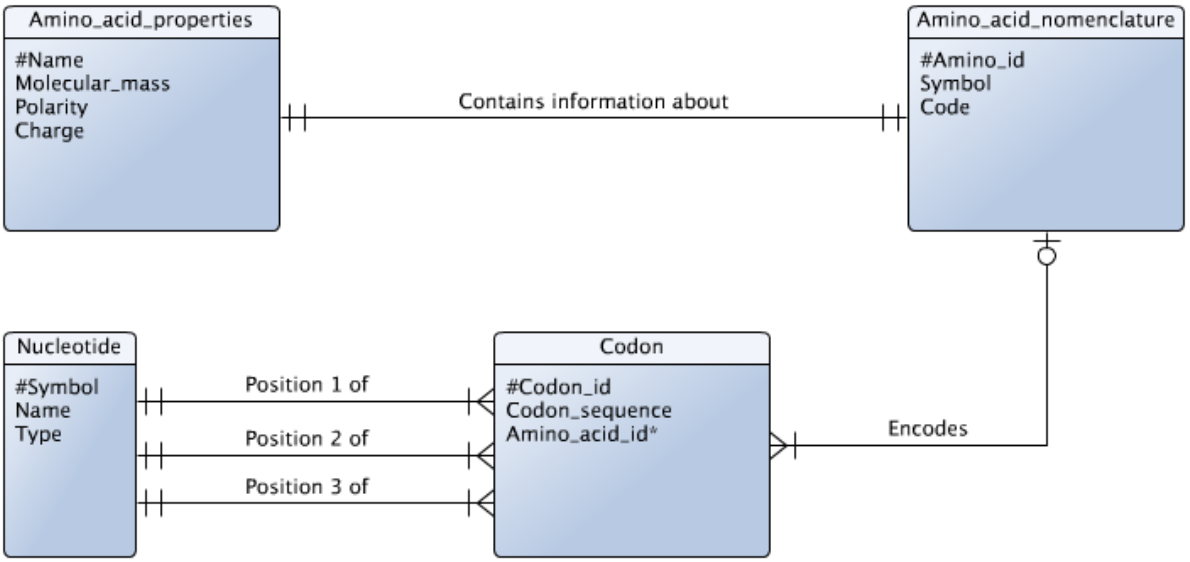
1

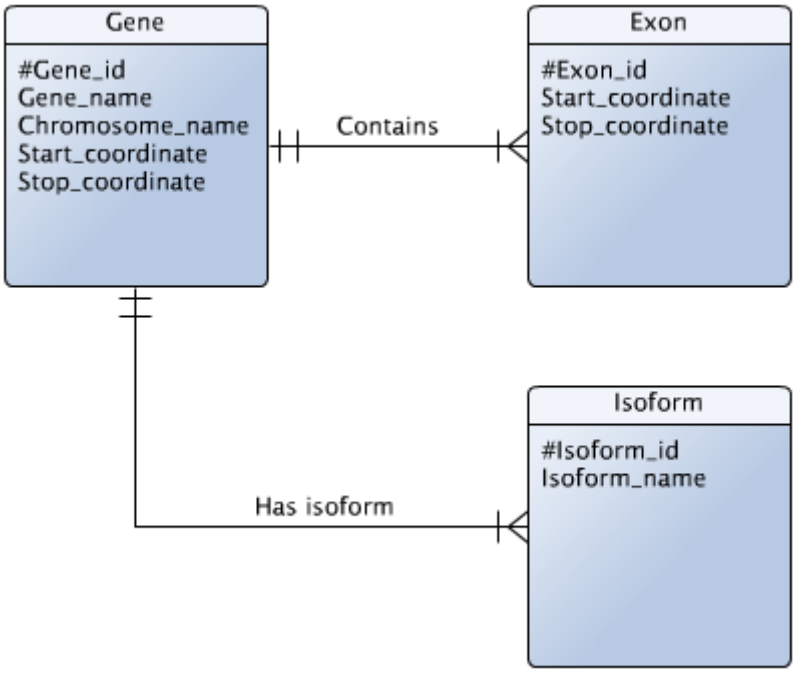
Where text is used on the lines it explains the relationship from left to right or from top to bottom.



