Dive into MongoDB & PostgreSQL

Contents: MongoDB, PostgreSQL, SQL and NoSQL databases

ML-learners · Follow

Published in SFU Professional Computer Science
13 min read · Feb 11, 2023

CR: https://www.toptal.com/database/database-design-bad-practices

> *In this post, we will use PostgreSQL and MongoDB as examples to introduce the differences between these two types of databases and their respective benefits. Read on to learn when and where you can apply them.*

Authors: Guanghua Yang, Jianan Li, Kelvin Li, Lexi Ke, Freya Li

This blog is written and maintained by students in the Master of Science in Professional Computer Science Program at Simon Fraser University as part of their course credit. To learn more about this unique program, please visit sfu.ca/computing/mpcs.

Table of Content

- **I. Introduction**
- **II. MongoDB**
  - Overview of MongoDB and NoSQL databases
  - MongoDB Basic Concepts
  - Get started with MongoDB
- **III. PostgreSQL**
  - Overview of PostgreSQL
  - How to use PostgreSQL
- **IV. Differences between MongoDB and PostgreSQL**
  - Data Model
  - Query language
  - Performance
- **V. Use cases**
  - MongoDB
  - PostgreSQL
- **VI. Conclusion**

I. Introduction

Are you aware that numerous service providers will collect your data while you use your computer and mobile device? Your present position will be utilized, for instance, when you use Google Maps to navigate; when you shop online, your credit card information will be used to pay the fees; etc. Do you know how is such a massive amount of data collected and stored to better the lives of individuals?

These pieces of information are kept in a place called a database. It's like a huge warehouse that can hold a lot of data and be used by many people at the same time. Because of this, the database is an important part of all information systems, no matter how large or small they are or what type of business they are for.

Since there is so much information in the database, we need to use a tool to work with it quickly and effectively. “Database Management System” is the name of this tool. The first generation of DBMS was released in the 1960s, and since then, various types have been made for different use. And the two most popular forms are relational databases, like PostgreSQL, and NoSQL databases, like MongoDB.

In this post, we will use PostgreSQL and MongoDB as examples to introduce the differences between these two types of databases and their respective benefits. Read on to learn when and where you can apply them.

II. MongoDB

1. Overview of MongoDB and NoSQL databases

MongoDB is a widely used cross-platform document-oriented NoSQL database. It uses a document-oriented data model with optional schemas, which allows it to store data in the form of JSON-like documents, hence providing more flexibility than traditional relational databases. There exist many other NoSQL databases that used different models: e.g. Redis used the key-value pair model, Cassandra and Bigtable used the wide-column model etc.

2. MongoDB Basic Concepts

2.1 Document

A document in MongoDB is a JSON-like piece of record that can consist of multiple fields, where each field can contain different types of data e.g. string, integer, date, or even a primary key referring to another document. It is equivalent to one row in

relational databases. Below is an example of a document, the “_id” field is an automatically generated primary key for this document.

```
{
   "_id": ObjectId("5f1437b936c67f5c04adf38a"),
   "name": "David Wilson",
   "age": 25,
   "address": {
   "street": "987 Broadway",
   "city": "New York",
   "state": "NY",
   "zipcode": 10010
   },
   "employee_since": ISODate("2022-01-01T00:00:00Z")
}
```

2.2 Collection

A collection in MongoDB consists of a group of related documents, it is equivalent to a table in a relational database. Yet the collection by default will not strictly enforce a fixed schema (the key difference between a collection and a table), which means the documents in one collection can have different fields.

3. Get started with MongoDB

3.1 Connect to MongoDB

3.1.1 MongoDB deployment

Before connecting to MongoDB, a MongoDB instance needs to be created first, it can be either an instance deployed using MongoDB Atlas or a standalone instance hosted on a user’s own machine.

MongoDB Atlas is a fully managed service in the cloud, allowing for easy setup without the need to manage the underlying infrastructure. Standalone MongoDB instances require software downloads and management infrastructure while providing the user with greater control and customization.

3.1.2 **Using MongoDB Shell**

The MongoDB Shell (mogosh) is one of the options to connect to users' MongoDB instances, it is a Node.js environment for interacting with MongoDB instances.

