Project Title:
INFRASTRUCTURE AND INTEGRATED TOOLS FOR PERSONALIZED LEARNING OF READING SKILL

Project Acronym:

D4.5 – Learning Analytics

Document identifier	**iRead_D4.5_Learning_Analytics_v06**
Date	**2020-01-0810**
WP	**WP-44**
Partners	**UCL, NTUA**
WP Lead Partner	**UCL**
Document status	**V07**

Grant Agreement number:
731724 — iRead H2020-ICT-2016-2017/H2020-ICT-2016-1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 731724

Deliverable Number	D4.5
Deliverable Title	Learning Analytics
Deliverable version number	v07
Work package	WP-4
Task	T4.6 Learning Analytics
Nature of the deliverable	Report (R)
Dissemination level	Public
Date of Version	2020-01-08

Author(s)	Dionysis Panagiotopoulos (NTUA), Yvonne Vezzoli (UCL)
Contributor(s)	Antonios Symvonis (NTUA), Manolis Mavrikis (UCL)
Reviewer(s)	Asimina Vasalou (UCL), Ioan Mihu (ULBS)
Abstract	In this deliverable, we describe how data collected during the operation of the iRead system are utilized to support learning analytic. Stored data can support the teacher in class by providing information on the playing/reading history of individual students (“activity reporting”) as well as the researcher who engages in explorative analysis. The data organization, the supported visualizations and the utilized tools are described in detail.
Keywords	Learning analytics, activity reporting, explorative analysis, profile visualization, log browser, data organization, Tableau software

Document Status Sheet

Issue	Date	Comment	Author
V0.1	2019-06-03	Initial outline/draft	D. Panagiotopoulos, A. Symvonis
V0.2	2019-06-10	Data Analysis in iRead, Data organization	D. Panagiotopoulos
V0.3	2019-06-17	Introduction, Requirements Analysis	M. Mavrikis, Y. Vezzoli
V0.4	2019-06-19	User activity reporting, Explorative analysis	D. Panagiotopoulos
V0.5	2019-06-21	Final pass of document; Ready to send for review	A. Symvonis
final	2019-06-30	Incorporating reviewer's comments	A. Symvonis
V0.6	2019-12-10	Update database structure, Profile Viewer, Logger Browser, Tableau tools	D. Panagiotopoulos
V0.7	2020-01-08	Final pass of document; Ready to send for review	A. Symvonis

Table of content

1 EXECUTIVE SUMMARY

In this deliverable we describe what data are collected during the operation of the iRead system, how they are organized, and how they can be utilized in order to support the needs of the iRead project.

Data in the iRead system consist of user persona information, user profile information (in the form of an instantiation of a language domain model), detailed user actions as a result of user engagement in some application (Navigo games or Amigo Reader), and iRead system actions. All these data reside in iRead's infrastructure, either in the user management and profile management components or (in the form of log entries) in the iRead's logger which is implemented as an Elasticsearch database.

Data stored/collected in the iRead system are used in order to fulfill two central objectives: Firstly, to support the teachers in their everyday engagement with the iRead system during its evaluation through the large scale pilots and, secondly, to support the data analysis phase of the project evaluation as well as to facilitate any explorative analysis efforts by our team researchers (mainly educational and linguistic partners).

Teacher support is provided through a developed "*Teacher Tool*" web application which consists of the "*profile viewer*" and the "*log browser*" tools. Both tools draw their data directly from iRead's Elasticsearch data logger.

Supporting data analysis for iRead's project evaluation phase as well as supporting explorative analysis is slightly more complicated. The complication comes from the facts that i) data concerning user actions must be first processed before they become suitable for analysis, and ii) the analysis must combine data concerning all the users of the iRead system. We have developed an Analytics System which is external to the iRead system and has a PostgreSQL database at its core. The analytics systems is design to draw in a timely fashion data from iRead's Elasticsearch logger, to process them and to insert them in to its analytics database. The full collection of data can become available to researcher through the use of two additional external analytics tools: Tableau Prep which brings the data to suitable form for analysis, and Tableau Desktop which can be used to visually present the results of the final data analysis as well as to support explorative analysis. Note that the Analytics System also include services to guarantee proper user deletion/anonymization, increasing in this way the safeguarding of personal data.

2 Introduction

Learning Analytics (LA) have been defined as the "measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs" (Siemens, 2011). The development of such tools has followed from the growing recognition that learning with technology is most likely to be effectively used when accompanied by appropriate teaching pedagogies and scaffolding (Sharma and Hannafin, 2007). Accordingly, it has been argued that the data provided through LA tools can raise teachers' awareness and reflection, informing their decision-making and consequent behaviors in the classroom (Holstain et al., 2017; Vatrapu et al., 2017; Verbert et al., 2013). LA tools typically allow the tracking of students' progress, the class' workflow and support class management (Martinez-Maldonado et al., 2017).

Research on LA has chiefly focused on enhancing the effectiveness of teachers' orchestration of the classroom and its inevitable variability and complexity (Dillenbourg et al., 2018; Prieto et al., 2011). For example, Martinez-Maldonado et al. (2017), draws a clear connection with orchestration, conceptualizing learning analytics as tools that can be used to address practical orchestration challenges in classrooms such as managing, adapting, scaffolding and assessing learning activities (see also Prieto et al., 2015).

It is worth recognizing that learning analytics tools are aimed at a wide range of stakeholders on both individual and organizational level (Chatti et al., 2013). Learning analytics stakeholders can be learners, instructors, teachers, trainers, administrators, developers, researchers, parents, and government bodies or other policy makers (Khalil and Ebner, 2015). In the context of iRead potential stakeholders are not only the teachers that are directly using the iRead technology in the classroom but also others who work in the educational sector e.g. holding a managerial position at a school and/or can use the outcome of this project to the benefit of the school or also the research and development team that may use learning analytics outcomes for research purposes or for further improvement of the iRead technologies.

As such, our focus on iRead has been twofold. First to design analytics with a focus on teachers' needs (Holstein et al., 2017; Dillenbourg, 2010) and second to design the infrastructure in a way that support another goal of learning analytics – that of helping stakeholders (mostly researcher and other educators) answer research questions or other inquiries that may help their decision making as discussed in the definition above. We will refer to the outcomes of these two goals separately. First, as a teacher "dashboard" i.e. a visual display that presents different indicators "about learner(s), learning process(es) and/or learning context(s) into one or multiple visualizations" (Schwendimann et al., 2017). For the second purpose, we will talk more generally as "learning analytics" to any other data analysis that can be performed beyond what is displayed in the dashboard.

3 Requirements analysis

3.1 Introduction

Designing learning analytics tools is not a straightforward process. A strong seam of research has begun to theorize the potential pedagogical activities that LA may support in classroom learning. Much of this research has engaged in theoretical work through the lens of education theories and existing concepts of educational practice (e.g. Martinez-Maldonado et al., 2017; Prieto-Alvarez et al., 2017). Whereas theoretical perspectives can provide input into how LA can support teaching activities for example, they cannot always capture the complex realities of education practitioners, their visions of learning and teaching. As such design-oriented approaches have started to emerge, where teachers and designers alike work together to uncover the new opportunities of LA (Holstain et al., 2017; Prieto-Alvarez et al., 2017).

Despite this recognition, researchers in the field of learning analytics (LA), usually involve teachers later in the process, for instance exploring how existing LA tools can be effectively used (e.g. Dollinger and Lodge, 2018; Mavrikis et al., 2016) or gathering their feedback on existing LA tools (c.f., Liaquat et al., 2018, Prieto-Alvarez et al., 2017). However, these can be viewed as an instance of the teacher adapting to the technology, rather than the other way around (Xhakaj et al., 2016; Dillenburg and Jermann, 2010).

A recent design model seeking to address this gap is the LATUX workflow (Martinez-Maldonado et al., 2015). In developing LA tools for educational contexts Martinez-Maldonado et al recognize the importance of “carrying out research to explore the new possibilities that learner data can offer for supporting instructors”. The LATUX workflow describes how an iterative user-centered design process is instantiated in the case of LA. The model identifies an initial problem identification phase for LA, followed by a series of iterative formative evaluation stages. The key challenge is to recognize teachers’ lack of experience in orchestrating learning with technology, let alone the lack of past engagement with learning analytics tools and, in some cases, lack of data literacy skills and understanding of the potential of LA in order to meaningfully provide requirements.

A systematic review by Nye (2014) showed how systems with higher adoption (e.g., Cognitive Tutor, ASSISTments, and ALEKS) tend to have some form of either monitoring or customization tools for teachers. Nye speculated that under consideration of teachers’ needs may be a significant factor affecting adoption and attrition. Building upon this idea, recent work by Holstein et al. (2017) has explored the potential of design methods while involving teachers in the early stages of technology design with the aim to design a dashboard that answers real-time teachers’ needs in the context of an intelligent-tutoring system (ITS).

Inspired by this perspective, in this WP we are following an analytics cycle that involves an early investigation of teachers’ requirements to inform system design. In what follows, we describe the methodology used to understand teachers’ needs. Following this, we briefly outline some of the key findings from this process and the rest of the deliverable focuses on the outcomes in terms of the implementation of these decisions.

3.2 Methodology

Taking a participatory design (PD) approach, we facilitated three workshops of around 1h15mins each, which were structured around three phases informed by Hornecker et al. (2006) i.e. 1. *exploring existing practices*, 2. *familiarization with technology* and 3. *exploring new possibilities*. Alongside our aspiration to promote the development of new scenarios through probing the 'whys' of particular pedagogical data uses, we faced the challenge of supporting teachers' understanding of a potentially complex design space characterized by different reading phases, learning outcomes, possible data, and so on.
The Inspiration Card Workshop method developed by Halskov and Dalsgård (2006) was employed to address both requirements. In particular, we were drawn to this method because of its tangible and material form, and its potential to scaffold conversations that were closely coupled with what technology could achieve. An Inspiration Card is an index card presenting a space for a title, image and description. While Inspiration Cards were conceived as a method for creating new opportunities that may transcend current practice, aligning with our aim, the original authors of this method explain that participants' prior knowledge, experience, and practices shape the workshop in different ways (Halskov and Dalsgård, 2006), in turn leading to different outcomes.

Because the iRead target users include primary school teachers, special education needs (SEN) teachers and SEN coordinators, we involved a set of participants from different contexts, who were supporting students with diverse literacy needs. In all three sessions participation was voluntary and all of the participants provided informed consent. Besides the practitioner-participants, each of the sessions additionally included two researcher-participants, one with expertise in literacy, and one researcher with knowledge of LA. Table 3-1 provides details of these sessions.

Session 1 4 practitioner- 2 researcher- participants	Participants were four SEN and personalised learning experts, i.e., two ex-teachers who now delivered professional teacher training for dyslexia, one teacher-student participating in the dyslexia training and one speech-language therapist
Session 2 2 practitioner- 2 researcher- participants	Participants included a teacher working on literacy in Year 2, and the head-teacher who had a literacy background and taught literacy sessions occasionally. These two literacy practitioners were based in a school with a strong track record in supporting students who struggled with literacy
Session 3 4 practitioner- 2 researcher- participants	Participants comprised a Year 1 literacy teacher, a Year 2 teaching assistant, a Year 3 teacher and the deputy head-teacher who was also the SEN coordinator. These practitioners were based in a school adopting holistic approaches to literacy through developing their own curriculum

Table 3-1: Participatory Design Workshops towards the design of the iRead Learning Analytics tools

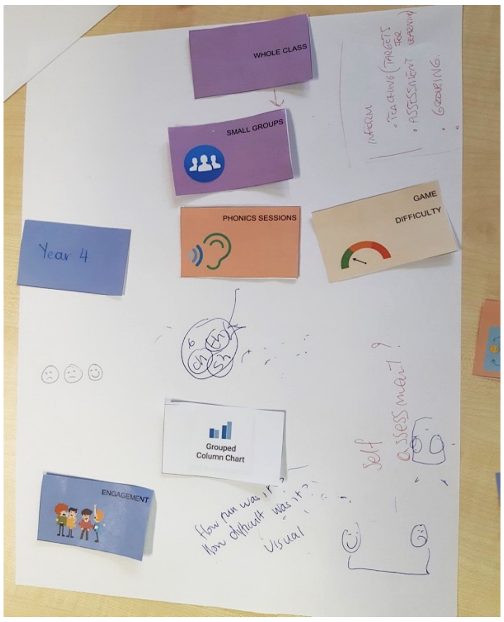


Figure 3-1: A scenario constructed with the Inspiration Cards

During each workshop, we displayed targetted inspiration cards while describing their meaning and answering possible questions raised by participants. This allowed researchers and participants to negotiate a shared interpretation of each card. Following the introduction of the cards, we invited participants to combine them in order to co-construct scenarios. In total this generated 14 new scenarios across the three workshops. Figure 3-1 shows a combination of cards that formed a scenario in one of the sessions where the teachers involved envisioned collecting data at a whole class level to inform the later organisation of smaller groups.

3.3 Summary of findings

Earlier we argued that a strong seam of research in LA has conceptualized this technology as one that supports orchestration in the classroom (Dillenbourg and Jermann, 2010). Our findings align with this research showing that teachers frame LA around design, planning and assessment activities that would complement their teaching (Dillenbourg and Jermann, 2010). Highlights include the contextualization of LA by teachers in relation to the challenges of the teaching profession, and in particular the limited time and increased workload issues they faced. LA were therefore seen to ameliorate these problems by suggesting pedagogical actions that would have the highest impact, either by affecting the most students, or those students who were at the highest risk. In addition, teachers show LA tools as a way to enhance existing practices, for example, specific teaching methods like reciprocal reading where students work together as a group predicting, classifying, questioning and summarizing a reading (for more details see Takala, 2006). Participant described the potential of the games and data to monitor and provide evidence-based awareness on the outcomes of the reciprocal reading sessions.

