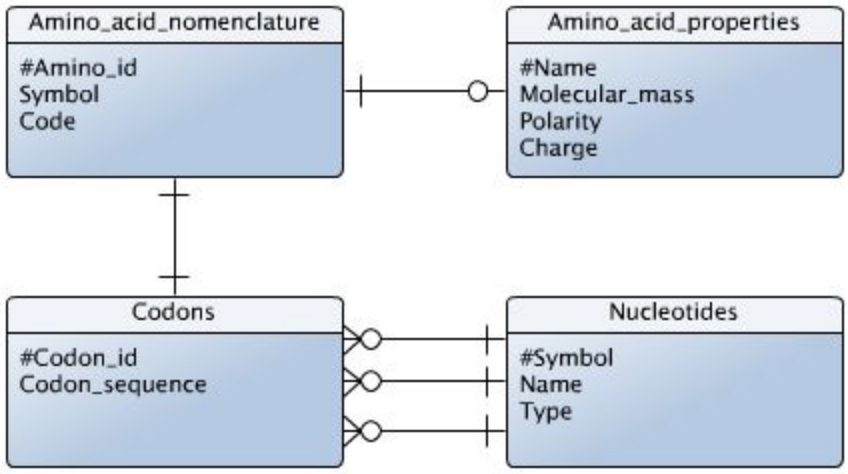
Compulsory Exercise #2

INF115, Spring 2016

Task 1

Using the four tables from compulsory exercise 1, reverse engineer an ER diagram to represent the entities and their attributes. Highlight public keys, and select appropriate relationships between entities.

I assume you meant “highlight **primary** keys”, not **public** keys.



Task 2

2.i: Identify the entities in the database description

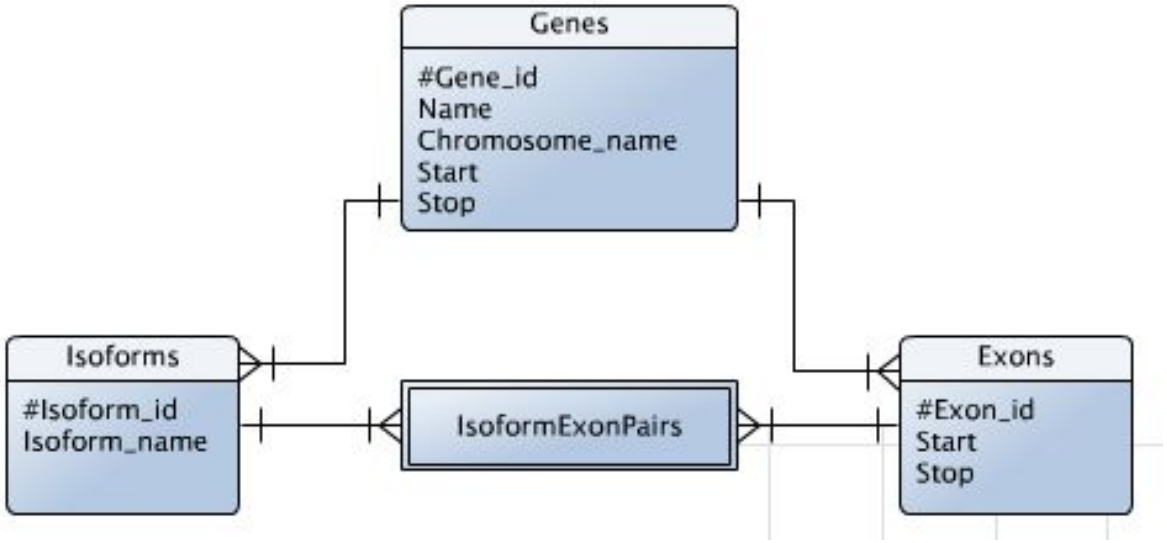
I identified the following entities

- Genes
- Exons
- Isoforms

Another entity which might be sensible to include is ***Chromosomes***. Having a table Chromosomes(#Chromosome_name, Chromosome_id) will save us a lot of storage space, as we can store the shorthand Chromosome_id in thousands of genes. We will additionally be able to verify that a chromosome name is correct if we store the valid ones in a table.

I did not include a chromosome table as the only mentioned related attribute was its name.

2.ii: Produce an ER diagram showing the relations between these entities, highlight primary keys, and select appropriate relationships between the entities.



2.iii: Convert the ER diagram in 2.ii to a set of tables conforming to the third normal form, include appropriate primary and foreign keys.