- Installation

The MongoDB Shell can be downloaded via the official download center: https://www.mongodb.com/try/download/shell

- Connect to cloud hosted deployment (MongoDB Atlas)

The following Shell command will allow the user to connect to its MongoDB Atlas deployment:

```
mongosh "mongodb+srv://cluster0.nuu4c9d.mongodb.net/<DataBase Name>" - username
```

- Connect to a local deployment

The following Shell command will allow the user to connect to its own deployment:

```
# standalone deployment on a local host
mongosh "mongodb://localhost:27017"

# standalone deployment on remote host
mongosh "mongodb://mongodb0.example.com:27017"
```

3.1.3 **Using MongoDB Driver**

Users can also connect to MongoDB directly from their application using the official libraries. Below are two examples of connecting to MongoDB using Node.js and Python provided in MongoDB's documentation:

```
const MongoClient = require('mongodb').MongoClient;
const uri = "mongodb+srv://kay:<Password>@cluster0.mongodb.net/test?w=majority"
const client = new MongoClient(uri, { useNewUrlParser: true });
client.connect(err => {
const collection = client.db("test").collection("devices");
// perform CRUD operations
```

```
client.close();
});
```

```
import pymongo
import dns
client = pymongo.MongoClient("mongodb+srv://kay:<Password>@cluster0.mongodb.net
db = client.test
```

3.2 MongoDB CRUD Operations

1. Create a document (collection)

Document can be inserted into a collection through MongoDB Shell using db.<collection>.insertOne(). Below is an example of inserting a document into the inventory collection. Note that if there wasn't a collection named "soundtrack" prior to this command being executed, this command will generate the "soundtrack" collection automatically.

```
db.soundtrack.insertOne(
{ name: "soundtrack01", status: "A", length: { time: 351, uom: "sec" } }
)
```

2. Query documents

Documents can be queried through MongoDB Shell using db.<collection>.find(), note that the keywords of the query are denoted by a dollar sign "$". Below is an example that retrieves all documents from the soundtrack collection where the "status" field is either "A" or "C":

```
db.soundtrack.find( { status: { $in: [ "A", "C" ] } } )
```

This is equivalent to this SQL statement:

```
SELECT * FROM soundtrack WHERE status in ("A", "C")
```

3. Update documents

Documents can be updated through MongoDB Shell using db.<collection>.updateOne(), note that the keywords of the query are denoted by a dollar sign “$”. Below is an example that updates the first document where the field “name” equals “soundtrack01”, the $set keyword will update the field “uom” to “min” and the field ”updated” to “P”.

```
db.soundtrack.updateOne(
{ name: "soundtrack01" },{ $set: { "uom": "min", "updated": "P" }}
)
```

4. Delete documents

Documents can be deleted through MongoDB Shell using db.<collection>.deleteOne(). Below is an example that deletes the first document where the field “name” equals “soundtrack01”

```
db.inventory.deleteOne( {"name": "soundtrack01" } )
```

5. Create a collection with a validation schema

Although by default the collection in MongoDB doesn’t enforce a schema, it can enforce a validation schema if provided with the requirements. When validation schema is enforced, documents that didn’t fulfill the requirements will not be able to be saved into the collection. Below is an example of a validation schema that requires two fields: “username” and “password” where the username has to be a string, and the password also has to be a string with at least 8 characters.

```
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["username", "password"],
      properties: {
        name: {
          bsonType: "string"
        },
        password: {
          bsonType: "string",
          minLength: 8
        }
      }
    }
  }
})
```

III. PostgreSQL

1. Overview of PostgreSQL

PostgreSQL is an open-source relational database management system (RDBMS) that is well-known for its advanced SQL features, scalability, high availability, and strong security and data integrity. It is a community-driven development project that emphasizes scalability and standard compliance. PostgreSQL provides a range of advanced SQL functions, including support for complex data types and transactions. The system is highly scalable and features built-in replication and connection pooling for high availability. It is capable of processing complex queries efficiently even at large scales and has robust security features like fine-grained access control and encryption. The user community of PostgreSQL is extensive and well-developed, providing ample resources for users.

