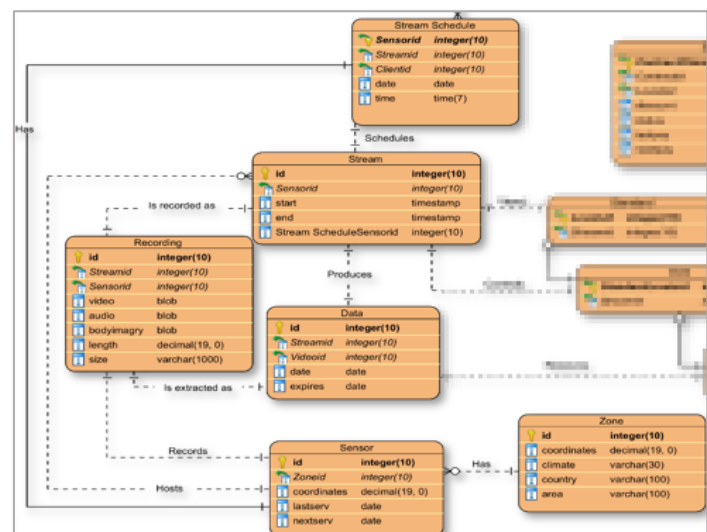
- **Standard** subscriptions simply view a stream.
- **Gold** subscriptions are able to view and control the angle of the view in a stream.
- **Platinum** subscriptions can control view angles and receive exported data from a stream.
- **Super Platinum** subscriptions can control view angles, receive data, and also own a private **Zone** with a private **Sensor**.



Streams, Recordings & Data

A **Sensor** is a device that is used to host a **Stream**. They are identified by a unique ID, located by coordinates, and have dates of regular servicing conducted by **Maintenance workers**. There can be many sensors in a **Zone** which is uniquely identified categorised by ID, has a climate type (forest, urban, alpine, etc.), a country and area, and general coordinates of its location.

To host a **Stream** on a public sensor (i.e. not one in a zone that is privately owned by a **Super Platinum** subscriber), a **Client** must book in a **Stream** via the **Sensor Schedule**. The schedule is recorded with the host's (one who made the booking) **ClientID**, the date and time of the booked stream. This is to privatise events where a client is able to host a stream and invite other participants without cross-over with other users. When a client books a stream, the **StreamID** is produced and can be used as a link or password to a booked event for subscribers to attend.

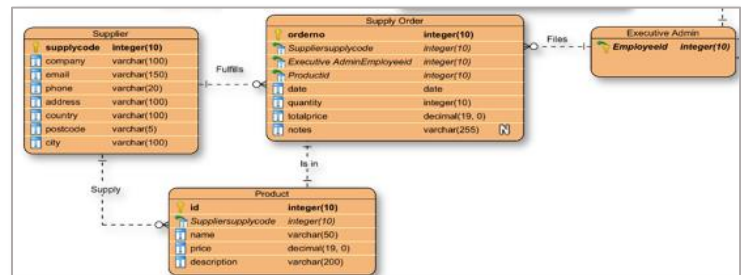


Sensors can record the data produced by a **Stream**. This includes a video file, audio file, and body imagery file as well as the size and time length of the over recording. These details of the **Recording** are conglomerated and extracted as **Data** to be received by **Platinum** and **Super Platinum** subscribers. The **Data** will expire after a week to make storage room in the sensor.

Suppliers and Supply Orders

An **Executive Admin** can make **Supply Order** which is fulfilled by a **Supplier**. An order can only be made from one **Supplier** at a time so that the **Products** within the **Supply Order** are referenced correctly by their **ProductID** that is extracted from the **Supplier's** catalogue and saved into the database.

Recorded in the **Supply Order** is the ID of that order, the unique supply code of the **Supplier**, the ID of the **Executive Admin** filing the order, and the ID of each product being ordered from the **Product** catalogue. The commonly purchased products are compiled into a table <**Products**>



where their price, code of **Supplier**, name, and description can be recorded. The total price of the **Supply Order** is a derived attribute retrieved from the price of each product in the **Products** table and calculated by the set quantity of each product. There is a section within the **Supply Order** where an **Executive Admin** can make any relevant notes.

