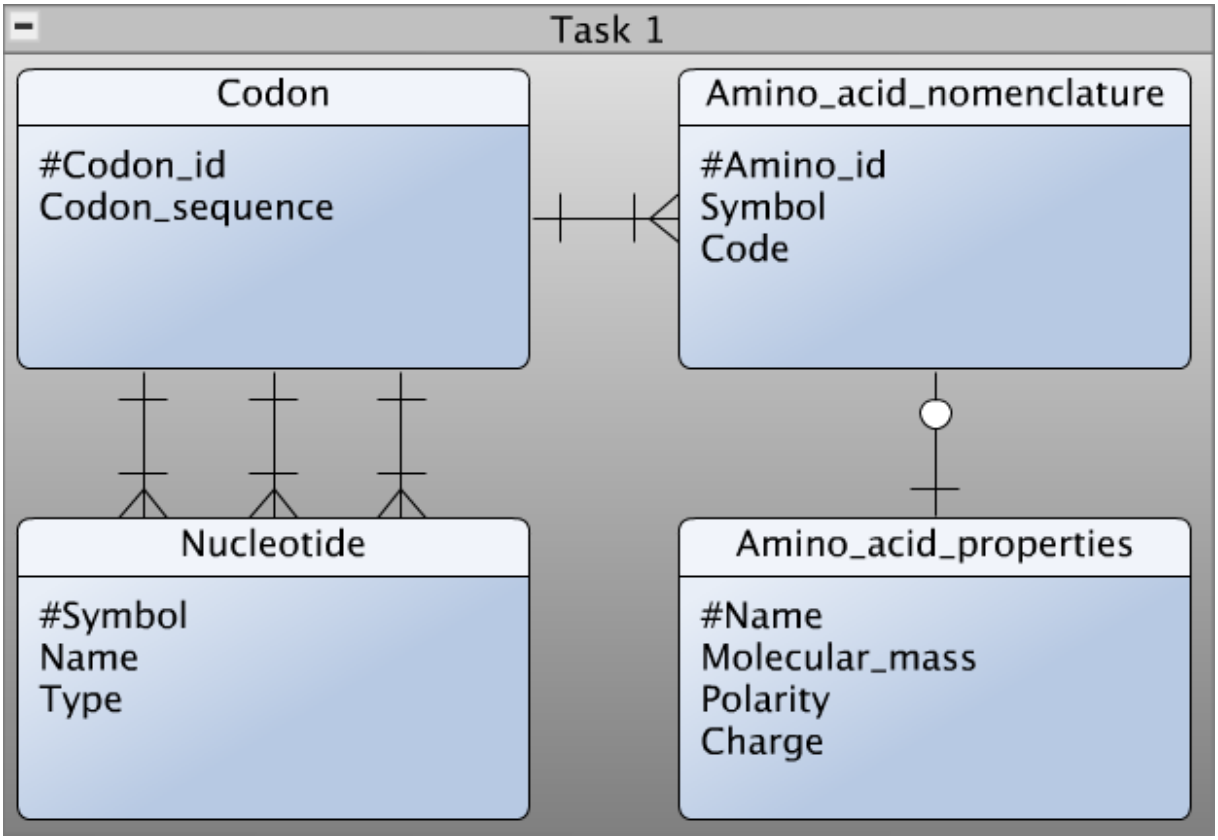
INF115 – Semester Task 2

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



Each Amino_acid_property codes to *one* Amino_acid_nomenclature, but some nomenclatures don't code for any properties. Further; each Codon codes for exactly *three* nucleotides and *one* nomenclature.