2. How to use PostgreSQL

2.1 Get started with PostgreSQL

Here are a few steps you need to know to get started with PostgreSQL.

- Installation: Download the latest version of PostgreSQL from the official website PostgreSQL: Downloads, and follow the instructions on the page.
- Database set up: After installation of PostgreSQL, you can choose to create a new database either through the default PostgreSQL database or a new one. To implement successfully, you can run the 'createdb' command in the terminal.
- Database connection: Use the 'psql' command to connect with the database. As requested, you will provide a PostgreSQL username and password for permission to enter.
- Workspace: Once connected to the database, we can then create tables for the first step, and then we can choose to insert data, execute queries, etc.
- Further learning: Structured Query Language (known as SQL) is the most standard language tool to use for relational databases. Therefore, it is recommended for users to learn SQL to get the most understanding of the usage of PostgreSQL.

2.2 *PostgreSQL CRUD Operation*

2.2.1 SQL version CRUD

SQL (Structured Query Language) Example for CRUD operations in PostgreSQL:

1. Create Table:

A table can be created through PostgreSQL using the <CREATE TABLE> command. This CREATE operation is to create a Table named "users" in a database. This table has 4 attributes, including id (as the primary key), name, email, and age.

```
CREATE TABLE users (
id SERIAL PRIMARY KEY,
name TEXT NOT NULL,
email TEXT NOT NULL UNIQUE,
```

```
    age INTEGER NOT NULL
    );
```

2. Query Record:

Records can be queried through PostgreSQL using the <SELECT> command. This READ operation returns all the data or records from the tables “users” in the database.

```
SELECT * FROM users;
```

3. Update Record:

Records can be updated through PostgreSQL using the <UPDATE> command. This UPDATE operation updates the age field to 30 for the user with id = 1 in the database.

```
UPDATE users
SET age = 30
WHERE id = 1;
```

4. Delete Record:

Records can be deleted through PostgreSQL using the <DELETE> command. Below is an example that delete will delete records from the database <users> that have its ‘id’ equal to 1.

```
DELETE FROM users
WHERE id = 1;
```

2.2.2 NoSQL version CRUD

1. Creating Table

In PostgreSQL, you can create a record by using the 'INSERT' statement to insert a JSON object into a table column that has been defined as the 'json' or 'jsonb' data type. For example:

```
CREATE TABLE customers ( id serial PRIMARY KEY, customer_info jsonb );
INSERT INTO customers (customer_info)
VALUES ('{"name": "John Doe", "email": "jdoe@example.com", "age": 30}');
```

2. Querying Records

To query records in PostgreSQL, you can use the 'SELECT' statement along with the '->>' operator to extract specific values from the JSON object stored in the table. For example:

```
SELECT customer_info->>'name' AS name, customer_info->>'email' AS email, custom
```

3. Updating Records

To update a record in PostgreSQL, you can use the 'UPDATE' statement along with the '->>' operator to modify specific values within the JSON object stored in the table. For example:

```
UPDATE customers
SET customer_info = customer_info || '{"age": 31}'
WHERE customer_info->>'name' = 'John Doe';
```

4. Deleting Records

To delete a record in PostgreSQL, you can use the 'DELETE' statement along with a 'WHERE' clause to specify the document you want to delete. For example:

```
DELETE
FROM customers
WHERE customer_info->>'name' = 'John Doe';
```

IV. Differences between MongoDB and PostgreSQL

1. Data Model

MongoDB uses a document-based data model, while PostgreSQL uses a relational data model.

MongoDB uses a document-oriented data model, which is based on a collection of JSON-style documents. In this model, each document can have a different structure, allowing for flexible and dynamic data modeling. Because of this, MongoDB is a good choice for companies requiring adaptable and dynamic data structures, like web and mobile apps, real-time analytics, and content management systems. This document-based model allows for easy aggregation and scalability that can handle complex queries using document nested structures and indexes.

