ASHESI UNIVERSITY COLLEGE

MYSQL DATABASE ENGINES REVIEW, ANALYSIS, COMPILATION AND CUSTOMIZATION

MAC-ANTHONY MANU

2013

Applied Project

ASHESI UNIVERSITY COLLEGE

MYSQL DATABASE ENGINES REVIEW, ANALYSIS, COMPILATION AND CUSTOMIZATION

By

MAC-ANTHONY MANU

Dissertation submitted to the Department of Computer Science

Ashesi University College

In partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science

APRIL 2013

Applied Project

Declaration

I hereby declare that this dissertation is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: ………………………………………………………………………

Candidate's Name: Mac-Anthony Manu

Date: …………………………..

I hereby declare that the preparation and presentation of the dissertation were supervised in accordance with the guidelines on supervision of dissertation laid down by Ashesi University College.

Supervisor's Signature: ………………………………………………………………..

Supervisor's Name: Mr. Kwadwo Gyamfi Osafo-Maafo

Date: …………………………………..

Acknowledgements

I am extremely grateful to my supervisor Mr. Kwadwo Gyamfi Osafo-Maafo for assisting me to choose a project that is very relevant to my future carrier and also for his immense contribution towards the successful completion of this project. I am also appreciative of the contributions of Mr. Aelaf Dafla and Dr. Nathan Amanquah who taught me lessons in Operating Systems and programming in C++ respectively. My sincere thanks to all the lectures who taught me in the numerous courses I have taken in Ashesi University College.

Abstract

This project is a review of several MySQL database storage engines. It shows what engines are available and the features they have. It also investigates the compilation and extension of these storage engines and it finally discusses and demonstrates how to add a customized engines.

Contents

Chapter One ... 1
1.1: Introduction ... 1
1.2: Objects of the Project ... 1
Chapter Two ... 2
2.1: A Review of a Selection of MySQL Engines ... 2
2.1.1: InnoDB Storage Engine ... 3
2.1.2: MyISAM Storage Engine ... 4
2.1.3: InnoDB Verses MyISAM ... 5
2.1.4: MEMORY Storage Engine ... 7
2.1.5: Comma-Separated Values (CSV) Storage Engine ... 8
2.1.6: ARCHIVE Storage Engine ... 9
2.1.7: BLACKHOLE Storage Engine ... 10
2.1.8: MySQL Engines that Store Raw Data ... 12
2.2: Related MySQL Projects ... 14
2.2.1: MySQL Cluster ... 14
2.2.2: Drizzle ... 15
2.2.3: MariaDB ... 16
2.2.4: Percona Server ... 17
2.2.5: OurDelta Server ... 18
2.3: Chapter Conclusions ... 18
Chapter Three ... 19
3.1: Implementing a Custom MySQL Engine ... 19
Chapter Four ... 24
4.1: Tests and Results ... 24
4.2: Time Taken to Make 10000 Insertions by the Storage Engines ... 24
4.3: Time Taken to Make 8000 updates by the Storage Engines ... 26
4.4: Time Taken to Make 8000 Deletions by the Storage Engines ... 27
Chapter Five ... 28
5.1: Conclusions and Recommendations ... 28
Bibliography ... 29
Appendix A ... 33
A.1: Screen Shots of Tests Performed on InnoDB Storage Engine ... 33

A.1.1: Update Operations 33
A.1.2: Delete Operations..... 33
A.2: Screen Shots of Tests Performed on MyISAM Storage Engine..... 34
A.2.1: Update Operations 34
A.2.2: Delete Operations..... 34
A.3: Screen Shots of Tests Performed on ARCHIVE Storage Engine 35
A.3.1: Plugging in ARCHIVE Engine to the Running Server 35
A.3.2: Prove of ARCHIVE Storage Engine Enabled. 35
A.3.3: Update and Delete Operations..... 35
A.4: Screen Shots of Tests Performed on MEMORY Storage Engine 36
A.4.1: Update Operations 36
A.4.1: Delete Operations..... 36
A.5: Screen Shots of Tests Performed on CSV Storage Engine 37
A.5.1: Update Operations 37
A.5.2: Delete Operations..... 37
A.6: Screen Shots of Tests Performed on BLACKHOLE Storage Engine 38
A.6.1: Plugging in ARCHIVE engine to the Running Server 38
A.6.2: Select, Update and Delete Operations 38
Appendix B 39
B.1: How to Get MySQL Source Files 39
B.2: Installing the Libraries Needed to Build MySQL Source Files 40
B.3: The Build Process..... 40
B.4: How to Start the Server 42
B.5: How to Start the MySQL Client..... 45

Chapter One

1.1: Introduction

MySQL is a relational database management system (RDMS). Its original author is Michael Monty Widenius and it was first released in 1996 (CrunchBase, 2009). It is known to be the world's most popular open source database management system. MySQL was first owned by MySQL AB but was bought by Sun Microsystems in January 2008 and was later purchased by Oracle.

MySQL is written in C and C++ programming languages. It has pluggable storage engine architecture. This allows storages engines to be plugged-in or plugged-out whiles the server is running. This project uses the source code for MySQL 5.5 server.

1.2: Objects of the Project

The objective of this project is to review several MySQL database storage engines, investigate the compilation and extension of these storage engines, and demonstrate how a custom MySQL storage engine can be developed.

Chapter Two

2.1: A Review of a Selection of MySQL Engines

MySQL 5.5 server came with about nine (9) pluggable storage engines, giving users the flexibility to choose the storage engine that best fits their specific need. Nevertheless, these engines defer in performance levels and scalability among other features. The engines include InnoDB, MyISAM, MEMORY, CSV, ARCHIVE, BLACKHOLE, MERGE, FEDERATED and the EXAMPLE storage engines.

Not all of these engines have the ability to store data. Some of these engines make use of the data stored by some other engine to create their data. The engines that are able to store their own data are InnoDB, MyISAM, CSV and ARCHIVE storage engines. MEMORY and MERGE storage engines mostly use data stored by other engines to create theirs. BLACKHOLE and EXAMPLE storage engines do not store any data at all. FEDERATED storage engine is a special engine for distributed database systems.

InnoDB "is the mostly widely used storage engine for Web/Web 2.0, eCommerce, Financial Systems, Telecommunications, Health Care and Retail applications built on MySQL" (Oracle, 2013) and has become the default storage engine as of MySQL 5.5.5 server. The following subsections of this chapter deal with some features and functionalities of these engines mentioned above. It is important to note that the sections will not discuss the performance capabilities of the engines but only their functionalities.

2.1.1: InnoDB Storage Engine

InnoDB has the features of a complete database engine. It is a transaction-safe ACID[1] compliant storage engine for MySQL. It has commit, rollback, and crash-recovery capabilities to protect the data of users. A feature that makes InnoDB transactional than the other engines is its row-level locking mechanism. This mechanism ensures that several read and write operations can be carried out on one table concurrently without destroying the integrity of the data in the table. Another advantage of the row-level locking mechanism is that it improves performance.

Another key feature of InnoDB is its ability to maintain data integrity by supporting foreign key referential-integrity constraints. Therefore a relationship can be defined between tables and this relationship ensures that only acceptable operations are carried out on the tables involved.

InnoDB also supports full-text search indexes. Usually, an index of the words in a document is created with references to their locations. A search is then map against the index and the word or phrase is retrieve from the exact location. The table below is a summary of the features supported by InnoDB.

[1] ACID stands for Atomicity, Consistency, Isolation and Durability.

Table 2.1.1: InnoDB Storage Engine Features (Oracle, 2013)

Storage Limits	64TB	Transactions	Yes	Locking granularity	Row
Multi-Version Concurrency Control (MVCC)	Yes	Geospatial data type support	Yes	Geospatial Indexing Support	No
B-tree indexes	Yes	T-tree indexes	No	Hash Indexes	No[a]
Full-text search indexes	Yes [b]	Clustered indexes	Yes	Data caches	Yes
Index caches	Yes	Compressed data	Yes [c]	Encrypted data [d]	Yes
Cluster database support	No	Replication support [e]	Yes	Foreign key support	Yes
Backup / point-in-time recovery [f]	Yes	Query cache support	Yes	Update statistics for data dictionary	Yes

[a] InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.
[b] InnoDB support for FULLTEXT indexes is available in MySQL 5.6.4 and higher.
[c] Compressed InnoDB table required the InnoDB Barracuda file format.
[d] Implemented in the server (via encryption functions), rather than in the storage engine.
[e] Implemented in the server, rather than in the storage engine.
[f] Implemented in the server, rather than in the storage engine.

2.1.2: MyISAM Storage Engine

MyISAM is the default storage engine prior to MySQL 5.5.5. MyISAM is not ACID complaint and non transactional and unlike InnoDB, it supports table-level locking. For this reason, it is able to carry out several read operations than write operations within a specific period (Oracle, January 2011). It is therefore the preferred engine for Web, data warehousing and other application environments (Oracle, 2013). The key features of MyISAM are shown in the table below.

Table 2.1.2: The key features of MyISAM

Storage Limits	256TB	Transactions	No	Locking granularity	Table
Multi-Version Concurrency Control (MVCC)	No	Geospatial data type support	Yes	Geospatial Indexing Support	Yes
B-tree indexes	Yes	T-tree indexes	No	Hash Indexes	No
Full-text search indexes	Yes	Clustered indexes	No	Data caches	No
Index caches	Yes	Compressed data	Yes [a]	Encrypted data [b]	Yes
Cluster database support	No	Replication support [c]	Yes	Foreign key support	No
Backup / point-in-time recovery [d]	Yes	Query cache support	Yes	Update statistics for data dictionary	Yes
[a] Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only. [b] Implemented in the server (via encryption functions), rather than in the storage engine. [c] Implemented in the server, rather than in the storage engine. [d] Implemented in the server, rather than in the storage engine.					

2.1.3: InnoDB Verses MyISAM

InnoDB and MyISAM are the two major storage engines for MySQL database. The first fact about these engines is that, MyISAM was developed and made available for public usage before InnoDB (Yang, 2009). However, in terms of functionality, InnoDB is much more capable than MyISAM. Basically, where data integrity and writer intensive operations become the priority (Yang, 2009), InnoDB should be used otherwise MyISAM is the best choice. The table below highlights the key feature differences between InnoDB and MyISAM.