2i

The entities are Gene, Exon and Isoform. Chromosome does not have to be an entity here as it has no attributes. Where text is used on the lines it explains the relationship from left to right or from top to bottom.

2ii



2iii

```
Gene (#Gene_id, Gene_name, Chromosome_name, Start_coordinate, Stop_coordinate)

Exon (#Exon_id, Gene_id*, Start_coordinate, Stop_coordinate)

Isoform (#Isoform_id, Isoform_name, Gene_id*)
```

All primary keys are minimal so the database is on 2NF. It is not on 3NF however; there are transitive functional dependencies in the Gene table. To remove them I split the table up:

```
Gene (#Gene_id, Name*)

Gene_properties (#Name, Chromosome_name, Start_coordinate, Stop_coordinate)
```

Assuming a one-to-one correspondence between isoform_id and isoform_name I must also split up the Isoform table:

```
Isoform (#Isoform_id, Isoform_name*)

Isoform_properties (#Isoform_name, Gene_id*)
```

The resulting database is now on 3NF:

```
Gene (#Gene_id, Name*)

Gene_properties (#Name, Chromosome_name, Start_coordinate, Stop_coordinate)

Exon (#Exon_id, Gene_id*, Start_coordinate, Stop_coordinate)

Isoform (#Isoform_id, Isoform_name*)

Isoform_properties (#Isoform_name, Gene_id*)
```

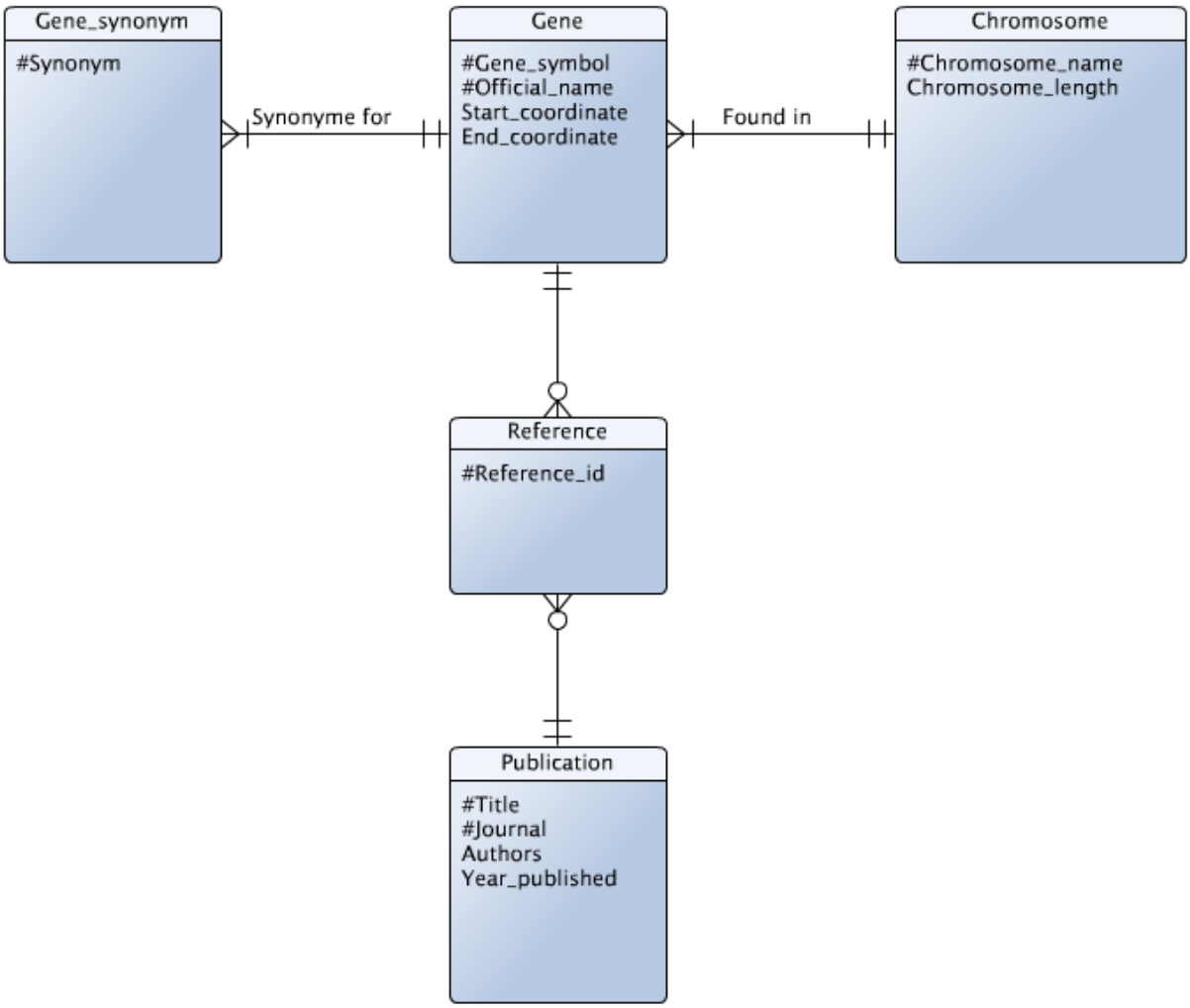
Because each gene_name and isoform_name would appear just once in the first 2NF draft I think the normalization to 3NF is unnecessary.

