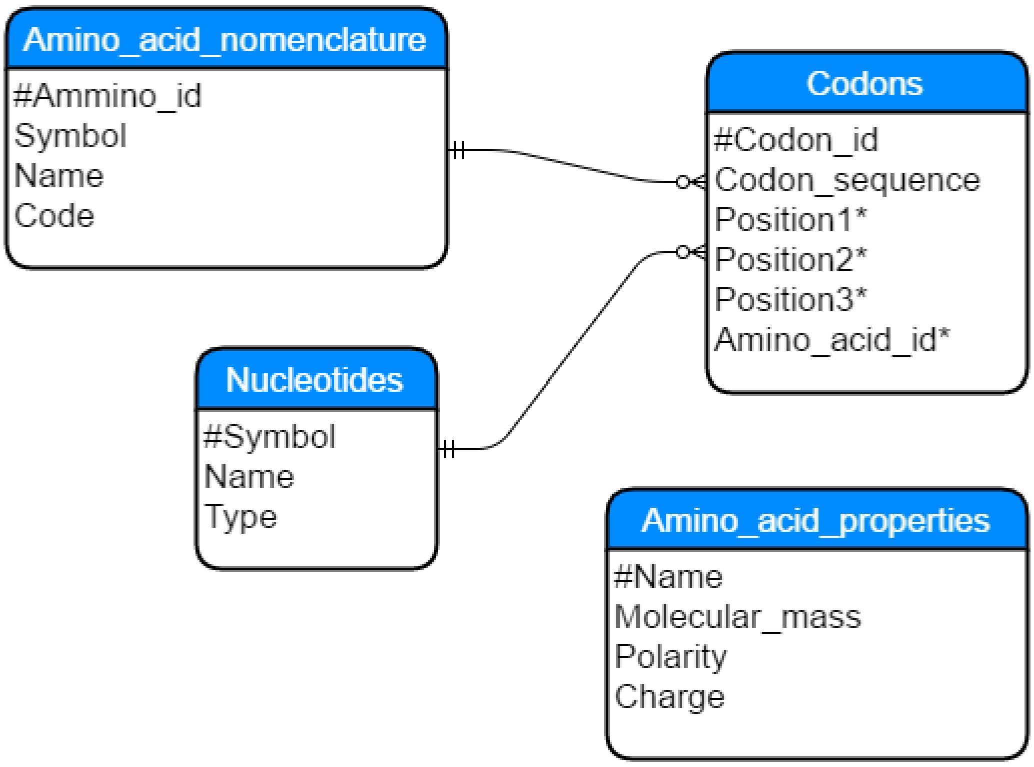
INF115 Compulsary 2

Kjetil Kjørstad, kkj004@student.uib.no

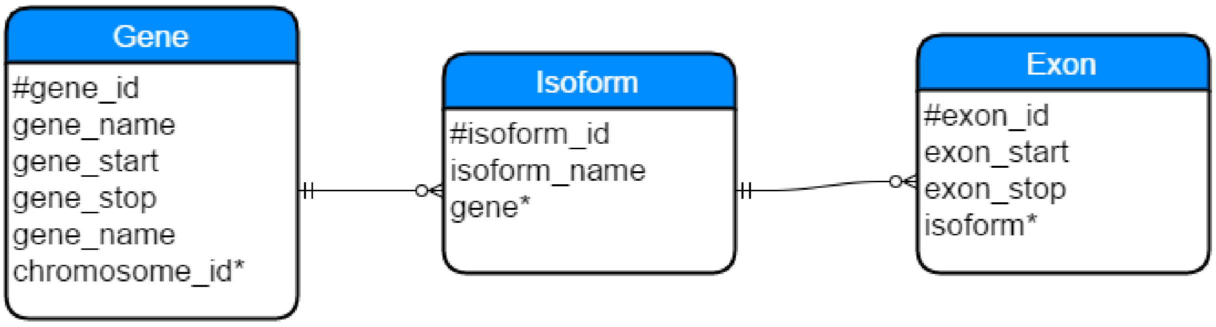
1)

Entities from exercise 1 was used according to the following structure:



2)

i. The logical entities are Genes, Exons and Isoforms. Seeing as a Gene also reference a Chromosome, this can also be thought of as a connected entity.
ii. A gene may be defined before Isoforms are mapped, therefore 0..many. Without expanding the model unnecessary, I would represent it as follows:



