# DAT601- DATABASE DESIGN METHODOLOGY

[DAT601- DATABASE DESIGN METHODOLOGY 1](#_Toc198893252)

[Conceptual to Relational 2](#_Toc198893253)

[Mapping rules 2](#_Toc198893254)

[Logical model 3](#_Toc198893255)

[Version 1.1 3](#_Toc198893256)

[Version 1.2 4](#_Toc198893257)

[Rational and explaination 4](#_Toc198893258)

[Normalisation 7](#_Toc198893259)

[Data dictionary 8](#_Toc198893260)

[Relations 8](#_Toc198893261)

[Attributes 9](#_Toc198893262)

[Derived attributes 17](#_Toc198893263)

[NaLER analysis report 17](#_Toc198893264)

[zone 17](#_Toc198893265)

[task 17](#_Toc198893266)

[Task\_type 18](#_Toc198893267)

[Drone\_data 18](#_Toc198893268)

[Account 18](#_Toc198893269)

[Contract 18](#_Toc198893270)

[Subscription 19](#_Toc198893271)

[Payment 19](#_Toc198893272)

[Merchant 19](#_Toc198893273)

[Supply 19](#_Toc198893274)

[Component 19](#_Toc198893275)

[Frame 20](#_Toc198893276)

[Battery 20](#_Toc198893277)

[Prop 20](#_Toc198893278)

[camera 20](#_Toc198893279)

[drone 20](#_Toc198893280)

[Staff 20](#_Toc198893281)

[Qualification 21](#_Toc198893282)

[Position 21](#_Toc198893283)

[Sales 21](#_Toc198893284)

[Location 21](#_Toc198893285)

[component\_type 21](#_Toc198893286)

[Component\_maintenance 21](#_Toc198893287)

[Final report 22](#_Toc198893288)

[Map ER model to relations 22](#_Toc198893289)

[First version 1.0 22](#_Toc198893290)

[Revised version 1.2 22](#_Toc198893291)

[Changes made to ERD 23](#_Toc198893292)

[Data modeling in information systems ERD and Logical model 31](#_Toc198893293)

[Data principles 32](#_Toc198893294)

# Conceptual to Relational

## Mapping rules

* Entities become relation tables.
* Attribute become table fields.
* Identify possible unique identifiers.
* Map relationships between entities defining them.
* Handle multi valued attributes by creating extra tables.
* Normalisation.
* Relationships with entities are turned into relations.
* Superclass

1. Create one relation to told common attributes and a relation for each sub-relation that hold the required information.
2. Create sub-relations only.
3. Create a masterclass to hold all the information requires NULLS.

* Normalise model for data integrity and business rules.
* Identify relationships using foreign keys.
* Define integrity constraints – check constraints e.g. NOT NULL, data types, PK, FK.

# Logical model

## Version 1.1

A diagram of a computer

AI-generated content may be incorrect.

## Version 1.2

A screenshot of a computer

AI-generated content may be incorrect.

Example the normalisation between the two versions… why or why not the relation is in its normal form and not high or lower. Why only 3NF??

### Rational and explaination

**merchant**

I first start with the merchant entity from the ERD and created a relation named merchant

And started to give the required attributes like name and email as there wasn’t really any unique identifier in the relation other than the email but they might change their email at some point that would be out of my control so I when with an ID to create a unique identifier for each row. (3NF)

1 merchant can have M supply

**Supply**

The supply entity in the ERD became the supply relation and it has attributes like the part\_name (Most likely coming from merchant database) and the price for that part\_name as there are no unique identifiers I have create one, ID. The relation relates to itself to create dependency so that one or more supplyID can relate to one ID. (3NF)

M supply for 1 component

**Component**

The component entity in the ERD became the component relation and it has attributes like name, used\_time and stock amount, their was no unique identifier so I create one ID. (3NF)

M components have 1 location

**Location**

Location comes the entity component from the ERD, location was to create to meet normalisation standards removing transitive dependencies and improving data integrity (3NF).

**Component subclasses**

I’m not sure if these are right for a logical model

1 component has M component\_maintenance

**Component\_maintenance**

The component\_maintenance entity for the ERD became the component\_maintenance relation including attributes like description and date. As there may be more than one component\_maintenance in one component\_maintenance, I couldn’t use date as the unique identity, so I created an ID that can reference itself. (3NF)

M component\_maintenance happens to 1 drone

**Drone**

The drone entity for the ERD became the drone relation and the relation has build\_date, total\_hours, and status. None of the attributes are unique so I created an ID for this purpose.

(3NF)

1 drone has M drone\_date

**Drone\_data - weak entity**

The drone\_data entity became the drone\_data relation with attributes to record the data like temp, humidity, ambient\_light\_strength, and air\_quality. There were no suitable unique identifiers for the relation. So, I used a composite key made up of accountID, taskID from their relations. This uniquely identifies each instance of drone\_data and forces the constraint that each drone\_data must relate back to a task performed and the account that needs the task done (BCNF).

M drone\_data has 1 account (composite key)

M drone\_date has 1 account (composite key)

M drone\_date has 1 task

**Account**

The account entity became the account relation and included attributes like contact\_person, email, and address. None of these attributes fit the description of a primary identifier so I created an ID attribute to act as such I possibly could have used the email as this is unique identifier, but it has a probability of possibly changing causing possible inconsistencies if constraint isn’t put in place.