On the other hand, PostgreSQL uses a relational data model, based on a collection of tables with defined schemas. In this model, data is organized into tables, where each row represents an individual record, and each column corresponds to a specific field of the record. Data in the tables is organized in a way that eliminates

redundant information and ensures data integrity through constraints and referential integrity. This characteristic makes PostgreSQL a powerful tool for companies that require strict data consistency and complex data manipulations, such as financial systems, data warehousing, and business intelligence software.

In general, the choice between MongoDB and PostgreSQL will depend on the specific requirements, including the size and complexity of the data being stored, the specific workload demands, and the balance between flexible data modelling and strict data consistency.

2. Query language

MongoDB and PostgreSQL have different query languages because of the different data models used that they choose.

MongoDB uses a query language based on the MongoDB query syntax, which is designed to support querying and manipulating the semi-structured data stored in MongoDB documents. MQL is used to search and retrieve data from MongoDB collections. The syntax is straightforward and easy to understand, even for non-technical users. This makes MongoDB a good choice for developers who need to quickly prototype and build applications.

Meanwhile, PostgreSQL uses a query language based on the Structured Query Language (SQL), which is a widely used and well-established standard for relational databases. The SQL query language supports a wide range of query types, including simple select statements, as well as more complex join operations, subqueries, and transactions. SQL also supports the use of subqueries, transactions, and stored procedures for more complex data manipulations. Queries in PostgreSQL can be executed using the SELECT statement, and results can be returned in a variety of formats, including tables, arrays, and key-value pairs.

3. Performance

MongoDB is faster for read-heavy or write-heavy workloads, while PostgreSQL is better suited for complex transactions and data analysis.

MongoDB is designed for high performance and scalability and is well-suited for handling large amounts of unstructured and semi-structured data. It achieves this performance through several optimizations, including its document-oriented data model, horizontal scaling through sharding, in-memory storage capabilities, and

indexing support. Its flexible data model also allows for easy aggregation and scalability, as well as efficient storage and retrieval of data.

In addition, fast data access and retrieval are made possible by in-memory storage and indexing, and MongoDB's query language offers a wide range of operators and functions for data filtering and transformation. Additionally, working with big datasets can be more performant thanks to MongoDB's usage of memory-mapped files.

PostgreSQL is widely known for its performance, stability, and reliability. It is designed to handle large amounts of structured data and provides a range of features for optimizing performance, including indexing, query optimization, and caching. PostgreSQL also allows for the efficient retrieval of data based on specific conditions. The query optimizer in PostgreSQL uses cost-based analysis to select the most efficient execution plan for each query, and the database also supports in-memory caching of data, which can greatly improve query performance.

Furthermore, PostgreSQL provides a range of advanced features, such as multi-version concurrency control, which enables effective management of multiple concurrent transactions, and the ability to create custom data types and aggregates. These features can be used to optimise data storage and retrieval for particular use cases.

V. Use cases

MongoDB:

MongoDB is a popular NoSQL database known for its scalability, high performance, and flexible data model.

> *Case 1: E-commerce Platforms*

MongoDB can be used to store and manage large amounts of customer data, such as product information, user profiles, and order history. Its support for efficient querying and indexing makes it well-suited for e-commerce platforms, where fast and responsive performance is critical. For example, an online retailer could use MongoDB to store and manage information about its products, customers, and orders, allowing for fast and personalized shopping experiences for users.

Real World Example: eBay: https://www.mongodb.com/blog/post/ebay-building-mission-critical-multi-data-center-applications-with-mongodb

> *Case 2: Real-time Analytics*

In this use case, MongoDB can be used to store and analyze large amounts of data in real-time. MongoDB's support for horizontal scaling and efficient indexing makes it well-suited for processing high-volume, high-velocity data streams. For example, a financial services company could use MongoDB to store and analyze stock market data in real-time, allowing traders to make informed decisions based on up-to-date information.

Real World Example: Weather Channel: https://www.mongodb.com/press/mongodb-chosen-technology-provider-deliver-accurate-data-weather-channel-hundreds-thousands

PostgreSQL

PostgreSQL is a sophisticated, open-source relational database solution designed for enterprise use. It offers the ability to run both SQL and JSON queries, allowing for both relational and non-relational data management.