Overall, emerging from our teacher workshops was the following three design principles for iRead:

1) Prioritize data and visualizations that have impact on teacher actionable choices;
2) Design for appropriation i.e. have the interface and any visualizations structured but also open enough to be able to be configured to answer specific questions over a period of time and targeted to specific parts of the domain;
3) Consider analytics tools as an opportunity for teacher reflection, design and redesign of activities and eventually professional development.

Table 3-2 and Table 3-3: Indicative usage queries of learning analytics tools as emerging from the PD workshops identifies illustrative examples of usage scenarios of the tools with the specific propositions that participants made while creating their scenarios.

#	What	Why	Pedagogical aim
1	Students struggling to achieve or performing well on a particular learning objective	Because of the lack of time to explore individual performance	To plan further learning activities on the main struggles at a classroom level or to avoid some activities and plan what objectives to target at a classroom level.
2	Visualization of trends within the class	To have an overview without consulting the individual performance	To decide if and when to move to the next learning objective at a classroom level
3	Visualization of specific features a child struggle with	To simplify the complex diagnosis process for a child with possible SEN	To target the specific features while the child progress with the game
4	Identification of a student who is particularly struggling and representation of the specific features the student struggle with	To speed up the time-consuming and stressful identification of struggling readers	To plan individualized support
5	Insights on how long struggling readers play	Because the intense work on fluency can impact students' concentration and engagement	To know about their concentration and engagement
6	Insights on the child's progress in a particular learning objective during an individualized intervention	To know the child's improvement on a lacking area identified before through traditional tests	To assess struggling readers' progress on a gap at an individual level
7	Report of students' performance in comprehension tasks	To be aware of their possible improvement during reciprocal reading sessions	To assess comprehension
8	Insights of the effects of the teaching methods used on students' learning	To reflect on the learning design and methods used with students	Teaching self-assessment and planning of future activities accordingly
9	Struggling readers' correct answers and progress against themselves on a learning objective	To know about struggling readers' strengths and learning achievements	To assess struggling readers' strengths and improvements against themselves
10	Time spent to complete particular combination of games and features	To re-challenge students individually or in competitions or simply to know about their performance for planning purposes	To enhance students' motivation and reinforce their learning

Table 3-2: Indicative usage scenarios of learning analytics tools as emerging from the PD workshops

#	Query
11	What kind of errors do students do more ('gaps' in features)?
12	Where the errors/gaps are in their learning path (e.g. are students doing well with the prerequisites of a certain feature f but they are still encountering specific difficulties with this feature)?
13	Which category of games are the most difficult (i.e. student make more errors in them e.g. most of the mistakes could be in the building games)?
14	Teachers talk about 'engagement' but it is not clear how to define this for iRead. Most likely some combination of time played and number of games and time 'on task'.

Table 3-3: Indicative usage queries of learning analytics tools as emerging from the PD workshops

4 Data analysis in iRead

4.1 Goals

Our goal is twofold: Firstly to provide all the necessary information to the teachers about the progress of their students and, secondly, to provide researchers with the means to evaluate impact of the iRead system to the users.

The iRead provides a variety of tools to teachers. These tools are accessible through a web application known as the *"Teacher Tool"*. Of particular interest is the *"activity reporting"* toolbox.

The *activity reporting* tools derive data in real time from the logger (elasticsearch[1]) database directly. Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. The available tools, in the activity reporting toolbox are the:

- The *Profile viewer*: The *profile viewer* displays a visualization of the user model, including its organization in linguistic levels, categories and features.
- The *Log browser*: The log browser provided an easy way to explore the recent actions of a user as they are recorded in the system log.

Explorative analysis is supported by a set of methods which retrieve data in regular time intervals from the logger, analyze and save them to a separate and independent database. The data in this new database can be used to answer a large set of questions related to the total performance of the students and to, eventually, assess the impact of the iRead system.

4.2 Data available in the iRead system

Each user, from the moment of the registration in the iRead, provides a set of personal information and by playing or reading, through the iRead applications (NAVIGO mini games, AMIGO e-Reader), she produces data related to her actions. For each registered student, iRead records the following information attributes:

- First name
- Last name
- e-mail (It could be an e-mail of her parent/guardian)
- Birthday
- Gender
- Country of residence
- School city
- School Name
- Languages spoken

[1] https://www.elastic.co/products/elasticsearch

Except of the personal information, we track the activity of a user, by keeping logs for her performance. This information refers to:

- Application usage (games, readers, web applications)
- Game report (from NAVIGO mini games application)
- Reader report (from AMIGO e-Reader)

For each application (web applications, game application etc.) we creates logs each time we use it (who used it, start time, finish time) so that we can gain insights about application usage. For each game activity, we track all user actions. We keep data about a) game actions/choices, b) time taken to make an action/choice, c) info regarding any help/feedback provided by the system and d) the final game outcome. For each reading session we keep info about the books assigned to (read by) the user and the issues she focused while reading.

Furthermore, we also log some of the system activities relevant to each user. Such activities concern the attempted change/reevaluation of the user competence of a specific feature in the user's profile (profile is the instantiation of the domain model for the specific user).

The logged data can be used to actually replicate the evolution of a specific user in iRead through time. So, it is conceptually possible to study and draw conclusions on user learning to read, to answer specific questions formulated by researchers, or to simply explore them hoping to identify hidden trends and to test various hypotheses.

4.3 Data organization

Elasticsearch database (online): Support for activity reporting

All logs generated by iRead and the associated applications are recorded in iRead's online logger database. The logs are raw, i.e., they are stored exactly as they are generated; no further processing is performed; In particular, for each game activity the log contains a "block of text" which collectively describes the whole game activity; no effort is made to break/parse it into individual actions. A sample game log is provided in Appendix I.

iRead's logger is an Elasticsearch database, which supplies us with tools to easily and quickly search the whole log collection. One of the most powerful tools that the Elasticsearch provides, is the ability to perform search queries for attributes stored deeply in the log structure. Real time user activity reporting is supported by the Elasticsearch database.

PostgreSQL database (offline): Support for data analytics

In order to meaningfully support explorative analysis, we need to combine information related to the users and their actions as well as to the domain model relevant to them. Moreover, the information related to their actions has to be parsed so that each unit action is distinguishable.

Towards this end, we employ a PostgrsSQL[2] database where we store structured information obtained by processing all logs initially recorded in the iRead's Elasticsearch database together with specific language domain information. During the evaluation of the iRead pilots, the PostgreSQL database will be daily populated with data extracted (after appropriate processing) form Elasticsearch's logs. Especially, for NAVIGO activity logs, the following unit/low-level information will be recorded:

- Which user played and when
- Which questions she had to answer and which words/sentences were involved in the activity (i.e., the exact educational content of the activity)
- Which events happened (correct answers, incorrect answers, requests for help, game feedback) and to which content elements (words/sentences) these events correspond to.

The collection of data allows to search for behavior patterns, like preference to particular games (as evidenced by success ratio), or to reveal effective (with respect to learning) sequences of language feature acquisition.

The PostgreSQL database can export/provide data to other applications, so researchers can perform explorative analysis using their favorite tools. In order to fulfill requirement related to the handling of personal data information, the data extraction supports data anonymization.

4.4 Data recording points

As we have already mentioned, we track the activity of a user through all her stay in iRead and the supported applications. We add a log for each of the following events:

- A user registers in the iRead system.
- A user requests to delete/anonymize her account from the iRead system.
- The account of user has been deleted/anonymized.
- A user changes her account's personal information.
- A user logs into an application.
- A user logs out from an application.
- A student completes a NAVIGO mini game.
- A reevaluation of a user's profile takes place.
- A student opens a book in the AMIGO e-Reader.
- An activity assignment takes place (initiated by a teacher).
- A book assignment takes place (initiated by a teacher).
- A teacher uses the profile viewer from the teacher tool web application
- A teacher performs admin actions of her class (adds/removes students)

The majority of these logs are recoding single events that don't need further analysis. Only the logs generated by NAVIGO game and the AMIGO e-Reader require further processing/parsing.

[2] https://www.postgresql.org/

4.5 A note on performance

The time required to answer a question based on data stored in a database depends on the complexity of the question and the amount of entities (users) involved. The need to relate the retrieved data with other data external to the log database (i.e., information about the user's profile) may deteriorate performance. In particular, replying to a question related to a single student requires making service calls to access a) the user's personal information, b) to access the user's profile, and c) to access all log records related to a user; with each call typically taking a few seconds to provide its reply. Replying to the same question for a group of students (i.e., a class) takes time proportional to the number of students in the group, which may be prohibited for an on-line system.

Due to these considerations, our real-time activity reporting toolbox supports single student queries. *Class-queries* can be answered by accessing data in the off-line PostgreSQL database where the more powerful relational database engine is evolved.

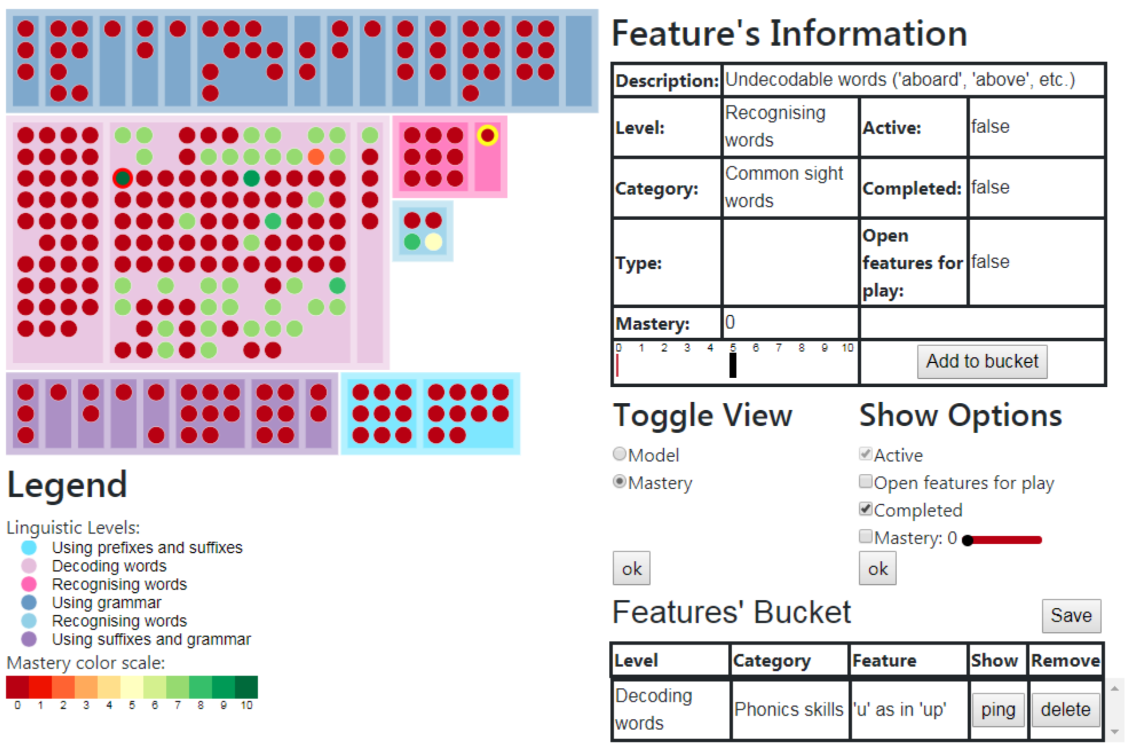
Consequently, questions #3, #5, #6, #9 and #10 in Table 3-2 can be answered in real time by accessing data the Elasticsearch database, while questions #1, #2, and #4 can be answered by referencing to the ProstgreSQL database. We note that it is not possible to answer questions #7 and #8 as they refer to aspects not covered ("comprehension task") or not modelled ("teaching method") by iRead. Similarly, the queries reported in Table 3-3 can be also answered by using the ProstgreSQL database in combination with explorative analysis tools.

5 User activity reporting

User activity reporting is available through the “teacher tool” web application and consists of *profile viewer* and the *log browser* tools. In this section, we provide an extensive description these tools.

5.1 Profile Viewer “REVISED”

The *profile Viewer* allows to a teacher to easily inspect the profile status of one of her students. The tool is composed of five parts; the graphical display of the profile, its legend, the feature information table, the view options and a table with teacher selected features.