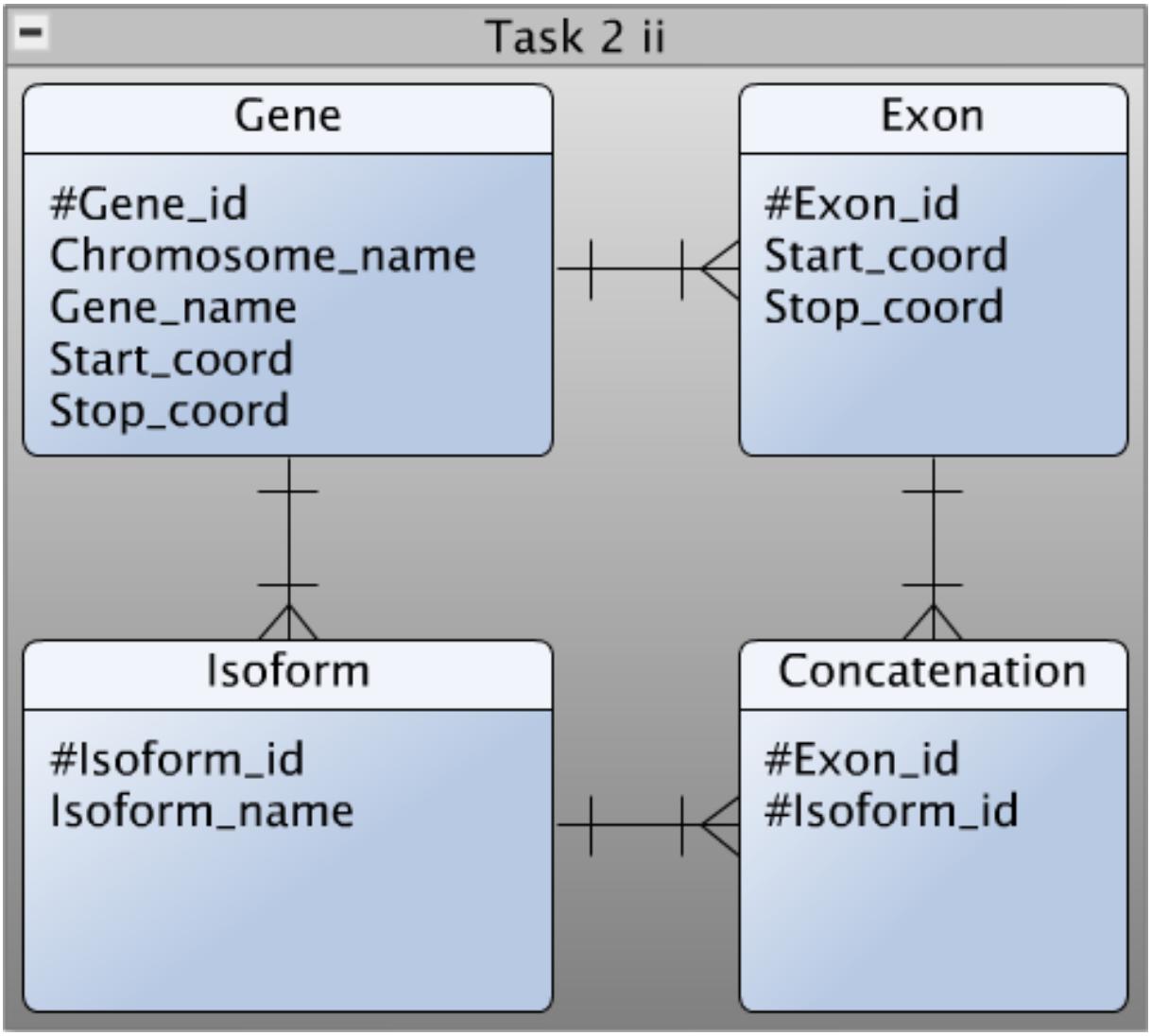
2) A database stores the information about the relations between Genes, Exons and Isoforms. A Gene has a unique gene_id and a name, the database will also store some information about the location of the gene, including the name of the chromosome on which it is located and the start and stop coordinate of the gene on the chromosome. Genes are always located on just one chromosome. Each gene contains one or more Exons, each with a unique exon_id and a start and stop coordinate. Every gene has one or more Isoforms, each with a unique isoform_id and an isoform_name, each Isoform is created from one or more of the Exons of a particular gene.

