Homework 3

Submit as a pdf upload via Learn@WU till end of Thursday 11.01.2018

Exercise 1:

(a) Which isolation level the transaction of User 1 can have in the following execution protocol? Explain.

```
User 1:  BEGIN;
User 1:  SELECT name FROM Customers WHERE cid = 1;
     name
-----------------------
     Anna

User 2:  BEGIN;
User 2:  UPDATE Customers SET name = 'Hannah' WHERE cid = 1;
User 2:  SELECT name FROM Customers WHERE cid = 1;
    name
-----------------------
   Hannah
User 2: COMMIT;

User 1:  SELECT name FROM Customers WHERE cid = 1;
    name
-----------------------
    Anna
```

(b) Which isolation levels the transaction of **User B** can have in the following command execution protocol? Explain your answer.

```
User A:  BEGIN;
User A:  UPDATE Customers SET name = 'Hannah' WHERE cid = 1;
User A:  SELECT name FROM Customers WHERE cid = 1;

  name
-----------------------
  Hannah

User B:  BEGIN;
User B:  SELECT name FROM Customers WHERE cid = 1;
  name
-----------------------
   Anna

User A: COMMIT;

User B:  SELECT name FROM Customers WHERE cid = 1;
  name
-----------------------
  Hannah
```

Exercise 2: The daily menu includes a single starter, a main course, and a dessert. The table Menu is defined as follows:

LunchMenu

Menu Id	**C**ourse	Course **T**ype
1	Salad	Starter
1	Schnitzel	Main course
1	Profiterole	Dessert
2	Garlic soup	Starter
2	Kaiserschmarrn	Main Course
2	Water melon	Dessert
3	Tomato soup	Starter
…		

Menu Id (M) is declared to be the PK in the table.

a) List all functional dependencies that hold in LunchMenu, given that each menu (M) contains exactly one course (C) of each type (T)?
b) Write down all superkeys of the table LunchMenus
c) Write down all candidate keys and explain how did you obtain them.
d) Is the table LunchMenu in BCNF? If not, bring it into the BCNF.
e) Is LunchMenu in 3NF? Justify.

Exercise 3: Decompose (respecting the lossless-join property) the relation Boxfights (**N**ickname, **R**eal Name, **W**eight, **A**ge, **G**roup, **C**ity, **D**ate, **P**lace) into BCNF using the procedure presented in Lecture 4. The following FDs are defined over NRWAGCDP:

N → R, N → W, N → A, CD →P, WA → G, P → C

Nickname	Real Name	Weight	Age	Group	City	Date	Place
Lights Out	James Toney	99	47	HW	München	22-10-2015	Olympiahalle
Iron Junior	Vincent Feigenbutz	76	18	JMW	München	05-09-2015	Zenith
Captain	Marco Huck	90	30	HW	München	22-10-2015	Olympiahalle
Lights Out	James Toney	99	47	HW	Karlsruhe	01-09-2015	Europahalle
Captain	Marco Huck	90	30	HW	München	05-09-2015	Zenith
Iron Junior	Vincent Feigenbutz	76	18	JMW	Karlsruhe	15-09-2015	dm-Arena
What the Heck	Owen Beck	105	39	HW	Karlsruhe	01-09-2015	Europahalle

Is your BCNF decomposition dependency preserving? If no, provide a dependency preserving 3NF decomposition.

Exercise 4. Redis. The online demonstration version at http://try.redis.io and the commands listed on the slide 20 of Lecture 5 should suffice to solve the tasks below.

a. Initialize the key S:1 to contain the set {1,3,4,6} and the key S:2 be the set {a, 4, b, 3}
b. With a single command initialize the key S:3 to contain the intersection of S:1 and S:2
c. Make the key S:4 contain an empty set.
d. Use the data type “sorted set” and a sequence of Redis commands to count three most frequent elements in the list of characters (letters) “(i f a j i 3 b f a 2 i j a 3 a)”.

Exercise 5. MongoDB. Refer to the documentation on the basic CRUD (**C**reate **R**etrieve **U**pdate **D**elete) operations of MongoDB https://docs.mongodb.com/manual/crud/. You can find links to executable examples after the words “For examples, see” in each CRUD category, in particular:
https://docs.mongodb.com/manual/tutorial/insert-documents/
https://docs.mongodb.com/manual/tutorial/query-documents/
https://docs.mongodb.com/manual/tutorial/query-embedded-documents/
https://docs.mongodb.com/manual/tutorial/query-arrays/

a. Represent the Suppliers-Catalog-Parts schema from the first Homework as a single document collection *catalog* in Mongodb.
The following commands can also be found at http://sqlfiddle.com/#!15/9334e #

create table Suppliers (sid **integer primary key**, sname **varchar**, address **varchar**);
create table Parts(pid **integer primary** k**e**y, pname **varchar**, color **varchar**);
create table Catalog(sid **integer**, pid **integer**, cost **real**, **primary key**(sid, pid));

insert into Suppliers **values**(1, 'Best Red East', 'Brigittenau'), (2, 'Green West', 'Hietzing'),
(3, 'Nordparts', 'Dobling');

insert into Parts **values**(1, 'Skypart', 'blue'), (2, 'Woodpart', 'green'),
(3, 'Grasspart', 'green'), (4, 'Sunpart', 'red'), (5, 'Firepart', 'red');

insert into Catalog **values** (1, 4, 120), (1, 5, 223), (1, 3, 523), (2, 2, 499), (2, 3, 320),
(2, 4, 161), (3, 1, 356), (3, 2, 650), (3, 3, 586), (3, 4, 184);

b. Implement the first four queries from the corresponding exercise of homework 1. Refer to the list of MongoDB query operators here:
https://docs.mongodb.com/manual/reference/operator/query/

i. Find the names of suppliers who supply some red part.
ii. Find the sids of suppliers who supply some red or green part.
iii. Find the sids of suppliers who supply some red part or are in Dobling.
iv. Find the sids of suppliers who supply some red part and some green part.

Exercise 6. SPARQL. In the addresses of pages of the English Wikipedia, replace the URL prefix https://en.wikipedia.org/wiki/ with "http://dbpedia.org/page/", to obtain the description of respective resource in the English DBpedia.
For instance:

- https://en.wikipedia.org/wiki/France ⇔ http://dbpedia.org/page/France
- https://en.wikipedia.org/wiki/Paris ⇔ http://dbpedia.org/page/Paris
- https://en.wikipedia.org/wiki/Austria ⇔ http://dbpedia.org/page/Austria

Starting with the examples in the section of Lecture 5 on SPARQL, **construct two SPARQL** queries of your liking.

HINTS:

- Check available properties from Wikipedia subject at respective DBpedia page (see above how to derive the DBpedia subject URI from the Wikipedia page address)
- Use some of the (numerous) properties in your SPARQL queries.
- Check out other SPARQL endpoints, e.g. Wikidata https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries/examples