Database Management System Project Ideas

Guidelines:

Groups should be of size at most five.

Final report submission and demonstration will be before endsem. There will be intermediate evaluations.

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

Some of the projects are open ended and you may decide the exact scope of the project.

Some 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 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.

Web Based Information System Design:

Project 1: IITKgp Bus Tracker

Design an information system to track the buses within IITKgp campus. On board GPS systems output will be provided. Goal is to build a smartphone interface to get updates about nearest bus, expected arrival time etc on a map to the students. The project is part of a larger project to develop a monitoring system for school kids in metro cities of India with sms/smartphone based alerting mechanism for parents.

Project 2: Disaster Management Database

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 coastal disaster management system in Eastern coast of India.

Project 3: National Freight Exchange

Transporters ferry goods all across the countries. Often trucks return empty from far off places. Logistics cost is reduced if a central exchange is provided where consignors and vehicles can search for prospective business and bid for consignments. Goal of this project is to build a freight exchange where truck owners and consignors can search and bid for consignments to be transported. It is part of a nationwide effort to build a optimal logistic system for Indian highways.

Project 4: BTP Report Archive Portal

Goal of this project is to build a web portal where the BTP/MTP/Term project reports can be archived in a searchable form. Various metadata like author, supervisor, keywords, category etc should be stored. Users should be able to submit report and get it authenticated by faculty advisor. Other users can search them later. It should run as a web service from the department webpage.

Project 5: Location Aware Digital Personal Assistant

The project aims to develop a personal assistant application which is an advanced calendar providing you reminders as well as recommendations for time management based on your activity history. It will provide you location aware reminders. Example, if you are near the library, it mind remind you to reissue a book. Location information might be obtained from your phone.

Project 6: Web Mashups

In this project you can connect a number of web services provided as APIs like google/bing map, flickr, twitter and combine them to provide interesting applications. Exact mashups may be decided by you. Examples include, geotwitter, integrating google map with twitter etc.

Physical Database Design(All these projects involve reading details of Postgresql documentation and compilation of the code. You need to first install and learn how to recompile postgresql)We suggest that you learn to navigate postgres code in Eclipse.

Project 1: Postgresql Buffer Manager

The aim of this project is to add functionality to the backend server of PostgreSQL. For this project, we focus on just one core module - the buffer manager. The actual coding for this assignment is minimal, given the highly organized and modular postgreSQL code (the code for buffer manager is cleanly separated from the rest of the code base). However, you have to figure out what code to add and where, which is non-trivial! There are three major parts to this project:

Understanding different buffer management strategies

Implement any new buffer replacement policy e.g., the pinned block policy

Compare performances of LRU, MRU and the new policy

Project 2: Postgresql query plan visualization and profiling

In this project, you will take up some existing queries, and then display the query execution plan generated. You may profile the plan under different indices and table sizes. Check if the plan changes if indices are changed. Following documentations are helpful.

- postgreSQL Performance Tips http://www.postgresql.org/docs/8.0/static/performance-tips.html
    - Read sections 13.1 and 13.2, which tell you how to use the "EXPLAIN" command, and how to query the statistics used by the query optimizer, respectively. See http://www.postgresql.org/docs/8.0/static/sql-explain.html for more info.
- The CREATE INDEX Command: http://www.postgresql.org/docs/8.0/static/sql-createindex.html
    - the command for creating indexes; you will also need "DROP INDEX"
- The SET Command: http://www.postgresql.org/docs/8.0/static/sql-set.html
    - this command lets you turn on and off operators that the query optimizer can consider. For example, the SQL command "SET enable_hashjoin TO off;" tells the query optimizer not to use Hash Joins.
- Runtime Configuration: http://www.postgresql.org/docs/8.0/static/runtime-config.html
    - see section 16.4.5.1 "planner method configuration" for a list of the operators you can disable and re-enable for the optimizer using the SET command.
- PostgreSQL System Catalogs: http://www.postgresql.org/docs/8.0/static/catalogs.html
    - note that PostgreSQL keeps its catalog information in tables. The tables/views that are likely to be most useful in this assignment are "pg_stats", "pg_class", "pg_indexes", and "pg_attribute". You can query these in *psql* just like regular tables. For example, the query "select * from pg_indexes where tablename = 'orders';" gives you information about the indexes defined on the orders table. You can see the statistics the optimizer has for the table 'orders' using the query: "select * from pg_stats where tablename = 'orders'" (note the single quotes around the tablename, which is a string attributed in the pg_stats view). The documentation for pg_stats (linked from the above page) explains what the numbers mean. You can get information on tables by querying the "pg_class" table and/or using the "/d" command in *psql*.

Project 3: In-memory Column Store

Columnar representations have become very popular for in-memory data due to the opportunities for compression that they offer. An in-memory columnar representation has been implemented as a patch on PostgreSQL already, and you can find more info at http://www.pgcon.org/2014/schedule/attachments/322_IMCS.pdf. But you can still implement a subset of the features described in the above presentation on your own.

Project 4: External Memory Join Algorithms

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

Project 5: High Dimensional Indexing

Implement high dimensional indexing schemes like R-Tree and KD-Tree and profile their performance on large data sets.

Project 6: Distributed 2-Phase Locking

Implement distributed 2-phase locking algorithm. Test on a small cluster you may form.

Large/Distributed Databases:

Project 1. Design of *non-relational , heterogeneous, distributed database systems*

The goal of the project is to design a non-relational 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 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)

Project 2: *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 cluster

2. Load data to Hive

3. Write and execute queries in HiveQL

4. Profile performances for different loads

Technologies used: Hadoop, Hive

Data API: Amazon public data sets (http://aws.amazon.com/datasets)

Project 3: *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.

Project 4: *Large scale graph processing on Hadoop*

The goal of the project is process large graphs in a map-reduce framework.

1. Install any graph processing systems e.g., ApacheGraph, Pregel (GoldenOrb), Giraph, or Stanford GPS, on Hadoop

2. Load a large graph from Stanford SNAP large graph repository

3. Run simple graph queries. Bonus for computing PageRank.

4. Profile performance

Project 5: Resilient Distributed Databases

Write an application in *Spark* to count the number of words in a text stream.