Bioinformatics Resources
Exercise Sheet 8 - Graded
due on July 19th, 9.30am

- Results can be sent to richter@in.tum.de before the deadline (Friday, July 19th, 9.30). Since you have ample time, any late submissions will not be accepted.
- A PDF file will likely be best for all descriptions and results. Code can be submitted in a separate file. A Jupyter Notebook with everything is also fine. Submit everything as an archive if you have multiple files.
- We request you to submit your coding solutions in Python3.
- Generally, make sure any program code executes. If you submit something that doesn’t even survive compilation because you have typos in the code, it will not be graded and you get 0 points on that task.
- If you don’t know how to fully solve a task submit your best effort (that still means it must compile and not have any obvious errors).
- If your results are too large for an email attachment (which they shouldn’t be…) either bring them to our offices or give them to Lothar at the start of the lecture.
- This is an individual submission with one submission per student. If I find significant overlap or other hints for copying the bonus will be voided.
- Remember to describe how you arrived at your results in all instances. Just the solution, even if correct, will not give you full points.
- If anything is completely unclear contact Lothar at the above email addresses.

Task 1 (7P)
Use the SQL schema from a previous exercise (https://rostlab.org/~tmhpred/EX8_schema.sql and also given on the last page) and give the SQL queries for the following questions. Don’t create views, just one SQL statement should suffice. You can add some dummy data to check whether your query works:

- Which students attend (hoeren) which lectures? Give student names and lecture titles (1P).
- Which professor has how many lecture hours (sws)? Give professor names and their total sws (1P).
  - Also list professors that exist in the database but do not give lectures (NULL values are fine in the output) (0.5P)
  - NULL values are not fine in the output. All professors that do not give lectures should have a value of 0. Just this statement is fine as a result to get all 2 points. (0.5P)
- Which students attend lectures for which they have not attended the required lectures (as specified by voraussetzen)? Give the students’ names, matnrnr and
vorhr of the lecture they are attending without having attended the predecessor (3P).
- How many students attend the lectures of each professor? Give the professor’s name and the total number of students that attend the lectures held by this professor. A student that attends multiple lectures by the professor can be counted multiple times. (1P)

Task 2 (5P)
- Given the following two tables in a relational database, what is the SQL statement to select the gate of a flight that travels from Munich to Tokyo, with one stop in-between. Again, use just one SQL statement, no views. Primary keys are IATA Code and Flight Number respectively. Flight numbers identify a flight from one airport to another airport without any stops in-between. (2P)

<table>
<thead>
<tr>
<th>Table “Airport”</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATA Code</td>
</tr>
<tr>
<td>MUC</td>
</tr>
<tr>
<td>LGA</td>
</tr>
<tr>
<td>JFK</td>
</tr>
<tr>
<td>EWR</td>
</tr>
<tr>
<td>NRT</td>
</tr>
<tr>
<td>HND</td>
</tr>
<tr>
<td>TXL</td>
</tr>
<tr>
<td>…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table “Connections”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight Number</td>
</tr>
<tr>
<td>42</td>
</tr>
<tr>
<td>1337</td>
</tr>
<tr>
<td>…</td>
</tr>
</tbody>
</table>

- Now model this data in neo4j. Give the corresponding create statements for some sample data, as well as the cypher query to answer the question from above. Use edges in your graph, don’t model everything as nodes. What is the query if you want your layover to be in New York and the airline for the second flight to be ANA? (3P)

Task 3 (5P)
Using the MongoDB demo code from the website (https://www.rostlab.org/sites/default/files/fileadmin/teaching/SS16/Bioinfres/getDataToMongo.py_.zip) perform the following tasks:
- Instead of letting MongoDB create a unique identifier for every document automatically, make use of the fact that the primary accession fulfills these criteria and use that as unique id for every document. Change the code that performs the initial insert accordingly. (0.5P)
- Update all entries by adding the taxonomy feature and the date field from the genbank entry to the mongodb documents. Don't rewrite the whole document, just add the fields to the already existing ones. (1.5P)
- Update all documents (in one command) so the sequence field is removed (since it currently contains truncated sequences, which doesn't make any sense). Print out the number of entries that are matched and the number of entries changed (both numbers should be 5) (3P).