1 account has many contract

M account has 1 staff

1 account has multiple payments

**Contract**

The contract entity became the contract relation with attributes like end\_date, discount, and total to create the required data structure for a contract agreement. There are no unique attributes. I considered a composite key that would include accountID, subscription\_type, and end\_date and I may change this later this would align with (BCNF). Currently it has an ID that acts as such (3NF).

M contract has 1 subscription

M contract has 1 staff

**subscription**

The subscription entity became the subscription relation with attributes like type, max\_drones, max\_area, max\_zone, and cost. As type is unique this became the primary key for the relation.

1 subscription belongs to M contract

**staff**

The staff entity became the staff relation with attribute values name, email. Email was considered as an identifier but went with my own created ID to limit the possibility of data inconsistencies that may occur if staff emails were to change. (3NF)

* Note

Possible changes to make turn ID, positionID into composite key (BCNF)

**salesperson**

The salesperson is a subclass from staff it comes from the salesperson entity in the ERD it has attribute customer\_count to record how many customers the staff member manages.

1 staff has M qualifications

M staff has 1 position

**qualification**

The qualification relation was created to differentiate between staff and is consistent with (3NF) principles. It has attributes values description and has an ID to act as a primary identifier.

M qualification has 1 staff (3NF)

**Position**

The position relation came for the executive entity attribute in the ERD. The position relation has attributes value position to hold a value, and a primary key ID. Staff can only hold one position at one time. This may be adjusted later to match qualifications so that staff can have many positions as they achieve qualifications. Basical an engineer would have their engineering degree and have a bachelor’s in commerce if they had one. So, engineers or general staff can work towards improving their qualifications and position titles.

1 position has many staff (3NF)

### Normalisation

First Normal Form (1NF)

Each row in the relation will have atomic values so only one value per attribute and each row is identifiable and unique.

* Focus on eliminating repeating groups ensure atomic values – data has its own single location.
* Each record unique

Second Normal Form (2NF)

Already in 1NF, and that all non-key attributes are fully dependent and relate to the relations unique identifier.

* Focus on full functional dependency – attributes must relate to their unique identifier if you know the value of the unique identifier you know the values of the depend on attributes.

Third Normal Form (3NF)

The relation is already in 2NF, and all non-key attribute relate to the unique identifier.

* Focus on removing transitive dependency

Boyce Cod Normal Form (BCNF)

The relation is already in 3NF, every determinant is a candidate key.

* Focus on candidate keys to uniquely identify relations e.g. a combination of attributes that are determined by each other.

# Data dictionary

## Relations

|  |  |  |  |
| --- | --- | --- | --- |
| **Relation Name** | **Start Volume** No. of rows loaded at the beginning | **Growth** e.g. no growth / 10% per year | **Comments** |
| Task | 30,000 |  |  |
| Zone | 8 |  |  |
| Subscription | 4 |  |  |
| Contract | NA |  |  |
| Task\_data | 30,000 |  |  |
| Drone\_data | 30,000 |  |  |
| Account | 1% gold, 0.5% platinum , 10 super platinum |  |  |
| Staff | NA |  |  |
| position | 4 |  |  |
| Sales | NA |  |  |
| Qualification | NA |  |  |
| Merchant | NA |  |  |
| Task\_type | 6 |  |  |
| Component\_type | NA |  |  |
| Component\_maintenance | NA |  |  |
| Drone | 7500 |  |  |
| Component | NA |  |  |
| Location | NA |  |  |
| Props | NA |  |  |
| Camera | NA |  |  |
| Frame | NA |  |  |
| Battery | NA |  |  |
| Payment | NA |  |  |