Feature's Information

Description:	Undecodable words ('aboard', 'above', etc.)		
Level:	Recognising words	Active:	false
Category:	Common sight words	Completed:	false
Type:		Open features for play:	false
Mastery:	0		
0 1 2 3 4 5 6 7 8 9 10		Add to bucket	

Toggle View

Model

● Mastery

ok

Show Options

✓ Active

Open features for play

✓ Completed

Mastery: 0

ok

Features' Bucket

Save

Level	Category	Feature	Show	Remove
Decoding words	Phonics skills	'u' as in 'up'	ping	delete

Legend

Linguistic Levels:
- ● Using prefixes and suffixes
- ● Decoding words
- ● Recognising words
- ● Using grammar
- ● Recognising words
- ● Using suffixes and grammar

Mastery color scale:

0 1 2 3 4 5 6 7 8 9 10

Figure 5-1: Overview of the *Profile Viewer*

The graphical display at the default view shows all the features of the profile. Circles represent features which are grouped within rectangles representing categories. Rectangles with the same color belong to the same linguistic level. The legend of the graphical view informs the teacher which color correspond to which linguistic level. Each feature has been drawn with a color which corresponds to its mastery (at the scale zero to ten, as it is shown in the legend). By changing the view options (“toggle view” and “show options”) different sets of features remains visible or different sets of features are highlighted. The two *toggle view* options are “model” and “mastery”. In “mastery” mode, the mastery of each feature is shown by appropriately coloring the corresponding circles, while in “model” mode features are uniformly colored as this mode is intended for obtaining information about the domain model. The *show options* allows the user to filter the features with the given attributes. The default view is shown in

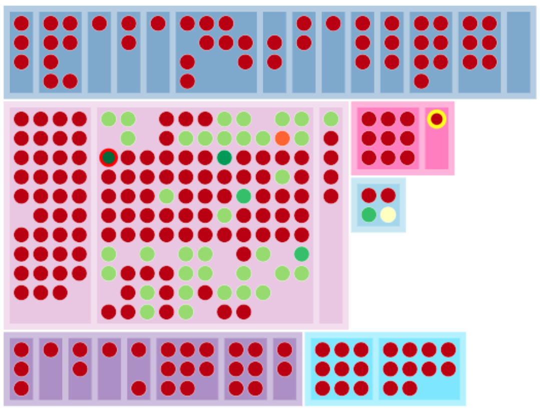
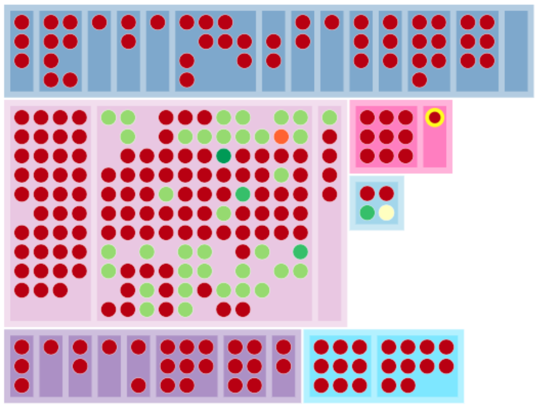


Figure 5-2, which displays all the features; active, inactive, completed (a feature is *completed* if the user has reached maximum mastery). The *active* features are always shown; the user has the option to hide only the



completed features (Figure 5-3). The teacher can also select to highlight only the features which are available for selection for future practice. These features have a thick blue border (see

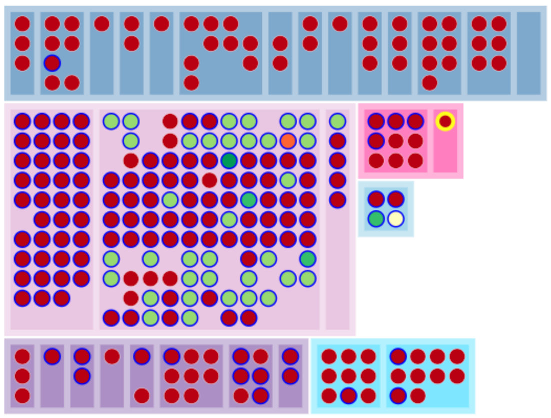


Figure 5-4). As *available feature* the system considers every feature which is *active but not completed*. An extra filter tool, which can

be used by the teacher, is to keep only features that correspond to a specific mastery level (see

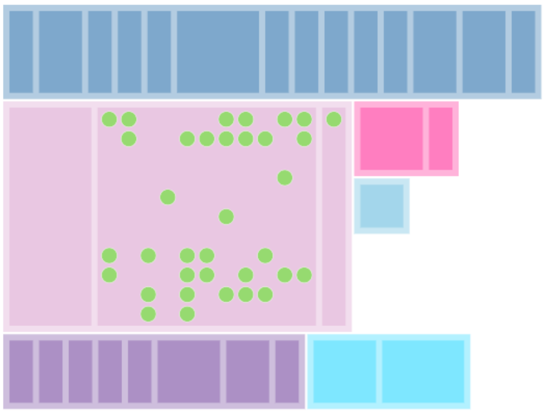


Figure 5-5).

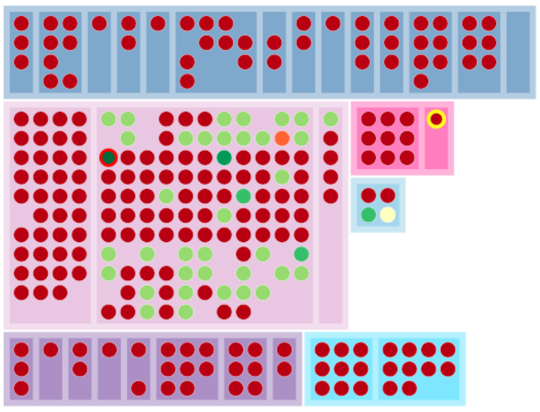


Figure 5-2: Default view

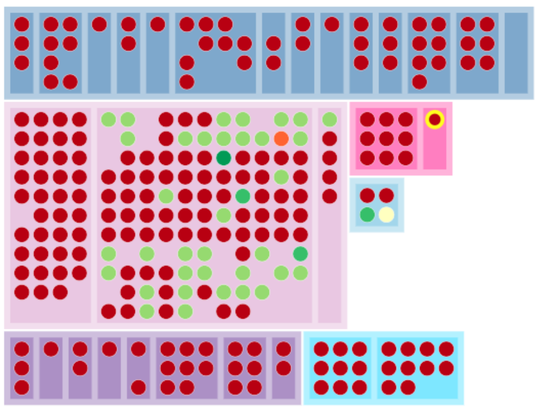


Figure 5-3: View which excludes completed features

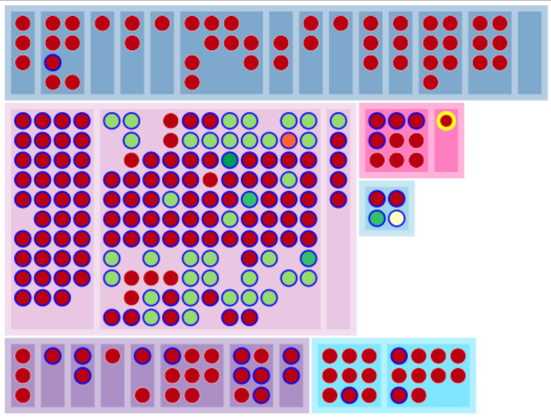


Figure 5-4: View which shows the available features highlighted

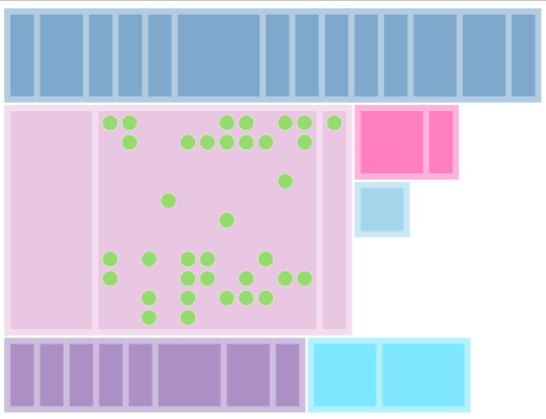
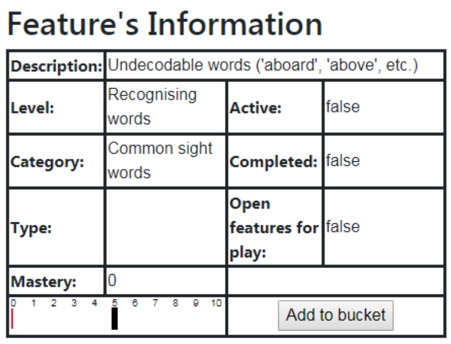


Figure 5-5: View which keep features with a specific mastery level

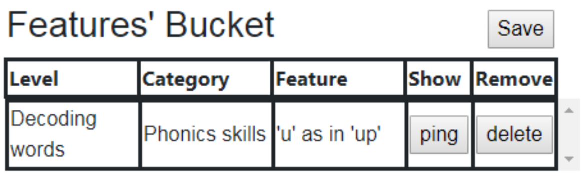
The teacher can select any feature and see all its information at the *feature's information* table. For each feature, the table contains its description, the linguistic level, the category and the type in which it belongs to. In addition, it displays the mastery level of the user, both as a number and as colored bar, which also has a black line which indicates the *threshold* of the feature. *Threshold* is the mastery limit which, when the user overcomes it, she is considered to have reached a mastery level sufficient to allow her to continue practicing with features which have this feature as a prerequisite. Additional information regarding whether the feature is active, completed or available for selection is also shown. Finally, the teacher can build a list of selected features which will then appear in red color in the graphical view. The teacher has the ability to save this list for the specific student, so each time she uses the *profile viewer* this list is prepopulated with her preferences.



Feature's Information

Description:	Undecodable words ('aboard', 'above', etc.)		
Level:	Recognising words	**Active:**	false
Category:	Common sight words	**Completed:**	false
Type:		**Open features for play:**	false
Mastery:	0		
0 1 2 3 4 5 6 7 8 9 10		Add to bucket	

Figure 5-6: The feature information table



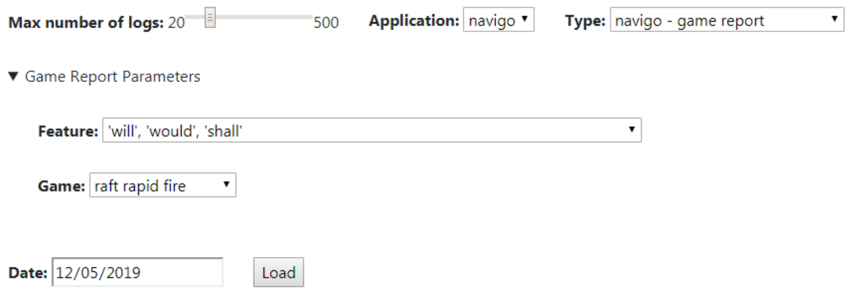
Features' Bucket Save

Level	Category	Feature	Show	Remove
Decoding words	Phonics skills	'u' as in 'up'	ping	delete

Figure 5-7: The selected feature list

5.2 Log browser *"REVISED"*

The *log browser* allows to a teacher to retrieve logs for one of her students from the Elasticsearch database. In this way, the teacher can have a quick, but quite informative glimpse over the student's recent activity. Logs can be filtered based on their *type* (*login*, *logout*, *gameplay* etc.), or by *feature*, *game* (for gameplay logs), or by time (selecting events that happened in a specific time interval) see Figure 5-8.



Max number of logs: 20 500 **Application:** navigo **Type:** navigo - game report

▼ Game Report Parameters

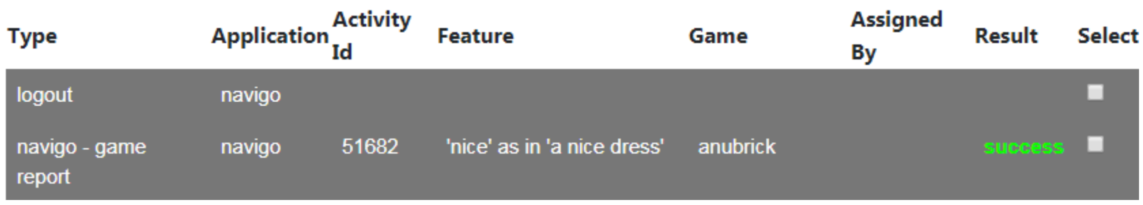
Feature: 'will', 'would', 'shall'

Game: raft rapid fire

Date: 12/05/2019 Load

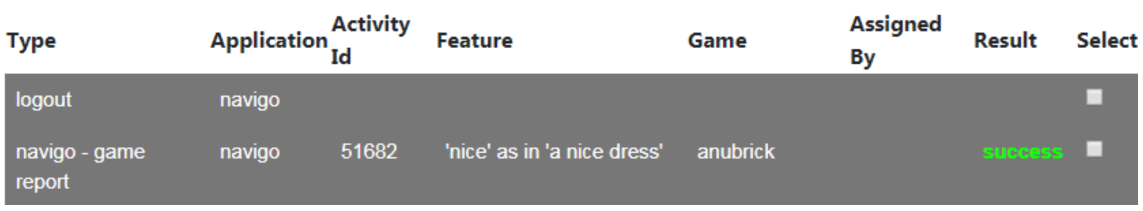
Figure 5-8: The search filters

The output of any filtering query to the Elasticsearch database is typically a set of log entries that have to be displayed.



Type	Application	Activity Id	Feature	Game	Assigned By	Result	Select
logout	navigo						☐
navigo - game report	navigo	51682	'nice' as in 'a nice dress'	anubrick		success	☐

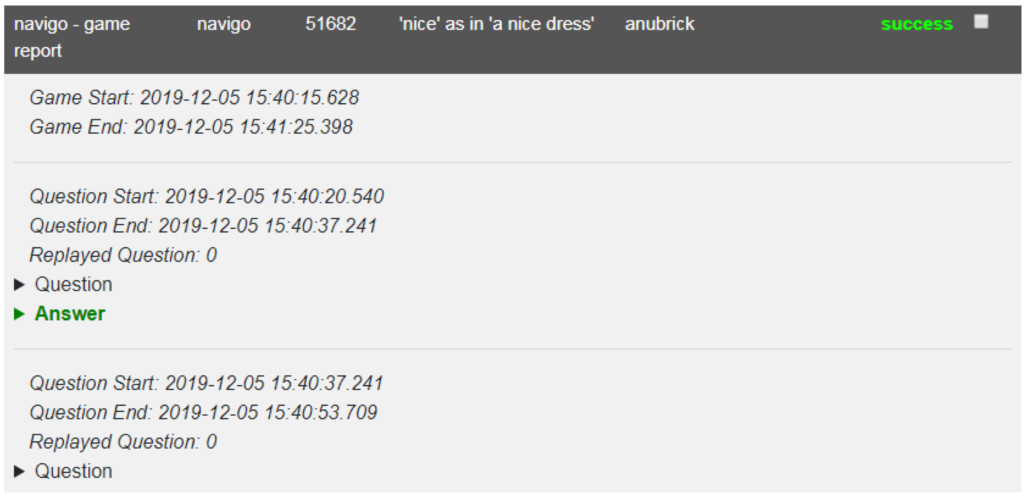
Figure 5-9 shows the output of such a filtering query consisting of two rows, each containing info about a single log entry. The log browser provides a nice way to expand and explore these entries. The basic information provided for each log entry consists of the *log-type*, *application*, *activity-id*, *feature*, *game* info about how the activity in the log was assigned (*assigned-by*) and the type of the activity *result*.



