A database is an organized collection of data. Databases can be designed to do just about anything with information – Track, organize, edit and produce reports. Many great applications and websites have a database as a main structural part.

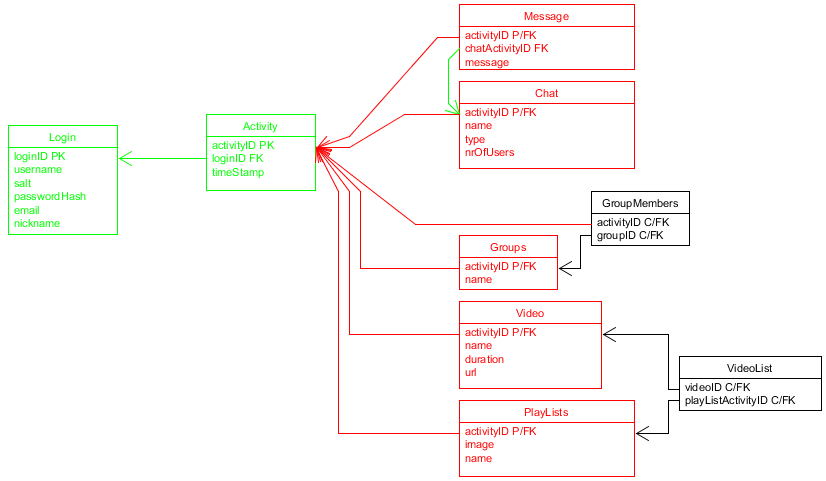
There are two types of database technologies: Relational Databases, which are great at organizing and retrieving structured data; and Non-Relational Databases, which are best used when the data is inconsistent, incomplete or simply massive.

For our project, we have chosen to go with a Relational Database, simply because pros such as: strict ACID support, data normalization, supports joins, limitless indexing, and being one of the most common used technologies\*, outweigh the of having a non-relational database, cons such as: working with joins can be difficult, low concurrency (no random reads and writes), slow mass updates, difficulty tracking schema changes.

As for engines, there are several choices that we considered, for a Relational Database, some of which are: Oracle Database, SQL Server and MySQL; and since all three of them were using dialects of the same language (SQL), it went down to the very basics when we took the decision on which to use.

As a final decision, we chose SQL Server 2014, because of the following: SQL Server executes and commits each instruction, unlike Oracle which requires explicit command to commit the changes; ease of use, since not only were we thought on how to use it, but also compared to Oracle, which give so many other settings and configurations that can be set to the wrong value; and performance.

The following diagram shows how our database currently looks.



As you can see, we have 3 types of tables: tables which hold critical information such as user details and actions (green), tables which specify the type of activity a user has done (red) and tables which’s main purpose is to handle a M-M relationship.

## 2.7. Interesting bits of code

### c. Database Triggers

Since one of the features, that would ease our programming task, we have decided in the begging was cascade deletion, and SQL Server 2014 does not allow such thing when a table is referenced, or references, by multiple foreign keys, we needed to find another way of dealing with phantom data. Shortly after the problem was encountered, we stumbled upon this ingenious way of doing it: Deletion Triggers.

What is a deletion trigger? Well, when a row is deleted from a table, instead of doing the default command, the server would execute this special piece of code, which specifies from which tables to delete and what.

The following image is one of the triggers present in our database.



As you can see, before deleting the entry that was commanded to be deleted, the engine would check in all of the mentioned tables if the entry’s foreign key, is referenced, then proceed to delete that entry, before continuing the check, and finally executing the commanded entry’s deletion.

Reference

\*According to James Serra, Big Data/Data Warehouse Evangelist at Microsoft in a presentation Published on Mar 15, 2016