> *Case 1: Web technology and NoSQL workloads*

To meet the demands of consumers, a modern website may have to handle a large number of requests, potentially ranging from tens of thousands to hundreds of thousands per second. The PostgreSQL community has made great efforts to address the scalability issues.

PostgreSQL can work seamlessly with a wide range of web frameworks, including but not restricted to: Ruby on Rails, PHP, node.js, Hibernate, Django (Python), and a lot more. The replication capabilities of PostgreSQL make it simple to expand websites to as many database servers as necessary.

Aside from functioning as a relational database, PostgreSQL also has the capability to perform as a NoSQL-style data storage system. There does not have to be a conflict between the relational environment and the document-oriented environment. Both are available in a single product.

Real-World Example: Reddit

> *Case 2: Financial Industry*

For the financial sector, PostgreSQL is ideal. Since PostgreSQL complies with all ACID requirements, it is perfect for OLTP (Online Transaction Processing) tasks. PostgreSQL's exceptional OLTP abilities made it a top choice, it is also a strong analytical database that integrates well with mathematical tools like MATLAB and R.

Real World Example: Capital One and American Express

VI. Conclusion

In this blog post, we first provided a brief overview of two popular databases: MongoDB and PostgreSQL, then contrasted their benefits and drawbacks, and last demonstrated various use cases for each. I hope this has given you a better grasp of them. MongoDB is a NoSQL database with a flexible database that is ideal for applications that need to constantly update unstructured data. The relational database PostgreSQL, on the other hand, supports complex query operations and can ensure data integrity when using structured data models. These two databases have their own strengths and weaknesses, and users need to choose according to the needs of the program.

If you want to know more about these two databases, you can refer to the following materials:

- https://www.mongodb.com/compare/mongodb-postgresql
- https://www.mongodb.com/nosql-explained/nosql-vs-sql

VII. References

https://www.mongodb.com/docs/manual/core/schema-validation/specify-json-schema/specify-allowed-field-values/

https://en.wikipedia.org/wiki/PostgreSQL

https://www.mongodb.com/docs/manual/crud/

https://www.mongodb.com/docs/atlas/driver-connection/

https://www.mongodb.com/docs/mongodb-shell/connect/

https://aws.amazon.com/rds/postgresql/what-is-postgresql/

https://www.cybertec-postgresql.com/en/postgresql-overview/solutions-who-uses-postgresql/