Type	Application	Activity Id	Feature	Game	Assigned By	Result	Select
logout	navigo						☐
navigo - game report	navigo	51682	'nice' as in 'a nice dress'	anubrick		success	☐

Figure 5-9: The initial view of the "log browser"

If the teacher wants to further probe into a log entry, she can click on its row to "expand" it. For game log entries, which can have a quite elaborate structure (see Appendix I for a typical game log entry) a summary is presented (see Figure 5-10) which contains elements that can be further expanded/collapsed, allowing the teacher to view only the information she considers important. At the high level, the game log entry provides information regarding: the game start and finish times, the basic question/instruction presented to the user and unit-activities (shown as an expandable list) that make up the game. By expanding the list of unit activities, she sees info about the specific answers provided by the student when presented with specific educational content. The game tracks the user's actions for each one of the questions. She can see her actions by clicking on "Answer" (see Figure 5-11). If the specific action corresponds to a correct answer the "head" row is colored green, otherwise it is colored red. Inside every answer block, we can see the flow of events in the order they happened. Four colors are used; green for correct answers, red for incorrect answers, blue for hints and feedbacks and black for start and end event. Each action event has a type, a timestamp and information about what happened and by whom (if relevant).



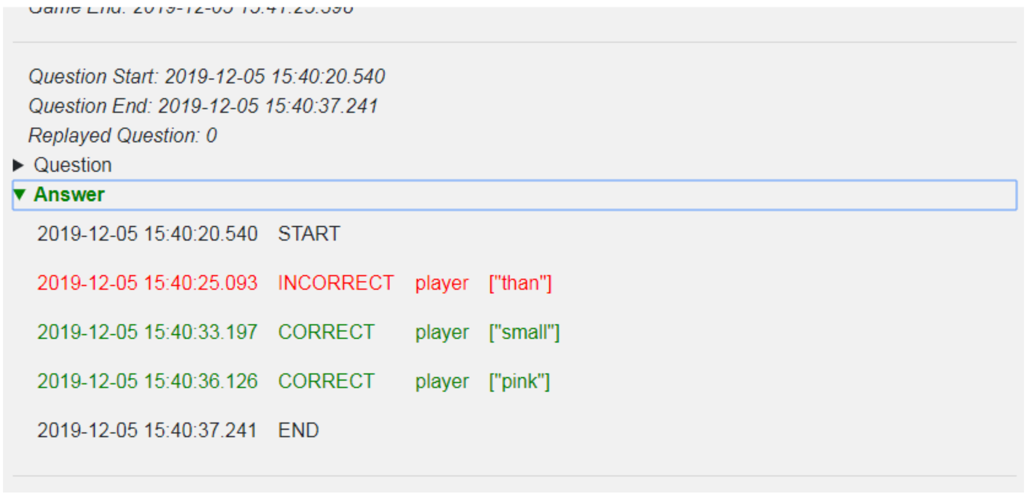
navigo - game report	navigo	51682	'nice' as in 'a nice dress'	anubrick	success	☐

Game Start: 2019-12-05 15:40:15.628
Game End: 2019-12-05 15:41:25.398

Question Start: 2019-12-05 15:40:20.540
Question End: 2019-12-05 15:40:37.241
Replayed Question: 0
▶ Question
▶ **Answer**

Question Start: 2019-12-05 15:40:37.241
Question End: 2019-12-05 15:40:53.709
Replayed Question: 0
▶ Question

Figure 5-10: The format of a game log



```
Game End: 2019-12-05 15:41:25.590

Question Start: 2019-12-05 15:40:20.540
Question End: 2019-12-05 15:40:37.241
Replayed Question: 0
▶ Question
▼ Answer
   2019-12-05 15:40:20.540   START
   2019-12-05 15:40:25.093   INCORRECT   player   ["than"]
   2019-12-05 15:40:33.197   CORRECT     player   ["small"]
   2019-12-05 15:40:36.126   CORRECT     player   ["pink"]
   2019-12-05 15:40:37.241   END
```

Figure 5-11: Browsing a sample game log entry

It should be emphasized that the log browser does not provide any functionality for processing the logs (such as collecting statistics, combining logs, etc.). Its purpose is to provide a simple way to browse the log entries and get a glimpse on the user’s recent activities and performance.

6 Data organization for analytics

In this section, we describe the structure of the system we have developed in order to support learning analytics and explorative analysis (referred to as the *“analytics system”*). The analytics system has access to the iRead universe and draws its data from it, however, it does not support (i.e., it does not provide data and services to) the on-line operation of any of iRead’s components. At the heart of the analytics system is a relational (PostgreSQL[3]) database (referred to as the *“analytics database”*) where all log related data (obtained by parsing iRead’s Elasticsearch log entries), data related to user information and data related to domain model knowledge and resources reside. PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. The parsing of the original log data into their constituent primitive parts eliminates the repetition of data and allows their organization in a way that makes complex queries feasible.

6.1 Design “REVISED”

The purpose of the analytics database is to collect all the logs data from the official iRead logger and avoid overlays and repetitions of information. The database (as it is shown in the Entity Relation Diagram of Figure 6-1) consists of four parts: The *users’ personal information* (blue tables), the *iRead resources* (green tables), the *log-related information* (beige tables) and *internal system information* (gray tables). Each part retrieves different type of data, with the log-related data part being the largest one.

User personal information

As we already described in section 4.2, at registration time each user provides her personal information. Additional to these data, the system produces information about relations between users, based on their roles. We support three types of users: students, teachers and admins. There are only a few admin users, each responsible to create and manage a group of users, which typically contains users from the same pilot/country. Teachers form a larger group, who are responsible to teach and guide their students. Between these three kinds of users there exist relations (usually representing conventions/restrictions adopted in the iRead pilots) based on their responsibilities. Every user in the iRead system has been created by an admin and every student has been assigned to a teacher.

We have also developed the notion of the “*class*” which is used to organize students into smaller groups. Each class is created by an admin and must have a teacher. Students can be added or removed from a class, but any student can be only assigned to a single class. In addition, all students in a class must use the same domain model and the same teacher. This makes a class to be a homogeneous set which, in turn, allows to draw meaningful answers on activity and performance related questions.

[3] https://www.postgresql.org/

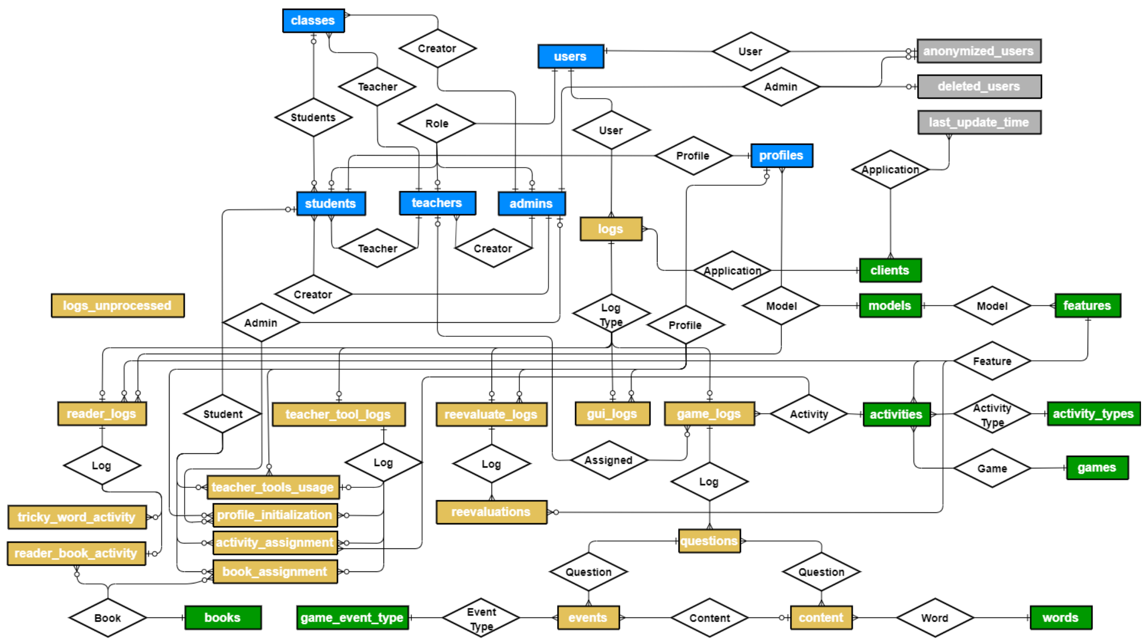


Figure 6-1: Entity Relation Diagram (ERD) of the analytics database

Last, but not least, every student has a single profile. Each profile is an instantiation of a specific domain model. At the user profile we store information related to the current status of the user and by using her profile we are able to track her progress and provide her with personalized content.

iRead resources

The analytics database also contains resources/data related to and extracted from the iRead system. These resources are independent from the users and their actions. They concern the domain models, the available game activities, the dictionaries, the books and iRead's clients/applications (Navigo games, Amigo reader etc.).

The domain models consist of groups of language features and dependency relations among them. The language features are individual literacy "items", which have to be mastered by each student. Each language has its own model (English, Greek, German, and Spanish). In addition, we provide models that teach English as a Foreign Language (EFL) as well as models for dyslexia in English and Greek.

Game activities are described as combinations of a *feature*, *an activity type* (accuracy, automaticity or blending) and *a game*. So, for each model there is a different set of activities. Activity types indicate different types of skills which are developed by practicing.

Besides the vast number of designed game activities, special *dictionaries* have been developed to provide content to the Navigo game and the Amigo reader. These dictionaries provide, for each word, additional information such as syllabification, phonetics, word structure, feature occurrences (domain model dependent), etc. This information is utilized to select appropriate content for each activity.

Log information

Each log entry, independently of its kind, contains some mandatory fields. These fields are: the log-*id* (assigned to it by the logger component), the *timestamp* (the time the event described in the log happened), the *username* (the user who produced the event), the *application-id* (from which application the event produced) and the *action-type*. Based on the application and the action-type we separate the log entries to different groups: game entries, reader entries, reevaluation entries, GUI entries (they record user actions performed web applications – '*Admin Tool*' and '*Online Pilot Tool*') and teacher tool usage entries (actions performed through the '*Teacher Tool*' web application).

In the game log entries, we store the *start* and *end* time of the game and the *end status* (whether the user completed the game successfully or unsuccessfully, whether she quitted or closed the application). Furthermore, we maintain information about the game-activity; whether it has been assigned by the student's teacher or automatically from the system. Then, we analyze the data associated to the actual game play. The questions and the answers are decomposed to their elements; text, audio files and content.

The content is drawn from the dictionaries, which consist of a large but static set of words. Each word can appear in multiple forms of content. *Content* can be considered to be a transformation of a word which provides the needed structure and data for a game activity. For example, a game that targets the syllabification needs words with their syllabification format.

The last part of the game log entries consists of the events produced by the user. For each game, while the user plays, she produces correct or incorrect answers. Each answer is a game event that consists of a word (content version) and the player's version of the word, i.e., the player's response. Besides the question, the user also gets feedback from the game. This feedback can guide her or explain to her a mistake she did. If the user made too many mistakes on a specific word, the game shows her the correct answer. All these cases correspond to game events which record the student's interaction with the games and the iRead system.

In the Amigo Reader's log entries, we store the *model id* and the *feature id* that correspond to user's activity through the Amigo application. Moreover, information related to which book is opened or closed, which book is added or removed to user's favorite list and which words are added or removed to user's tricky words are also stored to the database.

System information

We store the analytics database's update times (when and for which application we retrieved logs from the Elasticsearch database), and info about user deletion or user anonymization. In this way, we can ensure that the analytics database accurately reflects the state of the actual iRead system and that we don't keep personal information about users that requested to be deleted from iRead.

6.2 Implementation

The design we described in the previous section includes all the data available in the iRead system, including domain knowledge data, user info data and data due to action events (user playing/reading, GUI actions). In this section, we describe how we implemented the database, based on the design ER diagram. In total we have 36 tables in the database, to cover the 34 entities and the 26 relations between them. For a full definition of the database (SQL build queries), see Appendix II.b.

6.2.1 Schema "REVISED"

We will present the structure of each table. Colors correspond to the design groups. Attributes that are table primary keys appear underlined. An image of the full schema of the database can been found in Appendix II.a.

