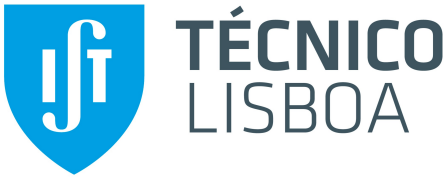

Instituto Superior Técnico

MEEC

Information Systems and Databases

SIBD Project - Part 2

Laboratory' shift: **b_SIBD77L03** — Lab 6 Monday (9:30 - 11:00)

Group 30:
Matilde Moreira, 84137—15hours—40%
João Cardoso, 84096—11hours—25%
Duarte Oliveira, 94192—12hours—35%

Laboratory Teacher:
Inês Filipe
Senior Lecturer:
Paulo Carreira

1^{st} *Semester*

2020-2021

1 Notes for the teacher

The creation of the database, which code is presented in file *schema.sql* was not only based on the original script of the project, but also on the model E-A given in the second part, making possible to add some cardinality restrictions and to decide the attributes and its type.

Initially, it was considered important to use the DROP TABLE command to wipe the database clean. The IF EXISTS constraint was passed as a command, in order to avoid generated errors, since if the tables did not exist at the base, it would not make sense to DROP them.

Subsequently, the CREATE TABLE command was used, in order, to create numerous tables. Within each command, we must define the field names according to the convenience of the database and determine the type of data that can be included in that same field. The fields correspond to attributes.

In order so this database can provide information, some actions were filtered in order to avoid the occurrence of possible errors, hence the need to define the integrity constants (IC's).

In the case of **IC-1** and **IC-2**, the solution found was to make the pair (id, voltage) of the bus_bar UNIQUE and finally use a FOREIGN KEY in the transformer table to pass the bus bar ids and voltages of the transformer, which have to be the same as the bus bar. **IC-3** and **IC-4** were resolved using the CHECK command of the two bus bar ids, so that they are different. **IC-5** is an integrity constant that cannot be solved simply with a FOREIGN KEY, NULL, UNIQUE or CHECK. Its solution is associated with the implementation of a TRIGGER. **IC-6** and **IC-7** were solved using the command UNIQUE.

For the definition of the tables, we tried to difficult the selection of the *queries*, founded at *queries.sql*, introducing allowed data, to also test limits. This data can be viewed in the *populate.sql* file.

Query A corresponds to the name listing of all analysts who analysed a particular element of the Power Grid, with a given ID. Knowing that the table of the association between the Analyst and the Incident comprises the PRIMARY KEYs of these 2 tables. The names can be obtained directly from the Association that comprises the relationship between (name, address) and (id, instant).

In relation to **Query B**, an analyst can analyse several incidents, so it is important to count the number of times an incident has occurred (count (id)). Subsequently, to know the name of the analyst who reported most often, creates a need to compare the number of incidents for each analyst with the others.

In **Query C**, it is asked to list all substations with more than one transformer. A substation can have more than one transformer, but a given transformer can only be in a given substation. To obtain all substations, whose number of transformers is greater than one, our solution consists of a grouping as substations by their geographical position and then counting the number of transformer ids associated with that same position. The condition is the number of transformer ids being greater than 1.

Finally, in **Query D** it is requested to find the names of the localities that have more substations than every other locality. The Substation table, in addition to containing information about the geographical position, also contains the name of the locality. There may be numerous substations with the same locality name, but the geographical position is unique, hence the need to group the names of localities, to then count the number of times that appears repeatedly. Subsequently, the location for which the number of associated substations is the maximum value is determined.