## Attributes

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task** | zoneID | The zone the task was done in | INT |  |  | Must exist in zone | NA | NA | FK | Zone | zoneID references zone(ID) |
|  | taskID | A taskID of another task | INT |  |  | Must exist in task | NA | NA | FK | task | taskID references task(ID) |
|  | Task\_type | The task type for the task | INT |  |  | Must exist in task\_type | NA | NA | FK | Task\_type | Task\_type references task\_type(ID) |
|  | ID | A possible unique identifier | INT | INDENTITY |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Start\_date | The start date and time for a task | DateTime |  |  |  |  | NA |  |  |  |
|  | End\_date | The end date and time for a task | DateTime |  |  |  |  | NA |  |  |  |
|  | Completed | To indicate if a task is complete | BIT | 1 |  |  |  | NA |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Zone** | ID | The possible unique identifier for the relation | INT | INDENTITY |  | Auto increment | No | NA | PK |  | Unique Identifier |
|  | Configuration | The configuration that the zone requires | Varchar | 50 |  |  | No | No |  |  |  |
|  | Description | Gives context to the name if required | Varchar | 100 |  |  | No | No |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task\_type** | ID | The primary identifier | INT |  |  | Unique identifier | NA | No | PK |  |  |
|  | Description | The description of the task\_type | Varchar | 50 |  |  |  | No |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Drone\_data** | droneID | FK from drone relation. | **INT** |  |  | Must exist in drone | NA | NA | FK | Drone |  |
|  | accountID | The account the drone\_data belongs to | **INT** |  |  | Must exist in account | NA | NA | PK | account | accountID references account(ID) |
|  | taskID | The task the drone was doing. | **INT** |  |  | Must exist in task | NA | NA | PK | task | taskID references task(ID) |
|  | Temp | The current recoding of temperature around the drone. | Float | single |  |  |  |  |  |  |  |
|  | Humidity | The current recording of humidity from the drone. | Float | single |  |  |  |  |  |  |  |
|  | Air\_quality | The current air quality recording around the drone. | float | single |  |  |  |  |  |  |  |
|  | Ambient\_Light\_strength | The current recording of the light strength. | float | single |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Drone** | ID | The possible primary identifier. | INT |  |  | Auto increment | NA | NA | PK | Drone | Unique identifier |
|  | Build\_date | The date the drone was built. | DateTime |  |  |  |  |  |  |  |  |
|  | Total\_hours | The total amount of hours the drone has flown. | Float | single |  |  |  |  |  |  |  |
|  | status | A status to represent if the drone can be used or not. | BIT |  |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component\_maintenance** | Component\_maintenanceID | A component\_maintenance record. | INT |  |  | Must exist in component\_maintenance | NA | NA | FK | Component\_maintenance | Component\_maintenance references component\_maintenance(ID) |
|  | componentID | The component being maintained | INT |  |  | Must exist in component | NA | NA | FK | Component | ComponentID references component(ID) |
|  | StaffID | The staff doing the maintenance | INT |  |  | Must exist in staff | NA | NA | FK | Staff | staffID references staff(ID) |
|  | droneID | The drone being worked on | INT |  |  | Must exist in drone | NA | NA | FK | Drone | droneID references drone(ID) |
|  | ID | The primary identifier | INT |  |  | Auto increment | NA | NA | Pk |  | Unique identifier |
|  | Description | A brief description of work | varchar | 255 |  |  | No | No |  |  |  |
|  | Date | The DateTime the component was maintained | DateTime |  |  |  | No | No |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component** | droneID | The drone that has the component | INT |  |  | Must exist in drone | NA | NA | FK | Drone | droneID references drone(ID) |
|  | locationID | The location of the component | **INT** |  |  | Must exist in location | NA | NA | FK | Location | locationID reference location(ID) |
|  | Component\_typeID | The type the component belongs to | **INT** |  |  | Must exist in component\_type | NA | NA | FK | Component\_type | Component\_type references component\_type(ID) |
|  | ID | The unique identifier for the relation. | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Name | The name of the component. | Varchar | 50 |  |  |  |  |  |  |  |
|  | Used\_time | The amount of flight hours a component travelled. | Float | Single |  |  |  |  |  |  |  |
|  | Stock\_amount | The derived total number of a type of component. | INT |  |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Frame** | componentID | The unique identifier for the frame | INT |  |  | Auto increment | NA | NA | FK | Component | componentID references component(ID) |
|  | Material | What composition the frame consists of. | Varchar | 50 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **battery** | componentID | The unique identifier for the battery | INT |  |  | Auto increment | NA | NA | FK | Component | componentID references component(ID) |
|  | ampere | The ampere rating the battery has | Varchar | 50 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **prop** | componentID | The unique identifier for the prop | INT |  |  | Auto increment | NA | NA | FK PK | Component | componentID references component(ID) |
|  | rating | The speed rating the prop are rated to | Varchar | 50 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **camera** | componentID | The unique identifier for the camera | INT |  |  | NA | NA |  |  |  |  |
|  | specification | The specification that the customer has |  |  |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Location** | locationID | The location in a location | INT |  |  | Must exist in location | NA | NA | FK | Location | locationID references location(ID) |
|  | ID | The possible unique identifier. | INT |  |  | Auto increment | NA | NA | PK |  |  |
|  | Note | The brief note about the location. | varchar | 50 |  |  |  | NA |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Supply** | MerchantID | A merchant that supplied the component | INT |  |  | Must exist in merchant | NA | NA | FK | Merchant | merchantID references merchant(ID) |
|  | supplyID | A supply in a supply | INT |  |  | Must exist in supply | NA | NA | FK | Supply | supplyID references supply(ID) |
|  | componentID | A component that has been supplied | INT |  |  | Must exist in component | NA | NA | FK | Component | componentID references component(ID) |
|  | ID | A unique identifier a supply | INT |  |  | Auto increment | NA | NA | PK |  |  |
|  | Part\_name | The merchants part number or name | Varchar | 50 |  |  |  |  |  |  |  |
|  | Price | The price of a supplied component | float | Single |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Merchant** | ID | The unique identifier for the merchant | INT | IDENTITY |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Name | The name of the merchant | Varchar | 50 |  |  |  |  |  |  |  |
|  | Email | The email address of the merchant | Varchar | 100 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Staff** | positionID | The position the staff has | INT |  |  | Must exist in position | NA | NA | FK | Position | positionID references position(ID) |
|  | ID | The unique identifier for staff | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Name | The name of the staff member. | Varchar | 50 |  |  |  |  |  |  |  |
|  | Type | Links the relation to its child relations. | INT |  |  |  |  |  |  |  |  |
|  | Email | The email address of the staff member. | Varchar | 100 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Salesperson** | staffID | FK relations to parent relation | Int |  |  | Must exist in staff | NA | NA | FK, PK | Staff | staffID references staff(ID) |
|  | Customer\_count | The amount of customer that the sales.staff has |  |  |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Qualification** | staffID | The staff that has the qualification | INT |  |  | Must exist in staff | NA | NA | FK | Staff | staffID references staff(ID) |
|  | ID | The unique identifier | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Description | A description of a qualification | Varchar | 50 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **position** | ID | The unique identifier for position | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Position | The name of the position | Varchar | 50 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **account** | staffID | A staff who manages the account | INT |  |  | Must exist in staff | NA | NA | FK | Staff | staffID references staff(ID) |
|  | ID | The unique identifier for the account | INT |  |  | Auto increment |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **payment** | accountID | An account that has made a payment | INT |  |  | Must exist in account | NA | NA | FK | Account | accountID references account(ID) |
|  | ID | The unique identifier for a payment | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Contact\_person | The persons to contact who has a subscription | Varchar | 50 |  |  |  |  |  |  |  |
|  | Email | The email of the account holder | Varchar | 50 |  |  |  |  |  |  |  |
|  | Address | An address location of the account | Varchar | 100 |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **contract** | staffID | A staff who overseen the contract | INT |  |  | Must exist in staff | NA | NA | FK | Staff | staffID references staff(ID) |
|  | accountID | An account that has a contract | INT |  |  | Must exist in account | NA | NA | FK | Account | accountID references account(ID) |
|  | Subscription\_type | The level of subscription the contract has | INT |  |  | Must exist in subscription\_type | NA | NA | FK | Subscription | subscriptionID references subscription(ID) |
|  | ID | The unique identifier for a contract | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | End\_date | The date a contract ends | DateTime |  |  |  |  |  |  |  |  |
|  | Discount | A discount applied to the contract | Float | Single |  |  |  |  |  |  |  |
|  | Total | The total amount for the contract | Float | Single |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **subscription** | Type | The type of subscription or level | INT |  |  | Unique | NA | NA | PK |  | Unique identifier |
|  | Max\_drones | The max drone count for a subscription | INT |  |  |  |  |  |  |  |  |
|  | Max\_area | The max area that is allocated to a subscription | INT |  |  |  |  |  |  |  |  |
|  | Max\_zone | The max zones that the subscription can operate in | Int |  |  |  |  |  |  |  |  |

