Databases

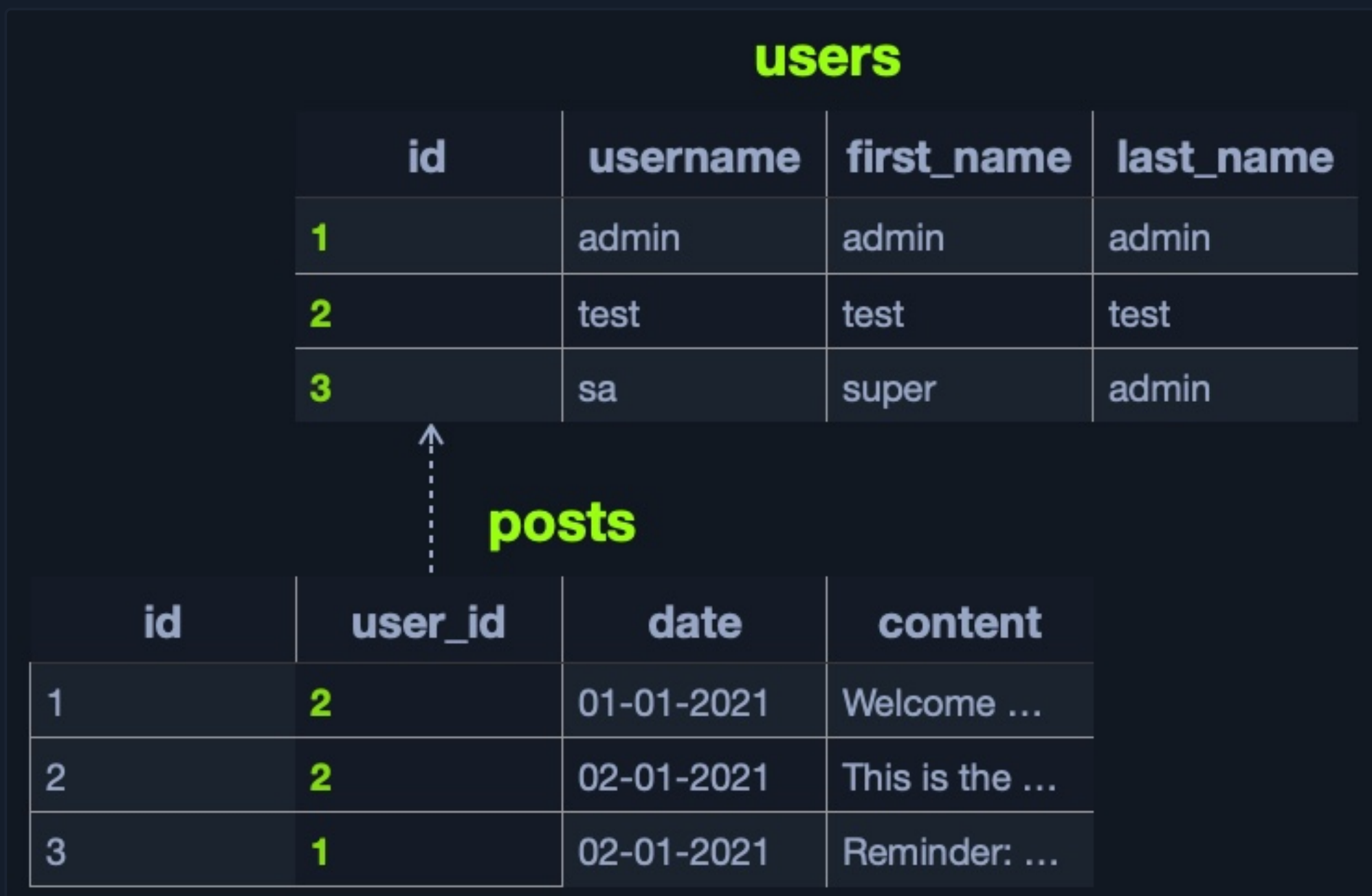
Web applications utilize back end databases to store various content and information related to the web application. This can be core web application assets like images and files, web application content like posts and updates, or user data like usernames and passwords. This allows web applications to easily and quickly store and retrieve data and enable dynamic content that is different for each user.

There are many different types of databases, each of which fits a certain type of use. Most developers look for certain characteristics in a database, such as `speed` in storing and retrieving data, `size` when storing large amounts of data, `scalability` as the web application grows, and `cost`.