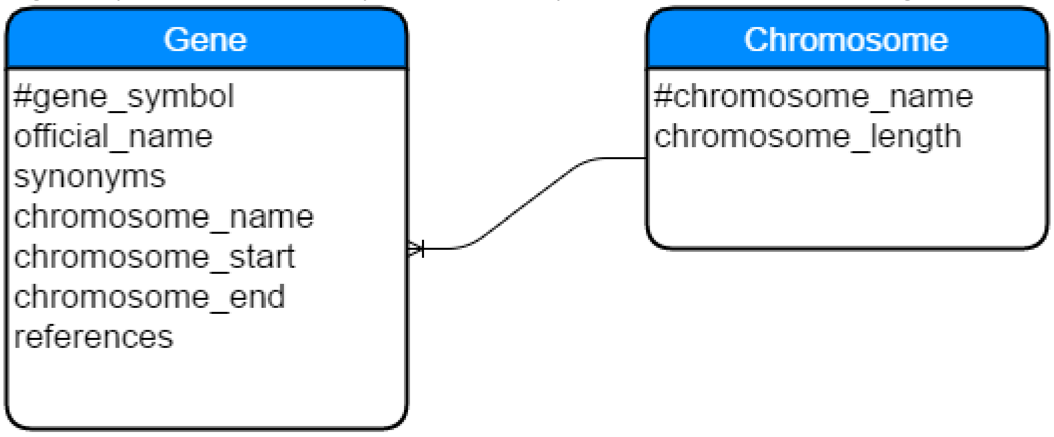
iii. Besides the unique key from 2NF, 3NF demands unique info for each key. Unclear assignment on how a set of tables are to be delivered in a pdf format, so SQL code will follow.

```
create table Gene
(
    gene_id varchar(255) unique,
    gene_name varchar(255),
    chromosome_name varchar(255),
    chromeosome_start long,
    chromosome_end long,
    primary key (gene_id)
);

create table Isoform
(
    isoform_id int not null,
    isoform_name varchar(255),
    gene varchar(255),
    primary key (isoform_id),
    foreign key (gene) references Gene (gene_id)
);

create table Exon
(
    exon_id int not null,
    isoform_start long,
    isoform_end long,
    isoform int,
    primary key (exon_id),
    foreign key (isoform) references Isoform (isoform_id)
);
```

3)

i. The logical entities are; Gene and chromosome

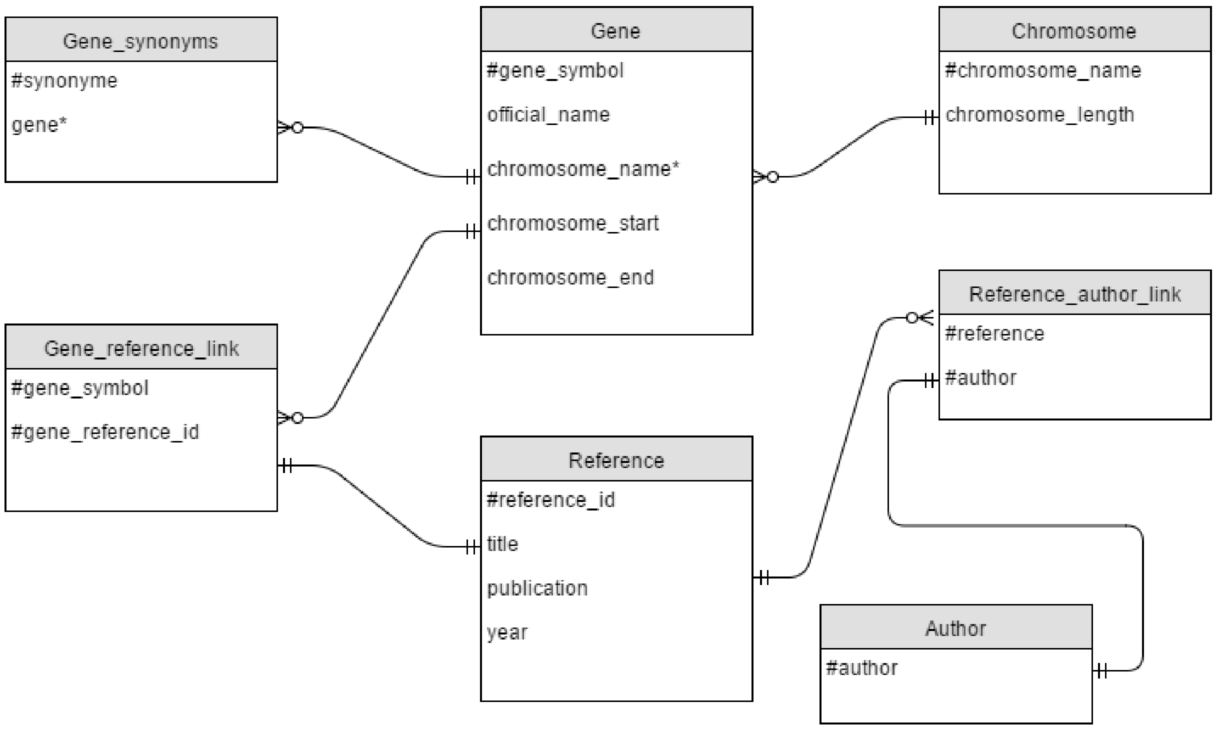
ii. Logical representation will only be one to many between chromosome and gene