| **Relation Name** | **Attribute** | **Description** | **Data type** | **Length** | **Value range** | **Validation Rules** | **Default Value** | **Nulls** | **Key?** | **References Entity** | **Integrity Constraints** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component\_type** | ID | The unique identifier for the component\_type | INT |  |  | Auto increment | NA | NA | PK |  | Unique identifier |
|  | Name | The name of the component\_type | Varchar | 50 |  |  |  |  |  |  |  |

## Derived attributes

| **Relation Name** | **Attributes** | **Derived from / calculation** |
| --- | --- | --- |
| **component** | Stock\_amount |  |
| **Customer\_count** | Salesperson |  |
| **total** | contract |  |
| **discount** | contract |  |

# NaLER analysis report

A natural language method for interpreting Entity Relationships

Natural language for interpreting entity relationships analysis is a method used in data modelling to break down logical data models and assess their design. The method translated ERD elements into statements. Assessing logical consistencies to ensure data integrity, and that the model meets the business rules or system requirements.

## zone

Each zone is uniquely identified by one configuration

One zone(configuration) must have a description

One zone(configuration) may have one or more task(ID)

## task

Each task is uniquely identified by one ID

One task(ID) must have one start\_date

One task(ID) must have one end\_date

One task(ID) must have one status

One task(ID) must have an area

One task identified by its ID can have one or more task identified by taskID and can have only one zone identified by zone\_configuration and only one task\_type identified by task\_type\_description

## Task\_type

Each task\_type is identified by description

## Drone\_data

Each drone\_date is uniquely identified by one accountID and taskID

one drone\_data(accountID, taskID) must have one temp

one drone\_data(accountID, taskID) must have one humidity

one drone\_data(accountID, taskID) must have one air\_quality

one drone\_data(accountID, taskID) must have one light\_strength

Each drone\_data identified by accountID and taskID must have one account identified by account(ID), and one drone identified by one drone(ID)

## Account

Each account is uniquely identified by one account(ID)

One account(ID) must have one contract\_person

One account(ID) must have one email

One account(ID) must have one address

Each account identified by an ID must have one staff identified by staff(ID)

Staff ---NEEDS MORE WORK

Each staff is uniquely identified by one ID

One staff(ID) must have one name

One staff(ID) must have one email

One staff can belong to many subclasses

Each staff subclass must have one staff identified by staff(ID)

One staff(ID) could have customer\_count

One staff(ID) must have one position

One staff(ID) could have one or more qualifications identified by engineer\_staffID

## Contract

Each contract is uniquely identified by one ID

One contract(ID) must have an end\_date

One contract(ID) must have a discount

One contract(ID) must have a total

Each contract identified by an ID must have one staff identified by staffID and one account identified by accountID and one subscription identified by subscription\_type

## Subscription

Each subscription is uniquely identified by type

One subscription must have one max\_drones

One subscription must have one max\_area

One subscription must have one max\_zone

One subscription must have one cost

## Payment

Each payment is uniquely identified by a one date

One payment(date) must have one amount

One payment must have one account identified by accountID

Component\_maintenance

Each component\_maintenance is identified by one ID

One component\_maintenance(ID) must have one description

One component\_maintenance(ID) must have one date

One component\_maintenance must have one or more component\_maintenance identified by component\_maintenanceID, must have one or more components identified by componentID, must have one or more staff identified by staffID, can happen to one or more drone identified by droneID.