The basic conversion from the ER diagram would give us..

- Genes (#Gene_id, Name, Start, Stop, Chromosome_name)
- Exons (#Exon_id, Start, Stop, *Gene_id)
- Isoform (#Isoform_id, Isoform_name, *Gene_id)
- IsoformExonPairs(*#Isoform_id, *#Gene_id)

Our basic conversion already upholds the first normal form (1NF), which states that there should be no nested tables.

A table is in second normal form (2NF) if it is in 1NF and every attribute is fully functionally dependent on the entire primary key (PK). Tables with a non-composite PK are automatically 2NF if they are 1NF, so we only have to show that IsoformExonPairs is in 2NF. There are no non-PK attributes in IsoformExonPairs table, which means that every attribute automatically depends on its PK (since its PK is the only attribute).

Showing that a table is in 3NF requires it to be in 2NF and that there are no transitive functional dependencies. I will show this is the case for each table:

Isoform (#Isoform_id, Isoform_name, *Gene_id)
There exists a transitive dependency if and only if **Gene_id* can be uniquely determined from the *Isoform_name*. This is only the case if *Isoform_name* is unique. The assignment didn't state if this attribute is unique, so we can assume that it isn't. Thus Gene_id is not fully functionally dependent on Isoform_name, and the isoform table is in 3NF.

IsoformExonPairs (#*Isoform_id, #*Exon_id)
The IsoformExonPairs table is automatically in 3NF since it doesn't have any non-PK attributes.

Exons (#Exon_id, Start, Stop, *Gene_id).
The Exons table is in 3NF if none of its attributes can fully determine any of the others.

We are not sure if *Start* and *Stop* count the location on the Gene or on the Chromosome, but it doesn't really matter which of the cases it is. There are many different chromosomes and genes, and there will be an exon that starts in position 0 of a gene, or at the start of a chromosså du burde begynne ome. We can thus assume that *Start* isn't unique. Similarly, *Stop* might end up not being unique either if two different exons stop at the same location of their respective chromosomes (or genes if we're counting location on genes).

We've just argued that *Start* and *Stop* aren't unique, so we cannot determine any other attributes solely from them. There are many Exons belonging to a single Gene through **Gene_id*, so we cannot determine any of the other attributes from that foreign key.

We've thus shown that none of the non-PK attributes determine the values of the others, removing the possibility of transitive dependencies. The Exons table is thus in 3NF.

Genes (#Gene_Id, Name, Start, Stop, Chromosome_name)
We start by reasoning that *Start* and *Stop* do not uniquely identify any Gene due to the same reasons that Exons.Start and Exons.Stop do not. None of the other attributes are therefore uniquely determined by *Start* or *Stop*. There are many genes on a chromosome, so Chromosome_name can't determine the other attributes either.

If the *Name* attribute is unique, then all the other attributes could be determined by it, which would mean that we have transitive dependencies. We would then have to divide the Genes table into 2 tables, such as...

- Genes (#Gene_Id, *Gene_name)
- Gene_nomenclature (#Gene_name, Start, Stop, Chromosome_name).

.. completing the conversion of our ER-diagram into 3NF.

Task 3

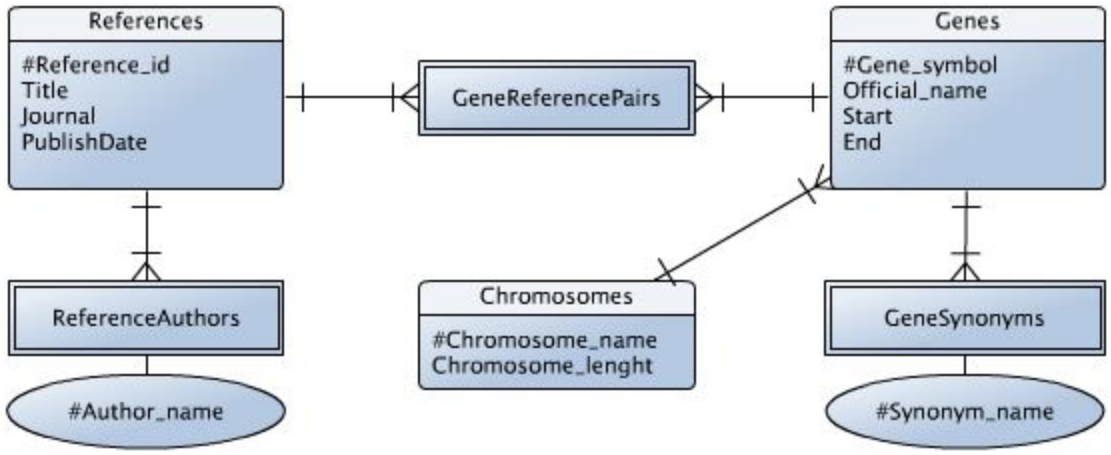
3.i: Identify the entities in the database description above

We cannot have tables in tables, so synonyms and references need to be broken up into tables. The entities I identified are..

