WEEKLY REPORT

April 30, 2019

ABDELMOUMENE Djahid
Université de Cergy-Pontoise

Contents

0.1 The objectives

The main objective of this project was to transform two datasets into a SQL database, so that their content can be easily exploited and used using the SQL operations.

These two datasets were unique because of their form, size and all the anomalies they contain, which had to be dealt with.

The two datasets are the EPMDataset with a size of 22MB (uncompressed) and containing 533 files. The other one was the OULAD dataset with a size of 442MB in 8 files

0.2 EPM Dataset

This dataset contains informations about 115 students over the course of 6 sessions on a simulation environment, as well as their final and intermediate grades.

0.2.1 The format

The data was disperesd over 533 csv (comma seperated values) and excel files, some of which were contained in different folders. The seperators for the csv files varied from semicolons to tabs. and some of excel files contained data in their column labels as well, which had to be scraped and formatted accordingly.

0.2.2 Data quality problems

There were many data quality issues that needed to be fixed in order for the data to be suitable for the transformation into a SQL database:

- **The sessions' data:** the data for the students' session was in a folder 'Processes' inside which there were other folders containing the files of each student, which had to be minimized into a single relation in the resulting database.
- **Null values:** some of the fields in the data contained null values, identified by a '?' character. These values had to be substituted into a null value.
- **The data in the grades' files columns:** some of the grade files contained data in the column names such as the max grade and the session and question numbers, these had to be inserted into the data itself.

- **multiple sheets in grade files:** some of the excel files contained multiple sheets, to fix this a new column (no_passage) was added to signify the sheet number, so that the data of the different sheets could be concatenated safely (eg. without losing any infomation).

0.2.3 Usage manual

To build the database's relation there is a SQL script named CREATE_TABLES.sql, which sould be executed in the desired database.

Afterwards a python script that generates another SQL file (INSERT_DATA.sql) should be executed as follows:

```
python gensql.py
```

And then the generated file sould be executed in the desired database.

To remove the relation along with their data, execute the DROP_TABLES.sql file.

Here is the UML diagram of the chosen database relations which was generated using SchemaSpy:

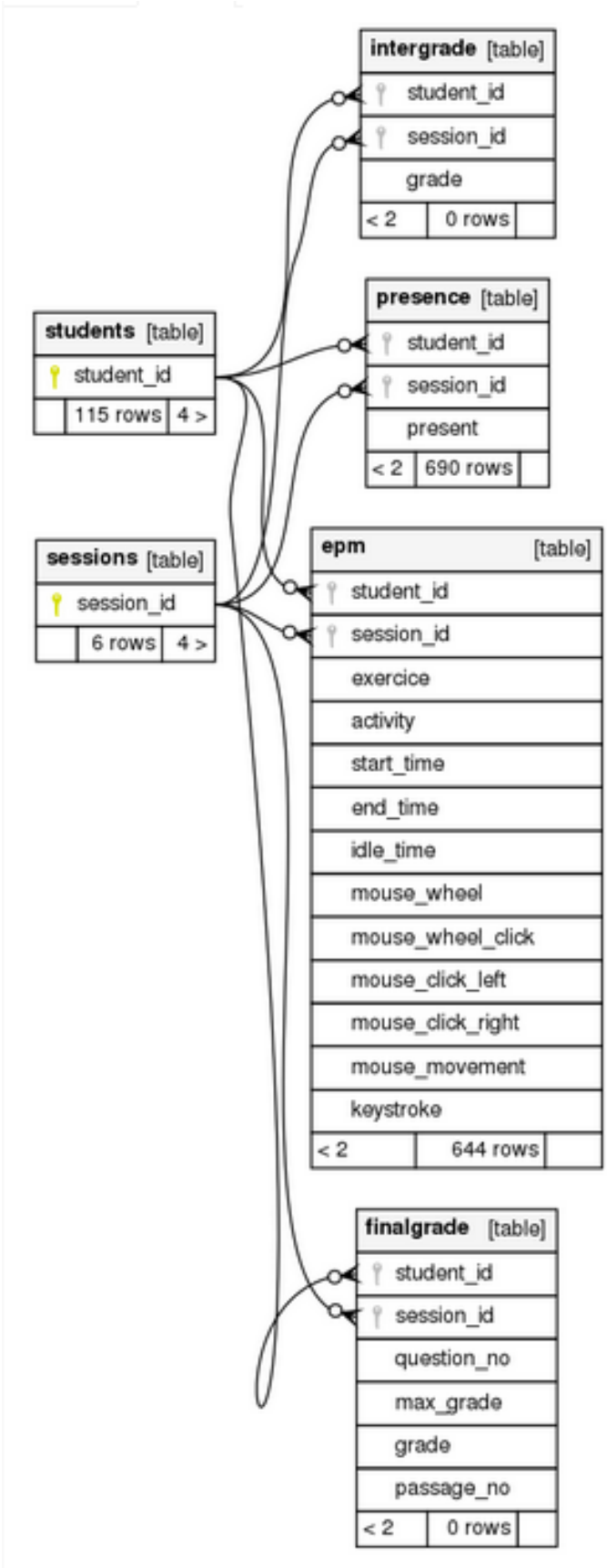
intergrade [table]
student_id
session_id
grade
< 2
0 rows
presence [table]
student_id
session_id
present
< 2
690 rows
students [table]
student_id
115 rows
4 >
sessions [table]
session_id
6 rows
4 >
epm [table]
student_id
session_id
exercice
activity
start_time
end_time
idle_time
mouse_wheel
mouse_wheel_click
mouse_click_left
mouse_click_right
mouse_movement
keystroke
< 2
644 rows
finalgrade [table]
student_id
session_id
question_no
max_grade
grade
passage_no
< 2
0 rows

