MAY 26, 2023

FROM CHEN TO CROW'S FOOT

DAT601 ASSESSMENT 2

KIRA BYRNE
STUDENT ID #1350995
NMIT

Contents

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

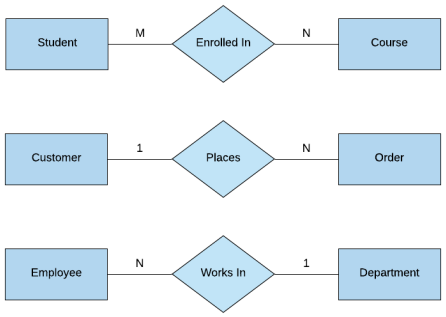
The Crows Foot notation displays entity and attributes in a tabular format. Each attribute is assigned a data type and length and their primary keys exist as foreign keys in applicable relationships with other entity tables.

Relationships, Cardinality, and Optionality

Cardinality and optionality extend from two entities to meet 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.



Chen's Notation Cardinality (Brumm, 2022)

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.

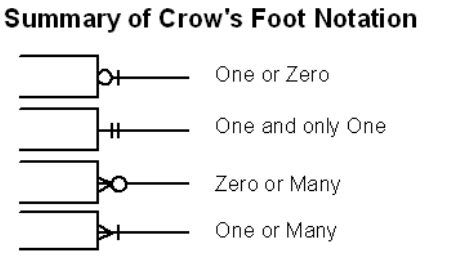


Figure 1 Crow's Foot Cardinality (Stewart, 2008)

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 exists 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 it's 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 → 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 (0NF) has not been normalised at any rate. From 0NF, 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 0NF to 1NF in a primary school database:

<0NF>

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 → 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 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 → 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

