# INF 115, Compulsory 2

Øyvind Hytten

April 10, 2016

# Task 1

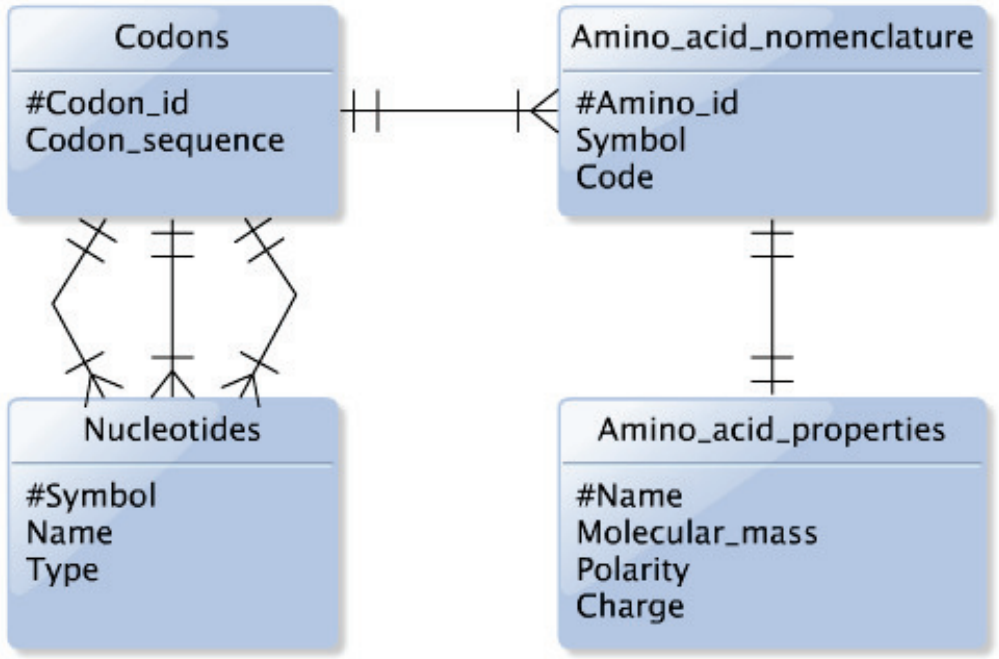


Figure 1: ER diagram from Compulsory 1

There isn't a many-to-many relation between Codons and Nucleotides, as a Codon consist of exactly three Nucleotides; hence there are rather three one-to-many relations (as a Nucleotide can be part of several different Codons).

As in the examples from the text book, I have omitted attributes which are foreign keys, instead representing them by relations (links).

I must admit I am not quite sure about the relation between Codons and Amino_acid_nomenclature, being less than familiar with biology. I suppose Codons and Amino acids aren't the same thing, which would suggest a one-to-many relation.

## Task 2 i

Entities in this scenario are Chromosome, Genes, Exons and Isoforms.

