

|  |
| --- |
| DAT601 Milestone 2 |
| Logical Database Design |
| Name: Rhylei Tremlett  ID: 13513865  Date: 28/05/2023 |

Table of Contents

[Conceptual Model to Logical Model 1](#_Toc136205191)

[Mapped 1](#_Toc136205192)

[Participation consideration 2](#_Toc136205193)

[Stable 2](#_Toc136205194)

[Normalisation 2](#_Toc136205195)

[First Normal Form (1NF) 2](#_Toc136205196)

[Second Normal Form (2NF) 3](#_Toc136205197)

[Third Normal Form (3NF) 3](#_Toc136205198)

[Boyce-Codd Normal Form (BCNF) 3](#_Toc136205199)

[Fourth Normal Form (4NF) 3](#_Toc136205200)

[Logical Entity Relationship Diagram 4](#_Toc136205201)

[Rationale 4](#_Toc136205202)

[Logical Data Dictionary 4](#_Toc136205203)

[NaLER Analysis 5](#_Toc136205204)

# Conceptual Model to Logical Model

There is a list of rules that you must adhere to when translating a conceptual entity-relationship model into a logical relational model. There are three techniques used to translate conceptual models into logical models, they are:

* Mapped
* Stable
* Mapped (participation consideration)

Each technique alters some of the translation rules, which in turn produces a slightly different model with differing benefits. First, I will list the common rules between all three techniques and then define the differences afterwards.

Firstly, all entities in the conceptual model become a relation of the same name in the logical model.

Next, all attributes are transferred over using the following rules:

1. All simple attributes are transferred without change.
2. Composite attributes are divided into multiple atomic attributes.
3. The entity’s unique identifier may be used as the primary key if it is found to be a suitable candidate key.

Secondly, relationships must be translated. Among the three techniques, the only relationship that remains the same is the Many-to-Many. To translate this relationship, both entities are transformed into relations, and a new relation is formed to represent the relationship. In this new relation, the primary keys of both entities are stored as foreign keys, which are combined to establish the relationship. Any attributes of the relationship in the conceptual model will be translated into the new relation that represents the relationship.

To translate Specialisation / Generalisation to the conceptual model, there are three methods to choose from. The most commonly used method is creating one relation for each sub-type and one for the super-type. Each sub-type contains the specific attributes of that sub-type. The super-type comprises all the common attributes of the entities.

The two lesser-used methods are to create one relation containing all sub-type and super-type attributes or one relation for each sub-type containing all of the specific and common attributes (no super-type). The first of the two methods results in many null values. The second method doesn’t create null values but does duplicate data, increasing the database’s size. However, it is highly convenient for querying.

## Mapped

The mapped technique is the most commonly used technique for this process. Its focus is to minimise the number of tables that are created. The trade-off for this is that more null values can be produced, and the model is unstable, meaning it is weak to fundamental changes in the problem domain.

The rules are:

For One-to-One relationships, no extra relation is created. Instead, the primary key of one relation is inserted into the other as a foreign key. The logical model does not care which one receives the key as it is a semantic issue. However, it is up to the designer to decide which relation feels more natural to receive the foreign key.

For One-to-Many relationships, no extra relation is needed. The relation that is on the ‘many’ side of the relationship is the one that receives the foreign key of the relation on the ‘one’ side. If the relationship has any attributes, they are given to the relation receiving the foreign key.

### Participation consideration

In this variation of the mapped technique, the participation of each relationship must be checked. If the participation is partial, then the rules from the stable technique may be applied.

## Stable

The stable technique is lesser-used, resulting in lesser performance because of having more relations. The stable technique creates a new relation for every relationship in the conceptual model. Creating a relation for all relationships maximises the number of tables, which lowers performance due to more tables creating more math. The stable technique also creates no null values, which is desirable. However, most consider that the performance trade isn’t worth it and opt to use mapped instead.

# Normalisation

## First Normal Form (1NF)

To be considered in the first normal form, a relation must have no repeating groups and only contain atomic values. To break this down more, a repeating group is a record of data repeating itself, therefore, not unique. To achieve no repeating groups, it is common to introduce or identify a primary key. This naturally results in unique records, as a primary key cannot repeat itself.

Atomic values (also known as simple values) are attributes broken down into atomic parts. A multivalued field, phone numbers, for example, could contain multiple numbers within the same field. Doing so violates the first normal form. Non-atomic values must be separated into atomic values to be in the first normal form.