Table 2.1.3 Comparison between InnoDB and MyISAM storage engines

Feature	InnoDB	MyISAM
ACID Transactions	Yes	No
Configuration ACID Properties	Yes	No
Crash Safe	Yes	No
Foreign Key Support	Yes	No
Row-Level Locking Granularity	Yes	No (Table)
Multi-Version Concurrency Control (MVCC)	Yes	No
Geospatial Data Type Support	Yes	Yes
Geospatial (R-Tree) Indexing Support	No	Yes
B-tree Indexes	Yes	Yes
Full-text search indexes	No	Yes
Clustered Index	Yes	No
Data Caches	Yes	No
Index Caches	Yes	Yes
Compressed Data	Yes [b]	Yes [a]
Read and Write to Compressed Table	Yes	No (read-only)
Encrypted Data [c]	Yes	Yes
Replication Support [d]	Yes	Yes
Backup / Point-in-Time Recovery [d]	Yes	Yes
Query Cache Support	Yes	Yes
Update Statistics for Data Dictionary	Yes	Yes
Storage Limits (Table Size)	64TB	256TB

[a] Compressed MyISAM tables are supported only when using the compressed row format.
[b] Compressed InnoDB tables required the InnoDB Barracuda file format.
[c] Implemented in the server (via encryption functions), rather than in the storage engine.
[d] Implemented in the server, rather than in the storage engine.

2.1.4: MEMORY Storage Engine

The MEMORY engine, formerly known as the Heap engine, creates its tables in the MEMORY of the computer. The content of the tables are usually from the database stored permanently on the physical disc drive. MEMORY tables use dynamic hashing for inserts (Oracle, 2013). The tables are highly volatile. The rows of a MEMORY table are destroyed when the server goes off. However, the tables continue to exit because their definitions are stored in .frm files on disk (Oracle, 2013).

MEMORY engines are useful in some situations. It is the appropriate engine to use if an application does a lot of reading from the database but does not need to update the database or does very few updates. In this use case keeping a copy of the data in the MEMORY of the computer will allow a quicker access to the data than if it being read from the physical hard disc drive. The table below shows the key features of the MEMORY Engine.

Table 2.1.4: Key features of the MEMORY Engine.

Storage Limits	RAM	Transactions	No	Locking granularity	Table
Multi-Version Concurrency Control (MVCC)	No	Geospatial data type support	No	Geospatial Indexing Support	No
B-tree indexes	Yes	T-tree indexes	No	Hash Indexes	Yes
Full-text search indexes	No	Clustered indexes	No	Data caches	N/A
Index caches	N/A	Compressed data	No	Encrypted data [a]	Yes
Cluster database support	No	Replication support [b]	Yes	Foreign key support	No
Backup / point-in-time recovery [c]	Yes	Query cache support	Yes	Update statistics for data dictionary	Yes
[a] Implemented in the server (via encryption functions), rather than in the storage engine. [b] Implemented in the server, rather than in the storage engine. [c] Implemented in the server, rather than in the storage engine. N/A means not applicable.					

2.1.5: Comma-Separated Values (CSV) Storage Engine

The CSV "storage engine stores data in text files using comma-separated values format" (Oracle, 2013). It became fully enabled in MySQL 5.1 server. When a command is issued to create a CSV table, three items are created. First item is a table format files created in the database directory by the server. It has the table name with .frm extension. The second item is plain text data file created by the storage engine. It also has the table name with .CSV extension. The last item is a "Metafile that stores the state of the table and the number of rows that exist in the table" (Oracle, 2013). It has the name of the table with CSM extension.

CSV files can be imported into any standard spreadsheet program and "it allows for the instantaneous loading of massive amounts of data into the MySQL server" (Schumacher, 2008). The CSV table offers a great way of bringing a spreadsheet into a real database for analysis, manipulation and extraction. The CSV engine however does not support both indexing and transactions. It does not support indexing because files can be taken in and out of the table directory without having to worry about rebuilding the directory. Like the MyISAM storage engine, CSV engine is not ACID complaint and non transactional.

2.1.6: ARCHIVE Storage Engine

The ARCHIVE storage engine stores large amount of data without indexing in highly compressed data tables (Oracle, 2013). The zlib library is used for the data compression. Table rows are compressed during insert operations and uncompressed on retrieval. Its architecture provides high inserting speed.

ARCHIVE tables do not support delete, replace or update operations. A select operation performs a complete table scan because ARCHIVE tables do not support indexing (Oracle, 2013). When an ARCHIVE table is created, a table format file is created by the server in the database directory. The table format file is named with the table name with an .frm extension. "The storage engine creates other files, all having names beginning with the table name. The data file has an extension of .ARZ. An .ARN file may appear during optimization operations" (Oracle, 2013).

The ARCHIVE storage engine is useful for storing enormous amount of data that is not frequently accessed (Refulz, 2011). The table below shows the key features of the ARCHIVE storage engine.

Table 2.1.6: Key features of the ARCHIVE storage engine.

Storage Limits	None	Transactions	No	Locking granularity	Table
Multi-Version Concurrency Control (MVCC)	No	Geospatial data type support	Yes	Geospatial Indexing Support	No
B-tree indexes	No	T-tree indexes	No	Hash Indexes	No
Full-text search indexes	No	Clustered indexes	No	Data caches	No
Index caches	No	Compressed data	Yes	Encrypted data [a]	Yes
Cluster database support	No	Replication support [b]	Yes	Foreign key support	No
Backup / point-in-time recovery [c]	Yes	Query cache support	Yes	Update statistics for data dictionary	Yes
[a] Implemented in the server (via encryption functions), rather than in the storage engine. [b] Implemented in the server, rather than in the storage engine. [c] Implemented in the server, rather than in the storage engine.					

Table … Key features of the ARCHIVE storage engine.

2.1.7: BLACKHOLE Storage Engine

BLACKHOLE storage engine successfully creates tables, accepts data but does not store this data. Retrieval operations as a result return empty result. INSERT triggers are accepted by this engine. UPDATE, DELETE and clauses such as 'FOR EACH ROW' do not apply since there are no rows

(Oracle, 2013). When a BLACKHOLE table is created, the server creates a table format file in the database directory. This table has the table name with frm extension. Unlike some of the engines discussed above, there are no other files associated with the table.

Although BLACKHOLE engines do not have data storage ability, they are useful in some use cases. One of the situations where BLACKHOLE engines are of use is in distributed database systems (Schneller, 2006). In this use case, there is usually one master server and several slave servers.

The binary log on the master server provides a record of the data changes to the database. The events contained in the master's binary log file are sent to the slave servers. The slave servers execute these events to make the same data changes that were made on the master server. Usually, the events are filtered and each slave server receivers only those events that are necessary to make its database update to date. Figure 2.1 shows a distributed database use case.

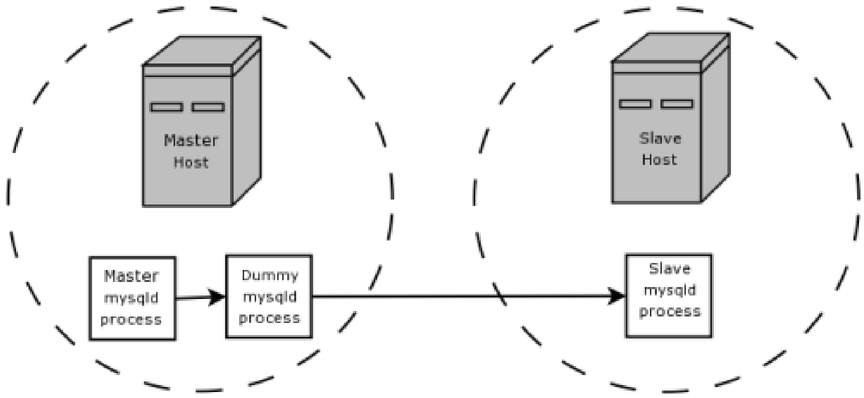


Figure 2.1.1 Distributed Database Design (Oracle, 2013)

However, replicating the data in the slave servers may not be necessary because what is actually desired is to update the binary log of the slave servers for data recovery. Therefore, whilst the master server tables are created with the InnoDB or MyISAM storages engines, the tables in the slaver servers can be created with BLACKHOLE storage engine. The key advantage of this use case is that disc spaces are reserved for other purposes. It is also appropriate to use BLACKHOLE storage engine to perform performance test or benchmarking especially when storing data is not necessary.

2.1.8: MySQL Engines that Store Raw Data

The interest of this project is in the primary data storage engines of MySQL. These are MyISAM, InnoDB, MEMORY and ARCHIVE. From the discussion of the storage engines above, it can be concluded that in terms of functionality, InnoDB and MyISAM carryout a lot of operations than the other engines. The table below presents some comparison between MyISAM, InnoDB, MEMORY and ARCHIVE storage engines.

Table 2.1.8: Comparing the data storage engines

Features	MyISAM	InnoDB	MEMORY	ARCHIVE
Storage Limits	256TB	64TB	RAM	None
Transactions	No	Yes	No	No
Locking Granularity	Table	Row	Table	Table
Multi-Version Concurrency Control (MVCC)	No	Yes	No	No
Geospatial Data Type Support	Yes	Yes	No	Yes
Geospatial Indexing Support	Yes	No	No	No
B-Tree Indexes	Yes	Yes	Yes	No
T-Tree Indexes	No	No	No	No
Hash Indexes	No	No [a]	Yes	No
Full-Text Search Indexes	Yes	Yes [b]	No	No
Clustered Indexes	No	Yes	No	No
Data Caches	No	Yes	N/A	No
Index Caches	Yes	Yes	N/A	No
Compressed Data	Yes [c]	Yes [d]	No	Yes
Encrypted Data [e]	Yes	Yes	Yes	Yes
Cluster Database Support	No	No	No	No
Replication Support [f]	Yes	Yes	Yes	Yes
Foreign Key Support	No	Yes	No	No
Backup / point-in-time recovery [g]	Yes	Yes	Yes	Yes
Query Cache Support	Yes	Yes	Yes	Yes
Update Statistics for Data Dictionary	Yes	Yes	Yes	Yes

[a] InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.
[b] InnoDB support for FULLTEXT indexes is available in MySQL 5.6.4 and higher.
[c] Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.
[d] Compressed InnoDB tables require the InnoDB Barracuda file format.
[e] Implemented in the server (via encryption functions), rather than in the storage engine.
[f] Implemented in the server, rather than in the storage engine.
[g] Implemented in the server, rather than in the storage engine.
N/A means not applicable

2.2: Related MySQL Projects

2.2.1: MySQL Cluster

This is a highly specialized distributed node architecture storage solution designed for fault tolerance and high performance (Bell, 2010) for the distributed environment. Data is stored and replicated on individual storage nodes. Each storage node executes a separated server and preserves a replica of the data.