NaLER Analysis

When conceptualising a database, using natural language to breakdown the entities and their relationships has the benefit of allowing ease of understanding to anyone involved in the project (Nussbaum, 2022). A Natural Language Entity Relationship analysis (NaLER) can provide insight to how the entities in the model should behave or connect. A NaLER analysis can be used to cross reference the relationships between entities as they are being created to make sure the cardinality and optionality are correct (E.g. *A student must be enrolled in one or many courses.*) The same goes for establishing the role for particular attributes of an entity (e.g. *Every client should be identified by their email address*).

See attached a spreadsheet containing a NaLER analysis on the logical model for SPACE. The same file can be found within Github Repository.

Database Dictionary

As seen in the report on the conceptual model (Byrne, 2023), a database dictionary is included with this logical model. To reframe, the purpose of a database diction is to define and maintain the integrity of the data by documenting the particulars of all attributes within each entity, resulting in a reference point for consistency and quality in the model.

Attached bellow is a database dictionary for this logical model. Included in this dictionary is a rundown of each attribute, its associated entity, a description of its purpose, data types, length restraints, applicable value ranges and validation rules, relevant key relations, applicable null values, and references to foreign entities.

See attached spreadsheet or file within Github Repository.

Transitioning the Model

Logical vs Physical Model

Once the logical diagram of the proposed database is completed, it will be analysed against the requirements and feasibility of the physical system. The key elements that differentiate a logical model from a physical model are representation, implementation, and complexity.

Logical models develop the concept of the data by providing data types, lengths, keys, and relationship constraints. Physical models need to be able to be developed into a live database and worked with by administrators (Lithmee, 2019).

At this stage, there are many things to consider when converting a logical model to a physical model. Ensuring that foreign keys correctly represent the relationships between tables, that queries or transactions can be made with a little complexity and runtime as possible, and that the data types correct are amongst the many questions that need to be asked.

Creating the Database

The proposed database for SPACES will be written in the SQL Server query language and managed using the SQL Server Management Studio application.

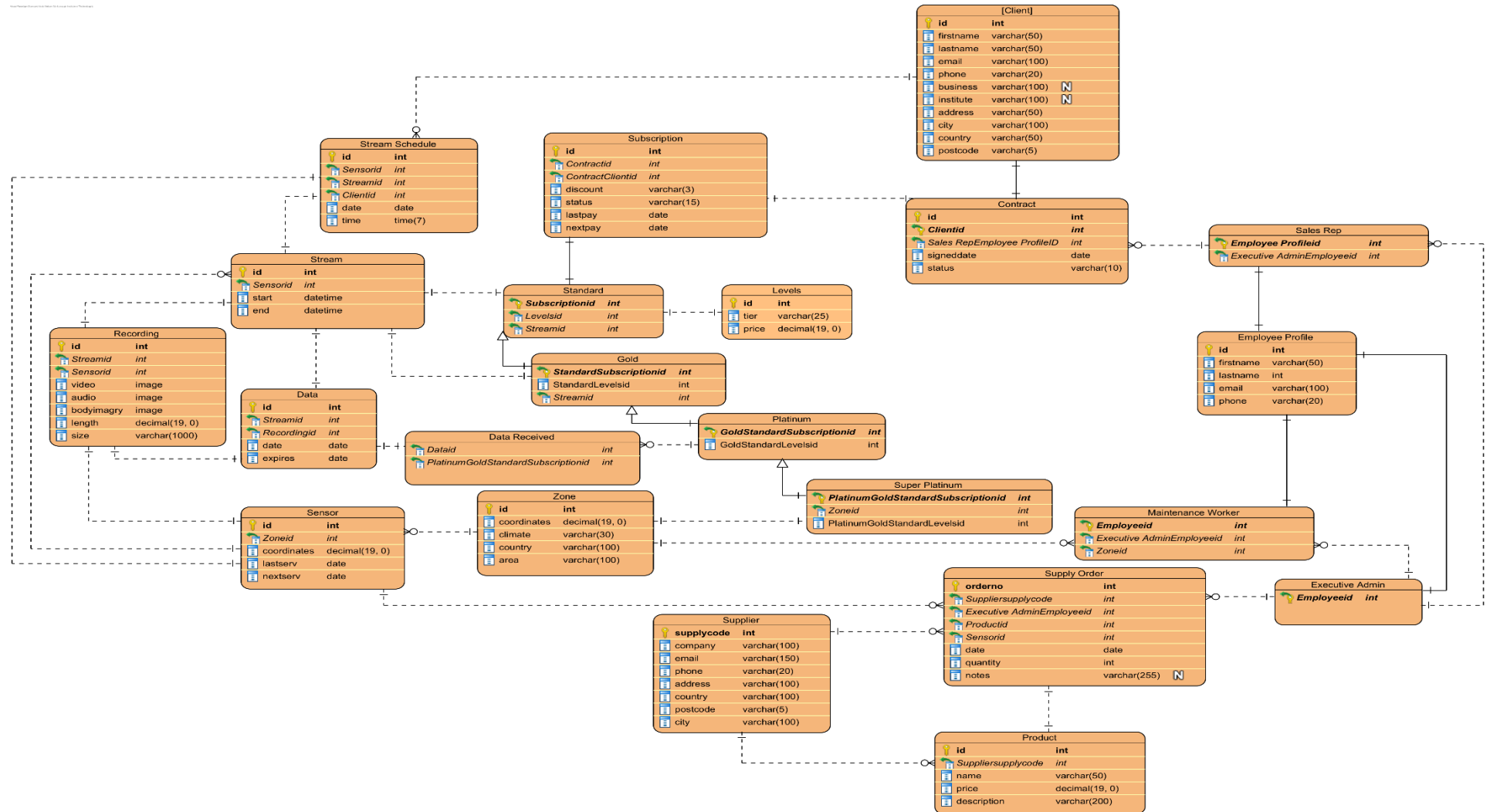
Each entity in the diagram is converted to a table containing the attributes as fields with their respective data types and character limits. As each field is added, the data type and length is considered in regard to how they may be used in queries and how the data type may affect this (e.g. a varchar datatype cannot be used in mathematical equations).

