Lecture #15 – Project Topics + Extra Credit

Andy Pavlo // Carnegie Mellon University // Spring 2016

TODAY'S AGENDA

Project #3 Topics
Extra Credit

PROJECT #3

Group project to implement some substantial component or feature in a DBMS.

Projects should incorporate topics discussed in this course as well as from your own interests.

Each group must pick a project that is unique from their classmates.

PROJECT #3

Project deliverables:

→ Proposal
→ Project Update
→ Code Review
→ Final Presentation
→ Code Drop

PROJECT #3 – PROPOSAL

Five minute presentation to the class that discusses the high-level topic.

Each proposal must discuss:
→ What files you will need to modify.
→ How you will test whether your implementation is correct.
→ What workloads you will use for your project.

PROJECT #3 – STATUS UPDATE

Five minute presentation to update the class about the current status of your project.

Each presentation should include:
→ Current development status.
→ Whether your plan has changed and why.
→ Anything that surprised you during coding.

PROJECT #3 – CODE REVIEW

Each group will be paired with another group and provide feedback on their code.

Grading will be based on participation.

PROJECT #3 – FINAL PRESENTATION

<u>10</u> minute presentation on the final status of your project during the scheduled final exam.

You'll want to include any performance measurements or benchmarking numbers for your implementation.

Demos are always hot too...

PROJECT #3 – CODE DROP

A project is **not** considered complete until:

→ The code can merge into the master branch without any conflicts.
→ All comments from code review are addressed.
→ The project includes test cases that correctly verify that implementation is correct.
→ The group provides documentation in both the source code and in separate Markdown files.

We will select the merge order randomly.

PROJECT TOPICS

Query Planner
Vectorized Execution
Logging & Checkpoints
Mat. Views & Triggers
Schema Changes
Constraints

User-defined Functions
Enhanced Indexes
Database Compression
Multi-Threaded Queries
Integrated Memcache

QUERY PLANNER

Peloton's still relies on Postgres' disk-oriented query planner. It also has to transform Postgres plans into Peloton compatible plans.

Project: Create a new main memory oriented query planner that emits Peloton plans.
→ Will want to consider using modern C++11 features and newer optimization techniques.
→ Do not need to support everything that Postgres supports but code should be easily extendible.

VECTORIZATION

SIMD instructions can greatly speed up the execution of scans and other operators.

Project: Implement support for vectorized query execution in Peloton.

→ Will want to focus on scan operators initially and extend to sorting/hashing.
→ Need to maintain/extend existing code so that you can measure the speed improvements.

LOGGING & CHECKPOINTS

Peloton currently supports a simple WAL scheme without checkpoints. It is not able to reload the database after restart.

Project: Optimize runtime logging. Implement checkpoints. Implement recovery from checkpoints + WAL.

→ You will want to use a fast parallel recovery method like in SiloR.

MATERIALIZED VIEWS & TRIGGERS

A materialized view is like a view that is updated whenever its underlying table is updated. Triggers are used to invoke changes.

Project: Implement support for incremental updates to mat. views and triggers in Peloton.

→ Will need to leverage Postgres' catalog infrastructure and then populate new data structures.
→ **Bonus:** Implement a pub/sub interface that can supports streaming updates to Kafka.

CONCURRENT SCHEMA CHANGES

A DBMS needs to be able to support updating the database schema (e.g., add/drop column) while it continues to execute txns and queries.

Project: Implement support for concurrent schema changes with low overhead.

→ You will want to use a lazy method that propagates changes to tiles only when they are updated.
→ Will want to extend internal catalog to keep track of different schema versions.

CONSTRAINTS

Constraints are important feature in DBMSs to ensure database integrity.

Project: Implement support for enforcing integrity constraints in Peloton.

→ Will want to start with simple constraints first.
→ Final goal will be to implement foreign key constraints.

USER-DEFINED FUNCTIONS

UDFs allow the developer to implement complex logic for evaluating tuples.

Project: Implement support for UDFs in Peloton for all query types.

→ Will need to extend Peloton's expression sub-system.
→ Can leverage existing Postgres catalog support.

ENHANCED INDEXES

We need to decide what is the best index to use for Peloton. Bw-Tree project is a good start.

Project: Implement additional indexes in Peloton and evaluate their performance.

→ B+tree, Bw-Tree, Skip List, ART Index
→ May need to enhance the index API to provide a more generic interface.
→ **Bonus:** Implement a concurrent inverted index for fast text searches.

DATABASE COMPRESSION

Compression enables the DBMS to use less space to store data and potentially process less data per query.

Project: Implement different compression schemes for table storage.
→ Deltas, Dictionaries, Naïve Block Compression
→ Will need to implement new query operators that can operate directly on this data.
→ **Bonus:** Implement the ability to automatically determine what scheme to use per tile.