iii. I will represent tables by SQL code. To make a 1NF table out of the diagram, one will need to split out every synonym as an extra row. Uniqueness is only needed for the entire row, but id and synonym will do the same check for this table (id + synonym as compound primary key)

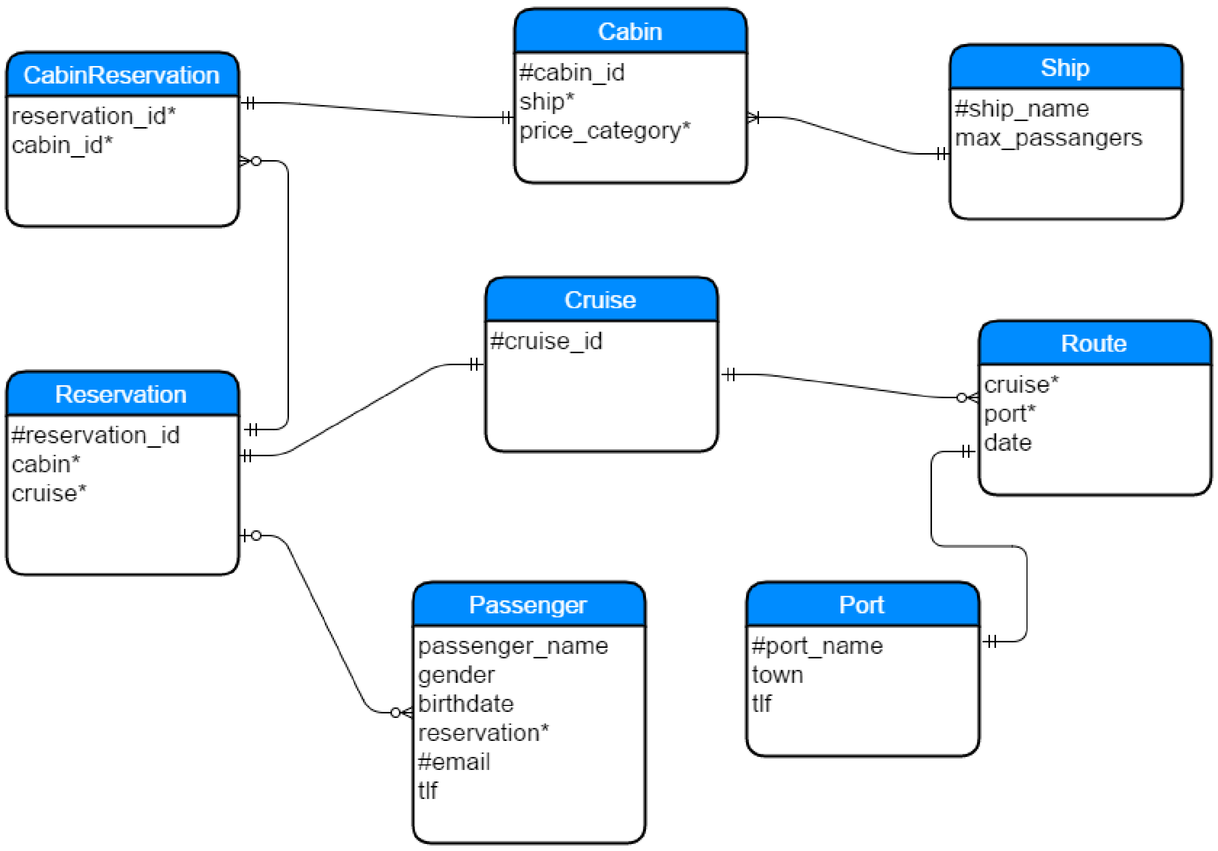
```
create table Gene
(
    gene_id varchar(255) not null,
    gene_name varchar(255),
    synonyms varchar(255) not null,
    chromosome_name varchar(255),
    chromosome_length long,
    chromeosome_start long,
    chromosome_end long,
    chromosome_references varchar(21844),
    primary key (gene_id, synonyms)
);
```

iv. To normalize the data to BCNF I have to split out synonyms and references, to avoid duplicating these. Found cooler representation of DB in draw.io.



