Applied research

By Sam Philipsen

Inhoud

Problem:

For my website I will need to store data about my user, which should easily be retrievable using my API. For this I am going to need a secure and fast available database. There are a lot of options out there, and I don't know which one is going to be the right one for me.

Main question:

What database programming language is best used for storing standard user information (name, email, password, id etc.) in my project.

Sub questions:

1. Does storing user data need a relational database, or non-relational database?
   Research methods:
   - <u>Community research</u>
     By looking at what other people have done already, I can see what they did and compare it to what I'm doing. This way I know the advantages and disadvantages of the different types of relational databases with the datasets that look like mine.
   - <u>Data analytics</u>
     Measuring and researching my dataset will give me more information on what it needs and doesn't need. If the things it needs match the advantages of a certain database type, then it gets me closer to my decision.
   - <u>Domain modeling</u>
     Similar to the previous method, this will give me more information about my dataset and see what database type gives the most advantages.

2. What kind of tables does a database need for storing user data?
   Research methods:
   - <u>Decomposition</u>
     By breaking my software system and data down into smaller pieces, it gives me a clearer overview of what I'm going to have to do with my data. This way I can already start on the normalizing of the dataset and determine the tables faster.
   - <u>Domain modeling</u>
     This will make me get a better understanding of how the database should look like, which in turn gives me a good overview of what tables the database will need.

3. What is the easiest and most secure database language to transfer data from?
   Research methods:
   - <u>Best good and bad practices.</u>
     Looking at what other people have done gives me a good idea on how the methods they used work. Using this knowledge, I should get a good idea on what the better database languages are.

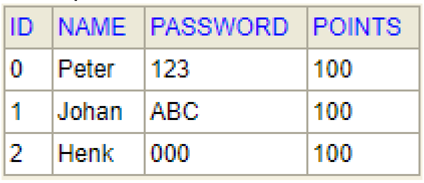
- <u>Document analysis</u>

  To get a better understanding of the most popular database access languages, I can read through their documentation. Once I know more about these languages, I can make a well built up reason for picking the right one.

Results:

1. <u>Does storing user data need a relational database, or non-relational database?</u>
   We can start answering this question by looking at what relational and non-relational databases are. (Pawlan, 2021)

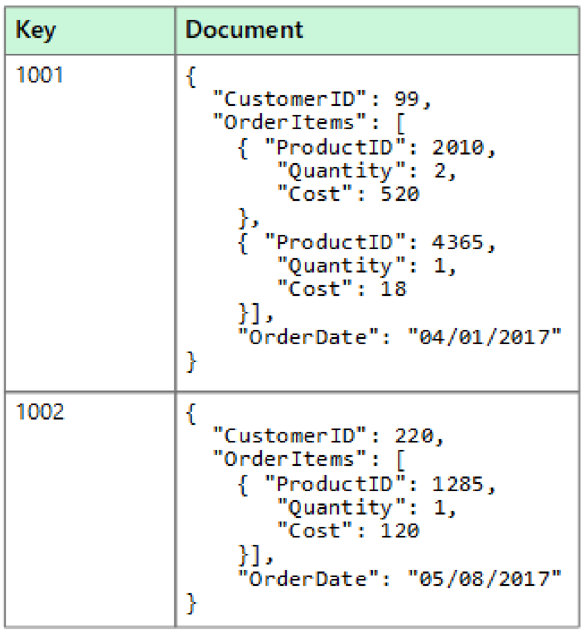
   Relational:
   A relational database stores its information in in tables with columns, rows and data points. The table can be the user, while the columns are things like 'name' and 'id'. You can easily retrieve user data with this type database. You can also connect two tables together with the use of keys.
   Example:



ID	NAME	PASSWORD	POINTS
0	Peter	123	100
1	Johan	ABC	100
2	Henk	000	100

Figure 1

   Non-relational:
   A non-relational database is not structured in the same way as a relational one. It is more stored as a long list of data than a structured table. This is handy for storing data that does not have to be very structured, but has to be accessed fast like a bunch of messages.
   Example:



Key	Document
1001	`{ "CustomerID": 99, "OrderItems": [ { "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }`
1002	`{ "CustomerID": 220, "OrderItems": [ { "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }`

Figure 2

   Analysis of my own dataset
   In my dataset, users have the following data attached to them: *ID, Name, Password, Points*.
   These values are related to the user and should never be assigned to a different user. They may also need to be constantly edited without much delay. In the future

it could also have a connection to a different dataset like games with other users in them. This basically already singles out the relational database as the one I will need to use.

Community research

The following question and its answers mostly talk about authentication purposes, but there are still a bunch of good points made on why a relational database is best for storing user data. For example, storing passwords, editing them retrieving them again is much handier with a relational database as you can easily access the columns of a user and edit them. (Kornelis, 2017)

2. What kind of tables does a database need for storing user data?
   System breakdown
   My system has a backend and frontend. The backend retrieves HTTP requests that come in and handles them accordingly (using my rest API). This currently only means CRUD operations of the user and passing those changes to the local memory database. Currently every aspect of the user's data can be changed. These include, but might change in the future, an ID, name, password, and points.