MySQL Cluster ensures the highest performance and availability possible by using multiple MySQL servers for load distributing and data storage. An update operation executed one on storage node is immediately made available to the rest of the nodes. The nodes use a sophisticated transmission protocol for data transfer across the network.

Basically, MySQL Cluster is made up of three components. These are the MySQL server component, the network database (NDB) component and the NDB cluster storage engine. MySQL Cluster mostly refers to MySQL server and the NDB component while NDB Cluster usually refers to the NDB Cluster technologies only. Using the NDB cluster storage engine as an interface, MySQL Cluster uses the MySQL server as the frontend to support standard SQL queries. In other words, the MySQL server processes the SQL commands and communicates to the NDB storage engine.

However, NDB cluster storage engine cannot be used without NDB Cluster components thought it is possible to use the NDB Cluster technologies without the MySQL server (Bell, 2010). The engine as well as the entire MySQL Cluster has been developed to achieve three primary objects. These are the assurance of the highest achievable performance, high availability and data redundancy. To achieve these objectives MySQL has functionality such as node recovery, logging, check pointing, system recovery, hot backup and restore, failover, partitioning, online operations and no single of point of failure.

2.2.2: Drizzle

Drizzle was developed out of MySQL 6.0 server. Drizzle developers took away the components of MySQL they considered as "bad" and built a new system out of the remaining good ones. The end product is a lightweight micro-kernel designed database system for cloud infrastructure and web applications. The database functionalities such as durability and relational properties are built into the kernel as default design. The kernel has been designed to be small, simple, clear and do little operations as much as possible. Drizzle therefore supports a number of pluggable interfaces to allow users extend the database by writing simple plug-ins (Drizzle Developers, 2010).

Drizzle has been "designed for massive concurrency on modern multi-cpu/core architecture" (Drizzle Developers, 2008). The design and architecture of Drizzle gives it the ability to be reconfigured to take advantage of the processing power of new servers with better technologies without disturbing the state of the database. This adaptive approach of integrating Drizzle with server infrastructure and making it a part of it is better than the historical situation where database servers dictate infrastructure of server computers and operating systems.

Drizzle uses InnoDB storage engine as the default storage engine (Otto, 2010). This makes Drizzle transactional and ACID compliant. Since the functionalities of InnoDB have been discussed above, much will not be said about it here. All other MySQL engines have been removed. Other items in MySQL that are not in Drizzle include **keywords** such as ENGINES and CLIENT, **commands** such as ALTER TABLE UPGRADE and SET NAMES and **objects** such as TIME and TINYBLOB. There are no FRM files and grant or privilege tables in Drizzle. "Drizzle does not currently have any plug-ins that implements stored procedures" (Drizzle Developers, 2010) and does not also have any plug-in that provides SQL triggers.

2.2.3: MariaDB

MariaDB is similar to MySQL in many ways except that it has improved features. It is based primarily on MySQL source code. The most important reason for the development of MariaDB is to continue to make MySQL open to users should Oracle decides to make MySQL fully commercial. The second reason is that, Michael Monty Widenius, the main author of the original version of MySQL who left MySQL Aktiebolag soon after MySQL was obtained by Sun Microsystems became dissatisfied with the quality of MySQL releases (Bartholomew, 2012).

MariaDB has all the standard storage engines of MySQL. In addition, it has the following storage engines: Aria, XtraDB, FEDERATEDX, OQGRAPH, SphinxSE, IBMDB21 and Cassandra. Aria is an upgraded version of MyISAM. XtraDB is MariaDB's InnoDB and it is the most fully featured MariaDB's storage engine. It is transactional and ACID compliant. FEDERATEDX is a version of MySQL's FEDERATED storage engine and Cassandra is similar to MySQL's NDB Cluster storage engine except that it functions with Cassandra Cluster from MariaDB.

The Open Query GRAPH computation engine (OQGRAPH) "allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions)" (MariaDB Developers, 2010). SphinxSE is built as a dynamically loadable .so plug-in. IBMDB21 is the same storage engine that was introduced in MySQL 5.1.33 and later removed in MySQL 5.1.54. "IBMDB21 storage engine is designed as a fully featured transaction-capable storage engine that enables MySQL to store its data in DB2 tables running on IBM i" (Oracle, 2013). It has similar functionalities as InnoDB. Among these functionalities are support for foreign key constraints, ACID-compliant, transactional, full crash recovery, radix-tree-based indexes, and the unique ability to enable DB2 for IBM i applications to see and update table data in instantaneously.

2.2.4: Percona Server

Percona Server and MariaDB have similar objectives. These are to provide a more efficient and a drop-in replacement for MySQL database. The default storage engine in Percona Server is XtraDB. This has similar properties and functionalities as the XtraDB used in MariaDB Server

(Percona Company, 2011). Both of these storage engines are enhanced versions of MySQL's InnoDB.

In general, Percona Server has extra features for developers, extra diagnostic features, and durability and reliability enhancements. It also has extra performance and scalability enhancements and extra features for database administrator (Percona Company, n.d.).

2.2.5: OurDelta Server

OurDelta Server is no longer maintained. It was built with patches from MySQL Server, MariaDB Server and Percona Server. It came with MySQL's Sphinx search engine, MariaDB's XtraDB, OQGRAPH, PBXT, FEDERATEDX storage engines (OurDelta, 2008). A version of Sphinx search engine called SphinxSE has been included in MariaDB. The features and functionalities of all of these engines have been discussed above.

2.3: Chapter Conclusions

The MySQL storage engine that has been improved and adapted in most MySQL projects discussed above is InnoDB. It is transactional, supports row-level locking and ACID compliant. Although the various engines has been developed for a specific use cases, the architectural design, features and functionalities of InnoDB storage seems to be desirable for users and developers.

Chapter Three

3.1: Implementing a Custom MySQL Engine

With the inclusion of the source code for the EXAMPLE storage engine, it is much easier to start developing a custom engine. The EXAMPLE engine has three files. These are the header file, the method implement file and a cmake text files. These can be found in the storage/engine directory of the MySQL 5.5 source tree.

The first step is to obtain a name for the engine. The name of the custom engine to be developed in the case is "macTXT". From the terminal, sed utility was used to copy and rename the three EXAMPLE storage engine files and also replace all instances of "EXAMPLE" and "example" in those files with the name of the new custom engine which in this case "macTXT". This process created the initial source files of macTXT. The procedure is shown below.

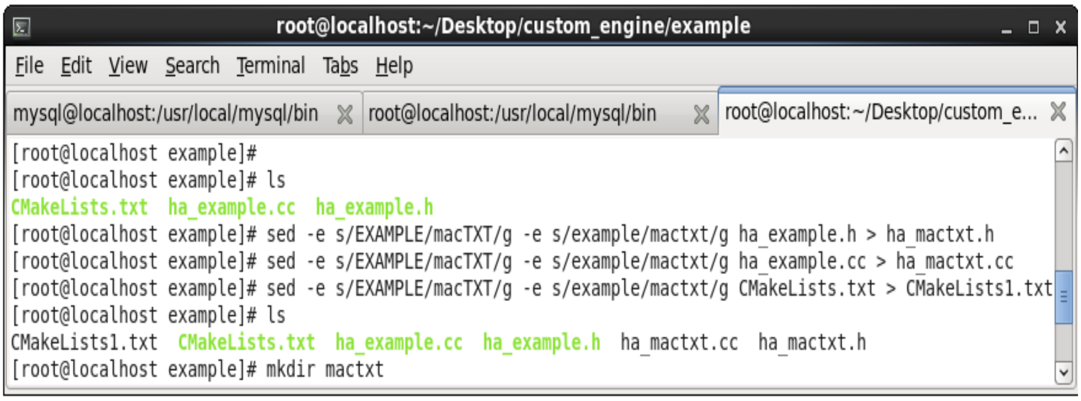


Figure 3.1: Creating Custom Engine Files

This produced a new custom engine which I compiled and plugged into the running server without any issues. The figure below shows macTXT successful added to the list of supported engines.

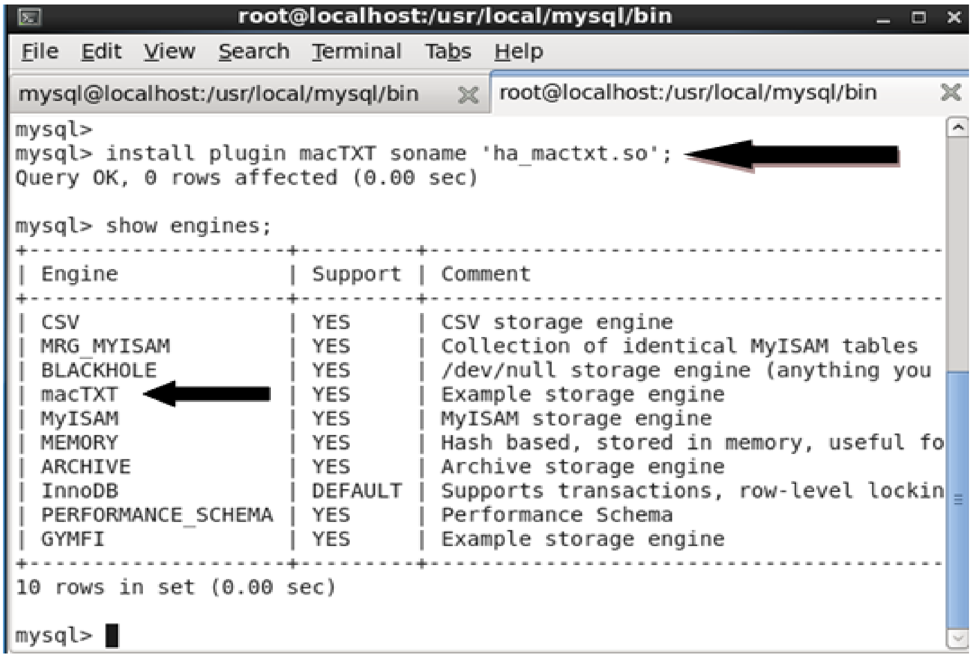


Figure 3.2: Plugging in Custom Storage Engine

At this point, macTXT has the same functionalities as the EXAMPLE storage engine. It can create tables but cannot store it. Before macTXT can do anything meaningful, Oracle recommends that the following methods defined in the EXAMPLE engine source code which are also now in macTXT engine source code are implemented:

- ✓ store_lock()
- ✓ enternal_lock()
- ✓ rnd_init()
- ✓ info(uinf flag)
- ✓ extra()
- ✓ rnd_next()
- ✓ an open method
- ✓ a close method

After the implementation of these methods above, support for the following methods must also be implemented:

- INSERT
- UPDATE
- DELETE
- Non-Sequential Reads
- Indexing
- Transactions

The ability of the engines depends on the number of operations it supports. For instance, as discussed in chapter two, the ARCHIVE storage engine does not support UPDATE and DELETE operations or any form of indexing because those methods have not been implemented.

