University of Zurich[UZH]

Seminar DSID0002

Data Management & Integration

Ivo Bantel, Bingjie Cheng, Simon Jantschgi, Yu Zhang*

Handed in: 27.11.2020

The following documentation gives a brief overview of the data and the undertaken steps. Data and code are available here.

Process description

1. Schemes To start, a relational graph was created as an overview. For this, the entities involved in the four datasets were pinpointed, i.e. participants, questions from the questionnaire task, and tasks from each individual behavioural experiment. Attributes were further assigned to each entity and relationships among the entities were also determined. An ER schema was generated then using the above information. On the basis of the ER schema, the relational graph was generated using ERdplus software. See Figures 1 & 2 for the relational and the ER scheme.

2. Reading in data In the first step of hands-on work, we crawl through the folder structure the data comes in and create a dictionary of paths and files. We then read in the respective data files and combine them into a data frame (using python's `pandas` module; s. `1-Data-reading-manipulation.ipynb` in the GitHub repository for details).

3. Structuring and relating data We relate the different data frames by ensuring they each contain unique IDs identifying them and relating the respective cases according to the ER schema created above.

4. Export to SQL To make Python interact with the PostgreSQL database, we use two modules from python: `sqlalchemy` and `psycopg2`. The `create_engine()` call from `sqlalchemy` recalls an engine configuration. Then we use `Engine` to provide access to a `Connection`, which can then invoke SQL statements (with `engine.connect()` as connection). When python and the SQL database are connected, we use the command: `pandas.DataFrame.to_sql()` to export the Python data to SQL.

*Alphabetical order; all contributed equally.