## Task 2 ii

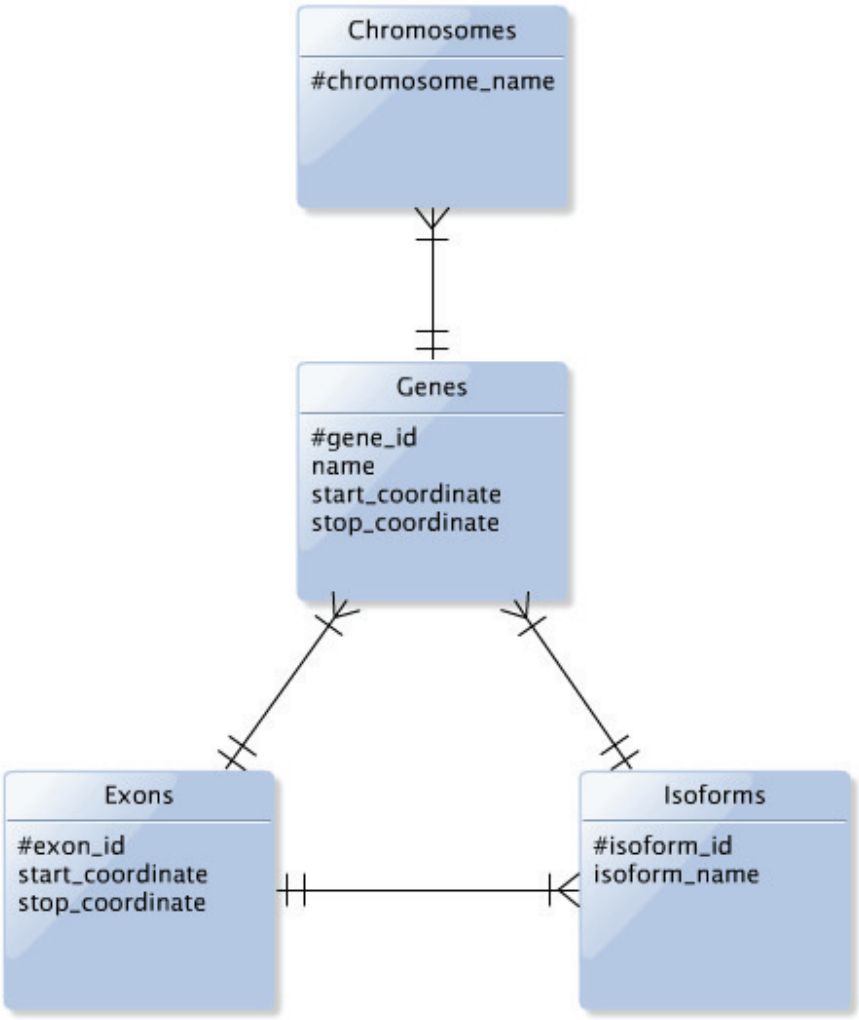


Figure 2: ER diagram for Chromosome, Genes, Exons and Isoforms

The task description provides a confusing explanation of this relationship, in particular the gene-exon-isoform dependence; perhaps by nature. The figure reflects my understanding of this relationship: that there is a direct relation as well as a transitive one between Genes and Exons.

# Task 2 iii

We need to split Genes into two separate tables to avoid a transitive dependency breaking 3NF.

- Chromosomes(#chromosome_name, gene_id*)
- Gene_names(#gene_id, name)
- Gene_properties(#gene_id, start_coordinate, stop_coordinate)
- Exons(#exon_id, start_coordinate, stop_coordinate)
- Isoforms(#isoform_id, isoform_name)

Finally, as we are dealing with a dynamic number of components ("Each gene contains one or more Exons" etc.), we should represent these pairings, or perhaps rather the exon content of a gene, as a separate two-column table in which both columns together constitute the primary key. Equivalently, we do the same for the gene-isoform- and isoform-exon relations.

- Genes_Exons(#gene_id*, #exon_id*)
- Genes_Isoforms(#gene_id*, #isoform_id*)
- Isoforms_Exons(#isoform_id*, #exon_id*)

# Task 3 i

The entities here are Chromosomes and Genes.

## Task 3 ii

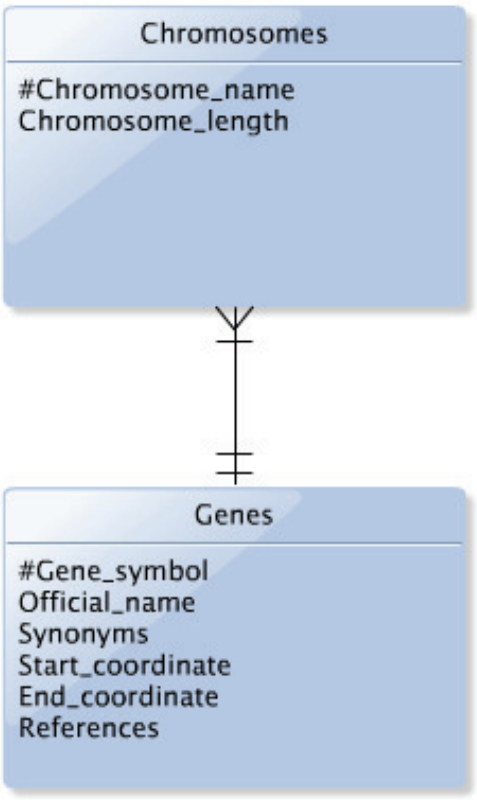


Figure 3: ER diagram for Chromosomes and Genes

## Task 3 iii

- Genes(Chromosome_name, Chromosome_length, #Gene_symbol, Official_name, #Synonym, Start_coordinate, End_coordinate, #Reference)

Note the singular form of Synonym and Reference. My thought is each combination of gene, synonym and reference make up a separate row; i.e. a gene with 3 synonyms and 4 references require in total 12 rows in this single table. And, since we have it all in a single table, there are no foreign keys.

I assume from the quotation marks in the task that a reference is represented as a single string value.