Table	users
Attributes	id, username, name, email, enabled, created_time, type
Description	The basic information for user in the iRead system.
Table	admins
Attributes	user_id, ui_language, country
Description	Admin users are related to one country/pilot.
Table	teachers
Attributes	user_id, ui_language, country, newsletter, school, creator_id
Description	Every teacher is created by only one admin user.
Table	students
Attributes	user_id, ui_language, country, newsletter, school, year_group, gender, birth_date, languages, teacher_id, creator_id, class_id
Description	Every student is created by only one admin user, she is assigned to only one teacher and she belongs to at most one class.
Table	profiles
Attributes	id, user_id, profile_id, model_id, year, competence, forced_active
Description	Each profile is associated with exactly one user. It is an instantiation of a domain model with initial values related to the "year" value. The competence is an array containing the competence values for each feature of the model. The forced_active is a Boolean array which indicate which features have been activated by force from the teacher or the admin of the user.
Table	classes
Attributes	id, class_id, name, school, season, creator_id, teacher_id
Description	A class has only one teacher and it has been created by an admin user. Their names are unique.
Table	clients
Attributes	id, client_name
Description	A client is an application that creates log-entries. Each client has its own id in the iRead system.

Table	models
Attributes	id, model_name
Description	A model has a name; EN_dm, GR_dm, EFL_ES_dm etc.
Table	features
Attributes	id, model_id, feature_id, linguistic_level, category, type, description, hr_linguistic_level, hr_category, hr_type, hr_description, difficulty, disabled
Description	Features are associate to distinct models. Each feature has a linguistic level, a category and, possibly, a type. There also exist human readable descriptions of each feature. Some of the features may be disabled either because there are no activities to play with or there is no content.
Table	activity_types
Attributes	id, activity_type_name, activity_type
Description	Each activity has a type. There are three groups of types; accuracy, blending and automaticity.
Table	games
Attributes	id, game_name
Description	The list of the available games.
Table	activities
Attributes	id, activity_id, input_type, difficulty, feature_id, activity_type, game_id
Description	Each activity is defined by the feature, the activity type and the game.
Table	words
Attributes	id, word_id, name, phonetic, syllables, pos, features_info, word_info
Description	A word has additional information about the phonetic, the syllabification, grammatical data etc.
Table	game_event_type
Attributes	id, type
Description	The possible types of a game event; start, end, correct, incorrect, hint, feedback.
Table	books
Attributes	id, book_id, language, grade, title, description, publisher, copyright, curriculum, topic, genre, file_url, cover_url
Description	A book has information related to the publisher, copyrights, its difficulty and paths to download it.
Table	logs
Attributes	id, user_id, logid, timestamp, action_type, application_id, log_type
Description	The basic information for each log entry. The logs are divided into four categories based on their origin application.
Table	game_logs
Attributes	log_id, start_time, end_time, end_state, assigned_by, activity_id
Description	A game log entry is further decomposed to questions, content, game_events tables. A game may be assigned either by the teacher of the student or automatically by the iRead system.
Table	questions

Attributes	id, log_id, question, question_audio, feedback_audio, feedback, question_number, question_replayed, question_data
Description	Each game contains a set of questions the user has to answer.
Table	question_content
Attributes	question_id, content_id, content_multiplicity
Description	A question is followed literacy content. This table links questions with content.
Table	events
Attributes	id, timestamp, source, data, event_comment, question_id, event_type, content_id
Description	The events that took place in relation to a game question.
Table	content
Attributes	id, feature_id, content_type, word_id
Description	The content is a transformation of a word to match to the game. A word is selected for a specific feature, which can be either the targeted feature or a distractor feature.
Table	reevaluate_logs
Attributes	log_id, profile_id
Description	A reevaluate log entry contains info about the change to the competence of a featur. The feature is consistent with the profile of the user that the log refers to.
Table	reevaluations
Attributes	id, log_id, feature_id, old_competence, new_competence
Description	A reevaluation indicates a change to the competence value of a specific feature of the user's profile.
Table	gui_logs
Attributes	log_id, profile_id, body
Description	These logs refer to the usage of the web applications (*Admin Tool* and *Online Pilot Tool*).
Table	teacher_tool_logs
Attributes	log_id, type
Description	Logs about actions from the Teacher Tool web application.
Table	profile_initialization
Attributes	log_id, profile_id, edit_by, old_competence, new_competence, forced_active
Description	Logs about the profile initialization from the teacher through the Teacher Tool
Table	teacher_tools_usage
Attributes	log_id, student, profile_id
Description	Logs about the usage of the internal tools of the Teacher Tool (*Profile Viewer* and *Log Browser*).
Table	book_assignment
Attributes	id, log_id, student, book_id, action_performed
Description	Logs related to teacher's book assignment to a student.
Table	activity_assignment
Attributes	id, log_id, student, activity_id, action_performed

Description	Logs related to teacher's activity assignment to a student.
Table	reader_logs
Attributes	id, log_id, model_id, feature_id, comment, type
Description	Logs about actions from the reader AMIGO application.
Table	tricky_word_activity
Attributes	id, log_id, word
Description	Logs about the addition or the removal of a word to the user's list of tricky words.
Table	reader_book_activity
Attributes	id, log_id, book_id
Description	Logs about the usage of a book or the addition/removal of a book to the user's favorite books.
Table	logs_unprocessed
Attributes	log_id, comment
Description	In case that system failed to analyze a log, the id of the log is stored for future analysis.
Table	anonymized_users
Attributes	user_id, admin_id, deleted_time
Description	The internal id (username is different, from the id) of the user, who has been anonymized after her request. The admin_id indicates who admin execute the anonymization.
Table	deleted_users
Attributes	user_id, admin_id, created_time, deleted_time
Description	A list with the users who has been deleted from the system. (The user_id is internal id and it is not the same with the username)
Table	last_update_time
Attributes	id, application_id, date
Description	A list with the dates that each application has been updated.

6.2.2 Organization of Java coding

On top of the database resides a set of Java projects, with each one being responsible for a different task. One project has been developed to contain the Data Access Objects (DAO) classes. For each table of the database there is a DAO Java class which implement the "add", "update", "delete", "get" transactions as well as every other required database query. In this way, we can manage the elements of the database with methods developed in Java. Along with these DAO classes, in the same project, we created classes to model the elements of each table as Java objects. A second Java project is needed to implement the calls to the iRead services. These methods are needed to retrieve data and logs from the iRead system. Besides accessing log data, we also need to initialize the resource and domain knowledge data (like dictionaries, activities, models etc.). For this purpose, a third project has been developed to populate the needed information, as it exists in the iRead system. Lastly, we develop a Java project which allows us to parse and analyze the data we retrieve from the iRead system. This project enables us to populate the analytics database in a daily basis. The utilities offered by this project are explained in the following section.

6.2.3 Data filtering

Updating user information

When a user is created, a log entry about his creation is added to the logger. Similarly, when the information of a user has been updated or the user is anonymized or deleted a log entry is added to the logger to report that change. When our system retrieves such a log entry, it has to update the corresponding information. In the case of a new user, a new row is added to the "users" table together with a row to one of the "admin", "teacher", or "student" tables based on the type of the new user. Besides this information, the relations "created by" and "teacher of" (in case of a student) must be updated properly by adding rows made up by the ids of the corresponding users, as they exist in our database. In a similar way, an update log will trigger a process to update the proper rows in these tables. The actions taken for the cases of an anonymization or a deletion are treated in detail in Section 6.3.

Retrieving log entries

Immediately after the start of the large scale pilots, the analytics system will retrieve in a daily basis all user action log entries from iRead's Elasticsearch logger. Every iRead log entry has five basic attributes: *logid*, *uid*, *timestamp*, *applicationid* and *actionType*. Consequently, for each log entry a new entry is created in the "logs" table on the analytics system. These five attributes populate the new entry. Further analysis takes place in the case where log refers to a Navigo game action (as determined by the *actionType* attribute). A sample of a Navigo game log entry is provided in Appendix I.

For each game log entry we create a new row in the "*game_logs*" table. This row contains information about when the game started, when it ended, how it did end (successfully, unsuccessfully, did the user close or quit the application?), in what way was the activity assigned to the student (by the teacher or automatically by the iRead system)? This latest information ("how was the activity assigned?") can be retrieved from the attribute *"comment" – "suggestedBy"*. If the activity was automatically proposed by the iRead system the value of the attribute is "*AUTO*", otherwise it contains the username of student's teacher. At last, we have to identify the activity that this game entry refers to. We get the relevant data from the attributes "*featureId*", "*activityType*", "*gameId*", "*activityId*" and "*modelId*". With these five values we identify the specific activity from the "*activities*" table. At this point, the new log entry is ready to be inserted in the analytics database.

The final and more demanding step is to parse the part of the Navigo game entry log that refers to the student's/player's actions. Attribute "*data*" contains the question and the feedback in text and in audio format (for audio format the names of the audio files are provided). Each question is accompanied by the correct and incorrect choices which were presented to the user. After having identified and recorded the questions presented to the user, we have to create entries for each one of the game events/actions the user produces. These actions are reported in attribute "*activities*" where we can find the correct and incorrect tries, whether the game provided any hints, or whether the user used any feedback. A correct answer can be provided either by the user or by the game in the form of feedback (in the case where the user failed to give the correct answer). An incorrect answer can be given only by the user. Hints can be given by the game or can be requested by the user for help. In each case, we isolate the event and connect it with the word it refers to. Note that this is not always possible, as the result of a try may provide not enough information to identify the corresponding content – word.

6.3 User deletion/anonymization

Besides the tools we have develop to keep the data up to date and analyze – parse iRead's log entries, the analytics system also supports user **deletion** and **anonymization**. When a user requests to be deleted from the iRead system and his administrator executes this request, a relevant log entry is added to iRead's logger. This log entry contains information regarding the identity of the deleted user and the process which was followed. A user can be fully deleted (delete personal information and delete all the logs of her) or she can be anonymized (delete personal information).

In case where a user is fully deleted, the analytics system also deletes from its database all entries which refer to her. This details to deleting her as a user (this also deletes her profile) as well as deleting all associated logs entries. There is only one trace of the user kept in the analytics database and more specifically in table "*deleted_users*" where we record the username, the registration time and the deletion time. These entries are necessary for documenting that the deletion request was successfully fulfilled.

In case of a request for anonymization, we delete the personal information about the user from all her database entries, but we do not delete the entry itself. We substitute her username with the new one which cannot be traced back to the original user. We also remove her relations (created by, teacher, classes) and assign her to some general users (which exist for anonymization purposes). Finally, we update all her logs with the new anonymous username and eliminate any relations between her logs and her teacher (assigned activities).

7 Graphical tools for analytics and explorative analysis "REVISED"

In this section, we describe the graphical tools we utilize in order to explore the analytics database to present the analysis results in visual form. These tools allow researchers to obtain their data directly from the analytics database, to manage their format (select specific attributes) and build manageable charts.

7.1 The Tableau Prep tool

The **tableau prep**[4] program is used to build data flows and sharing – managing them across an organization. This program can connect to the database and retrieve data. Its advantage is that it allows users to select which tables or attributes can be exported for further analysis. So, by using Tableau Prep data can be organized and, most importantly, data can be exported as anonymized data for further analysis. For the purposes of explorative analysis, personal data have no impact to the analysis results and, in order to minimize the risk of any data breach, they are not further exposed.

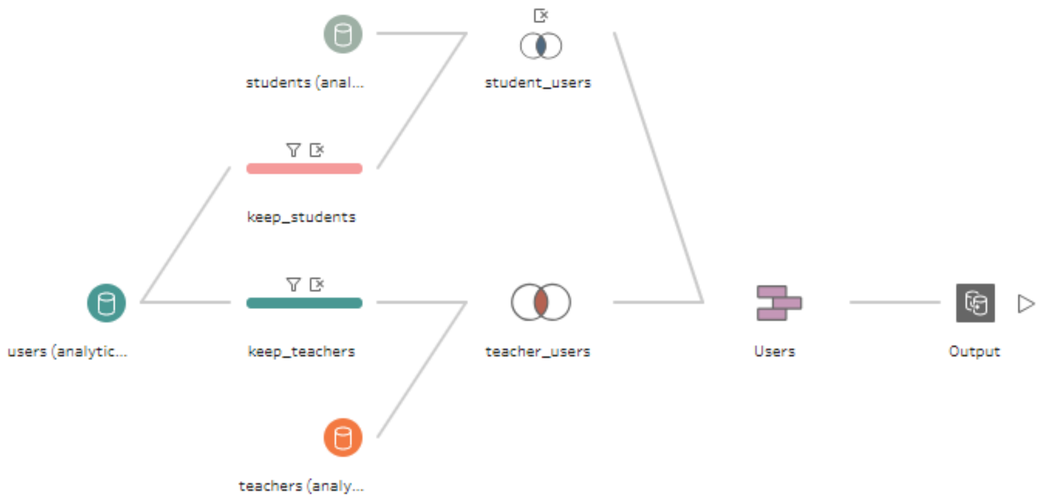


Figure 7-1: Tableau Prep data flow

When Tableau Prep starts, it needs to be connected to one or more databases. Tableau Prep supports a variety of databases, including PostgreSQL. To connect successfully to a database, the URL, the port, the name of the database and appropriate credentials have to be provided. After establishing a connection, Tableau Prep can retrieve the structure of the database as well as data from it. Each table can be used multiple times, data from different tables can be unified or joined to create new sets of data. Attributes can be removed (e.g., the users' names), they can be altered (in order to either eliminate/correct possible errors or to change the attributes format) or they can be filtered (to select a subgroup of the table rows).

[4] https://www.tableau.com/products/prep

Figure 7-1 shows an example data flow from the analytics database which yields a new restructured data set which can be used (or exported) by the Tableau Desktop tool to create diagrams and charts for analytics. In the shown example, as a first step two separate subparts of the *users* table have extracted by filtering their role (either the user is student or teacher). From these subparts the personal information of each user, like the first name, last name, email etc., have been removed. Then, each subpart joined with the corresponding table (*students* and *teachers*) to merge the additional information, based on their role. For the students an extra step for removing personal information has been added (to remove school name). These two sets of users are united to one single set. Lastly, we export the result to file output for further use. This flow describes an order of actions that creates data in the desired format. By re-running the data flow, refreshed data can be retrieved from the analytics database. Note that this flow is independent from the data; it is only related with the structure of the analytics database and the desired structure of the result.