To understand how these methods are implemented, the source code for the CSV was examined. The reason is because this engine is less complicated than InnoDB, MyISAM or MEMORY engines.

The following method calls during a five-row table scan of the CSV engines will be used to explain some functions of the methods listed above. It assumes that the table is opened.

- @code
- ha_EXAMPLE::store_lock
- ha_EXAMPLE::external_lock
- ha_EXAMPLE::info
- ha_EXAMPLE::rnd_init
- ha_EXAMPLE::extra
- ENUM HA_EXTRA_CACHE   Cache record in HA_rrnd()
- ha_EXAMPLE::rnd_next
- ha_EXAMPLE::rnd_next

- ha_EXAMPLE::rnd_next
- ha_EXAMPLE::rnd_next
- ha_EXAMPLE::rnd_next
- ha_EXAMPLE::extra
- ENUM HA_EXTRA_NO_CACHE End caching of records (def)
- ha_EXAMPLE::external_lock
- ha_EXAMPLE::extra
- ENUM HA_EXTRA_RESET Reset database to after open
- @endcode

Store_lock() is called before any reading from the table or writing into the table is done. Its mean function is to modify the table lock level. Examples are to change blocking write lock to non-blocking, ignore the lock (if we do not want to use MySQL table locks at all) or add locks for many tables.

external_lock() is called at the start of a statement or when a LOCK TABLES statement is issued.

The rnd_init() method is used to prepare a table for scanning. Its mean function is to reset counters and pointers to the start of the table.

The info(uinf flag) defines the type of operations supported by the storage engine. These include support for AUTO_INCREMENT, INDEXING and the ability of the engine to be transactional.

The functions of the extra() method is to provide extra hints to the storage engine on how to perform certain operations. For example, if a user mistakenly runs a delete operation without using the WHERE keyword, the engines must assume the user made a mistake although is a legal operation. Such operations should be treated accordingly. The extra() method therefore adds some level of intelligence to the functioning of the engine.

The first four methods in the table scan example above are called to initialize the table. The rnd_next() is called after the table initialization once for every row to be scanned until the server's search condition is satisfied or the end of the file has been reached.

The methods discussed above all will be called even if only one row is to be read. This shows how the methods depend on each other for every successful operation. There is no method in the source code runs alone. If the developer wants the engines to support UPDATE operations, then all methods related to this operation must be implemented.

Chapter Four

4.1: Tests and Results

This chapter shows the results obtained from performing INSERT, UPDATE and DELETE operations on InnoDB, MyISAM, ARCHIVE, MEMORY, CSV and BLACKHOLE storage engines.

Different python scripts each with a function that inserts ten thousand (10,000) rows into a table were used to test the performance of the various engines for INSERT operations. The function in each python script is called four times to make forty thousand insertions when a script is executed once from the terminal. To obtain the average insertion time, the time taken by each ten thousand insertion are added and the result is divided by four. Appendix A has the screen shots of this activity.

The time taken by UPDATE and DELETE operations are obtained by executing queries from the terminal. The same query is executed four times and the average time is obtained from the different execution times. The tables and figures below show the results obtained. Appendix A has the screen shots of these activities.

4.2: Time Taken to Make 10000 Insertions by the Storage Engines

Table 4.2: Average Time taken to make 10000 Insertions by some storage engines.

Average Insertion Times (sec)	
InnoDB	3.222495019
MyISAM	3.502286792
CSV	2.95816505
ARCHIVE	2.461804032
MEMORY	2.436110735
BLACKHOLE	2.197271168

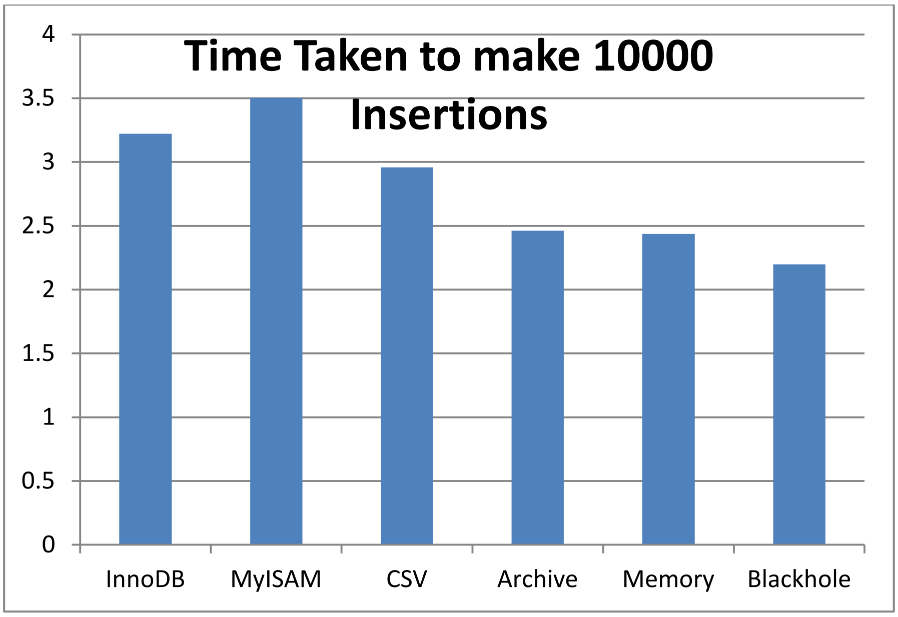


Figure 4.2: Graph of Average Insertion Times of Some Storage Engines

From the table and graph above, it can be observed that BLACKHOLE storage engine has the smallest insertion times and MyISAM has the highest insertion time. The outcomes of these tests show the effects that the functionalities of the engines discussed in chapter two have on their performance. Among the engines, it is only BLACKHOLE storage engine that accepts the data and does not save it. It is expected to run faster than the other engines.

MEMORY storage engine accepts and saves the data to the MEMORY of the computer but not to the physical disc. It is therefore expected to be fast as BLACKHOLE. Table 2.1.8 shows that unlike InnoDB and MyISAM, ARCHIVE engine does little to its data before it stores it. InnoDB and MyISAM have to prepare their data for indexing and other operation that

ARCHIVE does not support. And because InnoDB is transactional it is able to insert faster than MyISAM.

4.3: Time Taken to Make 8000 updates by the Storage Engines

Table 4.3: Average update times of some storage engines

average update times	
innodb	0.35
myisam	0.4075
MEMORY	0.0275
CSV	0.45

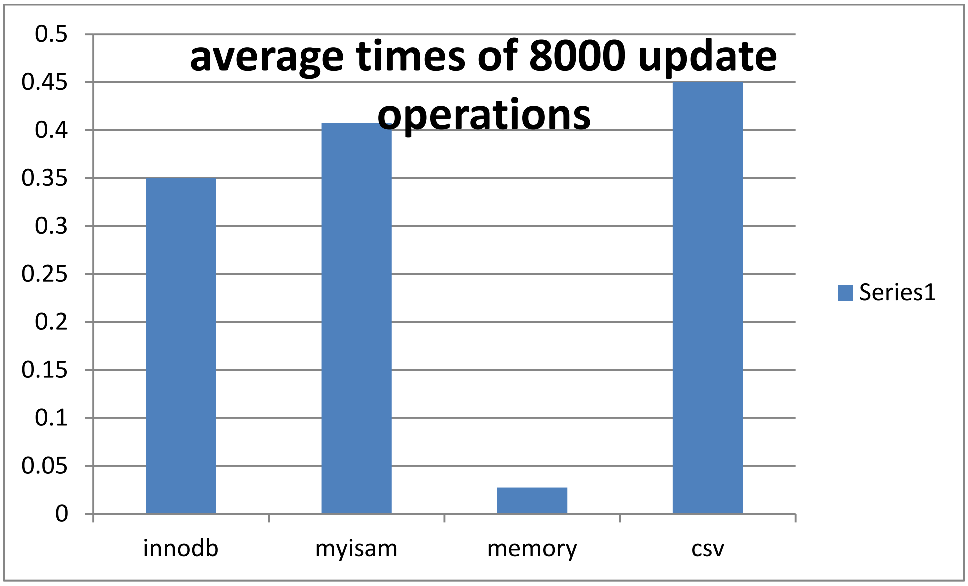


Figure 4.3: Graph of update times of some storage engine

The Graph above shows that MEMORY storage engine again make faster updates that the other engines. This is due to the same reasons for which it makes faster insertions. InnoDB makes faster updates than MyISAM because whereas InnoDB supports row-level locking, MyISAM supports table-level locking. CSV must do a complete table scan before it completes all of its updates. Besides, CSV does not support indexes. This explains why it is the slowest among the engines above. ARCHIVE storage engine did not show up because it does not support update operations.