3i

I propose the entities Gene, Gene_synonym, Chromosome, Reference and Publication.

3ii

I assume {Title, Journal} is a minimal primary key for the Publication table. Where the entity “is a relationship” I do not always write additional text on the connected lines. Where text is used on the lines it explains the relationship from left to right or from top to bottom.



3iii

`Gene_synonym(#Synonym, Gene_symbol*)`

`Gene (#Gene_symbol, #Official_name, Chromosome_name*, Start_coordinate, End_coordinate)`

`Chromosome (#Chromosome_name, Chromosome_length)`

`Reference (#Reference_id, Gene_symbol*, Title*, Journal*)`

`Publication (#Title, #Journal, Year_published, Authors)`

This database contains only atomic values (no lists, tables etc.) so it is on 1NF. The Gene table is not on 2NF because Start_coordinate is determined by any one of Gene_symbol and Official_name.

3iv

I mark the changes in red.

1NF -> 2NF:

Gene (#Gene_symbol, Official_name, Chromosome_name*, Start_coordinate, End_coordinate)

Now all the tables are on 2NF because all the tables with a primary key of just 1 attribute must be on 2NF and the Publication table is also on 2NF because the primary key is minimal.

2NF -> 3NF:

Because Start_coordinate is determined by Official_name we have a transitive functional dependency.

Gene (#Gene_symbol, Official_name*)

Gene_properties (#Official_name, Chromosome_name*, Start_coordinate, End_coordinate)

After the changes all the transitive functional dependencies are gone and we have 3NF.

3NF -> BCNF:

Any table on 3NF having just two columns must be on BCNF because any determinant must then determine both columns. Also any determinant in Reference and Publication determine all columns.

This leaves just Gene_properties left to be checked. First the determinant {Chromosome_name, Start_coordinate} looked like a problem because it determines the Official_name column. But then I realized it must also determine the End_coordinate column. Almost the same argument shows that {Chromosome_name, Start_coordinate} is also a superkey. All the other determinants in the Gene_properties are also superkeys. The table I got when I normalized to 3NF was therefore also on BCNF.