Whilst testing, each table that is created is also dropped in sequence of foreign keys. This allows the database to be wiped from the system and created fresh without issue. If a table has their primary key existing as a foreign key in other tables, those tables must be dropped first. Following this drop process will prevent errors where related tables still exist and are presenting data that doesn't exist. Each set of insert statements will be made like this as well. Data that is referenced will be inserted first and must correlate with related tables where a foreign key exists. Insert data is grouped based on the table for consistency and ease of searching. Relationship constraints are created a dropped like tables to implement business rules within the database.

Creation File

See attached in the Github repository the SQL file for creating the SPACE database and inserting test data.

Physical Model Diagram



Queries & Transactions

Transactions are a set of commands and/or queries that allow the database to execute multiple changes in an isolated set. Transactions differ from other queries (IBM, 2023) as they follow the ACID principle meaning they are:

Atomic – All changes are performed in a stand-alone execution.

Consistent – The data is consistent before and after the execution completes.

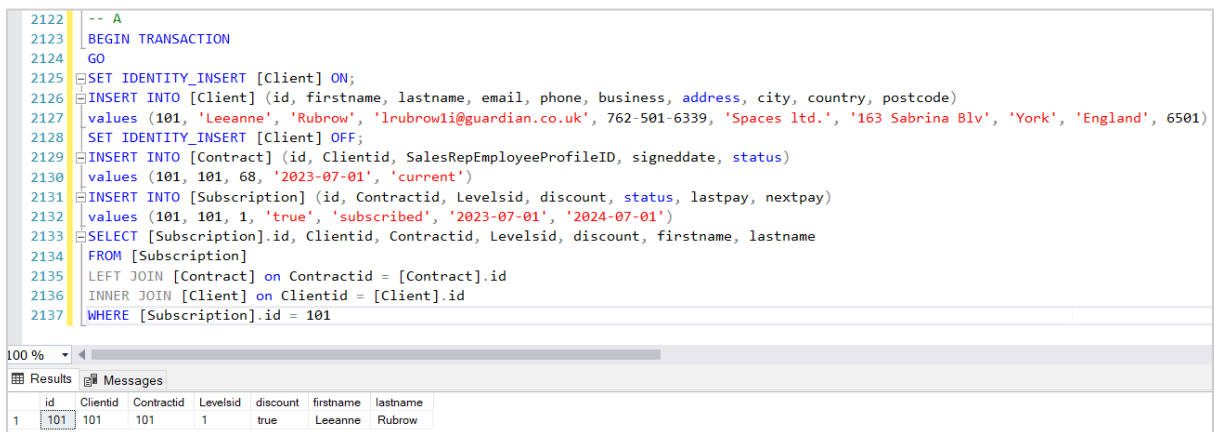
Isolated – All commands and queries in the transaction is separate from all other transactions.