4.4: Time Taken to Make 8000 Deletions by the Storage Engines

Table 4.4: Average Time taken to make 8000 deletions by some storage engines

average deletion times (sec)	
innodb	0.155
myisam	0.2825
MEMORY	0.03
CSV	0.165

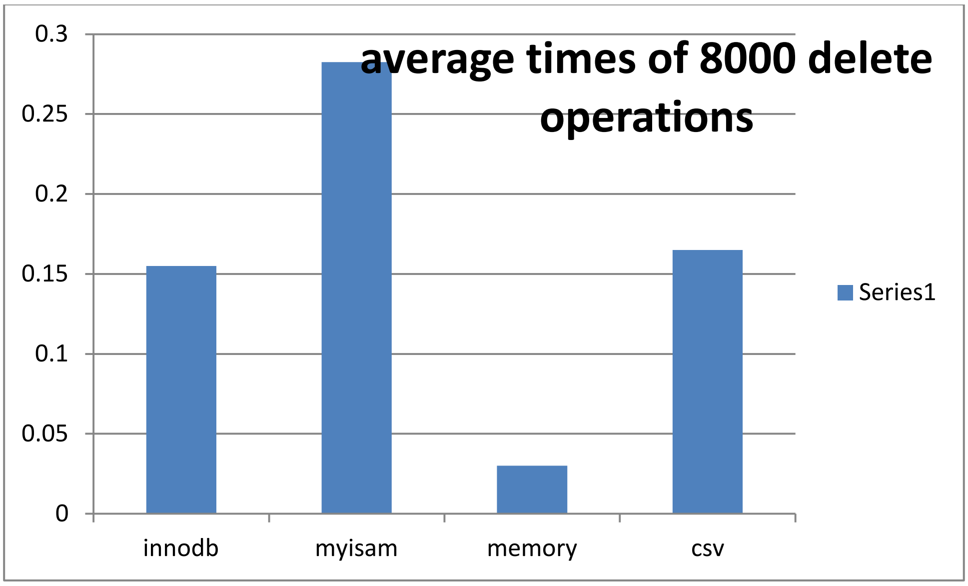


Figure 4.4.1: Graph of average time taken to make 8000 deletion by some storage engines

From the graph above, MEMORY has the fastest deletion times followed by InnoDB, CSV and MyISAM. MEMORY is the fastest for the same reason from which it make faster insertions as explained above. InnoDB's row-level locking, indexing and transactional properties, explains why it makes faster update and delete operations than MyISAM and CSV engines. CSV runs faster than MyISAM in this operation because its data items are not encrypted. It takes less time to compare the values in a query statement to the data items stored by the tables.

The graph below compares the execution times of the various operations. It shows that insert operations take much time to be completed than all other operations. This is followed by update operations and then deletions operations. The high inserting time for all engines is because much work is done to the data before storage. For instance, some engines must index the stored data in order to make retrieval operations easy to handle.

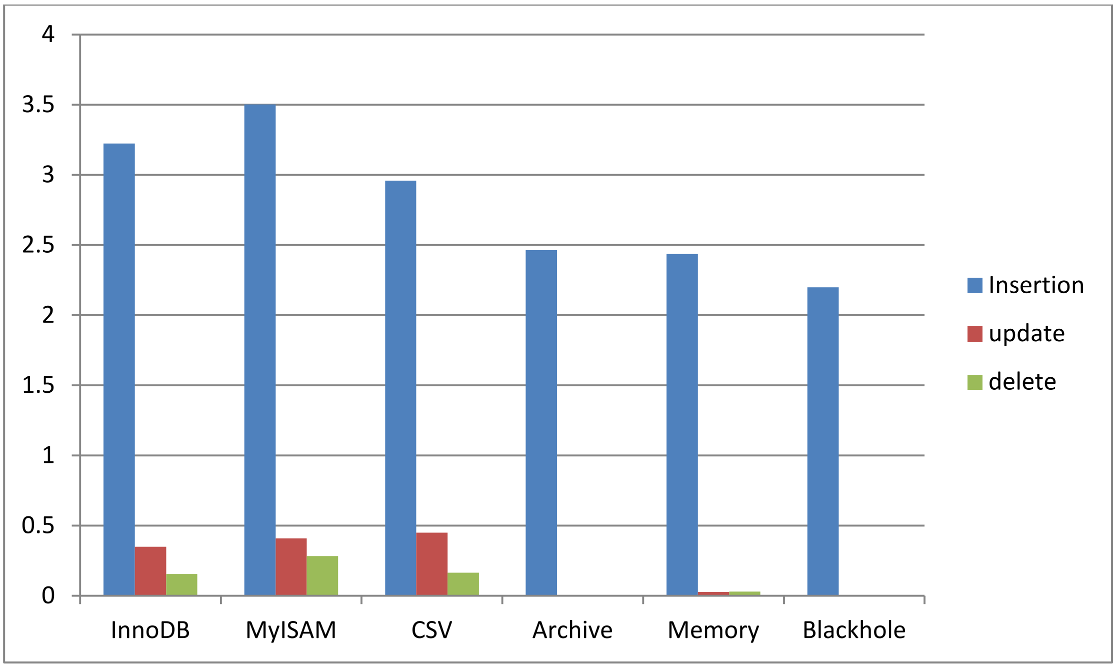


Figure 4.4.2: assessment of insert, update and delete operations execution times.

Chapter Five

5.1: Conclusions and Recommendations

This is an exciting applied project. It however requires knowledge, skills and interest in database management. Knowing how to write MySQL queries will be helpful but it is important to understand how database management system works. Because MySQL is written in C and C++, it is also important that one should have the ability to write codes in this programming language. Skills in python or PHP will also be needed.

Also, anyone interested in continuing this project or doing a similar project should be able to work in a Linux environment without much difficulties. That individual must know the file structure of the Linux operating system being used and since the terminal will be a good place to work from, knowing the basic and the most useful terminal commands will help.

Finally, it will be good if two people do a project like this, especially when it comes to writing a custom engine. The amount of detail involved in writing a custom engine cannot be handed by one person.

Bibliography

Bartholomew, D., 2012. *Monty Program: Whitepaper - MariaDB vs MySQL.* [Online]
Available at: http://montyprogram.com/whitepapers/mariadb-vs-mysql/download/
[Accessed 30 March 2013].

Bell, C., 2010. *Oreilly Answers: What Is MySQL Cluster?.* [Online]
Available at: http://answers.oreilly.com/topic/1862-what-is-mysql-cluster/
[Accessed 26 March 2013].

CrunchBase, 2009. *People: CrunchBase.* [Online]
Available at: http://www.crunchbase.com/person/michael-widenius
[Accessed 8 April 2013].

Drizzle Developers, 2008. *Overview: A Lightweight SQL Database for Cloud Infrastructure and Web Applications.* [Online]
Available at: https://launchpad.net/drizzle
[Accessed 28 March 2013].

Drizzle Developers, 2010. *Drizzle 2010.10 Documentation: Notable MySQL Differences.* [Online]
Available at: http://docs.drizzle.org/mysql_differences.html
[Accessed 28 March 2013].

Drizzle Developers, 2010. *Drizzle 2010.10 Documentation: What is Drizzle?.* [Online]
Available at: http://docs.drizzle.org/what_is_drizzle.html
[Accessed 28 March 2013].

MariaDB Developers, 2010. *AskMonty Knowledgebase: About OQGRAPH.* [Online]
Available at: https://kb.askmonty.org/en/about-oqgraph/
[Accessed 30 March 2013].

Oracle, 2013. *MySQL 5.1 Reference Manual :: Storage Engines :: The IBMDB2I Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.1/en/se-db2.html
[Accessed 30 March 2013].

Oracle, 2013. *MySQL 5.5 Reference Manual:: Storage Engines:: The Archive Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.5/en/archive-storage-engine.html
[Accessed 21 March 2013].

Oracle, 2013. *MySQL 5.5 Reference Manual:: Storage Engines:: The Blackhole Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.5/en/blackhole-storage-engine.html
[Accessed 22 March 2013].

Oracle, 2013. *MySQL 5.5 Reference Manual:: Storage Engines:: The CSV Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.5/en/csv-storage-engine.html
[Accessed 21 March 2013].

Oracle, 2013. *MySQL 5.5 Reference Manual:: Storage Engines:: The InnoDB Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.5/en/innodb-storage-engine.html
[Accessed 20 March 2013].

Oracle, 2013. *MySQL 5.5 Reference Manual:: Storage Engines:: The MEMORY Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.5/en/memory-storage-engine.html
[Accessed 20 March 2013].

Oracle, 2013. *MySQL 5.5 Reference Manual:: Storage Engines:: The MyISAM Storage Engine.* [Online]
Available at: http://dev.mysql.com/doc/refman/5.5/en/myisam-storage-engine.html
[Accessed 20 March 2013].

Oracle, January 2011. *MySQL 5.5: Storage Engine Performance Benchmark for MyISAM and InnoDB: A MySQL Technical White Paper,* s.l.: Oracle.

Otto, A., 2010. *Rackspace Blog: Rackspace and Drizzle: It's Time To Rethink Everything.* [Online]
Available at: http://www.rackspace.com/blog/rackspace-and-drizzle-its-time-to-rethink-everything/
[Accessed 28 March 2013].

OurDelta, 2008. *Patches/Sources.* [Online]
Available at: http://ourdelta.org/patches
[Accessed 1 April 2013].

Percona Company, 2011. *History: The Percona XtraDB Storage Engine.* [Online]
Available at: http://www.percona.com/docs/wiki/percona-xtradb:start
[Accessed 31 March 2013].

Percona Company, n.d. *Percona Server Feature Comparison.* [Online]
Available at: http://www.percona.com/software/percona-server/feature-comparison
[Accessed 31 March 2013].

Refulz, 2011. *Refulz Web Developer's Blog: Storage Engines in MySQL.* [Online]
Available at: http://php.refulz.com/storage-engines-in-mysql/
[Accessed 21 March 2013].

Schneller, D., 2006. *Daniel Schneller's Blog: MySQL Replication Using Blackhole Engine.* [Online]
Available at:
http://www.jroller.com/dschneller/entry/mysql_replication_using_blackhole_engine
[Accessed 22 March 2013].

Schumacher, R., 2008. *National ChungHsing University: A Look at the MySQL CSV Storage Engine.* [Online]
Available at: http://ftp.nchu.edu.tw/MySQL/tech-resources/articles/csv-storage-engine.html
[Accessed 21 March 2013].