- Genes
- Chromosomes
- Synonyms
- References
- Authors

We could argue if Authors should be an entity or not. The name of an author might not be an unique identifier of an Author. We would thus require a composite key, or adding an unique author id to fully qualify an author. Likewise, synonym name isn't an unique identifier, so we could either add a synonym_id, or make the entity a weak one. I choose to make both entities weak ones.

3.ii: Create an ER diagram to represent relations between entities identified in 3.i, specify the primary keys and select appropriate relationships between entities.



It isn't possible to make a weak entity with attributes in graphity, and neither is making the triple arrow that you've used in lectures. The circle means "attribute of", meaning that the weak entity GeneSynonyms and ReferenceAuthors are as following:

- **GeneSynonyms (#*Gene_symbol, #Synonym_name)**
- **ReferenceAuthors (#*Reference_id, #Author_name)**

3.iii: Convert the ER diagram in 3.ii to a set of tables conforming to the first normal form, but not the second normal form. Highlight primary and foreign keys.

Making the diagram into tables directly gives us...

- Chromosomes (#Chromosome_name, Chromosome_length)
- References (#Reference_id, Title, Journal, PublishDate)
- Genes (#Gene_symbol, Official_name, Start, End, *Chromosome_name)
- GeneReferencePairs (#*Reference_id, #*Gene_symbol)
- GeneSynonyms (#*Gene_symbol, #Synonym_name)
- ReferenceAuthors (#*Reference_id, #Author_name)

The second normal form (2NF) is breached if an attribute isn't fully dependent on a composite PK. We have 3 tables with composite PK's, but none of those have extra attributes, so all tables are in 2NF.

The easiest way to breach 2NF is to merge two tables, such as ***Genes*** and ***References***. We then get the new table

- GeneReferences (#Reference_id, Title, Journal, PublishDate,
  #Gene_symbol, Official_name, Start, End, *Chromosome_name)

Where all of the attributes are only dependent on the subset of the primary key from which it originally was dependent on before the table merge (e.g. Official_name is only dependent on #*Gene_symbol*, and Title is only dependent on #Reference_id).

3.iv: Further develop the table structure from 3.iii so that it conforms to Boyce Codd normal form (BCNF), again highlight primary and foreign keys.

Our basic conversion from 3.iii was already in 2NF, so let us check if it is in 3NF and BCNF.

We need to have multiple candidate keys and transitive dependencies to breach 3NF or BNCF.

GeneReferencePairs, GeneSynonyms, and ReferenceAuthors are already in BNCF since they do not have any non-PK attributes.

Chromosome is in BCNF since it only has 1 non-PK attribute, *Chromosome_length*, which may not be unique (i.e. 2 different chromosomes might have the same length).

References is in BCNF since *Title*, *Journal*, and *PublishDate* aren't sufficient to determine each other or the PK. They might be a candidate key together, which is fine since they determine all the other attributes (i.e. the primary key).