2 i) Identify the entities in the database description above.

There are three entities in the database description;

Gene, Exon, and Isoform.

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.

Gene (#Gene_id, Chromosome_id, Gene_name, Start_coord, Stop_coord)

Exon (#Exon_id, Gene_id*, Start_coord, Stop_coord)

Concatenation (#Exon_id*, #Isoform_id*)

Isoform (#Isoform_id, Gene_id*, Isoform_name)

3) Gene nomenclature can be confusing because the experimental nature of gene discovery can lead to the same gene being "discovered" and named a number of times by separate individuals. At some point the scientists will realize that they all discovered the same gene and a standardized name will be agreed upon, along with a new identifier called the gene symbol. All of the other names will be recorded as official synonyms for this gene to allow backward compatibility with the literature that has previously been published.

A database will record the following information about the genome of a particular species:

Gene_symbol, Offical_name, Synonyms, Chromosome_name, Chromosome_length, Start_coordinate, End_coordinate, References.

The synonym field contains a list of all of the previously used names. The References field contains a list of all of the publications that have cited this gene (each composed of: reference id, authors, title, journal, year published). A publication can cite more than one gene. Assume for this exercise that all genes have at least one synonym.

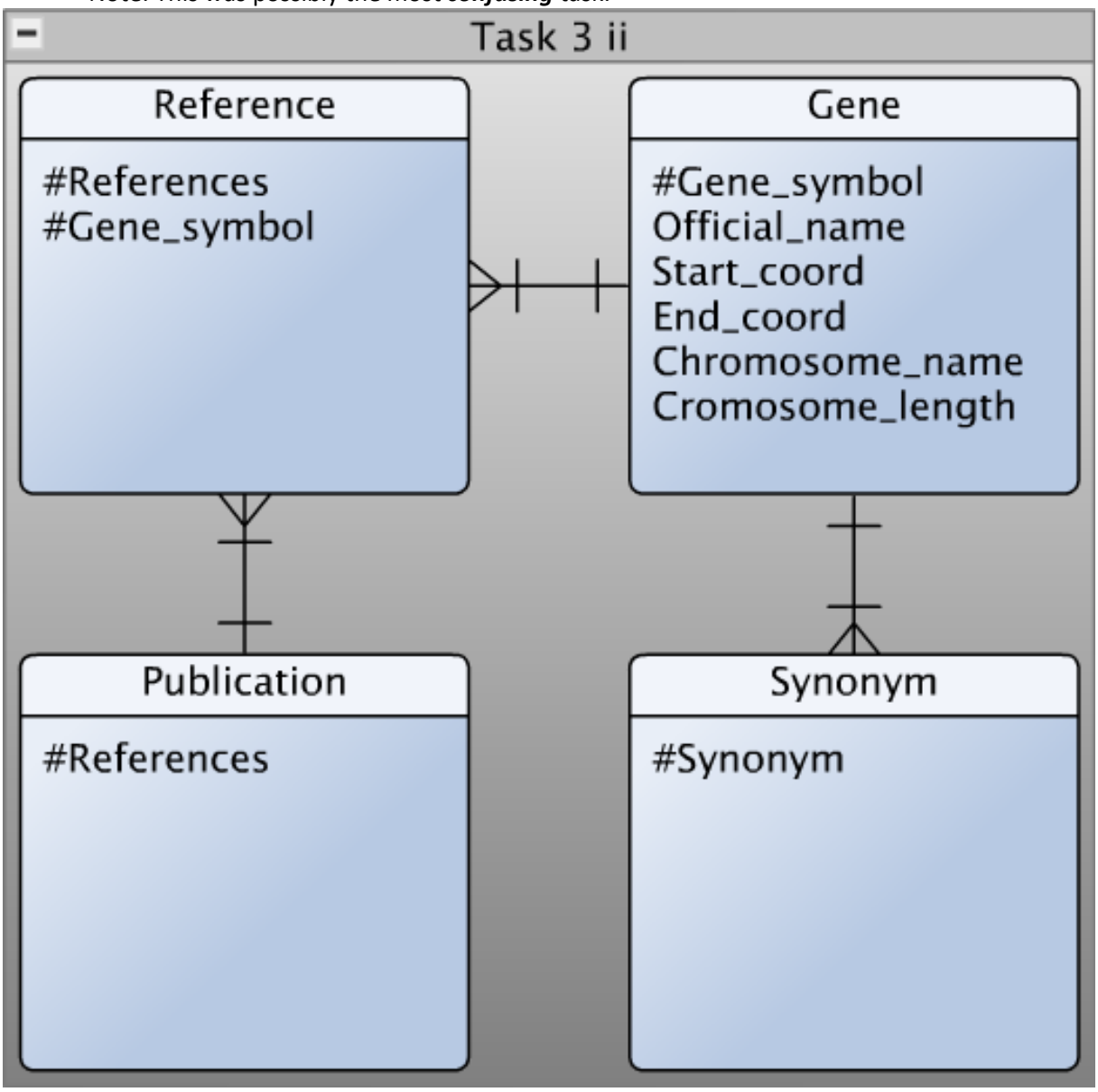
3 i) Identify the entities in the database description above.