Gene_synonym (#Synonym, Gene_symbol*)

Gene (#Gene_symbol, Official_name*)

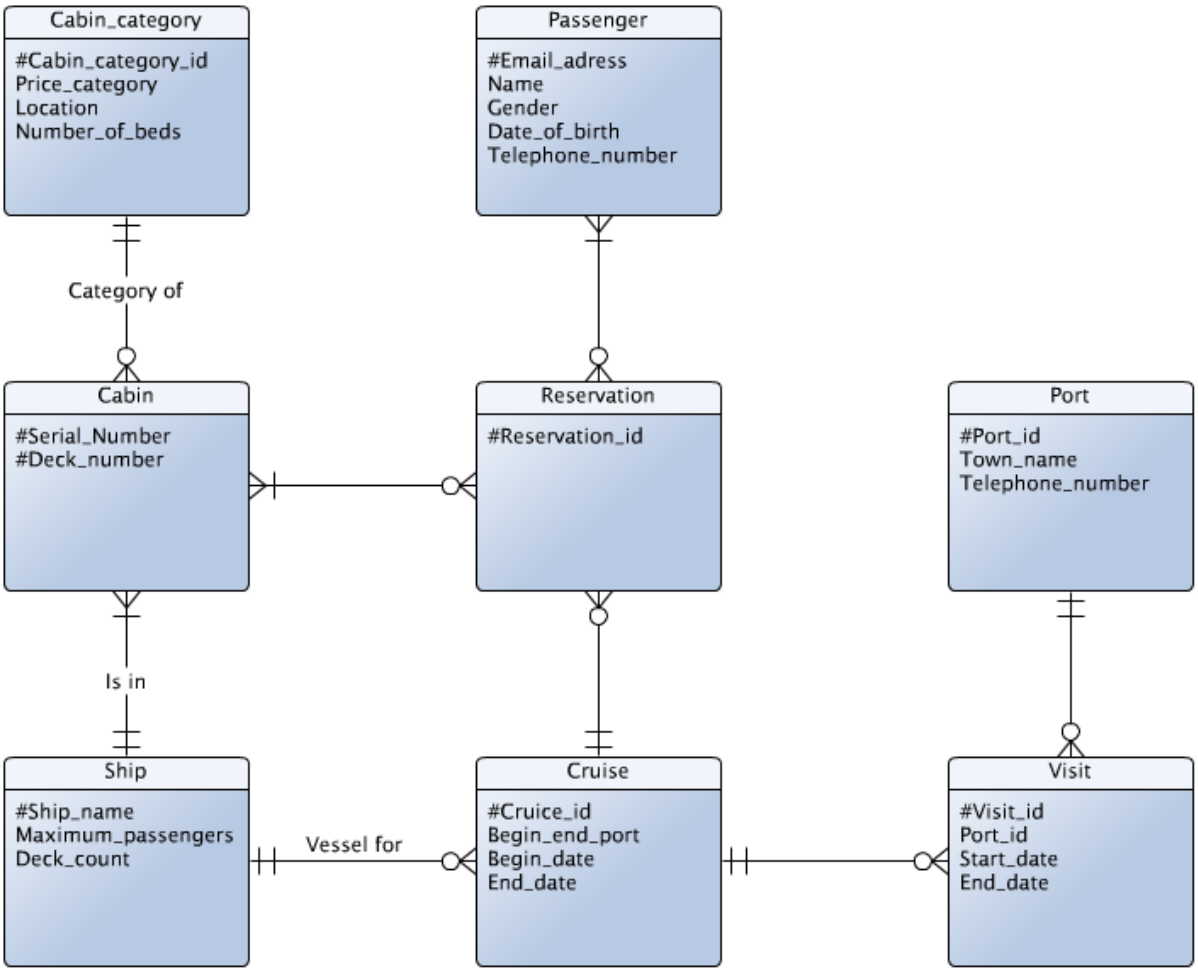
Gene_properties (#Official_name, Chromosome_name*, Start_coordinate, End_coordinate)