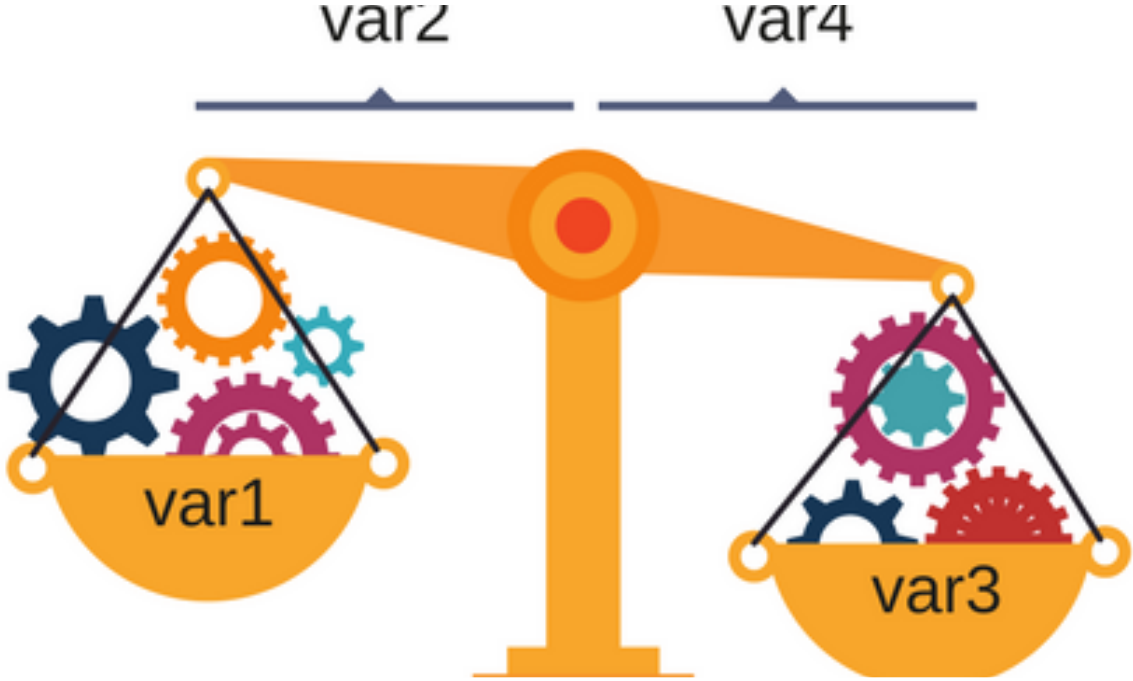
Mongodb Postgresql NoSQL Database Sql

Follow

Written by ML-learners

3 Followers · Writer for SFU Professional Computer Science

More from ML-learners and SFU Professional Computer Science



Kyoun Huh in SFU Professional Computer Science

Surviving In a Random Forest with Imbalanced Datasets

CONTRIBUTORS: Bilal Hussain, Kyoun Huh, Hon Wing Eric Chan, Sakina Patanwala

Feb 13, 2021 384 4

Rohan Harode in SFU Professional Computer Science

XGBoost: A Deep Dive into Boosting

Every day we hear about the breakthroughs in Artificial Intelligence. However, have you wondered what challenges it faces?

