Database Management System Project Ideas

Guidelines:

Project groups will be same as that of the mini-project groups.

Final report submission and demonstration will be done by April 15th, 2023. There will be intermediate evaluations.

Project will be evaluated based on the volume of work, and completeness.

Some broad project ideas are given below. You are free to choose other ideas too. Multiple groups may take up the same project idea.

You have to submit a concrete project proposal within a week and get it approved by us.

Deliverables: Report on the project (with group member names and roll numbers) and a demonstration.

A. Projects on Logical Database Design for Specialized Databases:

1. *Spatial Databases for Disaster Management*:

Design and populate a database of road networks, population, water level etc is provided from multiple data sources. The goal of the project is to provide an information system for decision support in disaster management tasks like evacuation, relief. It is part of a larger project for National Spatial Data Infrastructure specific to coastal disaster management system in Eastern coast of India. The database should be supported by a web interface. Data may be pulled from the Google Earth Engine.

Similar spatial database project ideas are listed in:
https://sites.google.com/view/summerofearthengine/projects

2. *High QPS Text Search Engine*

Objective: The project will use the Apache Solr framework running on ZooKeeper framework to develop scalable text search engines.

Methodology: Develop keyword based search engine. Do some benchmarking, to see how search performance scales in a very high query per second (QPS) setting. QPS can be obtained using either something like a multithreaded HTTP client using a ThreadPool and ConnectionPool, or by using a tool like Siege, that can simulate multiple concurrent HTTP requests on a server.

Solr is an open source enterprise search platform from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document (e.g., Word, PDF) handling. Providing distributed search and index replication, Solr is highly scalable.

Siege is an http load testing and benchmarking utility. It was designed to let web developers measure their code under duress, to see how it will stand up to load on the internet. Siege supports basic authentication, cookies, HTTP and HTTPS protocols. It lets its user hit a web server with a configurable number of simulated web browsers. Those browsers place the server "under siege."

Outcome: QPS data outcome using multithreaded http clients with the help of seige connection pool and thread pool.

3. *Large scale graph processing*

The goal of the project is process large graphs in a database

i. Install any graph processing systems e.g., ApacheGraph, Pregel (GoldenOrb), Giraph, or Stanford GPS,
ii. Load a large graph from Stanford SNAP large graph repository

iii. Provide interface to run simple graph queries. Bonus for computing PageRank.

iv. Profile performance

4. XML database

Build a XQuery interface for the Wikipedia/DBLP XML data. The interface should support structured as well as unstructured queries.

5. Multimedia database

Build a multimedia database consisting of text/structured data/music/images/video. Public data may be downloaded to populate the database. Basic queries should be supported.

6. Datalog query language

Datalog is a query language based on the relational calculus. Write datalog programs for a graph database to answer various graph queries.

B. Projects on Buffer Management

7. Simulating Buffer Manager Strategies

The of this project is to simulate a small buffer pool for simple Join/Selection queries on few small tables in the C/C++ language. Popular buffer manager strategies like LRU/MRU/CLOCK/Pinned blocks may be simulated. The strategies may be compared in terms of the number of disk i/o required.

You may use the SQLite C Library for more realistic simulation. https://sqlite.org/index.html

C. Indexing and Hashing

8. High Dimensional Indexing

Implement the R-Tree/KD-Tree for high dimensional indexing. You may choose a high dimensional data 9say, audio or image) and use the tree to index and search it.

9. Implementing Extendable Hashing

Implement the extendable hashing algorithm. Implement the data structures for the hash table. Assume only data expansion occurs. Benchmark the access time on a suitable dataset. You may compare it with the SQLite implementation.

10. Auto-admin for Index Creation

Given a set of query workloads and some table statistics along with a index storage budget write a program to decide the best indices to create. Auto-admin tools are often available to recommend indices, etc. Design a tool in SQLite that recommends a set of indices to build given a particular workload and a set of statistics in a database.

D. Query Processing

11. External Memory Join Algorithms

Implement external memory join computation algorithms and profile their performance on large data sets.

12. Metric Reporting

Build a wrapper/interface which collect query processing metrics like table statistics, CPU/memory usage in run time while executing a query. You may use the inbuilt commands of Postgres.

13. Rule based query rewriting

Specify some fixed rules for query optimization. Write a query rewriter for simple queries which takes as input a relational algebra query and returns its optimal version.

D. Distributed Databases

14. Design of *large, heterogeneous, distributed database systems*

The goal of the project is to design a large database running on a distributed map-reduce platform that can handle heterogeneous data obtained from different sources. The map-reduce distributed Hadoop platform will be used. The database will use Apache Hbase system as the table structure. It will integrate multiple data sources using the Protocol Buffer architecture.

Such databases are common in processing of large and unstructured data common in search engines and online social networks.

Steps:

1. Install Hadoop/Hbase on a laptop/server cluster

2. Load data to nodes

3. Write map-reduce operations

4. Pipe map-reduce outputs using Protocol Buffer

5. Run simple queries

Technologies Involved: Hadoop, Apache Hbase, Protocol Buffers

Data APIs (each group may select a separate data and appropriate queries):

Twitter, Amazon public data sets (http://aws.amazon.com/datasets)

15. *Datawarehouse on Hadoop*

The goal of the project is to run the Hive data warehouse system on Hadoop. And run aggregate/ reporting queries on large data sets in a map-reduce framework.

Steps:

1. Install Hive on Hadoop running on a laptop/server cluster

2. Load data to Hive

3. Write and execute queries in HiveQL

11. *Text search in a map-reduce framework*

The project will use the Apache Solr framework running on ZooKeeper framework.

In this project you will develop a keyword based search engine. Then do some benchmarking, to see how search performance scales in a very high query per second (QPS) setting. QPS can be obtained using either something like a multithreaded HTTP client using a ThreadPool and ConnectionPool, or by using a tool like Siege, that can simulate multiple concurrent HTTP requests on a server. Solr is an open source enterprise search platform from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document (e.g., Word, PDF) handling. Providing distributed search and index replication, Solr is highly scalable. Siege is an http load testing and benchmarking utility. It was designed to let web developers measure their code under duress, to see how it will stand up to load on the internet. Siege supports basic authentication, cookies, HTTP and HTTPS protocols. It lets its user hit a web server with a configurable number of simulated web browsers. Those browsers place the server “under siege.” Outcome is a high QPS data search benchmarking results using multithreaded http clients with the help of seige connection pool and thread pool.