Name : Meith Navlakha
SAP: 60004190068
ADBMS Practical

EXPERIMENT 1

Aim: Case study on professional and commercial database summary and comparison.

Case Study on: E-Commerce website (own project)

Framework: PHP

Database: MySQL

MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications.

Used in association with a scripting language such as PHP or Perl (both offered on our hosting accounts) it is possible to create websites which will interact in real-time with a mySQL database to rapidly display categorised and searchable information to a website user.

MySQL provides many things, which are as follows:

- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.

Features:

1) Optimized Searching

I have used mySQL along with PHP for my project. For searching a product by category I have written a SQL query using WHERE clause that just searches into the entire product table that has all the other products as well. Thus, it has to traverse through the entire product table to fetch the specified category products tsking a long time for product

information retrieval. making it highly time inefficient. Also, if there a minor spelling mistake then the search fails showing “no results found”.

Recommendation:

Currently I have used basic search for the website that just fetches the data when a query is fired. Instead I can use PHP with AJAX because it passes the request to an API that fetches the data and returns back in the JSON format. Firstly, this makes it platform independent as data is being received in JSON format. Also using AJAX we need not re-render the entire page. Only the segment of the page where data is being displayed is re-rendered. Thus, reducing the time and load on the server.

Secondly, can use ElasticSearch in Java for optimized searching. Elasticsearch uses a distributed system on top of Lucene StandardAnalyzer for indexing and automatic type guessing and utilizes a JSON based REST API to refer to Lucene features. Thus, using distributed system it much faster to fetch data than the basic SQL search which I implemented.

For the second problem the spelling mistake one, we can use node.js as it includes libraries that helps in coding **fuzzy search feature** i.e. if there’s a minor spelling mistake then it would direct to the most relevant route. This enhances the searching of our webpage. Also, node has tensorflow.js and other ML libraries which help in optimizing the searching featuring in the website.

2) Product Details Storing

MySQL uses RDBMS which in turn has rigidity with respect to the number of attributes for each instance(data record). I had made a table “Products” which had products of all the categories e.g. Necklace, Earing, Rings and so on where each category differs from the other in storing the dimensions. For example, Ring only requires one attribute “Diameter” while Earing requires attributes like “Size, Length and Height” and Category like Bridal Sets required even more. Thus, while storing the dimensions for different categories I had made different attributes into the tables which resulted into many empty cells for records of categories which did not have that attribute.

Recommendation:
To overcome this, we can use MongoDB being noSQL(non-relational) is more flexible with the schema and unstructured data. Thus, we can store data as key-value pair.
Moreover, mongoDB also supports storing value multiple datatypes we can just make a dimension key and store all the values in an array.

MySQL:
Product(ProductID , Name, ……. , Diameter, Size, Length, Height)

MongoDB:
Product
{
ProductID: Number,
Name: String,
….
Dimension : [Number],
}

3) Security

My project used MySQL which is highly vulnerable to SQL injection attacks. A hacker can skip the authentication process and manually inject SQL statements (or malicious payload) into the database. The database does not recognize the threat and executes the statement as if the application is sending the request.

Recommendation:

A) Certain measures that can be taken in order to lower the risk of an attack, such as using parameterized queries instead of a concatenated user input. Parameterized queries require the developers to define all the SQL code first and pass each parameter to the query later. This coding style enables

the database to differentiate between code and data, a feature that is not possible with dynamic SQL queries. This is particularly important in WHERE clauses and INSERT or UPDATE statements.

B) Setting up a Firewall. Use a Web Application Firewall (WAF) to filter all traffic between web applications and your database. A firewall protects all web-facing apps, so blocks the SQLi and similar cyberattacks.

Conclusion:

In this experiment I have suggested recommendations on 3 features in my E-Commerce project based on PHP & mySQL. Firstly Optimized search where we can use AJAX and REST API for dynamic search instead of re-rendering the entire page, and using fuzzy search in node.js which increases the searching capacity for the user or use ElasticSearch in Java which improves the searching time by using distributed databases. Second feature was product details storing where SQL has schema rigidity while mongoDB is flexible with the data schema. Lastly security, since my project in SQL is vulnerable to SQL injection attacks we can implement parameterized queries or placeholders or use Web Application Firewall (WAF) to block the SQLi attacks.