Task 4 (13P)
Here (https://rostlab.org/~tmhpred/ex8_pubmed.js) is a subset of Pubmed already parsed for importing into mongodb. Load the data into your database and add indices on the fields pmid, authors, created and mesh. Also mark pmid as unique. Now give the queries to answer the following questions on the mongodb shell. Supply them in a way that I can copy/paste them to my shell. For example, in a plain text file. Remember to test your queries for typos, otherwise they will not be graded. If you want a more readable output for debugging you can append .pretty() to every query:

- All articles in English (0.5P)
- All articles published in 2015 (0.5P)
- All articles with an author that has the last name "Rost" (0.5P)
- Number of articles with MESH terms "Heart" and "Verapamil" (0.5P)
- All articles from before 1970 that are in German (1P)
- Only the pmids for all articles in the journal PLoS ONE (1P)
- All articles in PubmedCentral (contain a pmc field) (1P)
- Articles where one or multiple of the MESH terms contain the string "Protein" (1.5P)
- The year of all articles where the last name of the first author is "Rost" (don’t return the whole document!) (1.5P)
- The number of articles for every journal (identified by the abbr field) (2.5P)
- From all articles created after 2010, the number of articles per year that contain the mesh term "Software". Sorted first by the number of articles (highest number first) and then sorted by the years (younger years appear first in list). Outputs are just the years and the number of articles in that year. (2.5P)

Task 5 (6P)
This task covers JavaScript. You can provide a .js file or include the code in one of your other submission documents.

- Write a function which returns a function that return a substring of length n from a provided string argument. The length n of the substring is set from the outer function. Use IIFE. (2P)
- Write a function that takes a single value or an array of values and returns either the square of a the single number or an array with all number squared for all inputs which are a number or a string representation of a number (4P)

Good luck,
L. Richter
CREATE TABLE studenten(  matrnr MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,  name varchar(60) NOT NULL,  semester TINYINT UNSIGNED,  PRIMARY KEY (matrnr) ) CHARACTER SET utf8;

CREATE TABLE vorlesungen(  vorlnr SMALLINT UNSIGNED NOT NULL,  sws TINYINT UNSIGNED,  titel varchar(60) NOT NULL,  PRIMARY KEY (vorlnr) );

CREATE TABLE assistenten(  persnr MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,  name varchar(60) NOT NULL,  fachgebiet varchar(60),  PRIMARY KEY (persnr) );

CREATE TABLE professorenn (  persnr MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,  name varchar(60) NOT NULL,  rang char(2),  raum SMALLINT UNSIGNED,  PRIMARY KEY (persnr) );

ALTER TABLE vorlesungen ADD COLUMN gelesenvon MEDIUMINT UNSIGNED NOT NULL;

describe vorlesungen;

ALTER TABLE vorlesungen ADD FOREIGN KEY (gelesenvon) REFERENCES professorenn (persnr);

describe vorlesungen;

ALTER TABLE assistenten ADD COLUMN chef MEDIUMINT UNSIGNED NOT NULL;

ALTER TABLE assistenten ADD FOREIGN KEY (chef) REFERENCES professorenn (persnr);

describe assistenten;

CREATE TABLE hoeren(  student MEDIUMINT UNSIGNED NOT NULL,  vorlesung SMALLINT UNSIGNED NOT NULL,  FOREIGN KEY (student) REFERENCES studenten (matrnr),  FOREIGN KEY (vorlesung) REFERENCES vorlesungen (vorlnr),  PRIMARY KEY (student, vorlesung) );

CREATE TABLE voraussetzen(  vorlesung_kind SMALLINT UNSIGNED NOT NULL,
vorlesung_elter SMALLINT UNSIGNED NOT NULL,
FOREIGN KEY (vorlesung_kind) REFERENCES vorlesungen (vorlnr),
FOREIGN KEY (vorlesung_elter) REFERENCES vorlesungen (vorlnr),
PRIMARY KEY (vorlesung_kind, vorlesung_elter)
);
CREATE TABLE pruefen(
    student MEDIUMINT UNSIGNED NOT NULL,
    vorlesung SMALLINT UNSIGNED NOT NULL,
    professor MEDIUMINT UNSIGNED NOT NULL,
    note DECIMAL(2,1) UNSIGNED NOT NULL,
    FOREIGN KEY (student) REFERENCES studenten (matrnr),
    FOREIGN KEY (vorlesung) REFERENCES vorlesungen (vorlnr),
    FOREIGN KEY (professor) REFERENCES professoren (persnr),
    PRIMARY KEY (student, vorlesung)
);