Durable - Changes to the data are persistent despite potential failures.

A set of transactions that could potentially be used in the live database have been tested.

See attached in the Github repository the SQL file to execute the transactions in the physical database.

A. Employees of SPACES are able to purchase a subscription to their own product. In this scenario, a sales representative of SPACES has purchased a standard subscription. Their personal details will be inserted into the client table, a contract will be made to establish their subscription, and their subscription details will include a 3% staff discount.



```
2122 -- A
2123 BEGIN TRANSACTION
2124 GO
2125 SET IDENTITY_INSERT [Client] ON;
2126 INSERT INTO [Client] (id, firstname, lastname, email, phone, business, address, city, country, postcode)
2127 values (101, 'Leeanne', 'Rubrow', 'lrubrow1@guardian.co.uk', 762-501-6339, 'Spaces Ltd.', '163 Sabrina Blv', 'York', 'England', 6501)
2128 SET IDENTITY_INSERT [Client] OFF;
2129 INSERT INTO [Contract] (id, Clientid, SalesRepEmployeeProfileID, signeddate, status)
2130 values (101, 101, 68, '2023-07-01', 'current')
2131 INSERT INTO [Subscription] (id, Contractid, Levelsid, discount, status, lastpay, nextpay)
2132 values (101, 101, 1, 'true', 'subscribed', '2023-07-01', '2024-07-01')
2133 SELECT [Subscription].id, Clientid, Contractid, Levelsid, discount, firstname, lastname
2134 FROM [Subscription]
2135 LEFT JOIN [Contract] on Contractid = [Contract].id
2136 INNER JOIN [Client] on Clientid = [Client].id
2137 WHERE [Subscription].id = 101
```

id	Clientid	Contractid	Levelsid	discount	firstname	lastname
101	101	101	1	true	Leeanne	Rubrow

B. Each Sales Representative can be listed against the clients they have sold a contract to. Identification, both unique and meaningful, can be retrieved on both the sales rep and client including their ID, full name, and address.

```

2122 -- B
2123 BEGIN TRANSACTION
2124 SELECT [Contract].SalesRepEmployeeProfileID, [Employee Profile].firstname AS SalesRep1stNme, [Employee Profile].lastname AS SalesRep2ndNme,
2125 [Client].id AS Clientid, [Client].firstname AS Client1stNme, [Client].lastname AS Client2ndNme, [Client].address AS ClientAddrss
2126 FROM [Contract]
2127 LEFT JOIN [Sales Rep] on [Contract].SalesRepEmployeeProfileID = [Sales Rep].EmployeeProfileid
2128 LEFT JOIN [Employee Profile] on [Contract].SalesRepEmployeeProfileID = [Employee Profile].id
2129 RIGHT JOIN [Client] on [Contract].Clientid = [Client].id
2130 ORDER BY [Contract].SalesRepEmployeeProfileID
2131