There are three entities in the database description;

Gene, Synonym, and Publication.

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

Note: This was possibly the most ***confusing*** task.



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 keys and foreign keys.

Gene (#Gene_symbol, #Chromosome_name, Chromosome_length, Official_name, Start_coord, End_coord)

Synonym (#Synonym, Gene_symbol*)

Reference (#Reference_id*, #Gene_symbol*)

Publication (#Reference_id, Title, Journal, Year_published)

Writer (#Author_id, #Reference_id*)

Author (#Author_id, Author_forename, Author_surname)

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.

Chromosome (#Chromosome_name, Chromosome_length)

Gene (#Gene_symbol, Chromosome_name*, Official_name, Start_coord, End_coord)

Synonym (#Synonym, Gene_symbol*)

Reference (#Reference_id*, #Gene_symbol*)

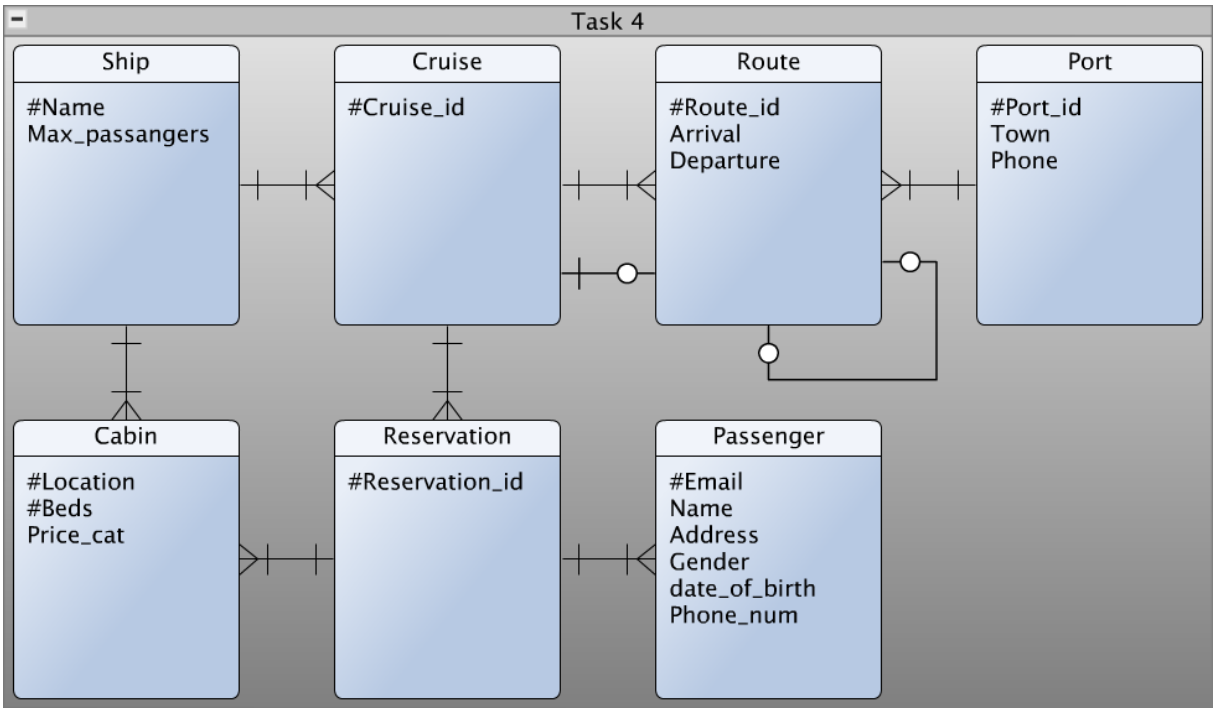
Publication (#Reference_id, Title, Journal, Year_published)

