DB 24 Handin-5 **Triggers and Indexes and Database applications** **Deadline 15.3 18:00**

The purpose of this hand-in is to demonstrate that you know how to use triggers to define complex constrain constraints, use views as named queries and discuss core principles in major indexing approaches and use indexes to speed up databases access. Moreover, to describe the core principles in B-tree family indexes and connect to a database from a program in Java or Python.

Please make sure to make yourself familiar with the reading material and lectures, and work with the presentation exercises before doing your hand-in. As detailed in "the road to success" on the course homepage, this benefits your learning and your understanding of what this hand-in asks you to do. The criteria for evaluation are detailed in the rubrics for this hand-in on Brightspace.

*Please note: You do **NOT** have to use MySQL in order to solve this hand-in, and you only need to hand in your statements in a readable format, but we expect that it is very helpful for you to try it out in your MySQL workbench.*

---

1. Decomposition

Consider the **relation R = {A, B, C, D, E, F, G, H, I, J}** and the set of functional dependencies

$$\begin{aligned} F = \{&\{A, B\} \rightarrow \{C\}, \\ &\{A\} \rightarrow \{D, E\}, \\ &\{B\} \rightarrow \{F\}, \\ &\{C\} \rightarrow \{B\}, \\ &\{F\} \rightarrow \{G, H\}, \\ &\{D\} \rightarrow \{I, J\}\} \end{aligned}$$

a. Decompose R to BCNF? (**Please begin with the decomposition into 3NF using the following solution as a starting point.**)

2NF:

$R1 = (A, D, E, I, J)$ Decomp around $A \rightarrow D, E$, Key A

$R2 = (B, F, G, H)$ Decomp around $B \rightarrow F$, Key B

$R3 = (A, B, C)$ Key A, B

3NF:

Decomp $R1$ around $D \rightarrow I, J$

$R11 = (A, D, E)$ Key A

$R12 = (D, I, J)$ Key D

Decomp $R2$ around $F \rightarrow G, H$

$R21 = (B, F)$ Key B

$R22 = (F, G, H)$ Key F

b. Is your decomposition lossless for BCNF? Please provide the steps to argue that why you decomposition is lossless or not.
c. Is it dependency preserving for BCNF? Please provide the steps to argue that why your decomposition is dependency preserving or not.

2. Multi Valued dependency

We store information about actors, their home addresses, and movies they have been in in a relation Actors(star_name, street, city, movie_title, movie_year). Assume that an actor can have a home in the same street address in two different cities.

The current state of the data is the following:

Carrie Fisher, High Road, Hollywood, Star Wars, 1977
Carrie Fisher, Park Dr, Malibu, Star Wars, 1977
Mark Hamill, High Road, Millers, Star Wars, 1977
Mark Hamill, Central Blvd, Palm Harbor, Star Wars, 1977
Mark Hamill, High Road, Millers, Star Wars, 1980
Mark Hamill, Central Blvd, Palm Harbor, Star Wars, 1980
Zoe Saldana, Research Blvd, Austin, Avatar, 2009
Zoe Saldana, Research Blvd, Austin, Avatar, 2022
Sigourney Weaver, Park Dr, Edison, Avatar, 2022

Consider the potential multivalued dependencies
star_name —>—> street, city and movie_title—>—> star_name.
Determine whether each MVD holds, may hold or does not hold and explain why.

3. NGOs

As we noticed, our NGO Database was suffering from redundancy as the same person would be inserted multiple times if they support multiple NGOs, and thereby we could potentially

have update and insertion anomalies. Therefore, we create two new relations, Supporter and Support.

```
CREATE TABLE `NGO`(
    `name`      VARCHAR(30)     NOT NULL    PRIMARY KEY,
    `based_in`  VARCHAR(30)     NOT NULL,
    `cause`     VARCHAR(40)     NOT NULL,
    `director`  VARCHAR(30),
    `phone`     CHAR(8)                     UNIQUE,
    `revenue`   INT
);

-- Create Supporter here
CREATE TABLE `Supporter`(
    `name`  VARCHAR(30)     NOT NULL,
    `email`     VARCHAR(50)         NOT NULL    CHECK(email LIKE '%_@_%._%') PRIMARY KEY,
    `phone`     CHAR(8) CHECK(`phone` REGEXP '[0-9]{8,}') UNIQUE,
    `address` VARCHAR(50) CHECK(`address` REGEXP '[0-9]+[ ]+[a-zA-Z ]+' OR `address` REGEXP '[a-zA-Z ]+[ ]+[0-9]+'),
    `zip_code` CHAR(4) CHECK(`zip_code` REGEXP '[0-9]{4,}'),
    `city`      VARCHAR(20),
    `birthday`  DATE    NOT NULL CHECK(`birthday` BETWEEN DATE '1922-01-01' AND '2002-01-01')
);
```

```
CREATE TABLE `Supports`(
    `ngo_name`  VARCHAR(30),
    `email` VARCHAR(50),
    `volunteer` BOOLEAN,
    `level` INT DEFAULT 0,
    FOREIGN KEY (ngo_name) REFERENCES NGO(name) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (email) REFERENCES Supporter(email) ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY(ngo_name, email)
);

CREATE TABLE `Donations`(
    `activity`  VARCHAR(30),
    `amount`    INT         NOT NULL,
    `date`      DATE,
    `email`     VARCHAR(50)  NOT NULL,
    `ngo_name`  VARCHAR(30) NOT NULL,
    FOREIGN KEY (email) REFERENCES Supporter(email)         ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (ngo_name) REFERENCES NGO(name)         ON UPDATE CASCADE ON DELETE CASCADE,
    PRIMARY KEY (email, ngo_name, date, amount)
);
```

a. Update

Update *level*, as the number of donations the supporter made to an NGO, for all entries in Supports using a single update command.

b. Triggers

Ensure that the 'level' of an NGO supporter is updated correspondingly whenever they make a donation to the NGO. Remember that level is defined to be the number of times a supporter has donated to the NGO. We assume that you cannot undo a donation, and therefore level cannot decrease.

c. Indexes

Create an index on Supports on the attribute ngo_name, so it is faster for an NGO to retrieve its supporters.

- What type of index will this be(primary or secondary)? if secondary, will it be dense, clustering or not?

- Let's make an assumption that this index is a 2-level clustered B-tree index; to find all the supporters of KDG, how many I/O operations would I need to do using this index? Describe what other assumptions you had to make to come up with this answer.

d. View

Suppose KDG has access to the information on their supporters and they would like to be able to share an overview of this data with others. Concretely, they want to show simple statistics of how many supporters they have at each level of support. Create a view that extracts this overview statistics.

4. Embedded databases

Using either Python or Java, as in week 6's exercises, connect to the NGO database defined by NGO_handin5.sql. Provide your code as part of your handin. (**Please also submit your source code file.**)

a. Change the program such that it finds the email of all the supporters who only support one NGO.

Hand in your answers in your group in a **single pdf file**. If possible, present your SQL code in a copyable format (then your TA can try it out), i.e. avoid screenshots. If you write in Latex then you can use the **verbatim** package.

We encourage you to **discuss any questions you may have in the discussion forum**, but **do not share solutions or solution attempts** in the forum.

Please note that late hand-ins will count **half** towards the grade.