Figure 1

0.3 OULAD Dataset

This dataset contains data about students and their interaction with another simulated environment, as well as some other useful information about them.

0.4 The format

The data was contained in 8 csv files which had for the most part consistent seperators and formats.

0.5 Data quality problems

There weren't as many problems for this dataset except for the size:

- **Null values:** some of the fields in the data had the same problem as the first dataset containing null values, identified by a '?' character. These values were substituted for null value.
- **the dataset size:** this dataset was massive and required a lot of processing time, mainly the studentVle table which contained over 10 million entries
- **Duplicates:** 2 million values in the studentVle table were duplicates.
- **id_student not completely unique:** The student ids in this dataset were not completely unique because student could attempt multiple times, this was solved by adding a new table for all the student ids to be referenced from.
- **The output sql files size:** The generated SQL files were too big, so they were split into batches of 1 million lines each (155MB)

0.5.1 Usage manual

To build the database's relation there is a SQL script named CREATE_TABLES.sql,which sould be executed in the desired database

To generate the data insertion scripts

```
python gensql.py
```

which will generate multiple SQL files which should be executed in this order:

```
INSERT_DATA_course.sql
INSERT_DATA_assessments.sql
INSERT_DATA_vle.sql
INSERT_DATA_students.sql
INSERT_DATA_studentInfo.sql
INSERT_DATA_studentAssessments.sql
INSERT_DATA_studentVle.sql
INSERT_DATA_studentVle_1.sql
INSERT_DATA_studentVle_2.sql
INSERT_DATA_studentVle_3.sql
INSERT_DATA_studentVle_4.sql
INSERT_DATA_studentVle_5.sql
INSERT_DATA_studentVle_6.sql
INSERT_DATA_studentVle_7.sql
INSERT_DATA_studentVle_8.sql
INSERT_DATA_studentVle_9.sql
INSERT_DATA_studentVle_10.sql
```

Here's the UML diagram of the resulting database:

0.6 Generating the diagrams

To generate the UML diagrams and other useful info SchemaSpy was used, which is a Java program that generates Schemas from the databases directly. To run it:

```
java -jar schemaspy-6.0.0.jar -t <type: pgsql for postgresql> -db <database name>
-host <host>:<port> -u postgres -p <password> -o ./schemaspy
-dp postgresql-42.2.5.jar -s public -noads
```

To view the resulting diagrams open the schemaspy/index.html with a web browser.