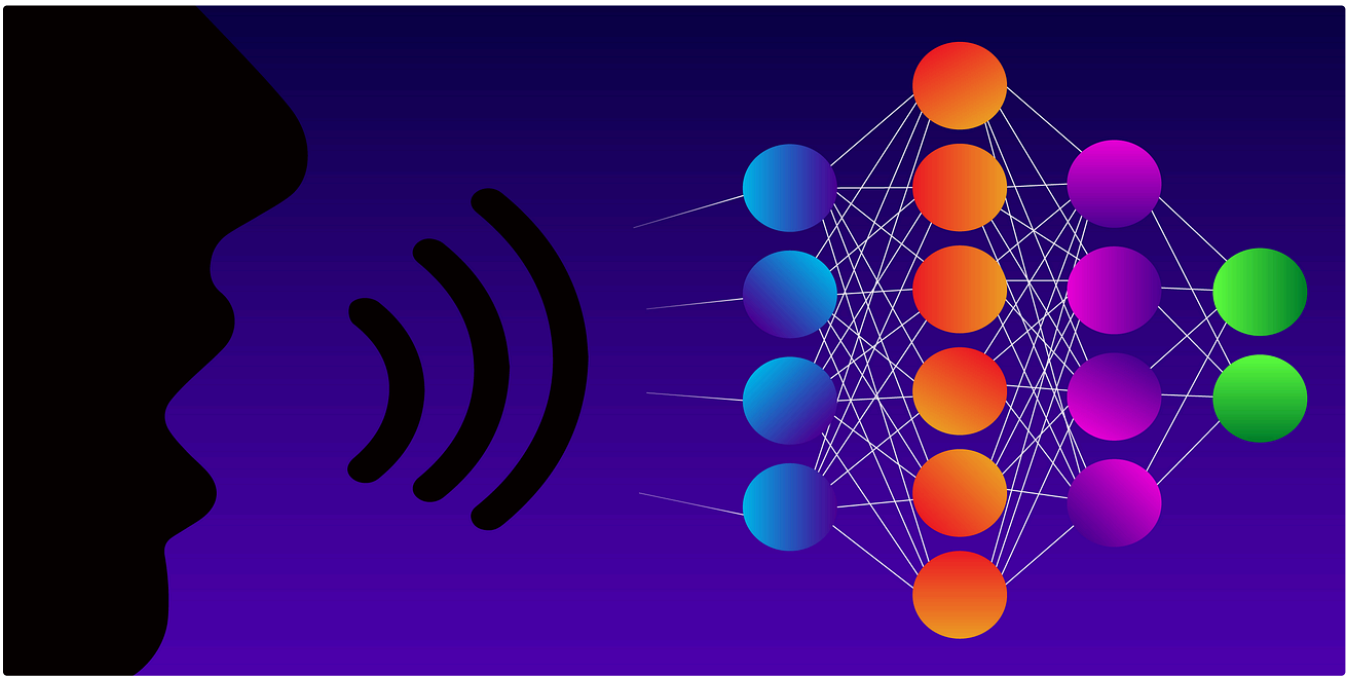
Feb 4, 2020 2.2K 3

Hong Nguyen in SFU Professional Computer Science

MLflow—a modern MLOps tool for data project collaboration

A comprehensive introduction to MLflow and its capabilities

Feb 11, 2023 3.8K 4



Fangyug in SFU Professional Computer Science

NLP: Word Embedding Techniques for Text Analysis

Fangyu Gu, Srijeev Sarkar, Yizhou Sun, Hengzhi Wu, Kacy Wu

Feb 4, 2020 1.1K 2

See all from ML-learners

See all from SFU Professional Computer Science

Recommended from Medium

Vipul Vyas

MongoDB Internal—Part 1

Inserting Data into MongoDB: Understanding the Process

Mar 13 7

Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

Jun 1 15.1K 231

Lists

ChatGPT
21 stories · 733 saves

data science and AI
40 stories · 203 saves

Natural Language Processing
1604 stories · 1165 saves

Afan Khan in JavaScript in Plain English

Microsoft is ditching React

Here's why Microsoft considers React a mistake for Edge.