7.2 *The Tableau Desktop tool*

The **Tableau Desktop**[5] program is used to build live visual analytics and interactive dashboards. This program allows to its users to create charts based on the output generated/provided by Tableau Prep or from directly from a database. Based on the same set of data, (tables, attributes, and data) each user can create different graphs that answer the analytic question of her interest.

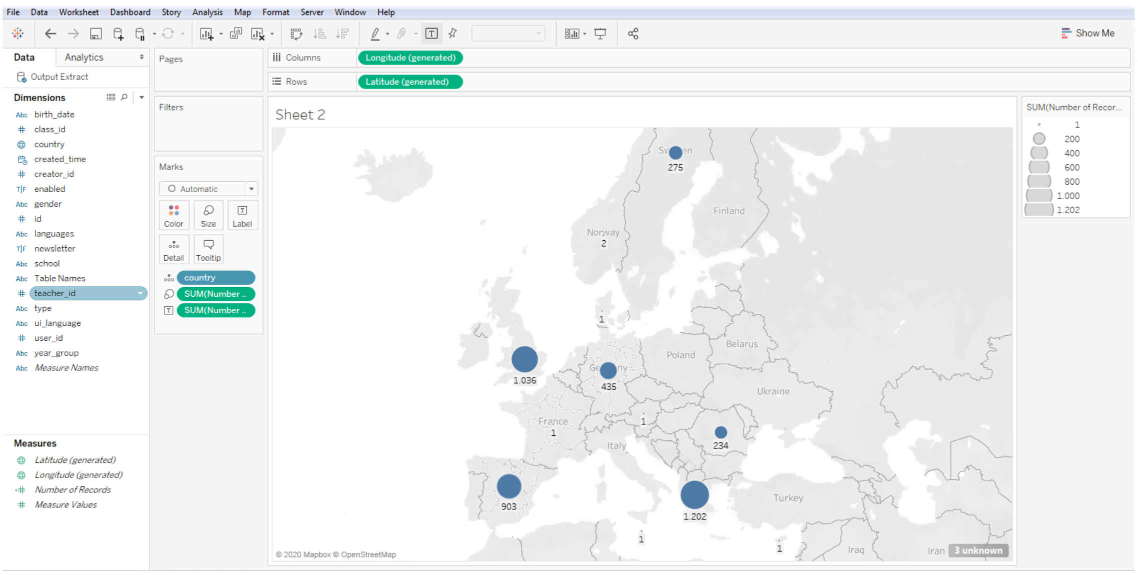


Figure 7-2: Tableau Desktop overview

During analysis with Tableau Desktop we import the flow we have already created with Tableau Prep. We can use all the attributes, with their data, to create charts and diagrams. Tableau Desktop separates them in *measurable* and *dimensional* attributes. With drag and drop functionality the researcher can select the attributes of interest and designate them either as *columns* or as *rows* of a chart. Tableau Desktop automatically builds the chart, with the selected style. In our example, as it is shown in Figure 7-2 ,

[5] https://www.tableau.com/products/desktop

based on the country of residence attribute of each user, the Tableau create circles above the corresponding countries to indicate how many users have been registered on each country. Under the circle, which size is depended to the number of users, the corresponding number is displayed.

The Tableau Desktop tool allows the researcher to build charts on demand, as well as to further customize them by allowing her to further filter the data used to build an existing chart. So, these dynamic charts can provide answers to multiple questions and focus to the specific parts of the data the researcher treats as most important.

8 Conclusion

iRead is a project which includes an evaluation phase consisting of several large-scale pilots. During these pilots and the extensive use of the iRead system, a vast amount of data will be generated, capturing the actual usage of the iRead system as well as the way users master the reading skill, as it is being abstracted in the developed language domain models.

In this deliverable, we have described *what data we collect* while the iRead system is in use, how these data are organized, and *how these data become available* to researchers. The developed Analytics System fulfills its main objective which is to support the analysis phase of the iRead project. In addition, it can become a valuable tool to the hand of educational and linguistic researchers who engage in explorative analysis.

References

Chatti, M.A., Dyckhoff, A.L., Schroeder, U., Thüs, H. 2013. A reference model for learning analytics. International Journal of Technology-Enhanced Learning 4 (5-6), 318-331.

Pierre Dillenbourg and Patrick Jermann. 2010. Technology for classroom orchestration. In K.M. and S.I. (eds.) New Science of Learning, Springer New York, 525-552.

Dillenbourg, P.; Zufferey, G., Alavi, H., Jermann, P., Do-Lenh, S., Bonnard, Q., Cuendet, S. and Kaplan, F. 2011. Classroom Orchestration: The Third Cycle of Usability. In Proceedings of the 11th Conference of Computer-Supported Collaborative Learning, 1, 510-517.

Dollinger, M., and Lodge. J.M. 2018. Co-creation strategies for learning analytics. In Proceedings of the 8th International Conference on Learning Analytics and Knowledge (LAK '18), 97-101.

Khalil, M., & Ebner, M., 2015. Learning Analytics: Principles and Constraints. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, 1326-1336.

Halskov, K., and Dalsgärd, P. 2006. Inspiration Card Workshops. In Proceedings of the 6th ACM Conference in Designing Interactive Systems (DIS'06), 2-11.

Holstein, K., McLaren, B. M. and Aleven, V. 2017. Intelligent tutors as teachers' aides. In Proceedings of the Seventh International Learning Analytics & Knowledge Conference on Learning Analytics (LAK '17), 257-266.

Hornecker, E., Halloran, J., Fitzpatrick, G., Weal, M., Millard, D., Michaelides, D., Cruickshank, D. and Roure, D.D. 2006. UbiComp in Opportunity Spaces: Challenges for Participatory Design. In Proceedings of the 9th Conference in Participatory Design, Trento, Italy, 47-56.

Liaqat, A., Axtell, B., Munteanu, C. and Demmans Epp, C. 2018. Contextual Inquiry, Participatory Design, and Learning Analytics: An Example. In Proceedings of the LAK 2018 Workshop on Participatory Design and Learning Analytics.

Martinez-Maldonado, Abelardo Pardo, Negin Mirriahi, Kalina Yacef, Judy Kay and Andrew Clayphan. 2015. The LATUX workflow. InProceedings of the Fifth International Conference on Learning Analytics And Knowledge (LAK '15), 1-10.

Martinez-Maldonaldo, Simon Buckingham-Shum, Bertrand Schneider, Sven Charleer, Joris Klerkx and Erik Duval. 2017. Learning Analytics for Natural User Interfaces. Journal of Learning Analytics, 4 (1), 24-57.

Mavrikis, M., Gutierrez-Santos, S., and Poulovassilis, A. 2016. Design and Evaluation of Teacher Assistance Tools for Exploratory Learning Environments. In Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, 168–172.

Nye, B. D. (2014). Barriers to ITS adoption: A systematic mapping study. In ITS 2014, 583-590. Springer International Publishing.

Roberto Martinez-Maldonaldo, Simon Buckingham-Shum, Bertrand Schneider, Sven Charleer, Joris Klerkx and Erik Duval. 2017. Learning Analytics for Natural User Interfaces. Journal of Learning Analytics, 4 (1). 24-57.

Prieto-Alvarez, C.G., Martinez-Maldonado, R., and Anderson, T. 2017. Co-designing in learning analytics: tools and techniques. In J.C.H. Lodge, J.C.H., L. Corrin ed. Learning analytics in the classroom: translating learning analytics research for teachers, Routledge.

Prieto, L.P., Dlab, M. H., Gutiérrez, I., Abdulwahed, M., and Balid, W. 2011. Orchestrating technology enhanced learning: a literature review and conceptual framework. Int. J. Technology Enhanced Learning, 3 (6), 583-598.

Prieto, L. P., Dimitriadis, Y., Asensio-Pérez, J. I. & Looi, C.-K. (2015). Orchestration in Learning Technology Research: Evaluation of a Conceptual Framework. *Research in Learning Technology,* 23 (1)**,** 1-15.

Sharma, P. and Hannafin, M.J. 2007. Scaffolding in technology-enhanced learning environments. Interactive Learning Environments, 15 (1). 27-46.

Schwendimann, B. A., Rodríguez-Triana, M. J., Vozniuk, A., Prieto, L. P., Boroujeni, M. S., Holzer, A., Gillet, D., Dillenbourg, P. (2017). Perceiving Learning at a Glance: A Systematic Literature Review of Learning Dashboard Research. IEEE Transactions on Learning Technologies, 10 (1), 30–41.

Siemens, G., *Learning and Academic Analytics,* 5 August 2011, http://www.learninganalytics.net/?p=131

Takala, M. 2006. The Effects of Reciprocal Teaching on Reading Comprehension in Mainstream and Special (SLI) Education. Scandinavian Journal of Educational Research, 50 (5), 559-576.

Xhakaj, F., Aleven, V., & McLaren, B. M. (2016). How Teachers Use Data to Help Students Learn: Contextual Inquiry for the Design of a Dashboard. In European Conference on Technology Enhanced Learning, 340-354. Springer International Publishing.

Vatrapu, R., Teplovs, C., Fujita, N., and Bull, S. 2011. Towards Visual Analytics for Teachers? Dynamic Diagnostic Pedagogical Decision Making. 93-98. In Proceedings of the First International Conference on Learning Analytics And Knowledge (LAK '11), 93-98.

Verbert, K., Duval, E., Klerkx, J., Govaerts, S. and Santos, J.S. 2013. Learning Analytics Dashboard Applications. American Behavioral Scientist, 57 (10), 1500-1509.

Appendix “REVISED”

I. Game log sample

```
{
   "logid": "AW7vdzA4-Vvpvk9eqMKX",
   "actionType": "GAMEPLAY",
   "applicationid": "iread_games",
   "uid": "<username>",
   "timestamp": "2019-12-10T10:58:21.130",
   "profileId": "<username>_EN_dm",
   "modelId": "EN_dm",
   "featureId": "22",
   "activityType": "B1a",
   "gameId": "9",
   "activityId": "15535",
   "comment": {
    "suggestedBy": "<teacher_username>/AUTO",
    "other": ""
   },
   "data": [
    {
     "question": "",
     "question_audio": [
      "EN_au_instr_37.mp3",
      "EN_0002207_cog.mp3"
     ],
     "feedback_audio": [
      "EN_au_feed_38.mp3",
      "EN_0002207_cog.mp3"
     ],
     "feedback": "",
     "context": "$$$",
     "answer": [
      "c",
      "o",
      "g"
     ],
     "word_info": [
      {
       "id": 2198,
       "featureId": "22",
       "name": "cog",
       "numberOfCharacters": 3,
       "phonetic": "/kɒg/",
        "graphemePhoneme": "[\"c-k\",\"o-ɒ\",\"g-g\"]",
        "numberOfPhonemes": 3,
        "syllables": "[\"cog\"]",
        "numberOfSyllables": 1,
        "cvForm": "-cvc-",
        "audioFile": "EN_0002207_cog.mp3",
        "featureInfo": {
         "featureId": 22,
         "matches": [
          {
           "start": 1,
           "end": 2
          }
         ]
        }
      }
     ]
    },
    {
     "question": "",
     "question_audio": [
      "EN_au_instr_37.mp3",
      "EN_0006378_long.mp3"
     ],
     "feedback_audio": [
      "EN_au_feed_38.mp3",
      "EN_0006378_long.mp3"
     ],
     "feedback": "",
     "context": "$$$",
     "answer": [
      "l",
      "o",
      "ng"
     ],
     "word_info": [
      {
        "id": 6357,
        "featureId": "22",
        "name": "long",
        "numberOfCharacters": 4,
        "phonetic": "/lɒŋ/",
        "graphemePhoneme": "[\"l-l\",\"o-ɒ\",\"ng-ŋ\"]",
        "numberOfPhonemes": 3,
        "syllables": "[\"long\"]",
        "numberOfSyllables": 1,
        "cvForm": "-cvcc-",
        "audioFile": "EN_0006378_long.mp3",
        "featureInfo": {
          "featureId": 22,
          "matches": [
            {
              "start": 1,
              "end": 2
            }
          ]
        }
      }
    ]
  },
  {
    "question": "",
    "question_audio": [
      "EN_au_instr_37.mp3",
      "EN_0006416_lot.mp3"
    ],
    "feedback_audio": [
      "EN_au_feed_38.mp3",
      "EN_0006416_lot.mp3"
    ],
    "feedback": "",
    "context": "$$$",
    "answer": [
      "l",
      "o",
      "t"
    ],
    "word_info": [
      {
        "id": 6395,
        "featureId": "22",
        "name": "lot",
        "numberOfCharacters": 3,
        "phonetic": "/lɒt/",
        "graphemePhoneme": "[\"l-l\",\"o-ɒ\",\"t-t\"]",
        "numberOfPhonemes": 3,
        "syllables": "[\"lot\"]",
        "numberOfSyllables": 1,
        "cvForm": "-cvc-",
        "audioFile": "EN_0006416_lot.mp3",
        "featureInfo": {
          "featureId": 22,
          "matches": [
            {
```

```
            "start": 1,
            "end": 2
          }
        ]
      }
    }
  ]
 }
],
"gameStart": "2019-12-10T10:58:21.130",
"activities": [
 {
  "startTime": "2019-12-10T10:58:37.278",
  "questions": [
   {
    "startTime": "2019-12-10T10:58:37.281",
    "dataIndex": 2,
    "replayedQuestion": 0,
    "correctAnswers": [
     {
      "timeStamp": "2019-12-10T10:58:59.098",
      "answer": [
       "l",
       "o",
       "t"
      ],
      "source": "player"
     }
    ],
    "incorrectAnswers": [],
    "hints": [],
    "endTime": "2019-12-10T10:58:59.098",
    "endState": "success"
   },
   {
    "startTime": "2019-12-10T10:59:05.613",
    "dataIndex": 0,
    "replayedQuestion": 0,
    "correctAnswers": [
     {
      "timeStamp": "2019-12-10T10:59:21.152",
      "answer": [
       "c",
       "o",
       "g"
      ],
      "source": "player"
```

```
          }
        ],
        "incorrectAnswers": [],
        "hints": [],
        "endTime": "2019-12-10T10:59:21.152",
        "endState": "success"
      },
      {
        "startTime": "2019-12-10T10:59:27.701",
        "dataIndex": 1,
        "replayedQuestion": 0,
        "correctAnswers": [
          {
            "timeStamp": "2019-12-10T10:59:39.230",
            "answer": [
              "l",
              "o",
              "ng"
            ],
            "source": "player"
          }
        ],
        "incorrectAnswers": [],
        "hints": [],
        "endTime": "2019-12-10T10:59:39.230",
        "endState": "success"
      }
    ],
    "endTime": "2019-12-10T10:59:39.231",
    "endState": "success"
  }
 ],
 "gameEnd": "2019-12-10T10:59:55.939",
 "endState": "success"
}
```