## Merchant

Each merchant is uniquely identified by one ID

One merchant(ID) must have one name

One merchant(ID) must have one email

## Supply

Each supply is uniquely identified by one ID

One supply must have one part\_name

One supply must have one price

One supply can have one or more supply(ID) identified by supplyID, only one merchant identified by merchantID, only one component identified by componentID

## Component

Each component is uniquely identified by ID

One component(ID) must have one name

One component(ID) must have one description

One component(ID) must have one used\_time

One component(ID) must have one stock\_amount

one component identified by ID can have only one component\_type identified by component\_type\_name, one drone identified by droneID, one location identifier by locationID

## Frame

Each frame is uniquely identified by componentID

One frame must have one material

One frame identified by componentID must only belong to one component identified by ID

## Battery

Each battery is uniquely identified by componentID

One battery identified by componentID must belong to one component identified by ID

## Prop

Each Prop is uniquely identified by one componentID

One prop identified by componentID must belong to one component identified by ID

## camera

each camera is uniquely identified by componentID

one camera identified by componentID must belong to one component identified by ID

## drone

Each drone is uniquely identified by ID

One drone(ID) must have one build\_date

One drone(ID) must have one total\_hours

One drone(ID) must have a status

## Staff

Each Staff is uniquely identified by one ID

One staff(ID) must have one name

One staff(ID) must have one email

One staff identified by ID can have only one position identified by position

## Qualification

Each qualification is uniquely identified by ID

One qualification must have a description

One qualification identified by qualification can have one or more staff identified by staffID

## Position

Each position is uniquely identified by position

## Sales

Each salesperson is uniquely identified by staffID

One salesperson can have one customer\_count

One salesperson identified by staffID belongs to one staff identified by ID

## Location

Each location is uniquely identified by ID

One location(ID) has one note

One location identified by ID can have one or more location identified by locationID

## component\_type

Each component\_type is uniquely identified by one name

## Component\_maintenance

Each component\_maintenance is identified by one ID

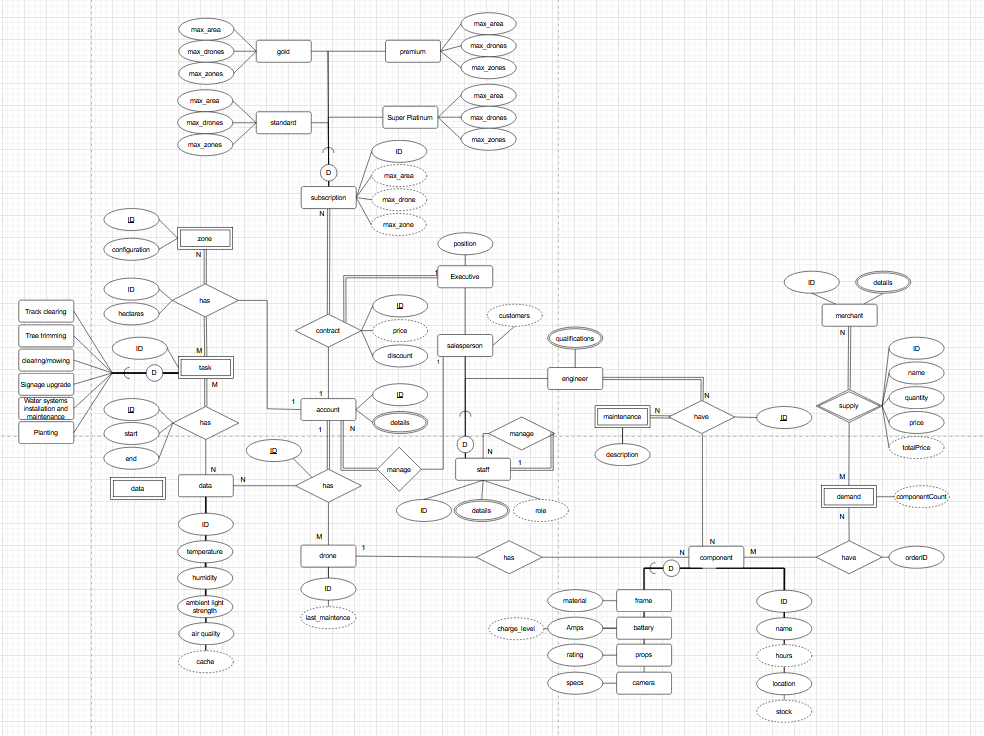
One component\_maintenance(ID) must have one description