```

	SalesRepEmployeeProfileID	SalesRep1stNme	SalesRep2ndNme	Clientid	Client1stNme	Client2ndNme	ClientAddrss
1	49	Heidi	Grinsted	53	Tiebout	Upton	88 Summer Ridge Park
2	49	Heidi	Grinsted	87	Paco	Bawcock	00401 Warrior Terrace
3	50	Felicia	Hyniewicz	82	Davis	Coverly	2054 Fairfield Trail
4	50	Felicia	Hyniewicz	83	Caron	Stapforth	489 Manufacturers Center
5	50	Felicia	Hyniewicz	98	Franklin	O'Cullinane	2207 Graedel Plaza
6	50	Felicia	Hyniewicz	18	Bradano	Hurles	6641 Graedel Way
7	51	Ruthann	Lapenna	11	Teodorico	Seaman	740 Grim Crossing
8	51	Ruthann	Lapenna	55	Joseph	Swinglehurst	7214 McCormick Street
9	52	Inga	Fallnee	69	May	Coan	5 Browning Alley
10	52	Inga	Fallnee	100	Bren	Muglestone	17 Alpine Alley
11	53	Filberto	Walster	99	Victoria	Asten	2 Mariners Cove Junction
12	54	Pepito	Lafaye	92	Aldridge	Oglevie	9 Butternut Parkway
13	54	Pepito	Lafaye	38	Cherlynn	Mayho	50 Mendota Alley
14	54	Pepito	Lafaye	43	Maren	Deevey	7 Oak Valley Junction
15	55	Leeanne	Rubrow	24	Werner	Tallow	74663 Parkside Terrace
16	56	Emmie	Manifold	31	Cedric	Woffenden	61877 Derek Junction
17	56	Emmie	Manifold	1	Nikki	Wrotham	585 Farragut Pass
18	56	Emmie	Manifold	74	Karry	Laven	62 Ramsey Terrace

B (cont.). This can also be filtered by employee id.

```

2123 BEGIN TRANSACTION
2124 SELECT [Contract].SalesRepEmployeeProfileID, [Employee Profile].firstname AS SalesRep1stNme, [Employee Profile].lastname AS SalesRep2ndNme,
2125 [Client].id AS Clientid, [Client].firstname AS Client1stNme, [Client].lastname AS Client2ndNme, [Client].address AS ClientAddrss
2126 FROM [Contract]
2127 LEFT JOIN [Sales Rep] on [Contract].SalesRepEmployeeProfileID = [Sales Rep].EmployeeProfileid
2128 LEFT JOIN [Employee Profile] on [Contract].SalesRepEmployeeProfileID = [Employee Profile].id
2129 RIGHT JOIN [Client] on [Contract].Clientid = [Client].id
2130 WHERE [Employee Profile].id = '49'
2131

```

	SalesRepEmployeeProfileID	SalesRep1stNme	SalesRep2ndNme	Clientid	Client1stNme	Client2ndNme	ClientAddrss
49	49	Heidi	Grinsted	53	Tiebout	Upton	88 Summer Ridge Park
49	49	Heidi	Grinsted	87	Paco	Bawcock	00401 Warrior Terrace

C. When a stream is complete, the audio and visual activity is recorded as data which is sent to platinum and super platinum subscribers. Included in the data is a video recording of the stream, the audio recording, and the body imagery as well as references to the stream, sensor, start and finish times, length of files, and their size. This is all saved under a data file that references back to the recording and the stream.

Below is a transaction showing a new stream being recorded and saved as data.

```

2122 -- C
2123 BEGIN TRANSACTION
2124 SET IDENTITY_INSERT [Stream] ON;
2125 insert into [Stream] (id, Sensorid, [date], [start], [end]) values (101, 56, '2023-07-18', '14:30', '20:40');
2126 SET IDENTITY_INSERT [Stream] OFF;
2127 SET IDENTITY_INSERT [Recording] ON;
2128 insert into [Recording] (id, Streamid, Sensorid, [video], audio, bodyimagry, [length], [size])
2129 values (101, 101, 56, 'BlackMirror.avi', 'LochHenry.wav', 'Joan.cad', '6:23:45', 6548);
2130 SET IDENTITY_INSERT [Recording] OFF;
2131 SET IDENTITY_INSERT [Data] ON;
2132 insert into [Data] (id, Streamid, Recordingid, [date], expires) values (101, 101, 101, '2023-07-18', '2023-08-18');
2133 SET IDENTITY_INSERT [Data] OFF;
2134 SELECT [Recording].id AS Recordingid, [Recording].Streamid, [Data].id AS Dataid, [Data].date
2135 FROM [Recording]
2136 LEFT JOIN [Data] on [Recording].id = [Data].Recordingid
2137 WHERE [Recording].id = '101'
2138
2139