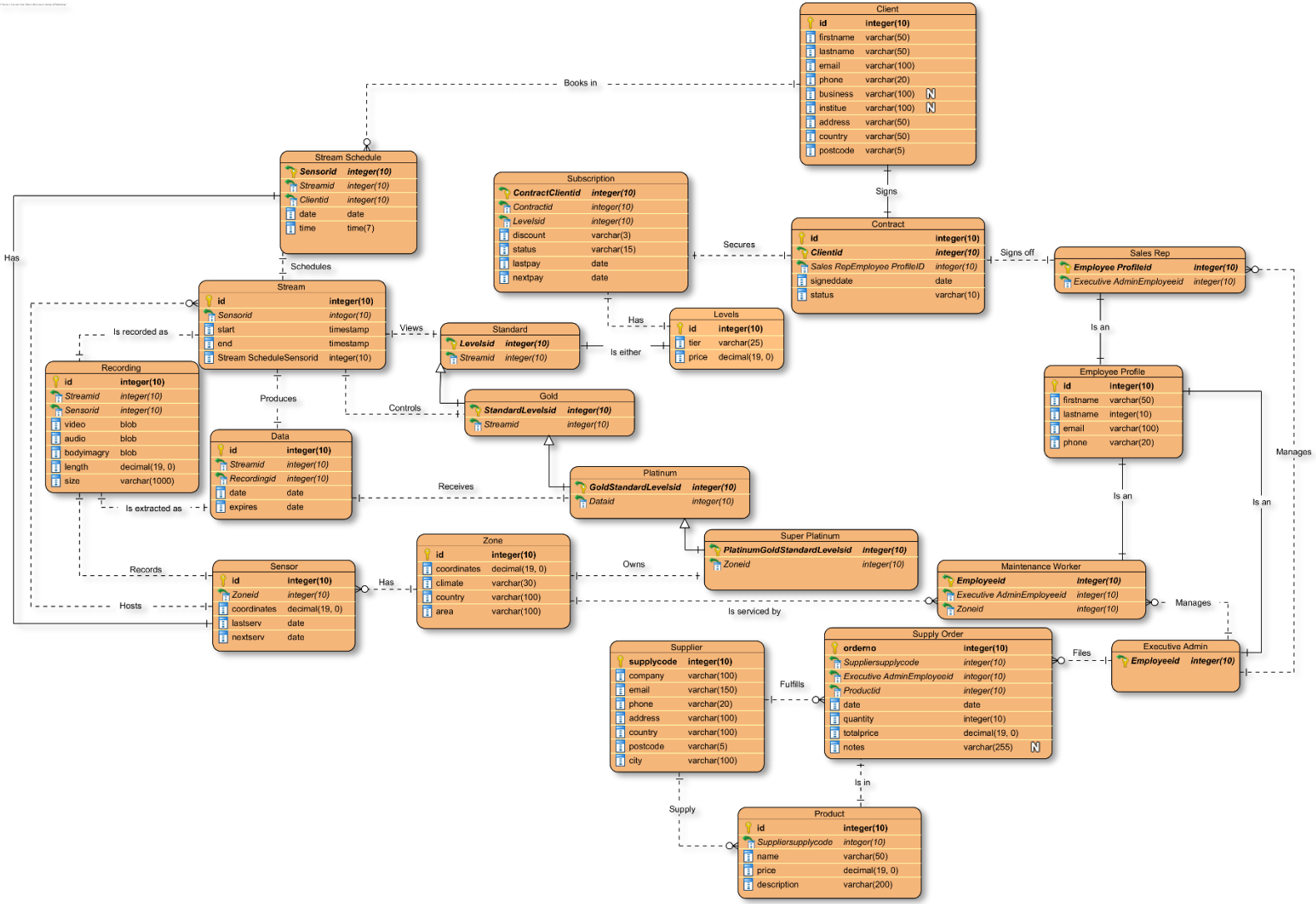
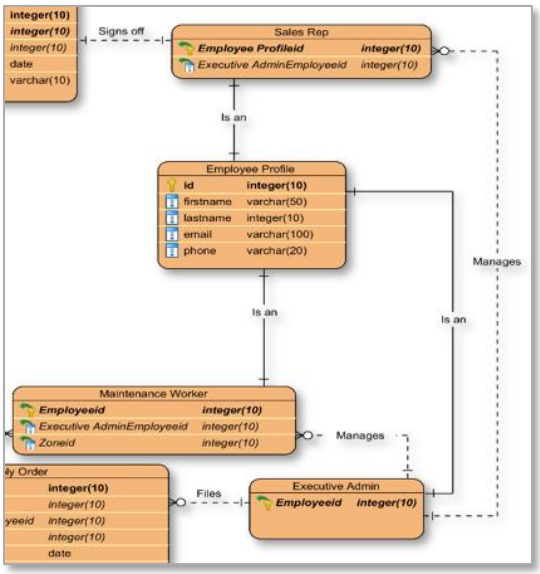


Diagram Rationale

Employees



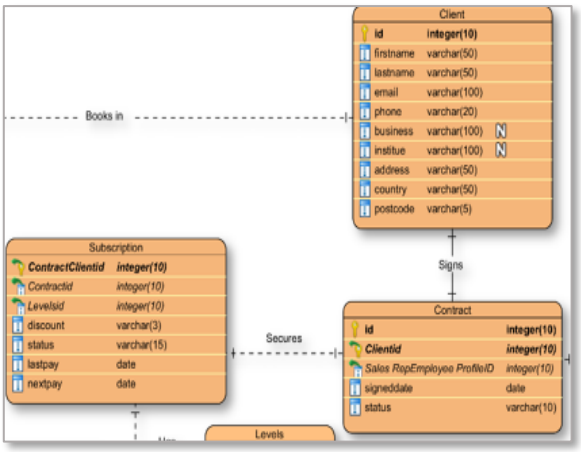
Each **Employee** has their details recorded in a profile (<Employee>). This profile includes their unique ID, name, and work-provided contact details.

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 o ff 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.