One component\_maintenance(ID) must have one date

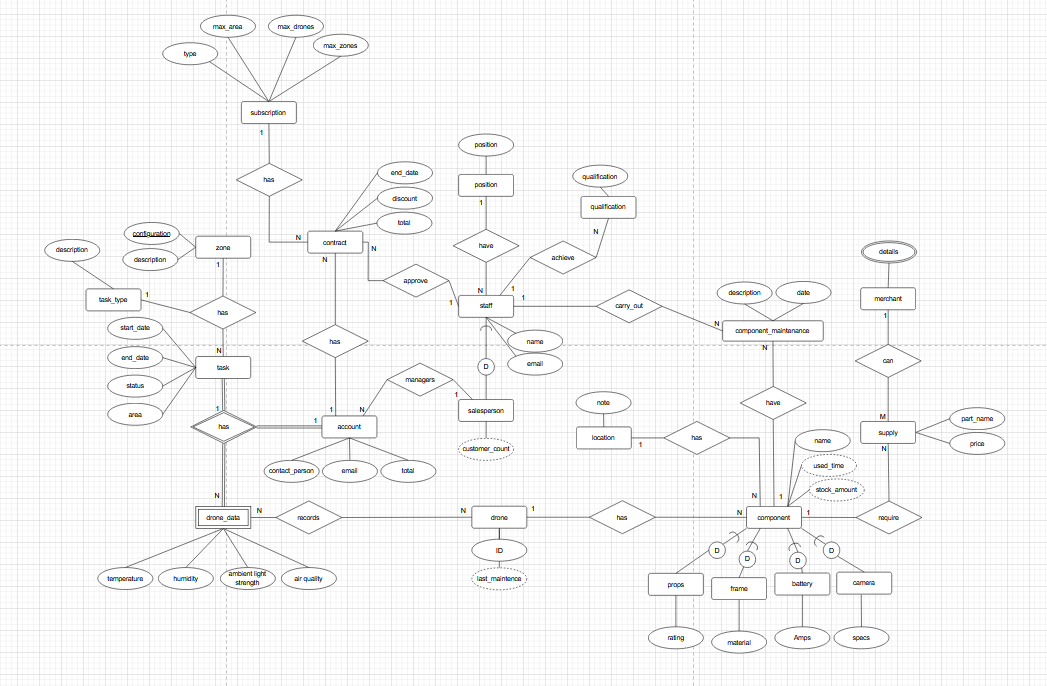
# Final report

## Map ER model to relations

### First version 1.0

****

### Revised version 1.2

****

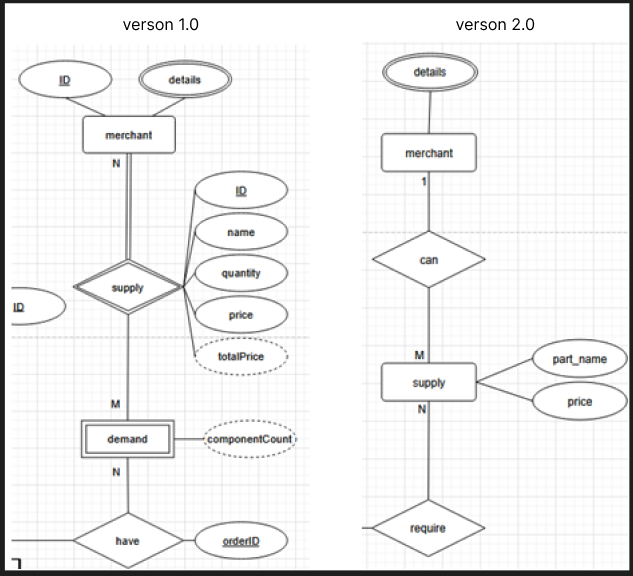
### Changes made to ERD

**ERD Changes made**

**Merchant and supply**

I had to change the structure in my extended Chen’s notation ERD in a few ways

First, my supply relation was incorrect I had it as a weak entity. This approach was wrong and I remove the weak relationship and turn it into an entity, it wouldn’t work because it would require an identifing relation with demand and merchant (BCNF) this would restrict the supply relation to much and force each supply to require a demand everytime. The demand relation needs to relate to the component relation and have nothing to do with supply.. it just gets in the way. But for my final ERD I have remove it all together because it isnt a requirement for this design.