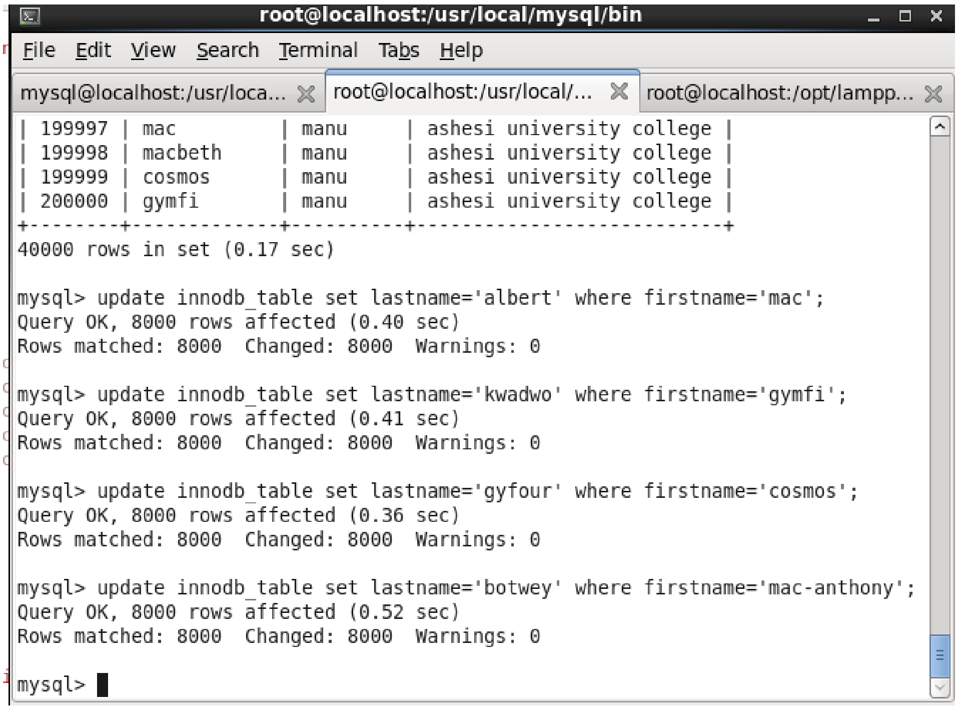
Yang, Y., 2009. *Kavoir: MySQL Engines: InnoDB vs. MyISAM – A Comparison of Pros and Cons.* [Online]
Available at: http://www.kavoir.com/2009/09/mysql-engines-innodb-vs-myisam-a-

comparison-of-pros-and-cons.html
[Accessed 26 March 2013].

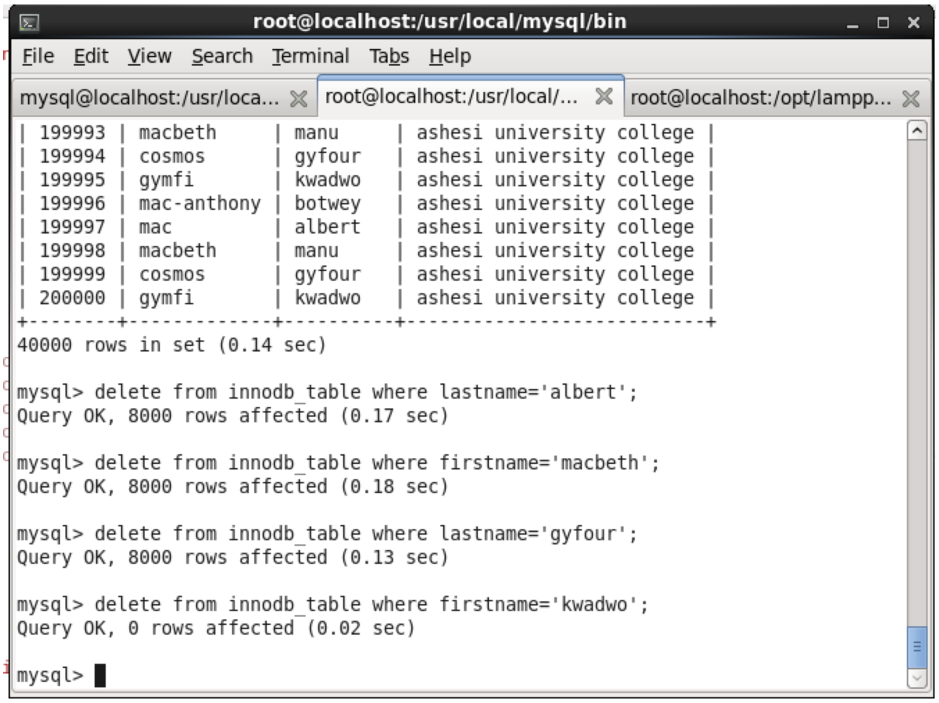
Appendix A

A.1: Screen Shots of Tests Performed on InnoDB Storage Engine

A.1.1: Update Operations

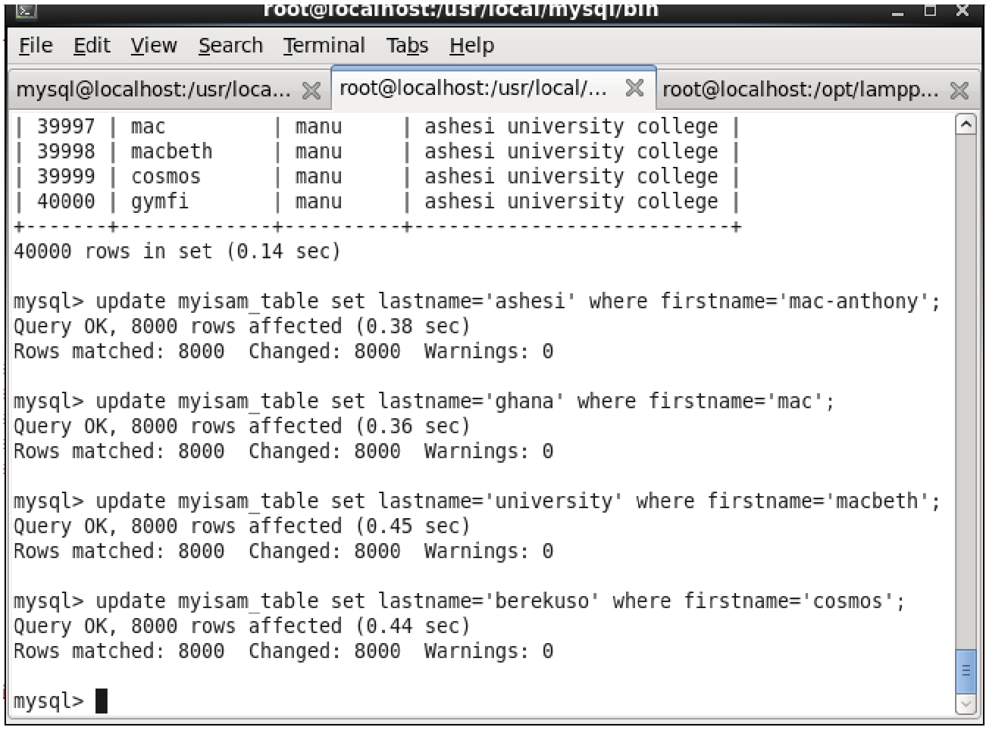


A.1.2: Delete Operations



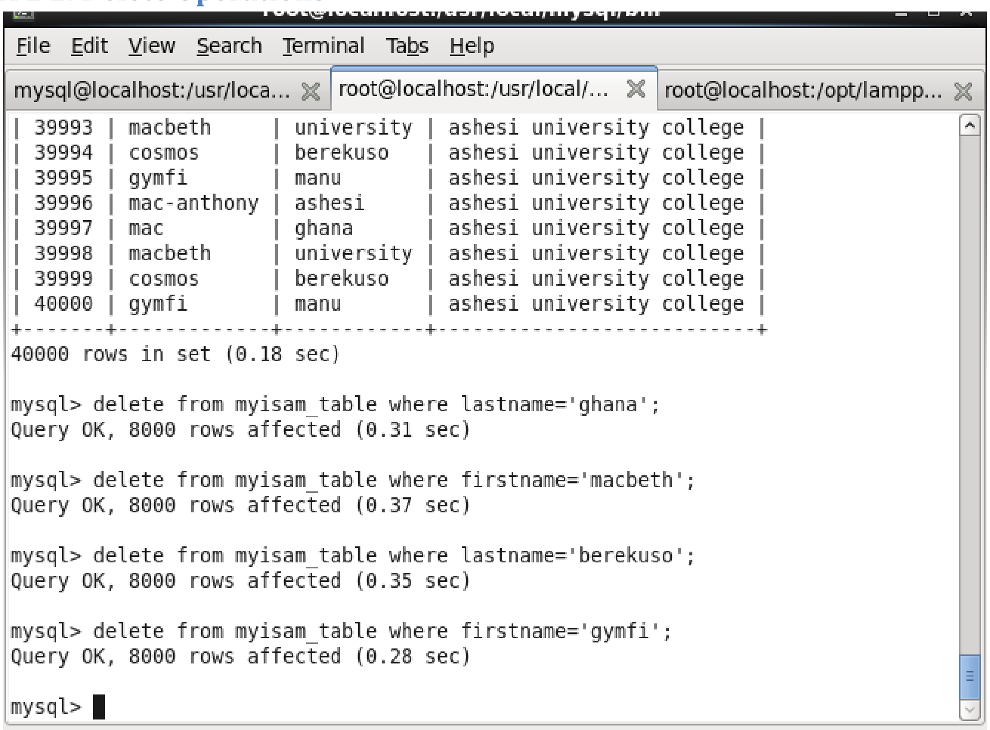
A.2: Screen Shots of Tests Performed on MyISAM Storage Engine

A.2.1: Update Operations



```
root@localhost:/usr/local/mysql/bin
File Edit View Search Terminal Tabs Help
mysql@localhost:/usr/loca...  root@localhost:/usr/local/...  root@localhost:/opt/lampp...
| 39997 | mac          | manu     | ashesi university college |
| 39998 | macbeth      | manu     | ashesi university college |
| 39999 | cosmos       | manu     | ashesi university college |
| 40000 | gymfi        | manu     | ashesi university college |
+-------+--------------+----------+---------------------------+
40000 rows in set (0.14 sec)

mysql> update myisam_table set lastname='ashesi' where firstname='mac-anthony';
Query OK, 8000 rows affected (0.38 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update myisam_table set lastname='ghana' where firstname='mac';
Query OK, 8000 rows affected (0.36 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update myisam_table set lastname='university' where firstname='macbeth';
Query OK, 8000 rows affected (0.45 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update myisam_table set lastname='berekuso' where firstname='cosmos';
Query OK, 8000 rows affected (0.44 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql>
```