Here is a list of rules for the first normal form:

* There must be no repeating groups (all records must be uniquely identified).
* All values must be completely atomic.
* There must be no duplicate columns.
* All values within a column must be of the same domain.

## Second Normal Form (2NF)

For a relation to meet the second normal form, it must be in the first normal form, and all non-prime attributes must be fully dependent on the primary key. In other words, there are no partial dependencies.

A non-prime attribute is any attribute that is not a candidate key or part of one.

## Third Normal Form (3NF)

For a relation to be in the third normal form, the relation must first be in the second normal form. The relation must also have no transitive dependencies, that is, when a non-prime attribute is the determinant of another non-prime attribute. Another way to phrase this is that all non-prime attributes must fully depend on the primary key.

## Boyce-Codd Normal Form (BCNF)

The Boyce-Codd normal form is an addition to the third normal form and is considered 3.5.

For a relation to be in BCNF, it must be these conditions:

* The relation is already in 3NF.
* For every functional dependency of the relation, the determinant should be a super key.

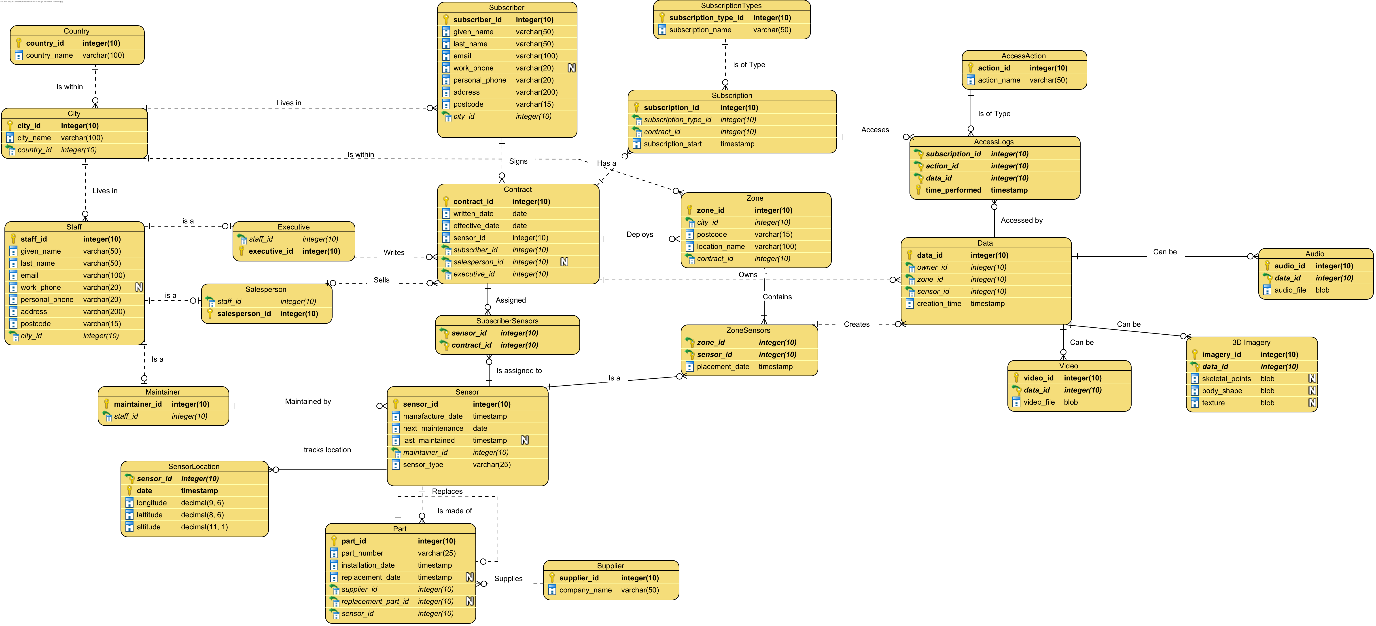
Another way to define the last condition is that for every dependency (X 🡪 Y), the determinant must be a prime attribute.

## Fourth Normal Form (4NF)

For a relation to meet the fourth normal form, it must already be in the BCNF and have no multi-valued dependencies.

A multi-valued dependency is when the determinant X determines multiple instances of Y.

# Logical Entity Relationship Diagram



Click this to open in the browser or photo viewer.

## Rationale

I normally don’t actively consider normalisation while I am designing. However, my instinctive design methods are to normalise data automatically. For the most part, my logical model was already normalised when I completed it, I chose to extract city and country out of the staff relation into their own tables, but I left postcode in the staff relation (this also applies to the subscriber relation).