II. Analytics database

a. Schema of the database

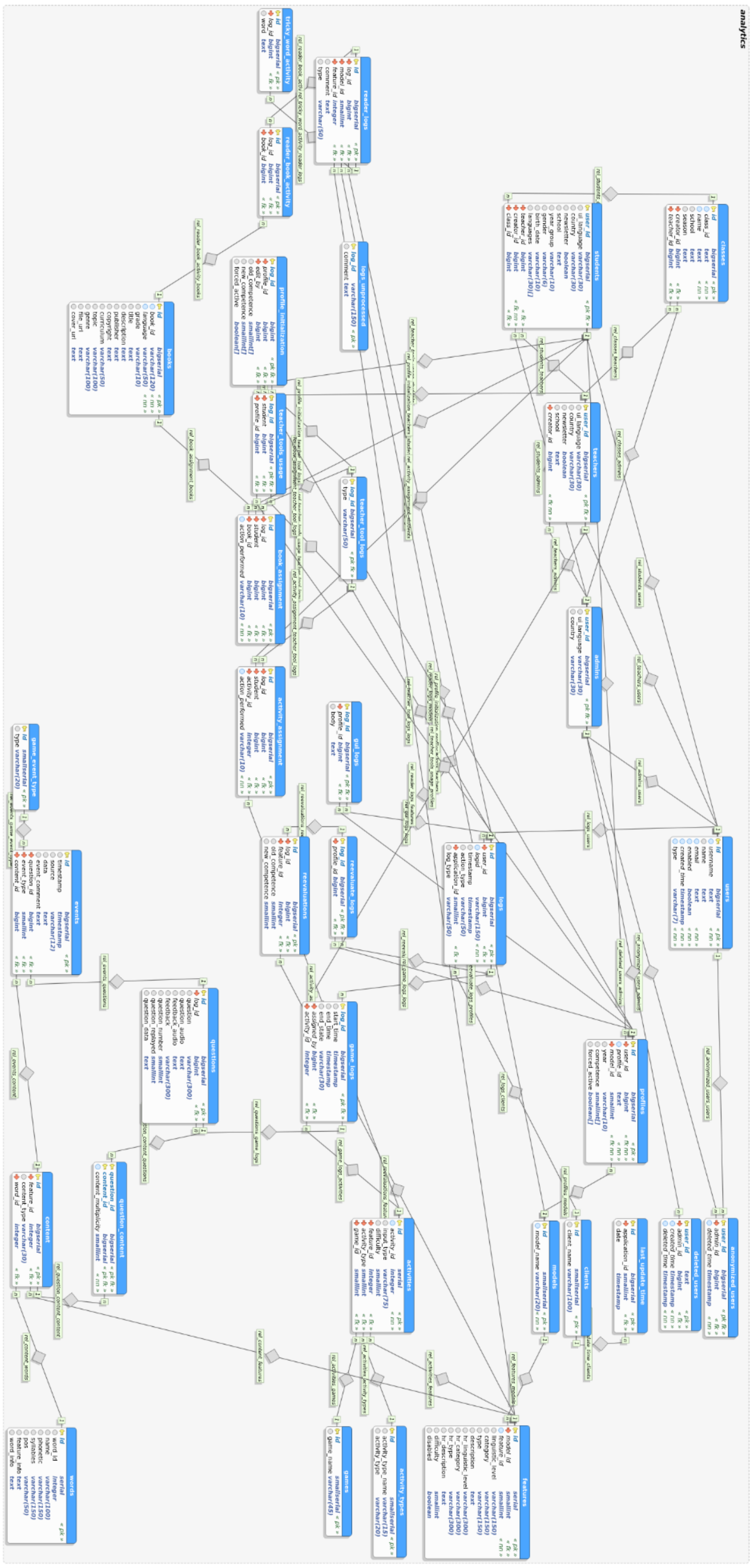


Figure II.a-0-1: Full schema of the analytics database

b. SQL build queries

```
CREATE SCHEMA analytics;

CREATE TABLE analytics.users(
	id bigserial NOT NULL,
	username text NOT NULL,
	name text,
	email text NOT NULL,
	enabled boolean NOT NULL,
	created_time timestamp NOT NULL,
	type varchar(7) NOT NULL,
	CONSTRAINT "USER_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.users.type IS 'student/teacher/admin';

CREATE TABLE analytics.admins(
	user_id bigserial NOT NULL,
	ui_language varchar(30),
	country varchar(30),
	CONSTRAINT "ADMIN_ID" PRIMARY KEY (user_id)
);

CREATE TABLE analytics.teachers(
	user_id bigserial NOT NULL,
	ui_language varchar(30),
	country varchar(30),
	newsletter boolean,
	school text,
	creator_id bigint NOT NULL,
	CONSTRAINT "TEACHER_ID" PRIMARY KEY (user_id)
);

CREATE TABLE analytics.students(
	user_id bigserial NOT NULL,
	ui_language varchar(30),
	country varchar(30),
	newsletter boolean,
	school text,
	year_group varchar(10),
	gender varchar(6),
	birth_date varchar(10),
	languages varchar(30)[],
	teacher_id bigint,
	creator_id bigint NOT NULL,
	class_id bigint,
	CONSTRAINT "STUDENT_ID" PRIMARY KEY (user_id)
);

CREATE TABLE analytics.classes(
	id bigserial NOT NULL,
	class_id text NOT NULL,
	name text NOT NULL,
	school text,
	season text,
	creator_id bigint,
	teacher_id bigint,
	CONSTRAINT "CLASS_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.logs(
	id bigserial NOT NULL,
	user_id bigint,
	logid varchar(150) NOT NULL,
	"timestamp" timestamp,
	action_type varchar(50),
	application_id smallint,
	log_type varchar(50),
	CONSTRAINT "LOG_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.logs.log_type IS
'gameplay/teacher_tool/reader/gui';

CREATE TABLE analytics.clients(
	id smallserial NOT NULL,
	client_name varchar(100),
	CONSTRAINT client_id PRIMARY KEY (id)
);

CREATE TABLE analytics.gui_logs(
	log_id bigserial NOT NULL,
	"profile_id" bigint,
	body text,
	CONSTRAINT "GUI_LOG_ID" PRIMARY KEY (log_id)
);
COMMENT ON COLUMN analytics.gui_logs."profile_id" IS 'for profile creations';

CREATE TABLE analytics.reader_logs(
	id bigserial NOT NULL,
	log_id bigint,
	model_id smallint,
	feature_id integer,
        comment text,
        type varchar(50),
        CONSTRAINT "READER_LOG_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.reader_logs.type IS
'book_activity/tricky_words_activity';

CREATE TABLE analytics.reevaluate_logs(
        log_id bigserial NOT NULL,
        profile_id bigint,
        CONSTRAINT "REEVALUATE_LOG_ID" PRIMARY KEY (log_id)
);

CREATE TABLE analytics.game_logs(
        log_id bigserial NOT NULL,
        start_time timestamp,
        end_time timestamp,
        end_state varchar(30),
        assigned_by bigint,
        activity_id integer,
        CONSTRAINT "GAME_LOG_ID" PRIMARY KEY (log_id)
);

CREATE TABLE analytics.activities(
        id serial NOT NULL,
        activity_id integer NOT NULL,
        input_type varchar(75),
        difficulty smallint,
        feature_id integer,
        activity_type smallint,
        game_id smallint,
        CONSTRAINT "ACTIVITY_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.logs_unprocessed(
        log_id varchar(150) NOT NULL,
        comment text,
        CONSTRAINT "LOG_UNPROCESSED_ID" PRIMARY KEY (log_id)
);

CREATE TABLE analytics.models(
        id smallserial NOT NULL,
        model_name varchar(20) NOT NULL,
        CONSTRAINT "MODEL_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.profiles(
        id bigserial NOT NULL,
        user_id bigint NOT NULL,
        profile_id text NOT NULL,
        model_id smallint NOT NULL,
        year varchar(10),
        competence smallint[],
        forced_active boolean[],
        CONSTRAINT "PROFILE_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.profiles.year IS 'this is profile init';

CREATE TABLE analytics.anonymized_users(
        user_id bigserial NOT NULL,
        admin_id bigint,
        deleted_time timestamp NOT NULL,
        CONSTRAINT "ANONYMIZED_ID" PRIMARY KEY (user_id)
);

CREATE TABLE analytics.deleted_users(
        user_id text NOT NULL,
        admin_id bigint,
        created_time timestamp NOT NULL,
        deleted_time timestamp NOT NULL,
        CONSTRAINT "DELETED_ID" PRIMARY KEY (user_id)
);

CREATE TABLE analytics.last_update_time(
        id bigserial NOT NULL,
        application_id smallint,
        date timestamp,
        CONSTRAINT "LAST_UPDATE_TIME_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.features(
        id serial NOT NULL,
        model_id smallint,
        feature_id smallint NOT NULL,
        linguistic_level varchar(150),
        category varchar(150),
        type varchar(150),
        description text,
        hr_linguistic_level varchar(300),
        hr_category varchar(300),
        hr_type varchar(300),
        hr_description text,
        difficulty smallint,
        disabled boolean,
        CONSTRAINT "FEATURE_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.activity_types(
        id smallserial NOT NULL,
        activity_type_name varchar(15),
        activity_type varchar(20),
        CONSTRAINT "ACTIVITY_TYPE_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.games(
        id smallserial NOT NULL,
        game_name varchar(45),
        CONSTRAINT "GAME_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.words(
        id serial NOT NULL,
        word_id integer,
        name varchar(100),
        phonetic varchar(150),
        syllables varchar(150),
        pos varchar(50),
        feature_info text,
        word_info text,
        CONSTRAINT "WORD_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.questions(
        id bigserial NOT NULL,
        log_id bigint,
        question varchar(300),
        question_audio text,
        feedback_audio text,
        feedback varchar(300),
        question_number smallint,
        question_replayed smallint,
        question_data text,
        CONSTRAINT "QUESTION_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.game_event_type(
        id smallserial NOT NULL,
        type varchar(20),
        CONSTRAINT "GAME_EVENT_TYPE_ID" PRIMARY KEY (id)
);
```

```
CREATE TABLE analytics.content(
        id bigserial NOT NULL,
        feature_id integer,
        content_type varchar(30),
        word_id integer,
        CONSTRAINT "CONTENT_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.content.feature_id IS 'the feature id for which the
word has been selected';
COMMENT ON COLUMN analytics.content.content_type IS 'if the word selected as a
correct or as a distractor one';

CREATE TABLE analytics.question_content(
        question_id bigserial NOT NULL,
        content_id bigserial NOT NULL,
        content_multiplicity smallint NOT NULL,
        CONSTRAINT "QUESTION_CONTENT_ID" PRIMARY KEY (question_id,content_id)
);

CREATE TABLE analytics.events(
        id bigserial NOT NULL,
        "timestamp" timestamp,
        source varchar(12),
        data text,
        event_comment text,
        question_id bigint,
        event_type smallint,
        content_id bigint,
        CONSTRAINT "EVENT_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.events.event_comment IS 'in case that the event
cannot match to a specific content add information about the event';

CREATE TABLE analytics.reevaluations(
        id bigserial NOT NULL,
        log_id bigint,
        feature_id integer,
        old_competence smallint,
        new_competence smallint,
        CONSTRAINT "REEVALUATION_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.teacher_tool_logs(
        log_id bigserial NOT NULL,
        type varchar(50),
        CONSTRAINT "TEACHER_TOOL_LOGS_ID" PRIMARY KEY (log_id)
);
```

```
COMMENT ON COLUMN analytics.teacher_tool_logs.type IS
'tool_usage/book_assignment/activity_assignment';

CREATE TABLE analytics.teacher_tools_usage(
        log_id bigserial NOT NULL,
        student bigint,
        profile_id bigint,
        CONSTRAINT "TEACHER_TOOLS_USAGE_ID" PRIMARY KEY (log_id)
);

CREATE TABLE analytics.book_assignment(
        id bigserial NOT NULL,
        log_id bigint,
        student bigint,
        book_id bigint,
        action_performed varchar(10) NOT NULL,
        CONSTRAINT "BOOK_ASSIGNMENT_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.book_assignment.action_performed IS
'assigned/unassigned';

CREATE TABLE analytics.activity_assignment(
        id bigserial NOT NULL,
        log_id bigint,
        student bigint,
        activity_id integer,
        action_performed varchar(10) NOT NULL,
        CONSTRAINT "ACTIVITY_ASSIGNMENT_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.activity_assignment.action_performed IS
'assigned/unassigned';

CREATE TABLE analytics.books(
        id bigserial NOT NULL,
        book_id varchar(120) NOT NULL,
        language varchar(50) NOT NULL,
        grade varchar(10),
        title text,
        description text,
        publisher text,
        copyright text,
        curriculum varchar(50),
        topic varchar(100),