   The frontend displays the user's information and provides ways of editing and creating user data via the React framework. It does this by making the HTTP requests that the API retrieves. The user can interact with the front end by playing games and losing or winning points based on the outcome. So right now, the points can be changed from the front end.

   Domain modeling
   My system will be used by people wanting to play the games hosted on the website. Considering there are going to be a lot of people on the site, there has to be a unique value in every user to identify an individual. An easy way to do this is by giving them a unique ID. Another way to identify people is by having them input a name.

3. What is the easiest and most secure database language to transfer data from?
   Best good and bad practices
   For this we look at how Facebook stores its data. Facebook uses a lot of different database engines, put together in a Polyglot Persistence system. This means it uses a lot of different databases and data models. These languages are all based on MySQL, and some of the data is also stored on the local memory/cache (Shivang, 2021). Using this many varieties of databases is not necessary for a system of my size, however. MySQL is still the primary database Facebook uses and most of the other databases are based on it as well.

   Document analysis
   Considering Facebook (and other popular systems like Twitter and Tesla) use MySQL. I have done more analysis on MySQL and how it works.

MySQL is a database management system that is used for relational databases. It’s multi-threaded, scalable and designed for heavy load production systems. It works, very simply, by retrieving a request form a user and returning a response. This request can be any CRUD (Create, Read, Update, Delete) operation. MySQL uses B-tree tables which, simply put, is a way to access and store user data really fast (Actian, 2018). Furthermore, MySQL supports a lot of different datatypes and is very secure (Kili, 2020).

This is all very simplified to avoid going into the deep specifics, as this is a very complicated topic.

Conclusion:

Question: Does storing user data need a relational database, or non-relational database?
Answer: To easily access a user's data and store or update it in a convenient, structured way, a relational database is the best answer for this. A non-relational database simply does not provide the necessary structure to easily retrieve and store a user's data. It also does not provide a way to join two tables together which is, while not a necessarily important for just storing user data, a vital part for the expansion of a database.

Question: What kind of tables does a database need for storing user data?
Answer: Not every database is the same, but there are some tables and data points a lot of user data needs. Some of the most common data points every user database needs is an ID, name and password.

Question: What is the easiest and most secure database language to transfer data from?
Answer: MySQL. It is simply one of the more modern, widely used databases that is being used by millions of systems worldwide. It provides the fast data access, security and scalability that nearly every dataset should need.

Recommendations:

If need a database for storing your user’s data, it is a good idea to get a relational database. To be more specific, a MySQL database. Every dataset has different tables and columns which it needs, but most user databases have at least a ‘User’ table with the columns ‘ID’, ‘Name’, and ‘Password’.

Bibliografie

Actian. (2018, 11 5). *Structure of a B-tree Table*. Opgehaald van Actian: https://docs.actian.com/ingres/10s/index.html#page/DatabaseAdmin/Structure_of_a_B-tree_Table.htm

Kili, A. (2020, 6 9). *What is MySQL? How does MySQL work?* Opgehaald van tecmint: https://www.tecmint.com/what-is-mysql-how-does-mysql-work/

Kornelis, H. (2017, 11 29). *Which type of database is best to store user details for authentication purposes?* Opgehaald van quora.com: https://www.quora.com/Which-type-of-database-is-best-to-store-user-details-for-authentication-purposes

Pawlan, D. (2021, 11 5). *Relational vs. Non-Relational Database: Pros & Cons*. Opgehaald van aloa Blog: https://aloa.co/blog/relational-vs-non-relational-database-pros-cons

Shivang. (2021, 11 5). *Facebook Database [Updated] – A Thorough Insight Into The Databases Used @Facebook*. Opgehaald van Scale your app: https://www.scaleyourapp.com/what-database-does-facebook-use-a-1000-feet-deep-dive/

Resources:

https://www.xplenty.com/blog/which-database/
https://aloa.co/blog/relational-vs-non-relational-database-pros-cons#:~:text=So%2C%20what's%20the%20difference%3F,of%20a%20laundry%20list%20order.
https://www.quora.com/Which-type-of-database-is-best-to-store-user-details-for-authentication-purposes
https://www.scaleyourapp.com/what-database-does-facebook-use-a-1000-feet-deep-dive/
https://www.scaleyourapp.com/what-database-does-twitter-use-a-deep-dive/
https://www.mysql.com/why-mysql/
https://www.tecmint.com/what-is-mysql-how-does-mysql-work/
https://docs.actian.com/ingres/10s/index.html#page/DatabaseAdmin/Structure_of_a_B-tree_Table.htm