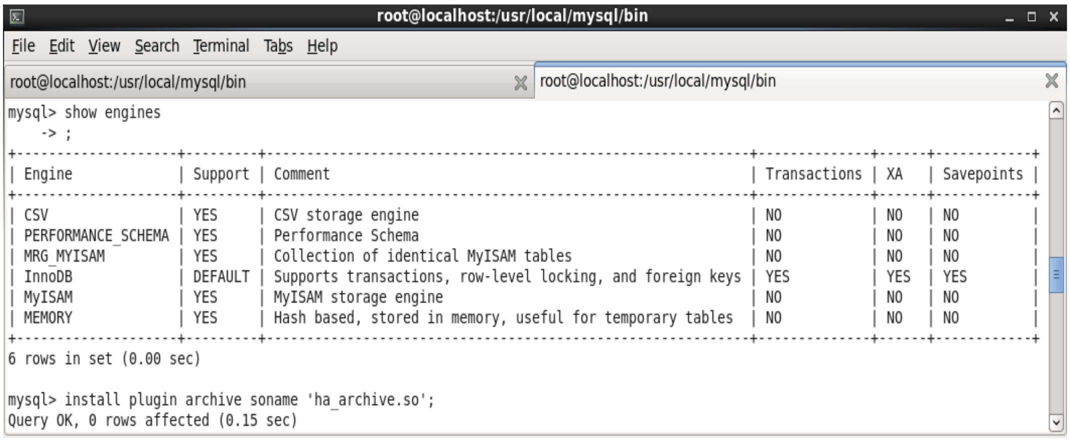
A.2.2: Delete Operations



```
root@localhost:/usr/local/mysql/bin
File Edit View Search Terminal Tabs Help
mysql@localhost:/usr/loca...  root@localhost:/usr/local/...  root@localhost:/opt/lampp...
| 39993 | macbeth      | university | ashesi university college |
| 39994 | cosmos       | berekuso   | ashesi university college |
| 39995 | gymfi        | manu       | ashesi university college |
| 39996 | mac-anthony  | ashesi     | ashesi university college |
| 39997 | mac          | ghana      | ashesi university college |
| 39998 | macbeth      | university | ashesi university college |
| 39999 | cosmos       | berekuso   | ashesi university college |
| 40000 | gymfi        | manu       | ashesi university college |
+-------+--------------+------------+---------------------------+
40000 rows in set (0.18 sec)

mysql> delete from myisam_table where lastname='ghana';
Query OK, 8000 rows affected (0.31 sec)

mysql> delete from myisam_table where firstname='macbeth';
Query OK, 8000 rows affected (0.37 sec)

mysql> delete from myisam_table where lastname='berekuso';
Query OK, 8000 rows affected (0.35 sec)

mysql> delete from myisam_table where firstname='gymfi';
Query OK, 8000 rows affected (0.28 sec)

mysql>
```

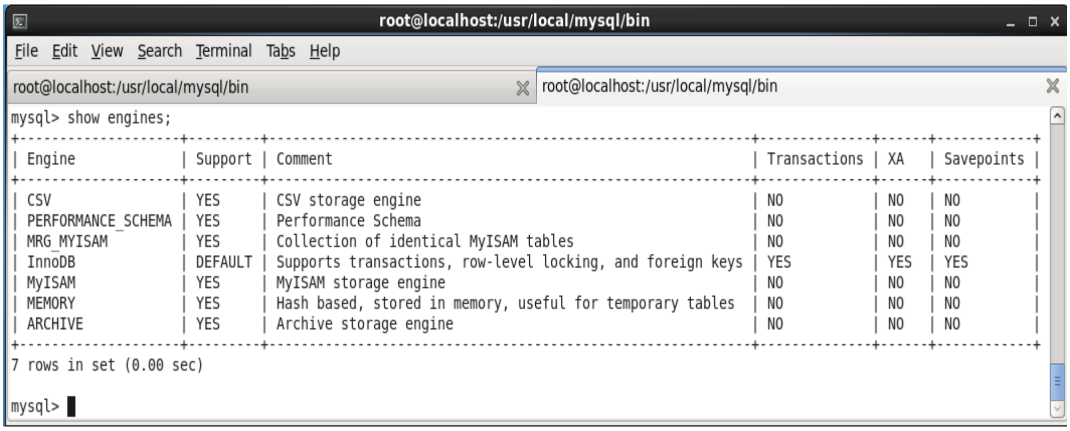
A.3: Screen Shots of Tests Performed on ARCHIVE Storage Engine

A.3.1: Plugging in ARCHIVE Engine to the Running Server



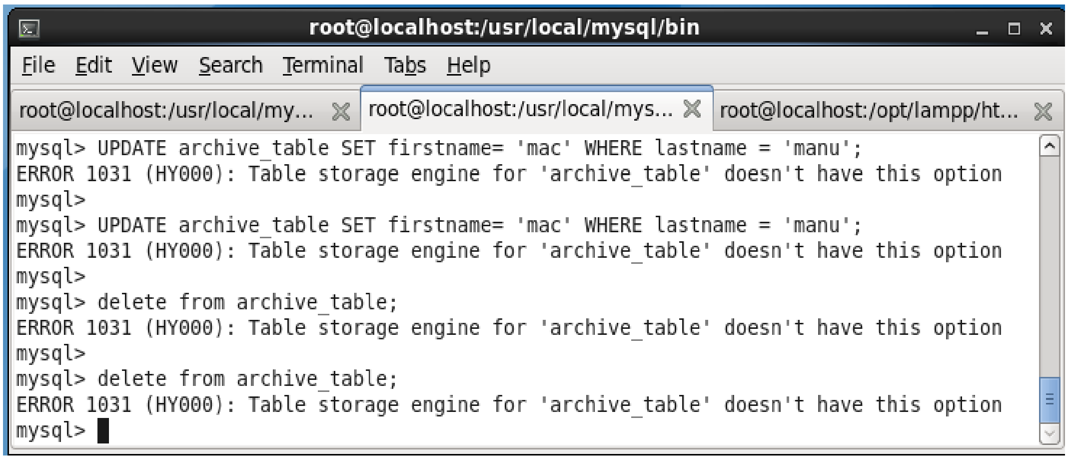

```
mysql> show engines
    -> ;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
6 rows in set (0.00 sec)

mysql> install plugin archive soname 'ha_archive.so';
Query OK, 0 rows affected (0.15 sec)
```

A.3.2: Prove of ARCHIVE Storage Engine Enabled.




```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
7 rows in set (0.00 sec)

mysql>
```

A.3.3: Update and Delete Operations




```
mysql> UPDATE archive_table SET firstname= 'mac' WHERE lastname = 'manu';
ERROR 1031 (HY000): Table storage engine for 'archive_table' doesn't have this option
mysql>
mysql> UPDATE archive_table SET firstname= 'mac' WHERE lastname = 'manu';
ERROR 1031 (HY000): Table storage engine for 'archive_table' doesn't have this option
mysql>
mysql> delete from archive_table;
ERROR 1031 (HY000): Table storage engine for 'archive_table' doesn't have this option
mysql>
mysql> delete from archive_table;
ERROR 1031 (HY000): Table storage engine for 'archive_table' doesn't have this option
mysql>
```

A.4: Screen Shots of Tests Performed on MEMORY Storage Engine

A.4.1: Update Operations

```
root@localhost:/usr/local/mysql/bin
File  Edit  View  Search  Terminal  Tabs  Help
mysql@localhost:/usr/loca...   root@localhost:/usr/local/...   root@localhost:/opt/lampp...
| 39997 | mac        | manu     | ashesi university college |
| 39998 | macbeth    | manu     | ashesi university college |
| 39999 | cosmos     | manu     | ashesi university college |
| 40000 | gymfi      | manu     | ashesi university college |
+-------+------------+----------+---------------------------+
40000 rows in set (0.11 sec)

mysql> update memory_table set lastname='ghana' where firstname='mac-anthony';
Query OK, 8000 rows affected (0.03 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update memory_table set lastname='berekuso' where firstname='mac';
Query OK, 8000 rows affected (0.03 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update memory_table set lastname='accra' where firstname='macbeth';
Query OK, 8000 rows affected (0.02 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update memory_table set lastname='university' where firstname='cosmos';
Query OK, 8000 rows affected (0.03 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> 
```

A.4.1: Delete Operations

```
root@localhost:/usr/local/mysql/bin
File  Edit  View  Search  Terminal  Tabs  Help
mysql@localhost:/usr/loca...   root@localhost:/usr/local/...   root@localhost:/opt/lampp...
| 39992 | mac       | berekuso   | ashesi university college |
| 39993 | macbeth   | accra      | ashesi university college |
| 39994 | cosmos    | university | ashesi university college |
| 39995 | gymfi     | manu       | ashesi university college |
| 39997 | mac       | berekuso   | ashesi university college |
| 39998 | macbeth   | accra      | ashesi university college |
| 39999 | cosmos    | university | ashesi university college |
| 40000 | gymfi     | manu       | ashesi university college |
+-------+-----------+------------+---------------------------+
32000 rows in set (0.10 sec)

mysql> delete from memory_table where firstname='cosmos';
Query OK, 8000 rows affected (0.03 sec)

mysql> delete from memory_table where lastname='manu';
Query OK, 8000 rows affected (0.03 sec)

mysql> delete from memory_table where firstname='mac';
Query OK, 8000 rows affected (0.03 sec)

mysql> delete from memory_table where lastname='berekuso';
Query OK, 0 rows affected (0.00 sec)

mysql> 
```

A.5: Screen Shots of Tests Performed on CSV Storage Engine

A.5.1: Update Operations

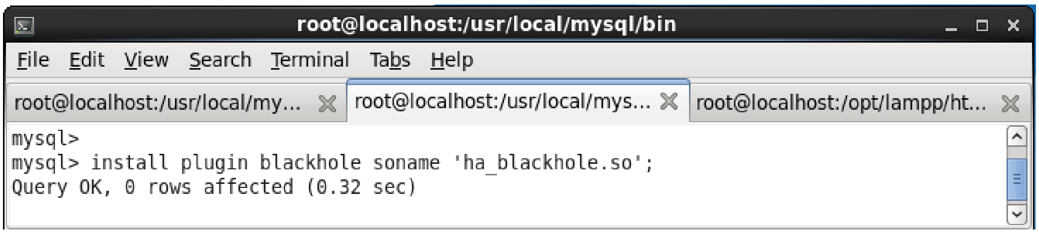
```
root@localhost:/usr/local/mysql/bin
File  Edit  View  Search  Terminal  Tabs  Help
mysql@localhost:/usr/local/mysql/bin        root@localhost:/usr/local/mysql/bin
|   1 | mac         | manu      | ashesi university college |
|   1 | macbeth     | manu      | ashesi university college |
|   1 | cosmos      | manu      | ashesi university college |
|   1 | gymfi       | manu      | ashesi university college |
+----+------------+---------+--------------------------+
40000 rows in set (0.32 sec)

mysql> update csv_table set lastname='ghana' where firstname='mac-anthony';
Query OK, 8000 rows affected (0.62 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update csv_table set lastname='berekuso' where firstname='mac';
Query OK, 8000 rows affected (0.44 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update csv_table set lastname='university' where firstname='macbeth';
Query OK, 8000 rows affected (0.39 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql> update csv_table set lastname='accra' where firstname='cosmos';
Query OK, 8000 rows affected (0.35 sec)
Rows matched: 8000  Changed: 8000  Warnings: 0

mysql>
```