Chromosome (#Chromosome_name, Chromosome_length)

Reference (#Reference_id, Gene_symbol*, Title*, Journal*)

Publication (#Title, #Journal, Year_published, Authors)

4



A cabin can be reserved by several reservations, as long as the reservations are for different cruises. Where the entity “is a relationship” I do not always write additional text on the connected lines. Where text is used on the lines it explains the relationship from left to right or from top to bottom.

5i

First of all, it is unintuitive to store the assignment number in the truck table, we would expect only information about the actual truck in a table called Truck.

Also with this setup we have to use an entire row in the Truck table each time a truck Is assigned a new job.

We could copy all the information about a truck for each row, but this is wasteful with regards to storage space. This would also make it harder to update the information (harder to maintain integrity) about a truck as we would have to update all the rows.

On the other hand, if we left some of the information about a truck blank in all but one row, we would still probably waste space. In this case it would also be more work extracting the information about a truck given an Assignment_number, as we would have to find out which row contains this information.

In 5iii I find another problem with this implementation.

5ii

The only column that decides any other is the Registration_number column. The only column that decides Assignment_number is itself. The functional dependencies are:

$$\text{Reg\_number} \cup X \rightarrow Y \quad \text{for all } X, Y \subseteq \{\text{Reg\_number, Reg\_year, Model, Max\_weight}\}$$

and

$$\text{Reg\_number} \cup \text{As\_number} \cup X \rightarrow Y$$

$$\text{for all } X, Y \subseteq \{\text{Reg\_number, Reg\_year, Model, Max\_weight, As\_number}\}$$

5iii

If we do not allow null values in Assignment_ number, then one candidate key is {Reg_number, As_number} because it is a minimal super key. There are no other minimal super keys.

This leads to another problem, a primary key cannot be allowed to contain null values. This means that if we want to add a truck that has no assignment yet, we cannot put null in the Assignment_number field. So we have to decide for some dummy value to put in it.

5iv

The table is already on 1NF:

```
Container_type (#Type_id, Type_name, Max_weight, Cubic_quantity, Nightly_rate)

Container (#Container_number, Type_id*)

Customer (#Telephone_number, Address)

Assignment (#Assignment_number, Telephone_number*, Container_number*, Start_date, End_date)

Truck (#Registration_number, Registration_year, Model, Maximum_weight, #Assignment_number*)
```

1NF -> 2NF, decompose Truck into two tables:

```
Truck (#Registration_number, Registration_year, Model, Maximum_weight)

Truck_assignment (#Assignment_number*, #Registration_number*)
```

I am assuming that any truck of a particular Model has the same Maximum_weight. I also assume that the Type_name of Container_type decides the Max_weight, Cubic_quantity and Nightly_rate.

2NF -> 3NF further decompose Truck into two new tables, also decompose Container_type:

```
Container_type (#Type_id, Type_name*)

Container_properties (#Type_name, Max_weight, Cubic_quantity, Nightly_rate)

Truck (#Registration_number, Registration_year, Model*)

Truck_properties (#Model, Maximum_weight)
```

3NF -> BCNF:

I cannot detect any determinants that are not super keys, so my normalization to 3NF also made it to BCNF. The final results are:

```
Container_type (#Type_id, Type_name*)

Container_properties (#Type_name, Max_weight, Cubic_quantity, Nightly_rate)

Container (#Container_number, Type_id*)

Customer (#Telephone_number, Address)

Assignment (#Assignment_number, Telephone_number*, Container_number*, Start_date, End_date)

Truck (#Registration_number, Registration_year, Model*)

Truck_properties (#Model, Maximum_weight)

Truck_assignment (#Assignment_number*, #Registration_number*)
```