MULTI-THREADED QUERIES

Peloton currently only uses a single worker thread per txn/query.

Project: Implement support for intra-query parallelism with multiple threads.

→ Will need to implement exchange operators and modify query plans to parallelize them.

→ **Bonus:** Add support for NUMA-aware data placement. Will need to update internal catalog.

INTEGRATED MEMCACHE

Memcache is most common system for caching data in large-scale apps. It doesn't need to be a separate system with an in-memory DBMS.

Project: Implement the Memcache API directly inside of Peloton and enable it to read/write directly to in-memory data.

→ GET/PUT can be implemented as a single-query txn with prepared statements.
→ **Bonus:** Rewrite the client connection handling code.

TESTING

We plan to provide a SQL-based regression test suite to check that your project does not break high-level functionalities.

Every group is going to need to implement their own unit tests for their code.

COMPUTING RESOURCES

The hardware from **MemSQL** has arrived.
→ Dual-socket Xeon E5-2620 (6 cores / 12 threads)
→ 16 GB DDR4

We are currently working on setting up the lab so that we can give everyone access.

If anybody has experience with Ubuntu MAAS, please let me know.

OLTP-BENCH

We already have a full-featured benchmarking framework that you can use for your projects.

It includes 15 ready to execute workloads
→ OLTP: TPC-C, TATP, YCSB, Wikipedia
→ OLAP: CH-Benchmark, TPC-H

http://oltpbenchmark.com/

PROJECT #3 PROPOSALS

Each group will make a **5** minute presentation about their project topic proposal to the class on **Monday March 14th**.

I am able during Spring Break for additional discussion and clarification of the project idea.

EXTRA CREDIT

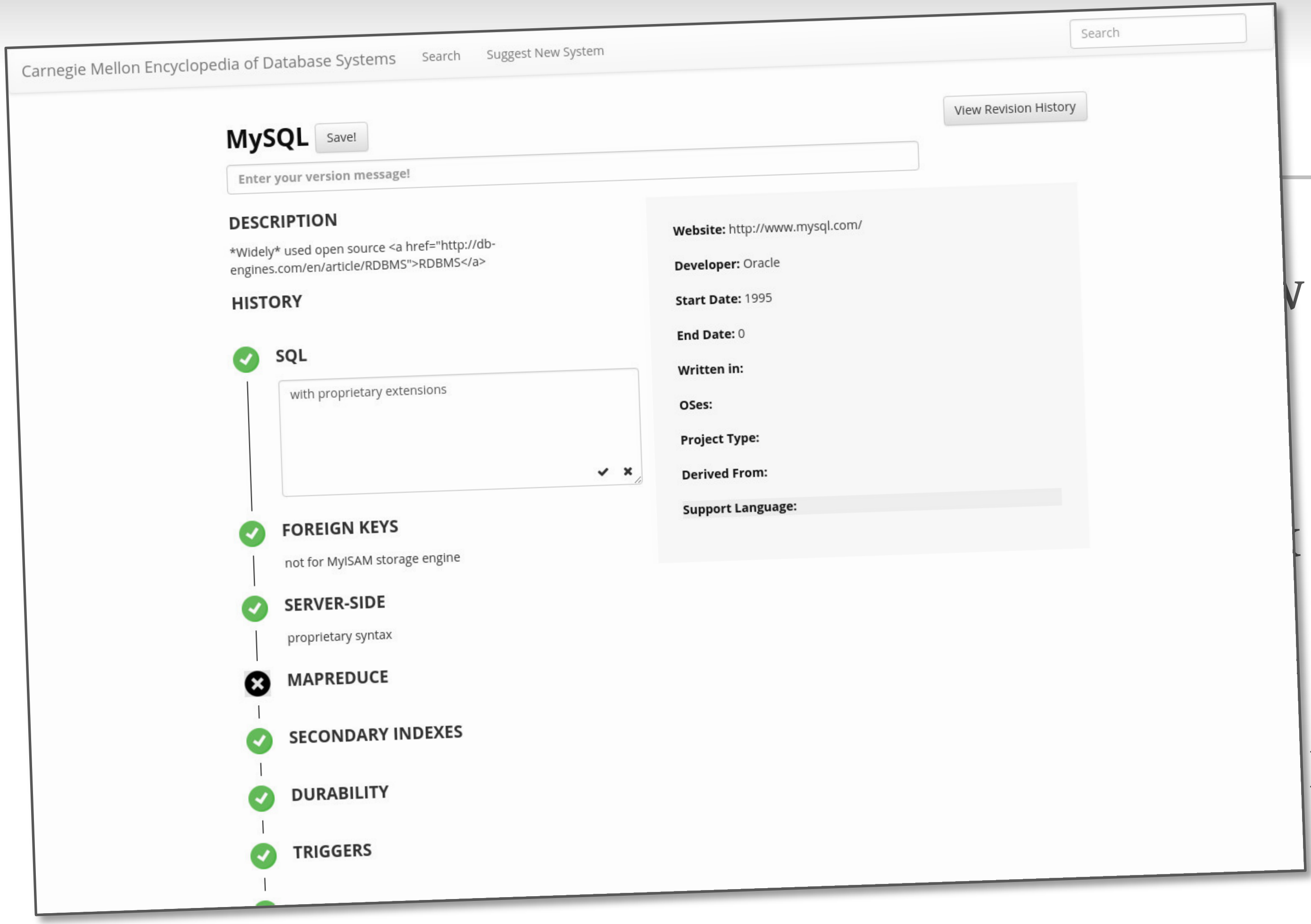
Each student can earn extra credit if they write a encyclopedia article about a DBMS.
→ Can be academic/commercial, active/historical.