Relational (SQL)

Relational (SQL) databases store their data in tables, rows, and columns. Each table can have unique keys, which can link tables together and create relationships between tables.

For example, we can have a `users` table in a relational database containing columns like `id`, `username`, `first_name`, `last_name`, and so on. The `id` can be used as the table key. Another table, `posts`, may contain posts made by all users, with columns like `id`, `user_id`, `date`, `content`, and so on.



We can link the `id` from the `users` table to the `user_id` in the `posts` table to easily retrieve the user details for each post, without having to store all user details with each post.

A table can have more than one key, as another column can be used as a key to link with another table. For example, the `id` column can be used as a key to link the `posts` table to another table containing comments, each of which belongs to a certain post, and so on.

The relationship between tables within a database is called a Schema.

This way, by using relational databases, it becomes very quick and easy to retrieve all data about a certain element from all databases. For example, we can retrieve all details linked to a certain user from all tables with a single query. This makes relational databases very fast and reliable for big datasets that have a clear structure and design. Databases also make data management very efficient.

Some of the most common relational databases include:

Type	Description
MySQL	The most commonly used database around the internet. It is an open-source database and can be used completely free of charge
MSSQL	Microsoft's implementation of a relational database. Widely used with Windows Servers and IIS web servers
Oracle	A very reliable database for big businesses, and is frequently updated with innovative database solutions to make it faster and more reliable. It can be costly, even for big businesses
PostgreSQL	Another free and open-source relational database. It is designed to be easily extensible, enabling adding advanced new features without needing a major change to the initial database design

Other common SQL databases include: `SQLite`, `MariaDB`, `Amazon Aurora`, and `Azure SQL`.

Non-relational (NoSQL)

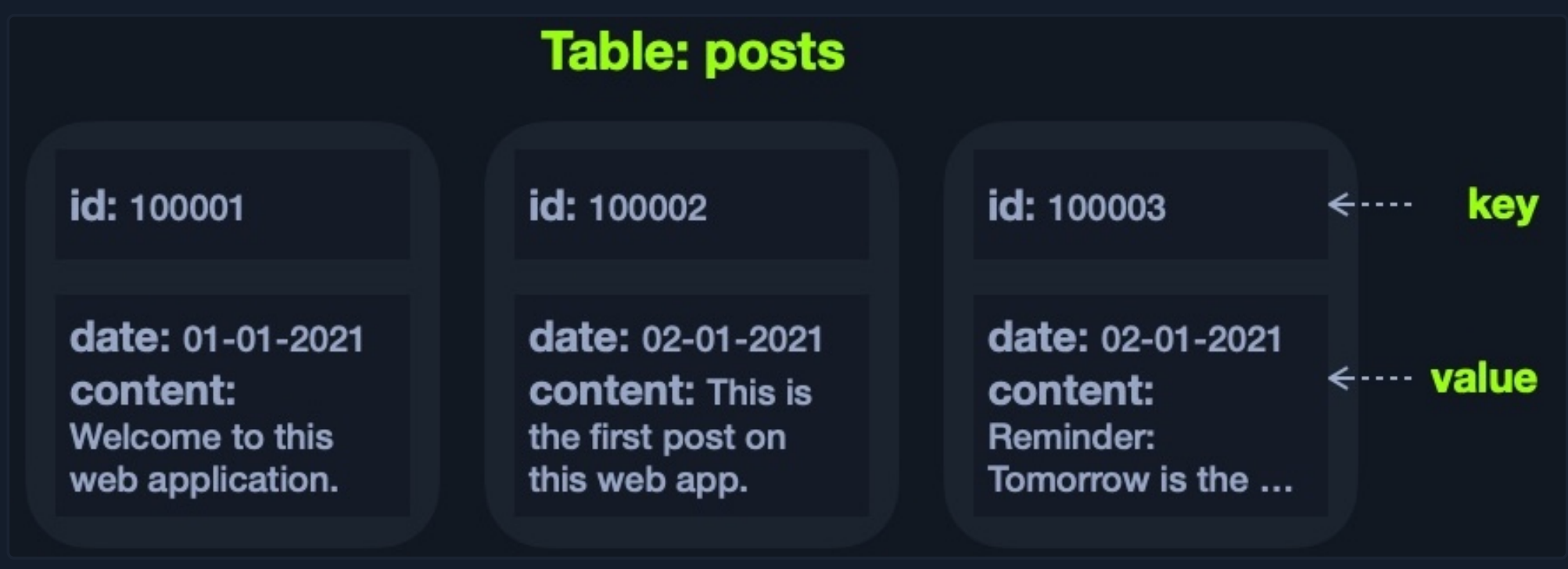
A non-relational database does not use tables, rows, columns, primary keys, relationships, or schemas. Instead, a `NoSQL` database stores data using various storage models, depending on the type of data stored.