Writer (#Author_id*, #Reference_id*)

Author (#Author_id, Author_forename, Author_surname)

4) Global_cruises offers luxury cruises worldwide, and is in need of a new system to handle ships, passengers and routes. The system must store information about each cruise ship. Each ship has a unique name. In addition, the maximum number of passengers will be stored for each ship. A cruise starts on a particular date from a particular port and follows a specific route. The system must store which harbors a cruise visits on each day. It must be possible to find out which date a cruise arrives at and leaves a particular port. For every port the town name and telephone number of the port office should be stored. Every cruise ship has a number of cabins (rooms) in 4 to 8 decks (floors). A cabin is identified by a deck number and a serial number, for example "4-17" means cabin 17 on deck 4. The cabins are assigned to different price categories depending on the number of beds and location on the ship. The system must also store information about the passengers and their reservations. Every passenger gets a unique email address, name, gender, date of birth and a telephone number. A reservation always applies to one particular cruise, but multiple passengers can be entered on one reservation. Passengers can book a number of cabins in a single reservation.

Create a data model (E / R diagram) for Global_cruises, specify the primary keys and select appropriate relationships between the entities.



Each passenger is associated with a reservation, which reserves one or more Cabins on a given cruise. Each ship is associated with one or more Cruises (assumed to only be one at a time, but for record-keeping purposes it's relevant to keep track) and each Cruise has a route, a route represents a destination and has an arrival and departure time, cruise and port, in addition, it also says tells what destination comes next (self-reference), first route's arrival field is null and final destination has no departure time. Finally, each route has a port.

5) A company rents out containers, and maintains the following database. Primary keys are marked with a hash ("#") and foreign keys are marked with an asterisk ("*"):

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)

The company wishes to expand so that it can provide transport services for the containers. The company will purchase a number of trucks and the various assignments will be continuously distributed on the available transportation. Multiple trucks can be involved in the transportation of one assignment. The following table has been proposed to extend the database in order to handle the new transportation requirements:

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

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

First, the Truck table doesn't have any Primary Key (easily fixed by making the Registration_number into a PK). Second, the Assignment_number FK is problematic because it prevents a given Truck from being on multiple assignments at the same time (in addition to the database being unable to store which Truck was involved in various assignments). Third, the Maximum_weight field is (presumably) dependent upon Model, which is just inefficient.

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

Maximum_weight is dependent on Model

Model is functionally dependent on Registration_number

Registration_year is functionally dependent on Registration_number

Assignment_number is functionally dependent on Registration_number

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

As shown in the previous subtask, only the *Registration_number* is a 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.

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*)

Connection (#Registration_number*, #Assignment_number*)

Model (#Model, Maximum_weight)