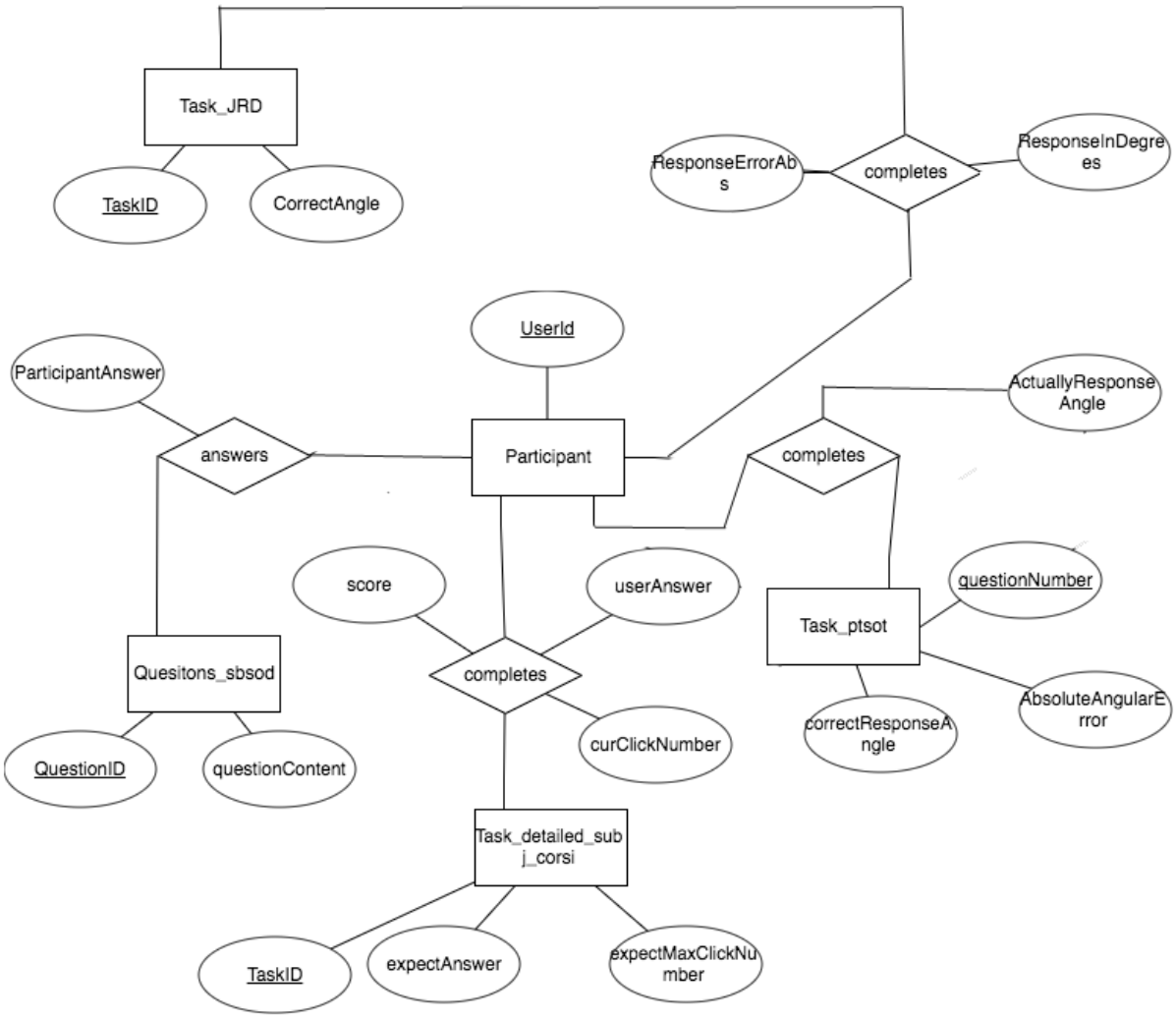

Figure 1: ER scheme

Data description

To give a better impression of the used data, we briefly detail the the four underlying, logically connected data sets below. Unique IDs are underlined.

`detailed_subj_xx`

- UserId: participant ID
- Correct: 1 = correct, 0 = wrong
- userAnswer: the answer user provided (the numbers indicates the labels of the squares)
- expectAnswer: the answer that is expected from the user (correct answer) (the numbers indicates the labels of the squares)
- curClickNumber: the current number of the squares that user have clicked
- expectMaxClickNumber: the max. number of the squares that user should click

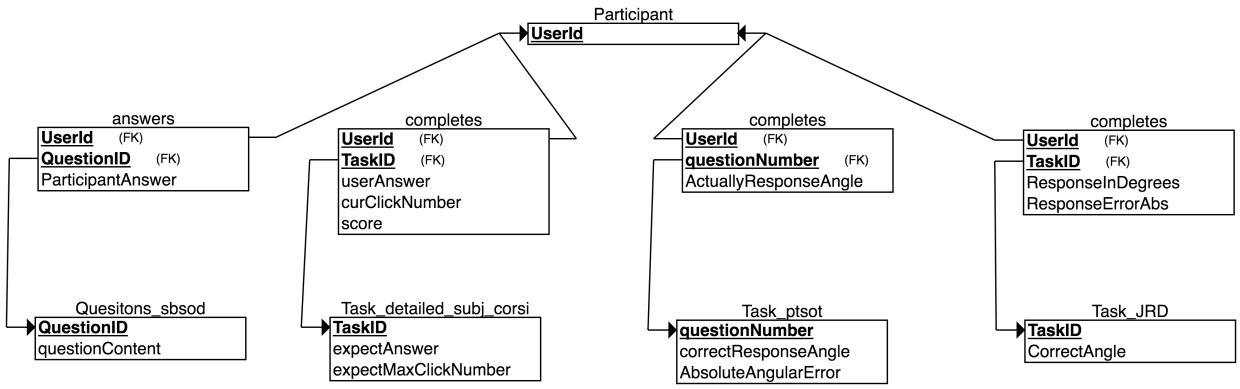

Figure 2: Relational scheme

`ptsot_results`

- participant ID: ID of participant; stored in the file name, not in the file itself
- questionNumber: number of question
- correctResponseAngle: correct response to question about angle
- ActuallyResponseAngle: given response to question about angle
- AbsoluteAngularError: difference between correct and given response

`sbsod`

- participant ID: ID of participant; stored in the file name, not in the file itself
- QuestionID: the ID of the questions. In total 15 questions
- questionContent: the question presented to the participant
- ParticipantAnswer: answer given, on a 1-7 likert scale

`UnityDataSave` **— incl.** `JDR` **and** `RouteTest`

- partID: ID of participant; stored in the file name, not in the file itself
- City_mapLMs, StandingAt, LookingAt, LandmarkToMove: content of the questions
- CorrectAngle: the correct angle (answer that is expected from the user)
- ResponseInDegrees: participants' answer
- ResponseErrorAbs: the absolute error between the correct angle and participants' answer