```

00 %

Results Messages

Recordingid	Streamid	Dataid	date
101	101	101	2023-07-18

D. Each client can view a stream that is produced by a sensor. Each sensor can be located by its coordinates and identified by a unique id. When a client joins a sensor's stream, their details are recorded in a stream schedule. Below is a transaction that list the sensor id, coordinates, and details of clients that are subscribed to each sensor.

```

2122 -- D
2123 SELECT [Sensor].id, [Sensor].coordinates, [Stream Schedule].Clientid, [Client].firstname, [Client].lastname, [Client].business
2124 FROM [Sensor]
2125 LEFT JOIN [Stream Schedule] on [Sensor].id = [Stream Schedule].Sensorid
2126 RIGHT JOIN [Client] on [Client].id = [Stream Schedule].Clientid
2127 WHERE [Client].business IS NOT NULL
2128

```

0 %

Results Messages

id	coordinates	Clientid	firstname	lastname	business
90	45.4726625	52	Ruben	Massy	Zulauf and Sons
1	27.845011	53	Tiebout	Upton	Nitzsche Inc
66	-0.421633	56	Crysta	Blumfield	Reynolds LLC
42	-7.221299	57	Maighdiln	Egarr	Brekke, Bauch and Zulauf
23	-22.2528986	57	Maighdiln	Egarr	Brekke, Bauch and Zulauf
65	35.547333	63	Nancy	Guttridge	Pouros Inc
25	-6.8230945	63	Nancy	Guttridge	Pouros Inc

E. (Could not complete without changing database structure)

F. It is probable that sales reps may want to see details of each stream that is currently live. This transaction produces a simple list of live sensors, the stream id, and details of the client viewing it.

```

2122 -- F
2123 SELECT sensorid, streamid, clientid, firstname, lastname from [Stream Schedule]
2124 LEFT JOIN [Client] on [client].id = clientid
2125 ORDER BY Streamid asc;

```

100 %

	sensorid	streamid	clientid	firstname	lastname
1	10	1	100	Bren	Muglestone
2	57	2	9	Marney	Burkwood
3	89	3	80	Engelbert	Bonifant
4	71	7	15	Menard	Emmanueli
5	82	7	62	Rona	Brodeau
6	66	8	14	Lester	Pennuzzi
7	90	9	95	Nydia	Worters
8	24	10	9	Marney	Burkwood
9	44	10	78	Stephana	Bullard
10	9	10	29	Rutherford	Castanaga
11	47	11	61	Nicky	Arondel
12	78	13	89	Tessa	Mengue

G. When an executive admin makes a supply order for a sensor, that sensor's id will be recorded against the order for auditing and reference purposes so that those materials are sent to the correct sensor. This transaction shows all orders made for a given sensor including details of the supplies ordered and the supply company they came from.

```

2122 -- G
2123 SELECT orderno, sensorid, [Supply Order].Suppliersupplycode, [Supplier].company, productid, [Product].name
2124 FROM [Supply Order]
2125 LEFT JOIN [Supplier] on [Supplier].supplycode = [Supply Order].Suppliersupplycode
2126 INNER JOIN [Product] on [Product].id = productid
2127 WHERE sensorid = '31';

```

100 %

	orderno	sensorid	Suppliersupplycode	company	productid	name
1	34	31	2173	Kwinu	71084	Steel
2	57	31	4990	Jabberbean	74994	Glass
3	70	31	6086	Nlounge	42981	Plexiglass
4	84	31	2333	Trudoo	81163	Granite

H. If a sensor is relocated or reallocated to different zone, their zone id, and coordinates will need to be updated as shown in this transaction.

2129 SELECT * FROM [Sensor]
 2130 WHERE id = '32'

100 %

	id	Zoneid	coordinates	lastserv	nextserv
1	32	46	45.156512	2022-10-15	2023-01-24

2122 -- H
 2123 BEGIN TRANSACTION
 2124 UPDATE [Sensor]
 2125 SET Zoneid = '55', coordinates = 37.83295626
 2126 WHERE id = '32'

2127
 2128
 2129 SELECT * FROM [Sensor]
 2130 WHERE id = '32'

100 %

	id	Zoneid	coordinates	lastserv	nextserv
1	32	55	37.83295626	2022-10-15	2023-01-24

I. If a client opts out of their contract or fails to pay for a subscription within a year of their last due payment, their subscription and contract details will be voided and removed from the system. Below is a transaction that will delete these details associated with the client id.