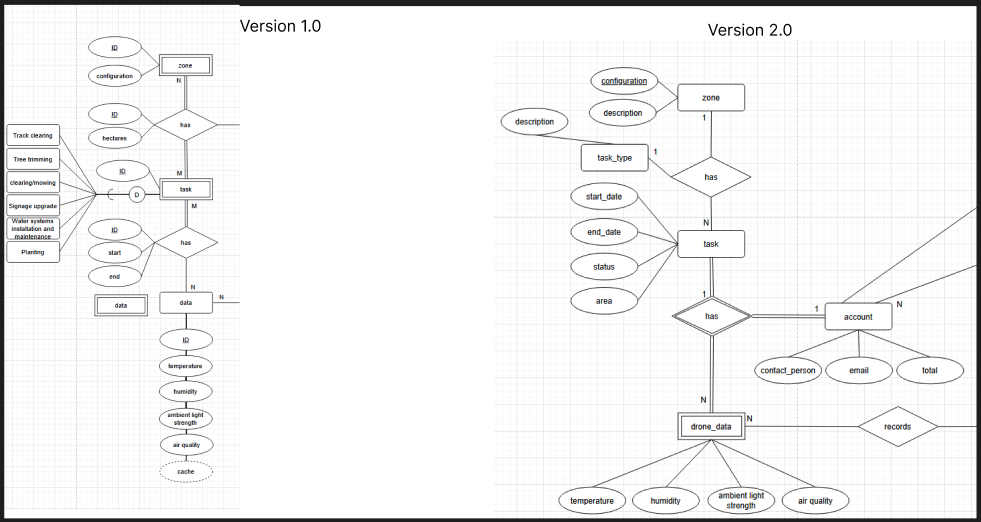
**component\_maintenance**

The second area I reworked was the component\_maintenance entity, in the orignal EERD I had the entity as weak this wouldn’t work as it would require an identifing relationship. The only way this could work is if a created a composite key with an identifier from drone and the identifier from component and used an attribute from the component maintenance like date (BCNF) combining these three would create a unique composite key for each record. But I just reworked it to be strong entity not requiring a composite key (3NF).

A diagram of a diagram

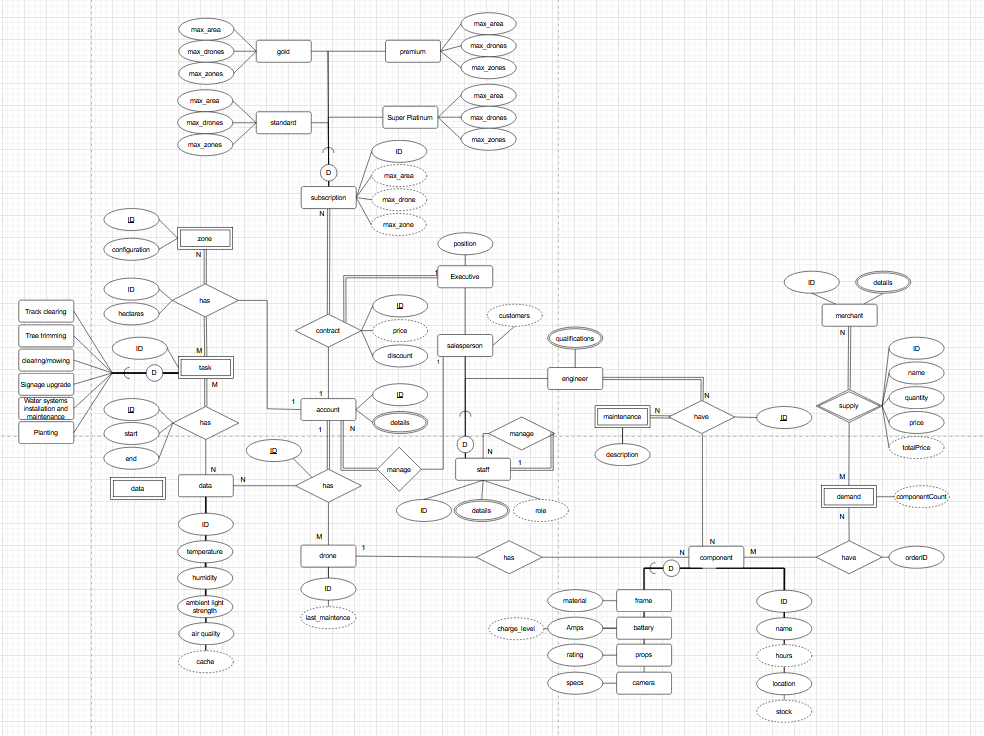
AI-generated content may be incorrect.

The third major area the needed reworking was the task, account, and drone\_data entities, drone\_data became a weak entity requiring an identifier from account and task to identify 1 record. Simplified task subclasses to be more reflective on how it relates. (BCNF)