Each article will use a standard taxonomy.
→ For each feature category, you select pre-defined options for your DBMS.
→ You will then need to provide a summary paragraph with citations for that category.

DBDB.IO

All the articles will be hosted on our new website (currently under development).
→ I will post the user/pass on Piazza.

I will post a sign-up sheet for you to pick what DBMS you want to write about.
→ If you choose a widely known DBMS, then the article will need to be comprehensive.
→ If you choose an obscure DBMS, then you will have do the best you can to find information.



Carnegie Mellon Encyclopedia of Database Systems   Search   Suggest New System   Search

View Revision History

MySQL Save!

Enter your version message!

DESCRIPTION

Widely used open source <a href="http://db-engines.com/en/article/RDBMS">RDBMS</a>

HISTORY

✓ **SQL**

with proprietary extensions

✓ **FOREIGN KEYS**

not for MyISAM storage engine

✓ **SERVER-SIDE**

proprietary syntax

✗ **MAPREDUCE**

✓ **SECONDARY INDEXES**

✓ **DURABILITY**

✓ **TRIGGERS**

Website: http://www.mysql.com/

Developer: Oracle

Start Date: 1995

End Date: 0

Written in:

OSes:

Project Type:

Derived From:

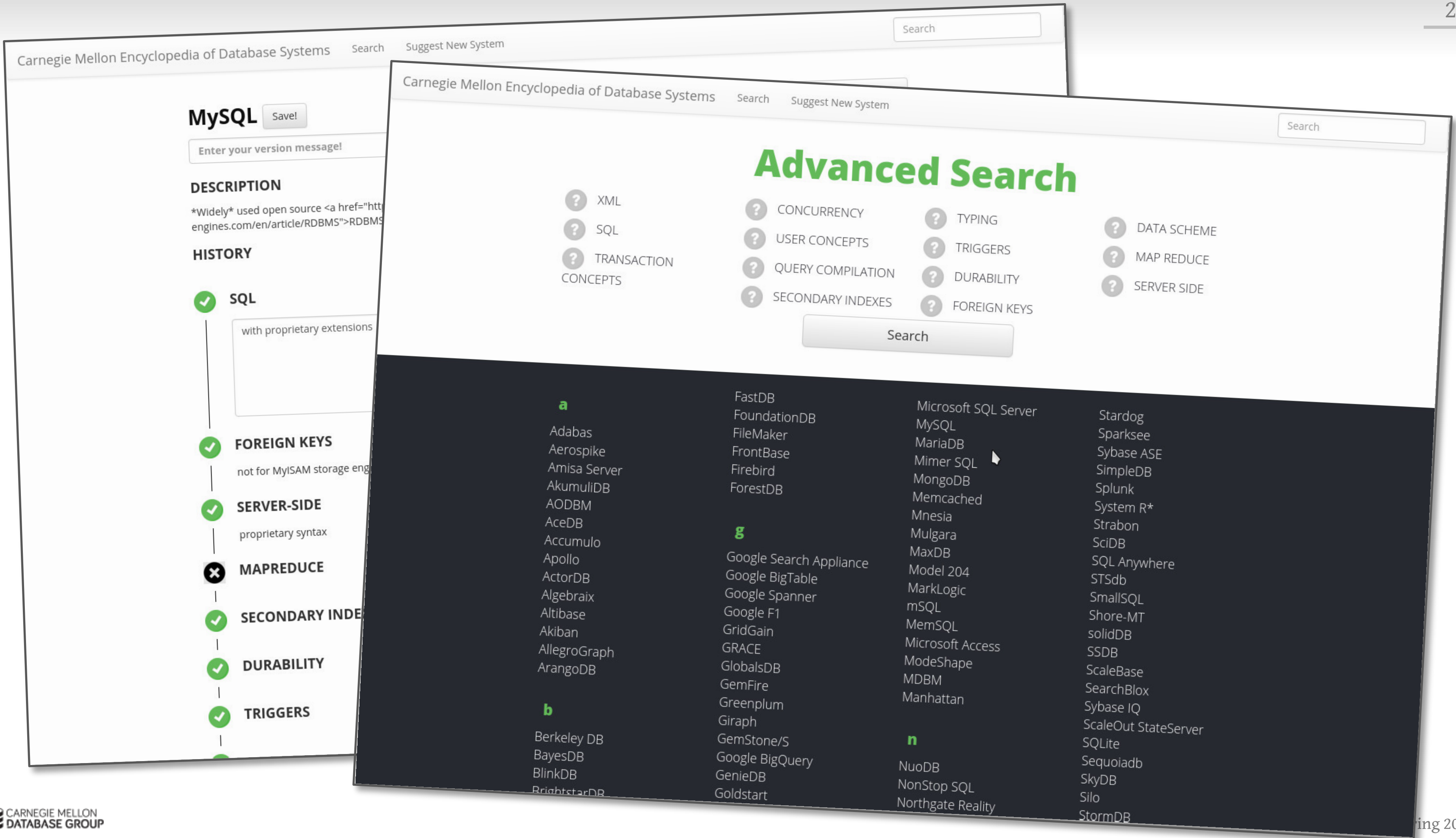
Support Language:

v

what

article

have



Carnegie Mellon Encyclopedia of Database Systems Search Suggest New System Search

MySQL Save!

Enter your version message!

DESCRIPTION

Widely used open source <a href="htt
engines.com/en/article/RDBMS">RDBMS

HISTORY

✓ SQL

with proprietary extensions

✓ FOREIGN KEYS

not for MyISAM storage eng

✓ SERVER-SIDE

proprietary syntax

✗ MAPREDUCE

✓ SECONDARY INDE

✓ DURABILITY

✓ TRIGGERS

Carnegie Mellon Encyclopedia of Database Systems Search Suggest New System Search

Advanced Search

XML
SQL
TRANSACTION CONCEPTS
CONCURRENCY
USER CONCEPTS
QUERY COMPILATION
SECONDARY INDEXES
TYPING
TRIGGERS
DURABILITY
FOREIGN KEYS
DATA SCHEME
MAP REDUCE
SERVER SIDE

Search

a
Adabas
Aerospike
Amisa Server
AkumuliDB
AODBM
AceDB
Accumulo
Apollo
ActorDB
Algebraix
Altibase
Akiban
AllegroGraph
ArangoDB

b
Berkeley DB
BayesDB
BlinkDB
BrightstarDB

FastDB
FoundationDB
FileMaker
FrontBase
Firebird
ForestDB

g
Google Search Appliance
Google BigTable
Google Spanner
Google F1
GridGain
GRACE
GlobalsDB
GemFire
Greenplum
Giraph
GemStone/S
Google BigQuery
GenieDB
Goldstart

Microsoft SQL Server
MySQL
MariaDB
Mimer SQL
MongoDB
Memcached
Mnesia
Mulgara
MaxDB
Model 204
MarkLogic
mSQL
MemSQL
Microsoft Access
ModeShape
MDBM
Manhattan

n
NuoDB
NonStop SQL
Northgate Reality

Stardog
Sparksee
Sybase ASE
SimpleDB
Splunk
System R*
Strabon
SciDB
SQL Anywhere
STSdb
SmallSQL
Shore-MT
solidDB
SSDB
ScaleBase
SearchBlox
Sybase IQ
ScaleOut StateServer
SQLite
Sequoiadb
SkyDB
Silo
StormDB

redis IBM DB2 cassandra Microsoft® SQL Server MySQL™

SQLite HyPer VOLTDB

riak APACHE HBASE presto Couchbase

JustOne NUODB ScaleBase cloudera® IMPALA MariaDB CUBRID™

membase AEROSPIKE WIREDTIGER

RAVENDB

mongoDB deepdb VERTICA Informix® HIVE

amazon DynamoDB SAP® HANA NETEZZA RocksDB

DATAllegro RethinkDB

Google Spanner FOUNDATIONDB SYBASE® ORACLE® BERKELEY DB Tokutek CouchDB relax

PostgreSQL PARACCEL Clustrix SUBWAY

memsql Hortonworks STINGER ScaleArc HYPERTABLE

ORACLE® ScaleDB LMDB db Shards

☠ PLAGIARISM WARNING ☠

This article must be your own writing with your own images. You may **not** copy text/images directly from papers or other sources that you find on the web.

Plagiarism will **not** be tolerated.
See CMU's Policy on Academic Integrity for additional information.

NEXT CLASS

Project #2 is due **March 9th @ 11:59pm**

Project #3 proposals are due **March 14th**

I will be available during my regular office hours next week.