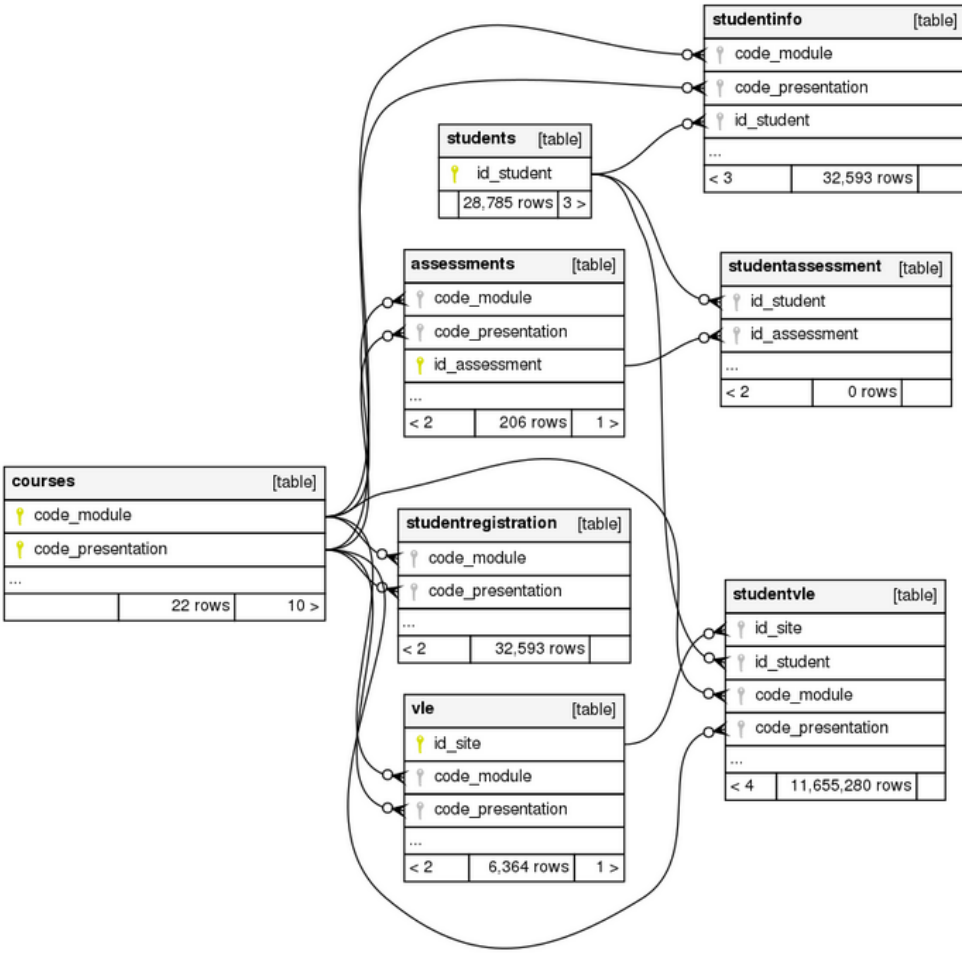
studentinfo [table]
code_module
code_presentation
id_student
...
< 3
32,593 rows
students [table]
id_student
28,785 rows
3 >
assessments [table]
code_module
code_presentation
id_assessment
...
< 2
206 rows
1 >
studentassessment [table]
id_student
id_assessment
...
< 2
0 rows
courses [table]
code_module
code_presentation
...
22 rows
10 >
studentregistration [table]
code_module
code_presentation
...
< 2
32,593 rows
studentvle [table]
id_site
id_student
code_module
code_presentation
...
< 4
11,655,280 rows
vle [table]
id_site
code_module
code_presentation
...
< 2
6,364 rows
1 >

Figure 2