2130 SELECT * FROM [Subscription]
 2131 WHERE Contractid = '88'

100 %

	id	Contractid	Levelsid	discount	status	lastpay	nextpay
1	88	88	4	true	subscribed	2022-09-29	2023-11-14

2122 -- I
 2123 BEGIN TRANSACTION
 2124 GO
 2125 DELETE FROM [Subscription] WHERE Contractid = '88';
 2126 DELETE FROM [Contract] WHERE id = '88'
 2127 Print 'Contract Deleted'
 2128 GO

%

Messages

(1 row affected)
 (1 row affected)
 Contract Deleted

Completion time: 2023-07-12T14:09:43.0610082+12:00

J. Executive admins of SPACES make supply orders for the maintenance workers that repair the sensors. This transaction will produce a list of each sensor and the total cost of all materials purchased.

```
2123 -- J
2124 SELECT [Supply Order].Sensorid, SUM([Supply Order].quantity*[Product].price) AS totalrepairprice
2125 FROM [Supply Order]
2126 LEFT JOIN [Product] on [Supply Order].Productid = [Product].id
2127 GROUP BY Sensorid
2128 ORDER BY Sensorid asc;
```

100 %

	Sensorid	totalrepairprice
1	1	3132
2	2	19176
3	4	86591
4	5	1898
5	7	18270
6	8	108000
7	10	30681

References

- Agar, R. (2021, May 19). *Stages and types of data models*. TDAN.com. <https://tdan.com/stages-and-types-of-data-models/28201>
- Brumm, B. (2022, September 26). *A guide to the entity relationship diagram (ERD)*. Database Star. <https://www.databasestar.com/entity-relationship-diagram/>
- Corbo, A. (2023, January 3). *What is data modeling?* Built In. <https://builtin.com/data-science/data-modeling>
- DataAcademy.in. (2020, May 1). *Conceptual, Logical & Physical Data Models* [Video]. YouTube. <https://www.youtube.com/watch?v=cY7WZYhyC3o>
- Decomplexify. (2021, November 21). *Learn database normalization - 1NF, 2NF, 3NF, 4NF, 5NF* [Video]. YouTube. https://www.youtube.com/watch?v=GfQaEYEc8_8&t=791
- Dybka, P. (2014, August 2). *Chen notation*. Vertabelo. <https://vertabelo.com/blog/chen-erd-notation/>
- Dybka, P. (2016, March 3). *Crow's foot notation*. Vertabelo. <https://vertabelo.com/blog/crow-s-foot-notation/>
- Gordon, K. (2007). *Principles of data management: Facilitating information sharing*. BCS, The Chartered Institute.
- IBM. (2023, January 7). *ACID properties of transactions*. <https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions>
- Lithmee. (2019, July 4). *What is the difference between logical and physical data model*. Pediaa. <https://pediaa.com/what-is-the-difference-between-logical-and-physical-data-model/>
- Nalimov, C. (2021, July 29). *Quick guide to physical data modeling*. Gleek.io. <https://www.gleek.io/blog/physical-data-modeling>

Nussbaum, B. (202, March 25). *How natural language processing (NLP) works in graph databases.*

GraphGrid. <https://graphgrid.com/blog/how-natural-language-processing-nlp-works-in-graph-databases/>

Rithika, S. (2022, April 1). *Understanding the stages & types of data models: A comprehensive guide*

101. Learn | Hevo. <https://hevodata.com/learn/types-of-data-model/>

Stewart, J. (2008, June 1). *Crow's feet are best.* The Data Administrator Newsletter.

<https://tdan.com/crows-feet-are-best/7474>

StudyTonight. (2023). *4th normal form (4NF) and mutli-valued dependency in database*

normalization. <https://www.studytonight.com/dbms/fourth-normal-form.php>

Yue, K. (2017). *The extended entity relationship model.* SCE Support Center.

https://dcm.uhcl.edu/yue/courses/itec3335/Fall2018/notes/model/Extended_ERDiagram.html