## Task 3 iv

- Chromosomes(#Chromosome_name, Chromosome_length)
- Genes(#Gene_symbol, Official_name, Start_coordinate, End_coordinate, Chromosome_name*)
- Synonyms(#Gene_symbol, Synonym)
- References(#Gene_symbol, Reference)

# Task 4

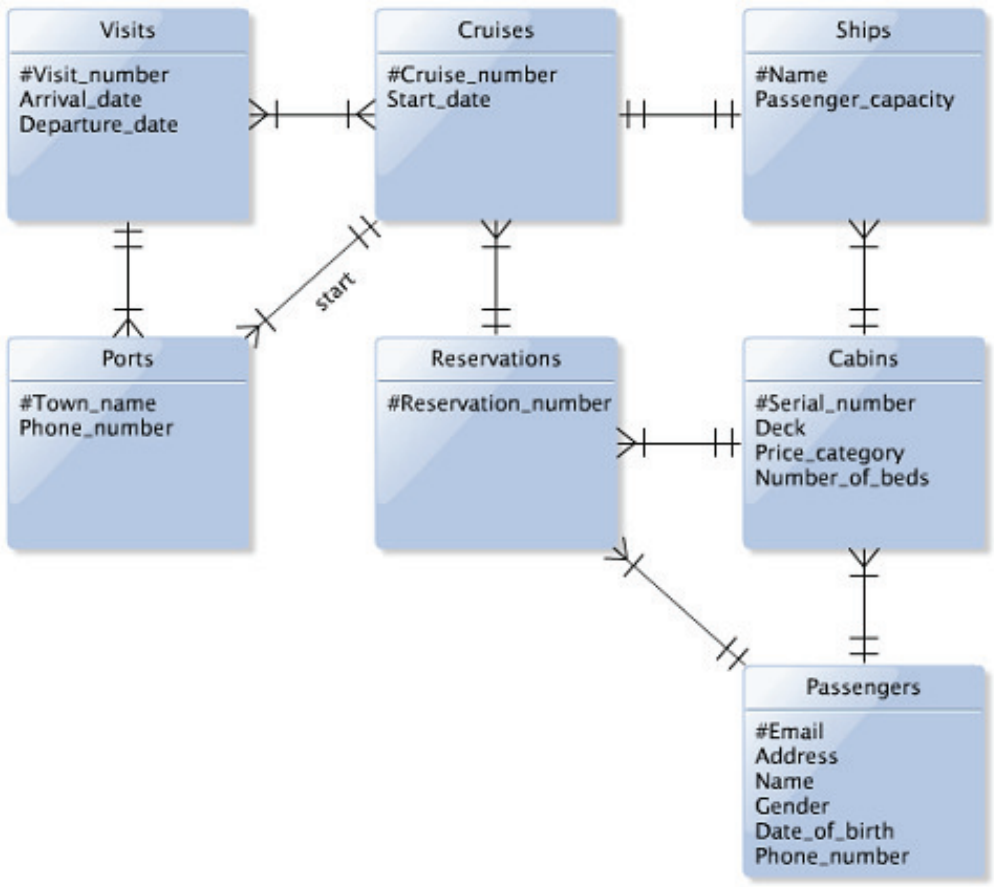


Figure 4: Global_cruises

In a functional database design Visits could be uniquely identified by for instance its Arrival_date and an identifier for the visiting Cruise (which would be a foreign key in Visits). However, in this ER representation such a foreign key is substituted with a link, and so we need an explicit primary key for Visits.

In a similar manner, Cruises could be identified by its start date and its route (assuming no two ships ever start on the same day and sail the same

route), but this would constitute a recursive issue with my proposal in the paragraph above.

## Task 5 i

This approach would require Assignment_number to be part of the primary key of Truck, and would also generate excessive amounts of overhead data in all other columns (for trucks that participate in more than one assignment, ever).

## Task 5 ii

Registration_number determines Registration_year, Model and Maximum_weight. One could perhaps argue that Model and Registration_year determine Maximum_weight, but due to the facts that a vehicle's registration year and its production year is not the same thing, and that a model can possibly have different cargo capacities from year to year (of production), we cannot be sure that two trucks of the same make and model, registered the same year, have the exact same capacity.

As such, Registration_number is the only determinant in Truck.

## Task 5 iii

Since we have a functional dependency from Registration_number to all other columns except Assignment_number, the one and only candidate key consists of these two.

## Task 5 iv

We would split the assignment reference into a separate table, let us call it TruckParticipation.

- Truck(#Registration_number, Registration_year, Model, Maximum_weight)
- TruckParticipation(#Registration_number*, #Assignment_number*)

However, I expect the aim for this task is to infer some other functional dependency as well, namely the one in which Registration_year actually means Production_year. To humor you I will make this substitution, with the added functional dependency it involves, which again would yield the following BNCF tables:

- Truck(#Registration_number, Production_year*, Model*)
- TruckProperties(#Production_year, #Model, Maximum_weight)
- TruckParticipation(#Registration_number*, #Assignment_number*)