My most notable action while designing was implementing the AccessLogs relation and the accompanying AccessAction relation. This was how I decided to implement the relationship of standard and gold subscribers accessing and controlling video streams.

# Logical Data Dictionary





# NaLER Analysis

1. Each Staff is uniquely identified by one staff\_id
2. Each Staff must have one given\_name
3. Each Staff must have one last\_name
4. Each Staff must have one email
5. Each Staff may have one work\_phone
6. Each Staff must have one personal\_phone
7. Each Staff must have one address
8. Each Staff must have one postcode
9. Each Staff must live in one City(city\_id)
10. Each Executive is uniquely identified by one executive\_id
11. Each Executive is a Staff(staff\_id)
12. Each Salesperson is uniquely identified by one salesperson\_id
13. Each Salesperson is a Staff(staff\_id)
14. Each Maintainer is uniquely identified by one maintainer\_id
15. Each Maintainer is a Staff(staff\_id)
16. Each Subscriber is uniquely identified by one subscriber\_id
17. Each Subscriber must have one given\_name
18. Each Subscriber must have one last\_name
19. Each Subscriber must have one email
20. Each Subscriber may have one work\_phone
21. Each Subscriber must have one personal\_phone
22. Each Subscriber must have one address
23. Each Subscriber must have one postcode
24. Each Subscriber must live in one City(city\_id)
25. Each City is uniquely identified by one city\_id
26. Each City must have one city\_name
27. Each City must be within one Country(country\_id)
28. Each Country City is uniquely identified by one country\_id
29. Each Country must have one country\_name
30. Each Subscription is uniquely identified by one subscription\_id
31. Each Subscription must have a SubscriptionType(subscription\_type\_id)
32. Each Subscription must link to one Contract(contract\_id)
33. Each Subscription must have one subscription\_start
34. Each SubscriptionType is uniquely identified by one subscription\_type\_id
35. Each SubscriptionType must have one subscription\_name
36. Each Contract is uniquely identified by one contract\_id
37. Each Contract must have one written\_date
38. Each Contract must have one effective\_date
39. Each Contract must be allocated one Sensor(sensor\_id)
40. Each contract may be sold by one Salesperson(salesperson\_id)
41. Each Contract must be written by one Executive(executive\_id)
42. Each Sensor is uniquely identified by one sensor\_id
43. Each Sensor must have one manafacture\_date
44. Each Sensor must have one next\_maintenance
45. Each Sensor may have one last\_maintenance
46. Each Sensor must be maintained by one Maintainer(maintainer\_id)
47. Each Sensor must have one sensor\_type
48. Each SubscriberSensor is uniquely identified by one subscriber\_id and sensor\_id
49. Each ZoneSensor is uniquely identified by one zone\_id and sensor\_id
50. Each ZoneSenor must have one placement\_date
51. Each Zone is uniquely identified by one zone\_id
52. Each Zone must be located within one City(city\_id)
53. Each Zone must have one postcode
54. Each Zone must have one location\_name
55. Each Zone must be deployed by one Contract(contract\_id)
56. Each SensorLocation is uniquely identified by one sensor\_id and date
57. Each SensorLocation must have one longitude
58. Each SensorLocation must have one latitude
59. Each SensorLocation must have one altitude
60. Each Part is uniquely identified by one part\_id
61. Each Part must have one part\_number
62. Each Part must have one installation\_date
63. Each Part may have one replacement\_date
64. Each Part must be supplied by one Supplier(supplier\_id)
65. Each Part may be replaced by another Part(replacement\_part\_id)
66. Each Part is part of a Sensor(sensor\_id)
67. Each Supplier is uniquely identified by one supplier\_id
68. Each Supplier must have one company\_name
69. Each AccessLogs is uniquely identified by one subscription\_id, subscription\_type\_id, data\_id and time\_performed
70. Each Data is uniquely identified by one data\_id
71. Each Data must be owned by one Contract(contract\_id)
72. Each Data must be created by one ZoneSensor(sensor\_id, zone\_id)
73. Each Data must have one creation\_time
74. Each Video is uniquely identified by one video\_id and data\_id
75. Each Video must have one video\_file
76. Each Imagery is uniquely identified by one imagery\_id and data\_id
77. Each Imagery may have one skeletal\_points
78. Each Imagery may have one body\_shape
79. Each Imagery may have one texture
80. Each Audio is uniquely identified by one audio\_id and data\_id
81. Each Audio has one audio\_file