Due to the lack of a defined structure for the database, `NoSQL` databases are very scalable and flexible. When dealing with datasets that are not very well defined and structured, a `NoSQL` database would be the best choice for storing our data.

There are 4 common storage models for `NoSQL` databases:

- `Key-Value`
- `Document-Based`
- `Wide-Column`
- `Graph`

Each of the above models has a different way of storing data. For example, the `Key-Value` model usually stores data in `JSON` or `XML`, and has a key for each pair, storing all of its data as its value:



The above example can be represented using `JSON` as follows:

Code: json

```
{
  "100001": {
    "date": "01-01-2021",
    "content": "Welcome to this web application."
  },
  "100002": {
    "date": "02-01-2021",
    "content": "This is the first post on this web app."
  },
  "100003": {
    "date": "02-01-2021",
    "content": "Reminder: Tomorrow is the ..."
  }
}
```

It looks similar to a dictionary/map/key-value pair in languages like `Python` or `PHP` 'i.e. `{'key':'value'}`', where the `key` is usually a string, the `value` can be a string, dictionary, or any class object.

The `Document-Based` model stores data in complex `JSON` objects and each object has certain meta-data while storing the rest of the data similarly to the `Key-Value` model.

Some of the most common `NoSQL` databases include:

Type	Description
MongoDB	The most common `NoSQL` database. It is free and open-source, uses the `Document-Based` model, and stores data in `JSON` objects
ElasticSearch	Another free and open-source `NoSQL` database. It is optimized for storing and analyzing huge datasets. As its name suggests, searching for data within this database is very fast and efficient
Apache Cassandra	Also free and open-source. It is very scalable and is optimized for gracefully handling faulty values

Other common `NoSQL` databases include: `Redis`, `Neo4j`, `CouchDB`, and `Amazon DynamoDB`.

Use in Web Applications

Most modern web development languages and frameworks make it easy to integrate, store, and retrieve data from various database types. But first, the database has to be installed and set up on the back end server, and once it is up and running, the web applications can start utilizing it to store and retrieve data.

For example, within a `PHP` web application, once `MySQL` is up and running, we can connect to the database server with:

Code: php

```php
$conn = new mysqli("localhost", "user", "pass");
```

Then, we can create a new database with:

Code: php

```php
$sql = "CREATE DATABASE database1";
$conn->query($sql)
```

After that, we can connect to our new database, and start using the `MySQL` database through `MySQL` syntax, right within `PHP`, as follows:

Code: php

```php
$conn = new mysqli("localhost", "user", "pass", "database1");
$query = "select * from table_1";
$result = $conn->query($query);
```

Web applications usually use user-input when retrieving data. For example, when a user uses the search function to search for other users, their search input is passed to the web application, which uses the input to search within the database(s).

Code: php

```php
$searchInput =  $_POST['findUser'];
$query = "select * from users where name like '%$searchInput%'";
$result = $conn->query($query);
```

Finally, the web application sends the result back to the user:

Code: php

```php
while($row = $result->fetch_assoc() ){
    echo $row["name"]."<br>";
}
```

This basic example shows us how easy it is to utilize databases. However, if not securely coded, database code can lead to a variety of issues, like SQL Injection vulnerabilities.

Questions

Answer the question(s) below to complete this Section and earn cubes!

+ 1 What type of database is Google's Firebase Database?

Submit your answer here...

Submit

Hint

Previous Next

Go to Questions

Table of Contents

Cascading Style Sheets (CSS)

JavaScript

Front End Vulnerabilities

Sensitive Data Exposure

HTML Injection

Cross-Site Scripting (XSS)

Cross-Site Request Forgery (CSRF)

Back End Components

Back End Servers

Web Servers

Databases

Development Frameworks & APIs

Back End Vulnerabilities

Common Web Vulnerabilities

Public Vulnerabilities

Next Steps

Next Steps

My Workstation

OFFLINE

Start Instance

1 / 1 spawns left