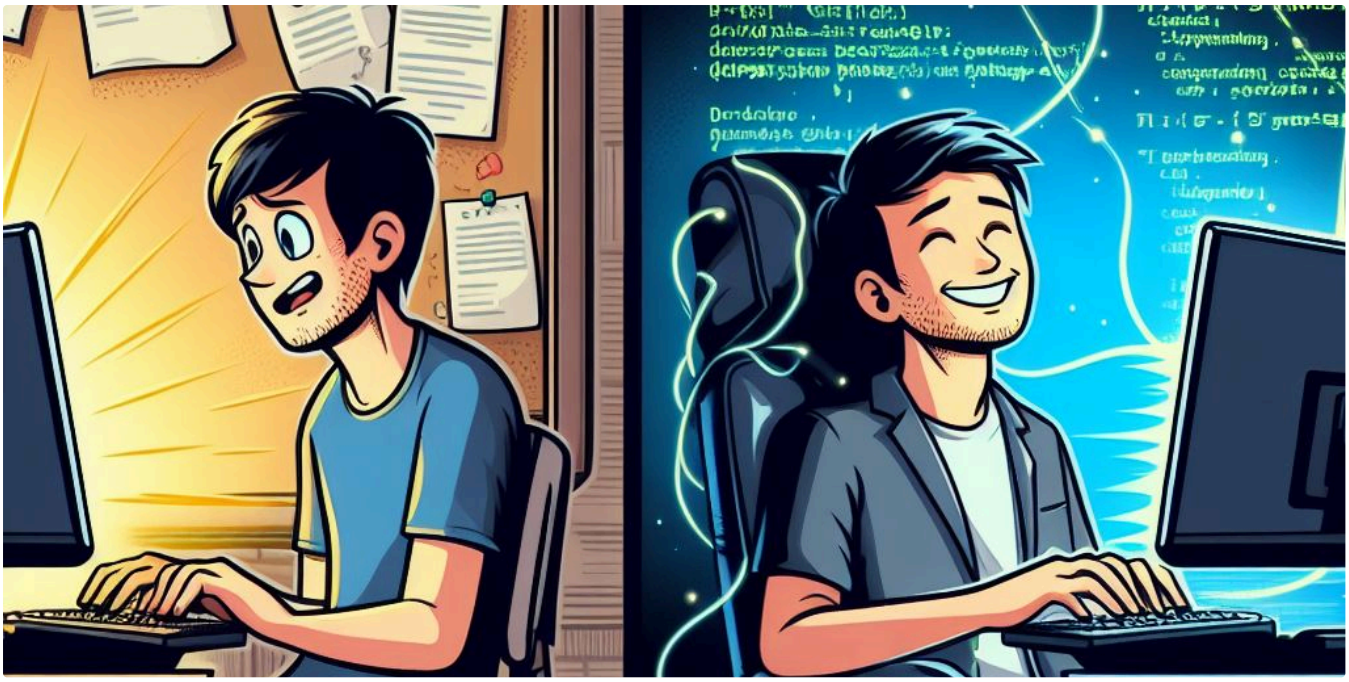
Jun 6 2.9K 66

Daniel Craciun

Stop Using UUIDs in Your Database

How UUIDs can Destroy SQL Database Performance

May 17 2.2K 91

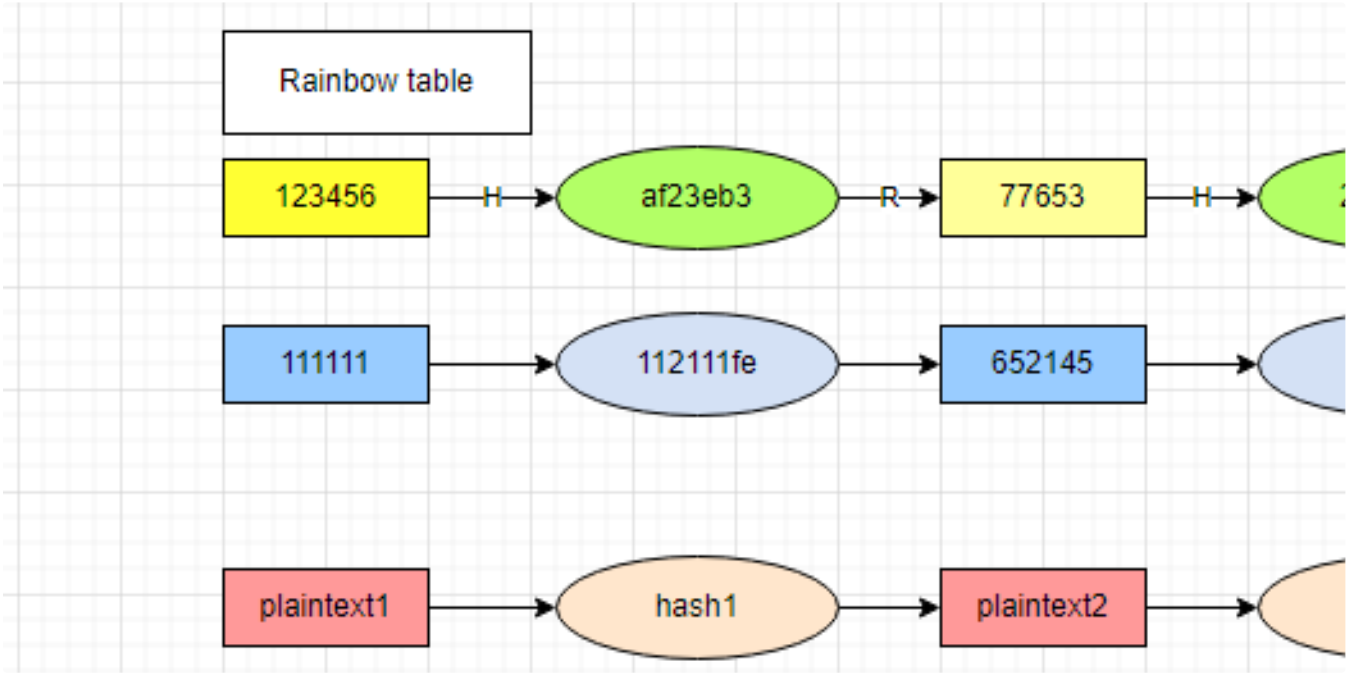


Abhay Parashar in The Pythoneers

17 Mindblowing Python Automation Scripts I Use Everyday

Scripts That Increased My Productivity and Performance

2h ago 4.6K 35



LORY

A basic question in security Interview: How do you store passwords in the database?

Explained in 3 mins.