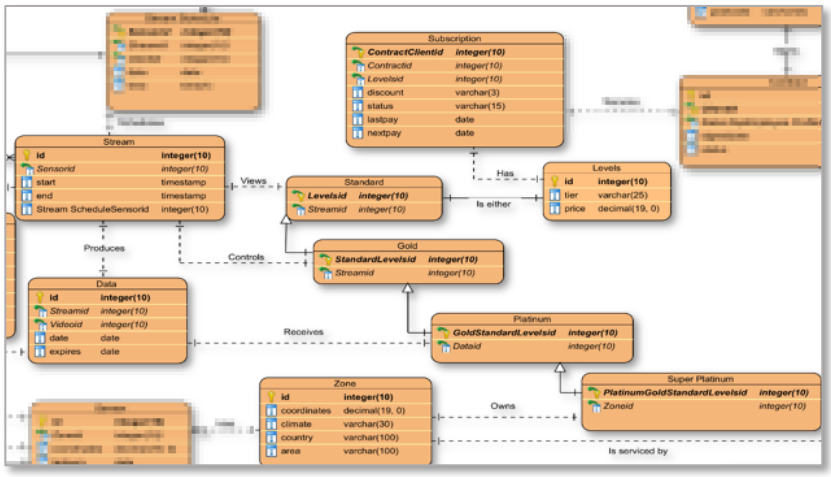


A **Client** is related indirectly to their subscription through a **Contract** which holds details about their involvement with SPACES. The reason for this is that if the client decides to unsubscribe for a time, their partnership with SPACES is not made void.

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



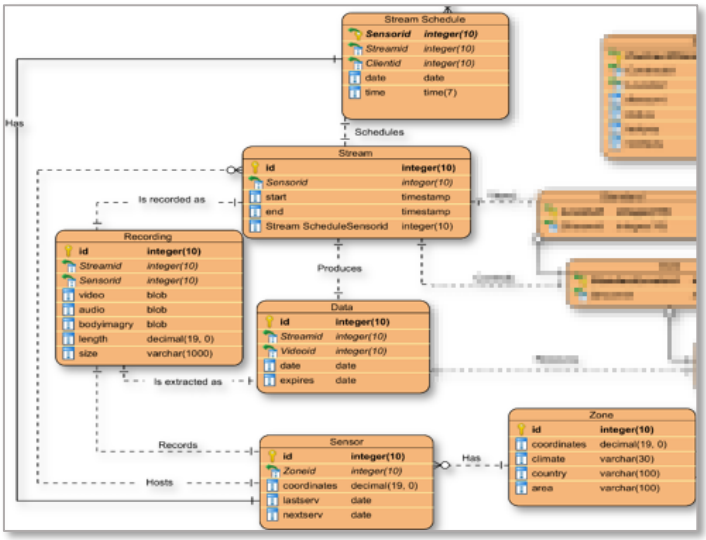
A **Client** must select a tier for their **Subscription**. Each tier is identified, tier-label, has their own price, and abilities as recorded in the **Levels** table (<Levels>).

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**. The 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.

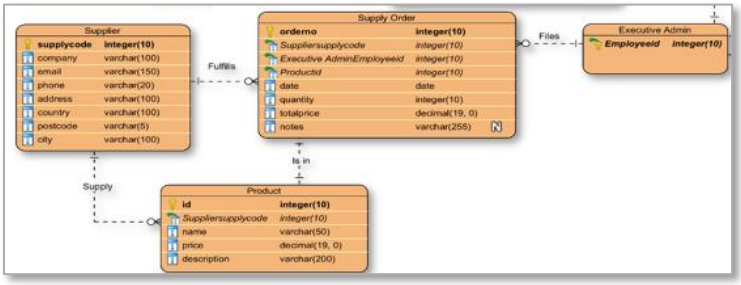


To host a **Stream** on a public sensor (i.e. not one in a zone that is privately owned by a **Super Platinum** subscriber), a **Client** must book in a **Stream** via the **Sensor Schedule.** The schedule is recorded with the host's (one who made the booking) **ClientID**, the date and time of the booked 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:

NaLER Analysis.xlsx

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:

DBD Attributes Assessment 2.xlsx

References

Brumm, B. (2022, September 26). *A guide to the entity relationship diagram (ERD)*. Database Star. https://www.databasestar.com/entity-relationship-diagram/

Byrne, K. (2023). *Spaces and the chen notation*. Nelson Marlborough Institute of Technology.

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. (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.

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/

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