        genre varchar(100),
        file_url text,
        cover_url text,
        CONSTRAINT "BOOK_ID" PRIMARY KEY (id)
);
COMMENT ON COLUMN analytics.books.language IS 'English/Greek/Spanish/German';

CREATE TABLE analytics.reader_book_activity(
        id bigserial NOT NULL,
        log_id bigint,
        book_id bigint,
        CONSTRAINT "READER_BOOK_ACTIVITY_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.tricky_word_activity(
        id bigserial NOT NULL,
        log_id bigint,
        word text,
        CONSTRAINT "TRICKY_WORD_ACTIVITY_ID" PRIMARY KEY (id)
);

CREATE TABLE analytics.profile_initialization(
        log_id bigint NOT NULL,
        profile_id bigint,
        edit_by bigint,
        old_competence smallint[],
        new_competence smallint[],
        forced_active boolean[],
        CONSTRAINT "PROFILE_INITIALIZATION_ID" PRIMARY KEY (log_id)
);

ALTER TABLE analytics.admins ADD CONSTRAINT "USER_ADMIN" FOREIGN KEY
(user_id)
REFERENCES analytics.users (id) MATCH FULL

ALTER TABLE analytics.teachers ADD CONSTRAINT "USER_TEACHER" FOREIGN KEY
(user_id)
REFERENCES analytics.users (id) MATCH FULL

ALTER TABLE analytics.teachers ADD CONSTRAINT "TEACHER_ADMIN" FOREIGN KEY
(creator_id)
REFERENCES analytics.admins (user_id) MATCH FULL

ALTER TABLE analytics.students ADD CONSTRAINT "USER_STUDENT" FOREIGN KEY
(user_id)
REFERENCES analytics.users (id) MATCH FULL

ALTER TABLE analytics.students ADD CONSTRAINT "STUDENT_TEACHER" FOREIGN KEY
(teacher_id)
REFERENCES analytics.teachers (user_id) MATCH FULL
```

```
ALTER TABLE analytics.students ADD CONSTRAINT "STUDENT_ADMIN" FOREIGN KEY
(creator_id)
REFERENCES analytics.admins (user_id) MATCH FULL

ALTER TABLE analytics.students ADD CONSTRAINT "STUDENT_CLASS" FOREIGN KEY
(class_id)
REFERENCES analytics.classes (id) MATCH FULL

ALTER TABLE analytics.classes ADD CONSTRAINT "CLASS_CREATOR" FOREIGN KEY
(creator_id)
REFERENCES analytics.admins (user_id) MATCH FULL

ALTER TABLE analytics.classes ADD CONSTRAINT "CLASS_TEACHER" FOREIGN KEY
(teacher_id)
REFERENCES analytics.teachers (user_id) MATCH FULL

ALTER TABLE analytics.logs ADD CONSTRAINT "LOG_USER" FOREIGN KEY (user_id)
REFERENCES analytics.users (id) MATCH FULL

ALTER TABLE analytics.logs ADD CONSTRAINT "LOG_CLIENT" FOREIGN KEY
(application_id)
REFERENCES analytics.clients (id) MATCH FULL

ALTER TABLE analytics.gui_logs ADD CONSTRAINT "GUI_LOG_LOG" FOREIGN KEY
(log_id)
REFERENCES analytics.logs (id) MATCH FULL

ALTER TABLE analytics.gui_logs ADD CONSTRAINT "GUI_LOG_PROFILE_ID" FOREIGN
KEY ("profile_id")
REFERENCES analytics.profiles (id) MATCH FULL

ALTER TABLE analytics.reader_logs ADD CONSTRAINT "READER_LOG_LOG" FOREIGN
KEY (log_id)
REFERENCES analytics.logs (id) MATCH FULL

ALTER TABLE analytics.reader_logs ADD CONSTRAINT "READER_LOG_MODEL_ID"
FOREIGN KEY (model_id)
REFERENCES analytics.models (id) MATCH FULL

ALTER TABLE analytics.reader_logs ADD CONSTRAINT "READER_LOG_FEATURE_ID"
FOREIGN KEY (feature_id)
REFERENCES analytics.features (id) MATCH FULL

ALTER TABLE analytics.reevaluate_logs ADD CONSTRAINT "REEVALUATE_LOG_LOG"
FOREIGN KEY (log_id)
REFERENCES analytics.logs (id) MATCH FULL
```

```
ALTER TABLE analytics.reevaluate_logs ADD CONSTRAINT
"REEVALUATION_LOGS_PROFILE_ID" FOREIGN KEY (profile_id)
REFERENCES analytics.profiles (id) MATCH FULL

ALTER TABLE analytics.game_logs ADD CONSTRAINT "GAME_LOG_LOG" FOREIGN KEY
(log_id)
REFERENCES analytics.logs (id) MATCH FULL

ALTER TABLE analytics.game_logs ADD CONSTRAINT "ASSIGNED_TEACHER" FOREIGN
KEY (assigned_by)
REFERENCES analytics.teachers (user_id) MATCH FULL

ALTER TABLE analytics.game_logs ADD CONSTRAINT "GAME_LOG_ACTIVITY" FOREIGN
KEY (activity_id)
REFERENCES analytics.activities (id) MATCH FULL

ALTER TABLE analytics.activities ADD CONSTRAINT "ACTIVITY_FEATURE" FOREIGN KEY
(feature_id)
REFERENCES analytics.features (id) MATCH FULL

ALTER TABLE analytics.activities ADD CONSTRAINT "ACTIVITY_ACTIVITY_TYPE"
FOREIGN KEY (activity_type)
REFERENCES analytics.activity_types (id) MATCH FULL

ALTER TABLE analytics.activities ADD CONSTRAINT "ACTIVITY_GAME" FOREIGN KEY
(game_id)
REFERENCES analytics.games (id) MATCH FULL

ALTER TABLE analytics.profiles ADD CONSTRAINT "PROFILE_USER" FOREIGN KEY
(user_id)
REFERENCES analytics.students (user_id) MATCH FULL

ALTER TABLE analytics.profiles ADD CONSTRAINT "PROFILE_MODEL" FOREIGN KEY
(model_id)
REFERENCES analytics.models (id) MATCH FULL

ALTER TABLE analytics.anonymized_users ADD CONSTRAINT "ANONYMIZED_USER"
FOREIGN KEY (user_id)
REFERENCES analytics.users (id) MATCH FULL

ALTER TABLE analytics.anonymized_users ADD CONSTRAINT "ADMIN_ID_DELETE"
FOREIGN KEY (admin_id)
REFERENCES analytics.admins (user_id) MATCH FULL

ALTER TABLE analytics.deleted_users ADD CONSTRAINT "ADMIN_ID_DELETED"
FOREIGN KEY (admin_id)
REFERENCES analytics.admins (user_id) MATCH FULL
```

```
ALTER TABLE analytics.last_update_time ADD CONSTRAINT "LAST_UPDATE_CLIENT"
FOREIGN KEY (application_id)
REFERENCES analytics.clients (id) MATCH FULL

ALTER TABLE analytics.features ADD CONSTRAINT "FEATURE_MODEL" FOREIGN KEY
(model_id)
REFERENCES analytics.models (id) MATCH FULL

ALTER TABLE analytics.questions ADD CONSTRAINT "QUESTION_GAME_LOG" FOREIGN
KEY (log_id)
REFERENCES analytics.game_logs (log_id) MATCH FULL

ALTER TABLE analytics.content ADD CONSTRAINT "CONTENT_FEATURE" FOREIGN KEY
(feature_id)
REFERENCES analytics.features (id) MATCH FULL

ALTER TABLE analytics.content ADD CONSTRAINT "CONTENT_WORD" FOREIGN KEY
(word_id)
REFERENCES analytics.words (id) MATCH FULL

ALTER TABLE analytics.question_content ADD CONSTRAINT
"QUESTION_QUESTION_CONTENT" FOREIGN KEY (question_id)
REFERENCES analytics.questions (id) MATCH FULL

ALTER TABLE analytics.question_content ADD CONSTRAINT
"CONTENT_QUESTION_CONTENT" FOREIGN KEY (content_id)
REFERENCES analytics.content (id) MATCH FULL

ALTER TABLE analytics.events ADD CONSTRAINT "EVENT_QUESTION" FOREIGN KEY
(question_id)
REFERENCES analytics.questions (id) MATCH FULL

ALTER TABLE analytics.events ADD CONSTRAINT "EVENT_EVENT_TYPE" FOREIGN KEY
(event_type)
REFERENCES analytics.game_event_type (id) MATCH FULL

ALTER TABLE analytics.events ADD CONSTRAINT "EVENT_CONTENT" FOREIGN KEY
(content_id)
REFERENCES analytics.content (id) MATCH FULL

ALTER TABLE analytics.reevaluations ADD CONSTRAINT "REEVALUATION_FEATURE"
FOREIGN KEY (feature_id)
REFERENCES analytics.features (id) MATCH FULL

ALTER TABLE analytics.reevaluations ADD CONSTRAINT
"REEVALUATION_REEVALUATION_LOG" FOREIGN KEY (log_id)
```

```
REFERENCES analytics.reevaluate_logs (log_id) MATCH FULL

ALTER TABLE analytics.teacher_tool_logs ADD CONSTRAINT
"TEACHER_TOOL_LOGS_LOG" FOREIGN KEY (log_id)
REFERENCES analytics.logs (id) MATCH FULL

ALTER TABLE analytics.teacher_tools_usage ADD CONSTRAINT
"TEACHER_TOOLS_USAGE_TEACHER_TOOL_LOGS" FOREIGN KEY (log_id)
REFERENCES analytics.teacher_tool_logs (log_id) MATCH FULL

ALTER TABLE analytics.teacher_tools_usage ADD CONSTRAINT
"TEACHER_TOOLS_USAGE_STUDENT" FOREIGN KEY (student)
REFERENCES analytics.students (user_id) MATCH FULL

ALTER TABLE analytics.teacher_tools_usage ADD CONSTRAINT
"TEACHER_TOOLS_USAGE_PROFILE" FOREIGN KEY (profile_id)
REFERENCES analytics.profiles (id) MATCH FULL

ALTER TABLE analytics.book_assignment ADD CONSTRAINT
"BOOKS_ASSIGNMENT_STUDENT" FOREIGN KEY (student)
REFERENCES analytics.students (user_id) MATCH FULL

ALTER TABLE analytics.book_assignment ADD CONSTRAINT
"BOOK_ASSIGNMENT_LOG_ID" FOREIGN KEY (log_id)
REFERENCES analytics.teacher_tool_logs (log_id) MATCH FULL

ALTER TABLE analytics.book_assignment ADD CONSTRAINT
"BOOK_ASSIGNMENT_BOOK" FOREIGN KEY (book_id)
REFERENCES analytics.books (id) MATCH FULL

ALTER TABLE analytics.activity_assignment ADD CONSTRAINT
"ACTIVITY_ASSIGNMENT_STUDENT" FOREIGN KEY (student)
REFERENCES analytics.students (user_id) MATCH FULL

ALTER TABLE analytics.activity_assignment ADD CONSTRAINT
"ACTIVITY_ASSIGNMENT_ACTIVITY" FOREIGN KEY (activity_id)
REFERENCES analytics.activities (id) MATCH FULL

ALTER TABLE analytics.activity_assignment ADD CONSTRAINT
"ACTIVITY_ASSIGNMENT_LOG" FOREIGN KEY (log_id)
REFERENCES analytics.teacher_tool_logs (log_id) MATCH FULL

ALTER TABLE analytics.reader_book_activity ADD CONSTRAINT
"READER_BOOK_ACTIVITY_READER_LOG" FOREIGN KEY (log_id)
REFERENCES analytics.reader_logs (id) MATCH FULL
```

```
ALTER TABLE analytics.reader_book_activity ADD CONSTRAINT
"READER_BOOK_ACTIVITY_BOOK" FOREIGN KEY (book_id)
REFERENCES analytics.books (id) MATCH FULL

ALTER TABLE analytics.tricky_word_activity ADD CONSTRAINT
"TRICKY_WORD_ACTIVITY_READER_LOG" FOREIGN KEY (log_id)
REFERENCES analytics.reader_logs (id) MATCH FULL

ALTER TABLE analytics.profile_initialization ADD CONSTRAINT
"PROFILE_INITIALIZATION_LOG_ID" FOREIGN KEY (log_id)
REFERENCES analytics.teacher_tool_logs (log_id) MATCH FULL

ALTER TABLE analytics.profile_initialization ADD CONSTRAINT
"PROFILE_INITIALIZATION_TEACHER" FOREIGN KEY (edit_by)
REFERENCES analytics.teachers (user_id) MATCH FULL

ALTER TABLE analytics.profile_initialization ADD CONSTRAINT
"PROFILE_INITIALIZATION_PROFILE" FOREIGN KEY (profile_id)
REFERENCES analytics.profiles (id) MATCH FULL
```