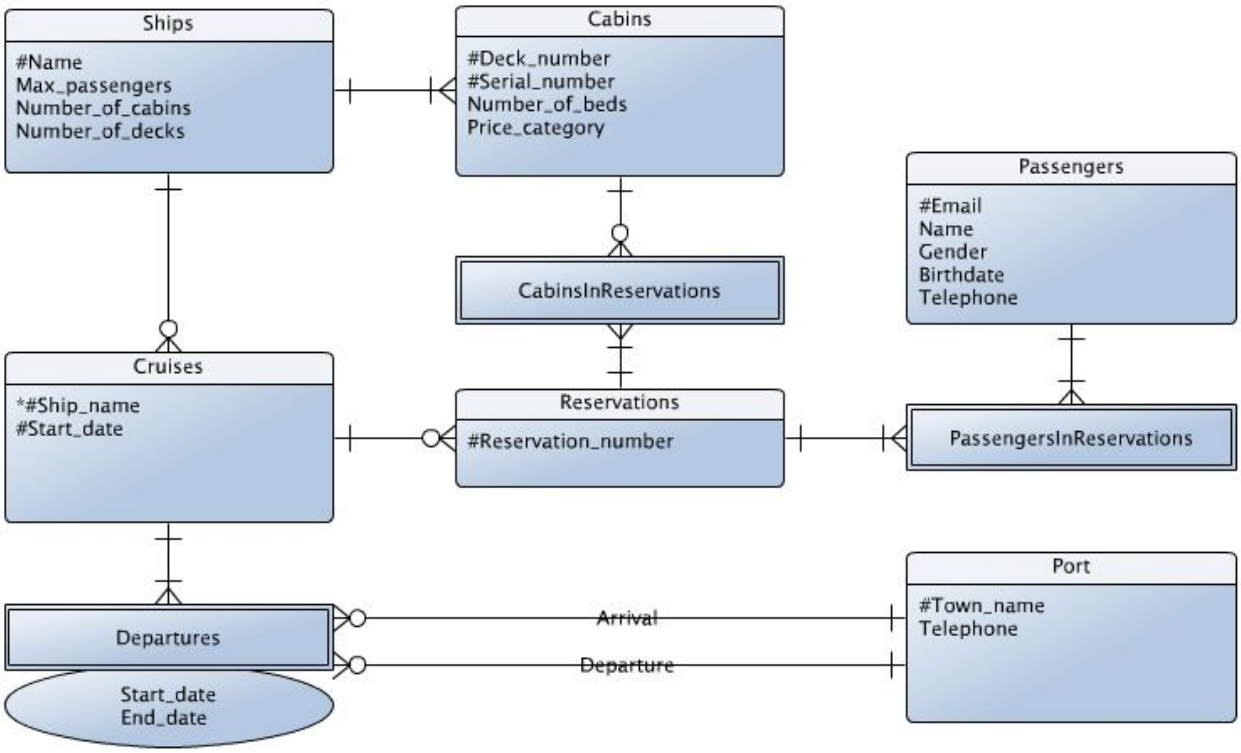
Genes is in BCNF since *Official_name*, *Start*, and *Stop* may not be unique and thus aren't primary key candidates. None of the attributes values can be determined from the foreign key **Chromosome_name*, since there are many Genes on a chromosome.

We've thus shown that none of the non-PK attributes depend on one another, making the tables 3NF. The PK doesn't depend on other attributes either, making all the tables BCNF.

Task 4

Create an ER diagram for Global cruises, specify the primary keys and select appropriate relationships between entities.

The assignment text stated that passengers could book a number of cabins in one reservation. I was unsure if you meant that we should store an actual number of cabins they wanted, or the specific cabins they desired. I assumed the latter.



The attributes of the weak entity Departures are represented in the ellipse under the entity. Cruises was not displayed as a weak entity due to lack of the triple arrow in graphity, so Cruises.Ship_name* refers to Ships.Name. ***We assume that the date refers to dateTime, and that a ship does not start several cruises in the exact same moment.*w**

We can find the route of the ship by doing a select of the departures of a cruise ordered by the start dates. The first such departure is the start port of the cruise.

Task 5

5.i Explain first why this solution proposed by the Truck table is problematic

The truck table ..

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

… is problematic for several reasons. A truck can be part of many assignments, and a single assignment may be transported by several trucks. In simpler terms, the relation between the Truck table and Assignment should be a many-to-many, but they've set it up to be one-to-many.

Having a truck cover different assignments would force us to duplicate all its non-foreign keys. This is undesirable, especially since some of the attributes must compose the primary key, and we don't allow duplicate primary keys. We have yet to decide on a primary key, so I'll just refer to the yet to be decided primary key as #Truck_primary_key.

A simple solution to the issue of many-to-many relation would be to create a weak entity between Truck and Assignment, such as..

TruckAssignments (#*_Truck_primary_key_, #*Assignment_number).

We can assume that *Truck.Maximum_weight* depends on the *Truck.Model*, so we're essentially repeating this data for each truck of the same model. Making a table ...