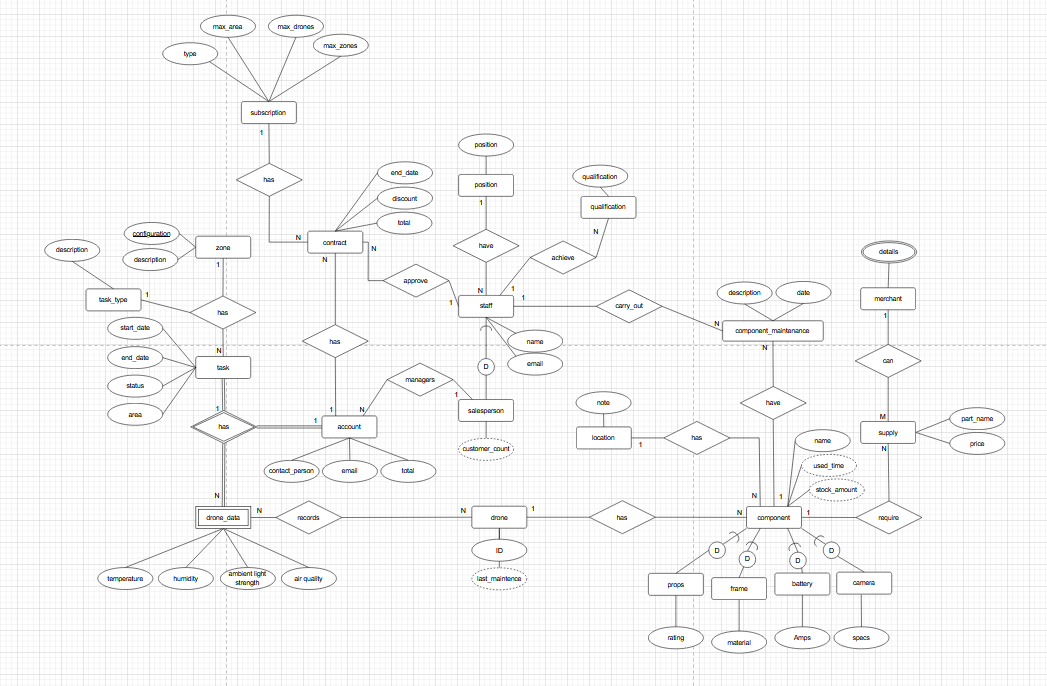


**Map ER model to relations**

First version 1.0

****

Second version most current 2.0

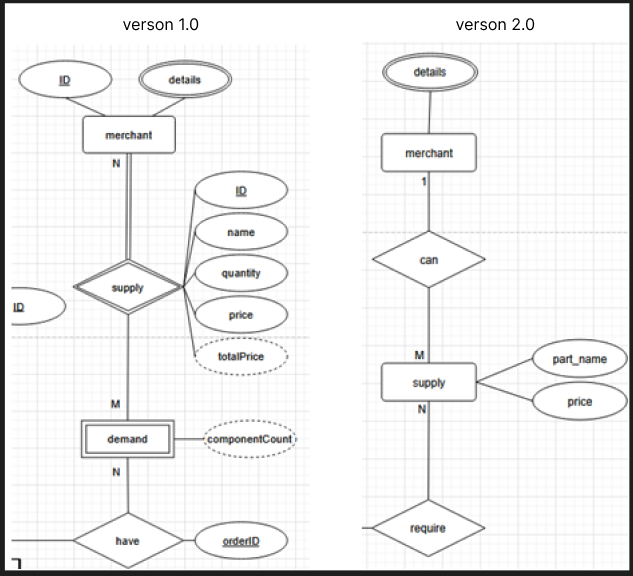
****

**ERD Changes made**

**Merchant and supply**

I had to change the structure in my extended Chen’s notation ERD in a few ways

First, my supply relation was incorrect I had it as a weak entity. This approach was wrong and I remove the weak relationship and turn it into an entity, it wouldn’t work because it would require an identifing relation with demand and merchant (BCNF) this would restrict the supply relation to much and force each supply to require a demand everytime. The demand relation needs to relate to the component relation and have nothing to do with supply.. it just gets in the way. But for my final ERD I have remove it all together because it isnt a requirement for this design.



**component\_maintenance**

The second area I reworked was the component\_maintenance entity, in the orignal EERD I had the entity as weak this wouldn’t work as it would require an identifing relationship. The only way this could work is if a created a composite key with an identifier from drone and the identifier from component and used an attribute from the component maintenance like date (BCNF) combining these three would create a unique composite key for each record. But I just reworked it to be strong entity not requiring a composite key (3NF).

A diagram of a diagram

AI-generated content may be incorrect.

The third major area the needed reworking was the task, account, and drone\_data entities, drone\_data became a weak entity requiring an identifier from account and task to identify 1 record. Simplified task subclasses to be more reflective on how it relates. (BCNF)

A screenshot of a computer

AI-generated content may be incorrect.

## Data modeling in information systems ERD and Logical model

Data modeling in information systems is a curical process where system briefs or other forms of project descriptions get turned into visual data structures. Using elements like entities and relationships, the model can be used to elicit information from other stakeholder that the system has to make sure the the model will work and that business rules are adheard to. The two models Extend Chen’s notation and the logical model serve different purposes but are both vital in the design process. Extended Chen’s notation is perfect for getting all the vital attributes and entities and the relationships that the have with one another in a visual diagrammtic format. This gives the data modeller a great observers view of the design to make sure that all elements are their and accounted for. Essentially, it acts as the blueprints for the database and schema. To add context heres an analogy, its like building a house without the plans you couldn’t. No one would know what to do and the bank wouldn’t even give you the time of day to get the loan. That’s the ERD, it provide clarity and keeps the system stakeholders in the decision curve and the system meets its requirements.

After the data has been modelled at a conceptual level and the design approved or meets the requirements, a logical model is created from the conceptual giving more life and meaning. Elements from the conceptual level get converted to their logical counter parts, like entities become relations and attributes broken down into single atomic values. Functional dependencies are created using primary unique identitiers and referential integrity contraints are applied and tested using NaLER and checked against business rules to make sure the database system schema is robust enough and has the correct constaints. If any relations have transitive dependencies that haven’t been normalised, normalise them. Once the logical model is full normalised to the required level.

Data principlesAccountability

An organisation has policies and procedures for staff management and auditability.

Transparency

An organisation documents its policies, processes and activites including its data governance program and make them available to all people.

Integrity

The data governance policy should be set up to make sure that the data is authentic and reliable.

Protection

The data policy provide a reasonable level of protection.

Compliance

The data policy needs to comply with applicable laws and other relevant policies.

Availability

The system is efficient, information or data assets is store and index as required and robust backup and recover policies are created for efficient disaster recovery.

Retention

A data retention policy is needed to outline the retention period appropriate time and explain what happens after the retention period.

Disposition

A data policy to outline how data will be distroyed in accordance with law and the retention policy.