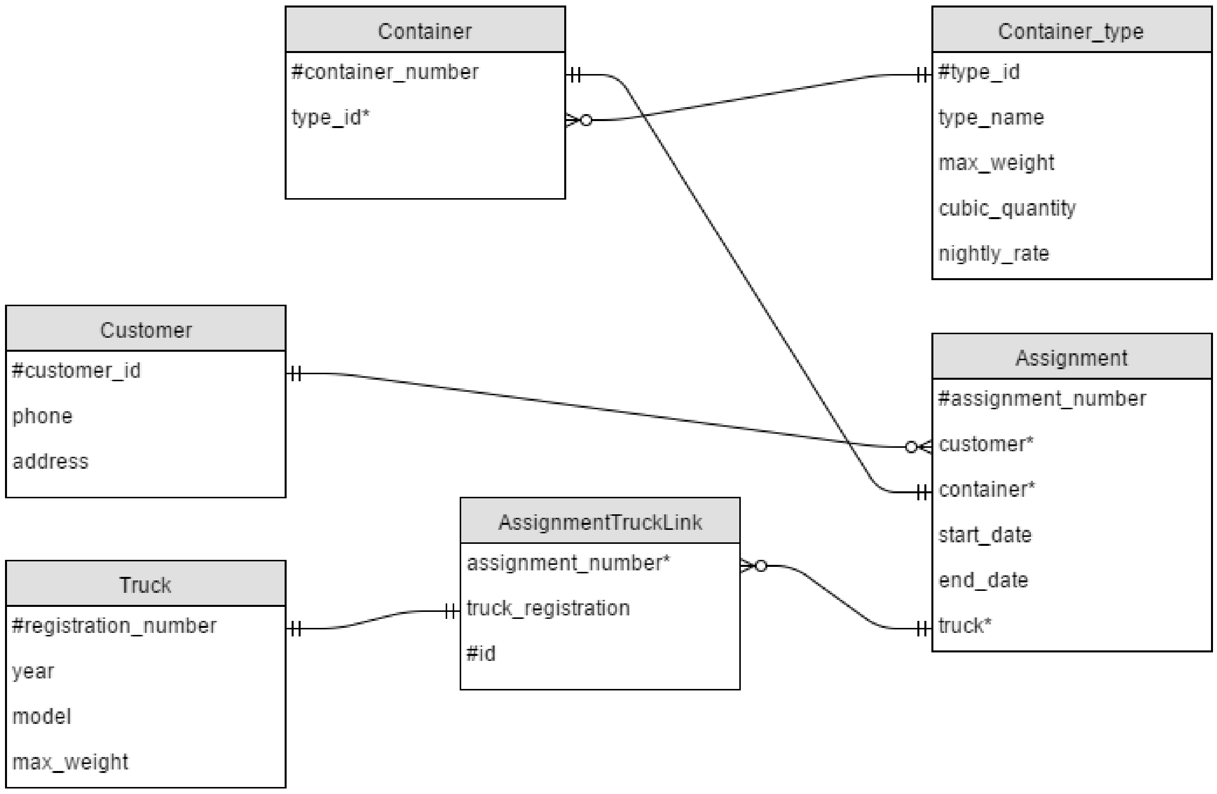
4)

Assumptions: No further specification of prices, no changing of rooms specified, reservation is the root data driver. Draw.io is difficult to control, so relationship is a bit messy.



5)

i. Problems with the proposed solution: The "Truck" record will need to be continuously updated and loosing history if assignment number is stored in truck-table, or massively duplicated if a new truck-row is posted for each assignment. Without a primary key the table will not conform to any known viable rdb structure.
ii. R(registration_number -> reg_year, mode, max_weight). This means that everything except assignment_number is functional dependent upon registration_number.
iii. Candidate key is registration number.
iv. Normalized as simple as possible


Container
#container_number
type_id*
Container_type
#type_id
type_name
max_weight
cubic_quantity
nightly_rate
Customer
#customer_id
phone
address
Assignment
#assignment_number
customer*
container*
start_date
end_date
truck*
AssignmentTruckLink
assignment_number*
truck_registration
#id
Truck
#registration_number
year
model
max_weight