TruckModels (#Model, Maximum_weight)

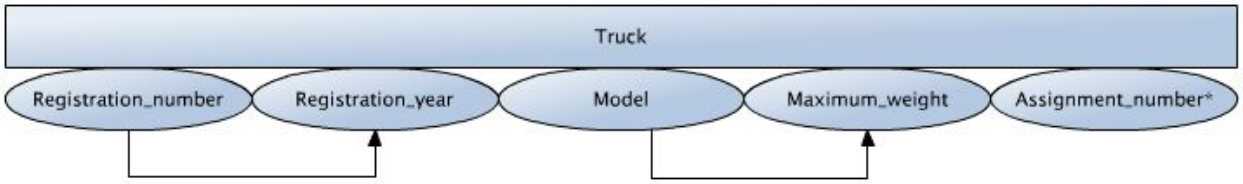
… solves this issue, giving us 3 final tables solving our dependency and relational issues:

Truck (Registration_number, Registration_year, *Model)
TruckModels (#Model, Maximum_weight)
TruckAssignments (#*_Truck_primary_key_, #*Assignment_number).

This also makes it very clear what should be the primary key of Truck, as the only unique attribute is the Registration_number, making that the only viable candidate key.

5.ii Write down the functional dependencies of the Truck table.

The Registration_year can be inferred from the Registration_number.Different trucks of the same Model should have the same Maximum_weight, but we cannot map anything from the rest of the attributes.



5.iii Determine the candidate key(s) for the Truck table

We already determined that Registration_number is the only viable candidate key in Task 5.i, but I'm going to outline it more formally here. The only determinants are Model and Registration_number, and the only field which must be unique is Registration_number; giving us the following superkeys:

1. Registration_number
2. Registration_number, Maximum_weight
3. Registration_number, Registration_year
4. Registration_number, Model
5. Registration_number, Registration_year, Model
6. Registration_number, Registration_year, Maximum_weight
7. Registration_number, Model, Maximum_weight
8. Registration_number, Registration_year, Model, Maximum_weight
9. Registration_number, *Assigment_number
10. Registration_number, Maximum_weight, *Assigment_number
11. Registration_number, Registration_year, *Assigment_number
12. Registration_number, Model, *Assigment_number
13. Registration_number, Registration_year, Model, *Assigment_number
14. Registration_number, Registration_year, Maximum_weight, *Assigment_number
15. Registration_number, Model, Maximum_weight, *Assigment_number
16. Registration_number, Registration_year, Model, Maximum_weight, *Assigment_number

The candidate key(s) must be a minimal superkey, meaning that there cannot be any superkey S such that the candidate key C is a subset of S. We have 16 superkeys S1-S16, of which S2-S16 cannot be candidate keys since S1 $\subset$ S2, …. , S1 $\subset$ S16.

We've thus shown that S1 = { Registration_number } is the only possible candidate key.

5.iv Perform normalization to BCNF for the whole table (the original table expanded to incorporate transportation). Show primary keys and foreign keys in the final result.

The following tables..

1. Container_type (#Type_id, Type_name, Max_weight, Cubic_quantity, Nightly_rate)
2. Container (#Container_number, *Type_id)
3. Customer (#Telephone_number, Address)
4. Assignment (#Assignment_number, *Telephone_number, *Container_number, Start_date, End_date)
5. Truck (#Registration_number, Registration_year, *Model)
6. TruckModels (#Model, Maximum_weight)
7. TruckAssignments (#*Truck_primary_key, #*Assignment_number).

All of the following tables are in BCNF, since all the tables are in 3NF, and there are no multiple candidate keys.

To show that there are no multiple candidate keys, we can go through all the tables and see that none of the non-PK attributes are unique, e.g. many TruckModels might have the same Maximum_weight.

We assume that different Container_type.Type_name isn't unique. If Type_name is unique, then BCNF and 3NF is violated in my proposed solution. To make it work if Type_name isn't unique, split Container type into 2 tables, such as

Container_type (#Type_id, *Type_name)
Container_details (#Type_name, Max_weight, Cubic_quantity, Nightly_rate)

To prove that the tables are in 3NF, we show that there are no transitive dependencies:.

- Table 1 satisfies 3NF since different container types can have the same name, weight, cubic quantity and nightly rate.
- Table 2 is trivial since many containers may be of the same type.
- Table 3 is in 3NF as different customers can have the same address.
- Table 4 is also OK, as different assignments might start or end at the same date, and a customer and container may be part of many assignments.
- Table 5-7 are in 3NF, as outlined in Task 5.i and Task 5.ii.

All of the tables satisfy 2NF as every attribute is automatically fully functionally dependent on the primary key (since there are no composite primary keys). The tables are also in 1NF since there are no nested tables.

We've thus shown that that the proposed tables are in BCNF.