A.5.2: Delete Operations

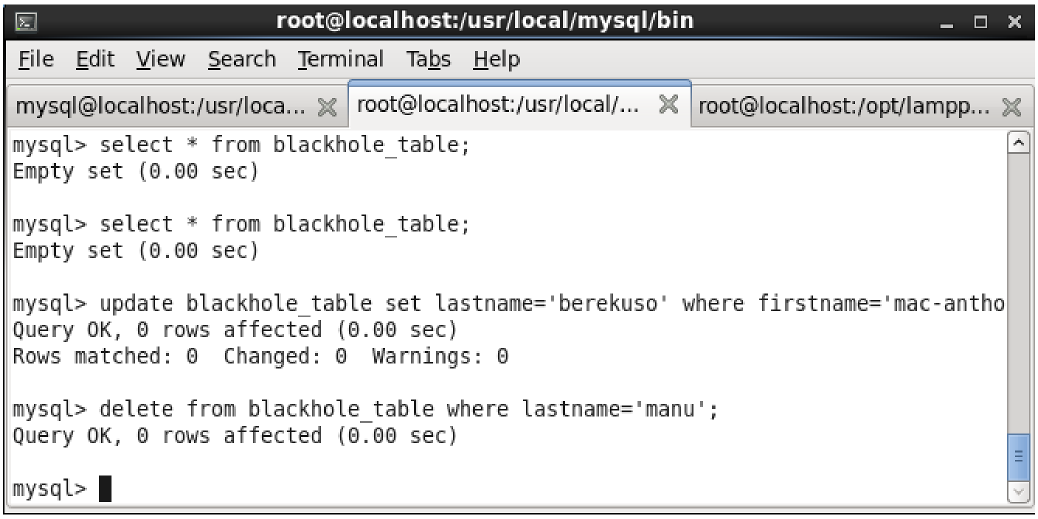
```
root@localhost:/usr/local/mysql/bin
File  Edit  View  Search  Terminal  Tabs  Help
mysql@localhost:/usr/local/mysql/bin        root@localhost:/usr/local/mysql/bin
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
|   1 | gymfi       | manu        | ashesi university college |
+----+------------+-----------+--------------------------+
40014 rows in set (0.22 sec)

mysql> delete from csv_table where lastname='ghana';
Query OK, 8000 rows affected (0.17 sec)

mysql> delete from csv_table where lastname='manu';
Query OK, 8014 rows affected (0.15 sec)

mysql> delete from csv_table where lastname='berekuso';
Query OK, 8000 rows affected (0.21 sec)

mysql> delete from csv_table where lastname='university';
Query OK, 8000 rows affected (0.13 sec)

mysql>
```

A.6: Screen Shots of Tests Performed on BLACKHOLE Storage Engine

A.6.1: Plugging in ARCHIVE engine to the Running Server



A.6.2: Select, Update and Delete Operations



Appendix B

This appendix shows the tools, commands, and resources needed to obtain the MySQL source code. It also includes the process of building and installing the MySQL server.

B.1: How to Get MySQL Source Files

There a number of ways of getting the source code of MySQL. It can actually be obtained from someone if that is all right. However, if you would like to get the latest version of the source code, it will have to be obtained with a software management system. For this project bazaar was used. With a few commands, bazaar allows you to get the latest version of MYSQL source from launchpad. Launchpad has various teams working on a variety of open source software projects.

MySQL was developed and is being maintained on a Linux system. For some reasons that will be stated shortly, it is safer and appropriate to also modify and build the source code on a Linux system. The chosen Linux system must have the necessary configuration and should support the libraries needed for the build process. CentOS was used for this project,.

To get the source, open the terminal and execute the following commands:

```
shell> bzr init-repo <name_of_folder>/mysql-server
shell> cd <name_of_folder>/mysql-server
shell> bzr branch lp:mysql-server/5.5 mysql-5.5
```

B.2: Installing the Libraries Needed to Build MySQL Source Files

After getting the source code, the first thing to do is to compile and link the files. However, a number of libraries are needed to enable a successful build process. Without any of these libraries, the build process will pose to be very difficult than it should be.

The first tool needed is a good make program. "Although some platforms come with their own make implementations, it is highly recommended that you use GNU make 3.75 or newer." To install the most up-to-date version of a tool, connect to the internet and install from the terminal. Cmake was used in this project and the command to install it is yum install cmake. In addition to a good make program the most recent versions of autoconf, automake, libtool, m4 and bison must be installed. A c++ compiler must also be installed.

B.3: The Build Process

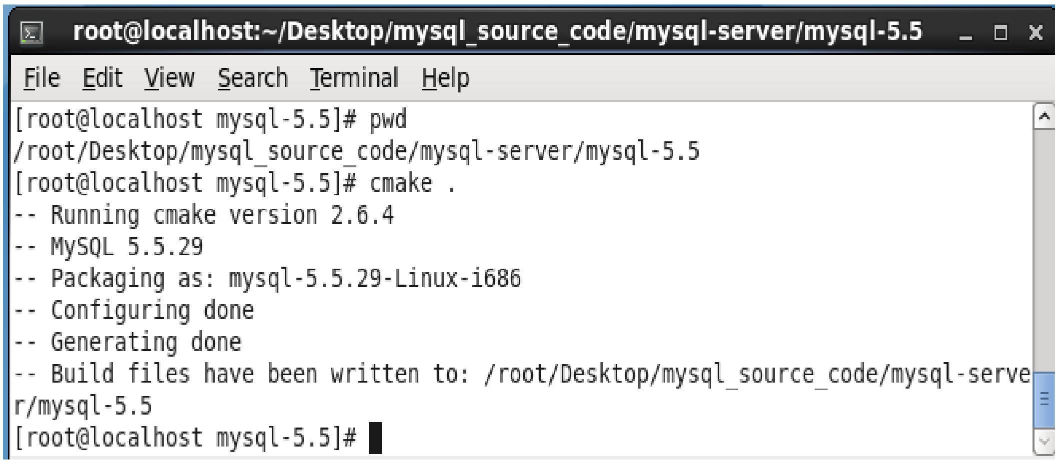
The first step is to configure the build process. To do so, get into the top-level source directory and run the cmake command.

```
cmake .
```

Configuration parameters can be added to this command. For EXAMPLE,

```
cmake . -DWITH_ARCHIVE_STORAGE_ENGINE=1
```

This adds ARCHIVE to the list of statically compiled storage engines. The image below shows the terminal output after executing cmake.



To the build and link the files, run the make command whiles in the top-level source directory. That is:

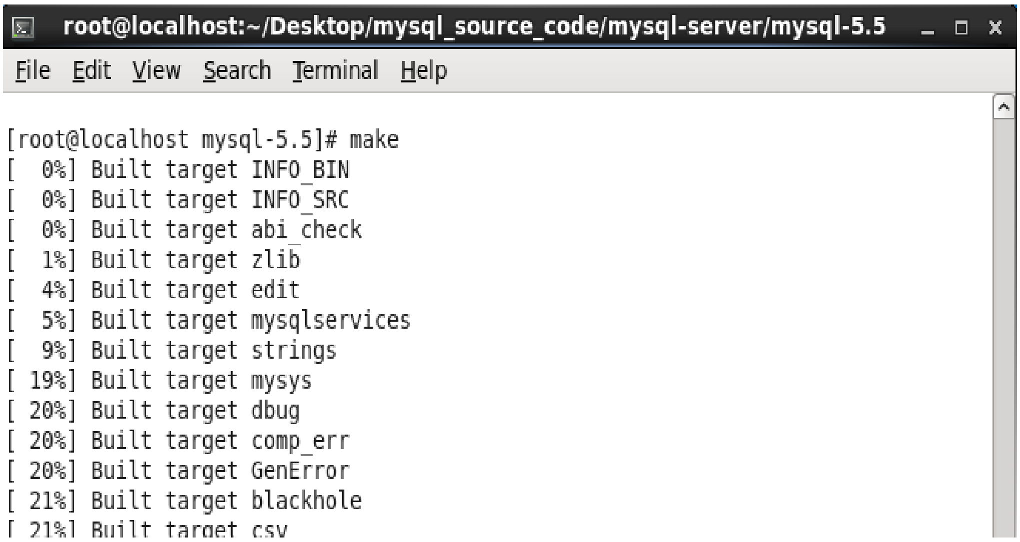
```
make
```

or

```
make VERBOSE=1
```

This shows how compiler is invoked.

The terminal below shows an ongoing build process.



The next step is to install the files. The make install command does this.

```
make install DESTDIR="/some/absolute/path"
```

If no destination is specified, the default path in the CmakeCache.txt file will be used. The path is usually /usr/local/mysql. The image below show the end of a make install process and the begin of another.

```
root@localhost:~/Desktop/mysql_source_code/mysql-server/mysq
File  Edit  View  Search  Terminal  Help
-- Up-to-date: /usr/local/mysql/sql-bench/copy-db
-- Up-to-date: /usr/local/mysql/sql-bench/test-alter-table
-- Up-to-date: /usr/local/mysql/sql-bench/test-ATIS
-- Up-to-date: /usr/local/mysql/sql-bench/graph-compare-results
-- Up-to-date: /usr/local/mysql/sql-bench/server-cfg
-- Up-to-date: /usr/local/mysql/sql-bench/test-alter-table
-- Up-to-date: /usr/local/mysql/sql-bench/compare-results
-- Up-to-date: /usr/local/mysql/sql-bench/test-select
-- Up-to-date: /usr/local/mysql/sql-bench/test-insert
-- Up-to-date: /usr/local/mysql/sql-bench/bench-init.pl
-- Up-to-date: /usr/local/mysql/sql-bench/innotest1b
-- Up-to-date: /usr/local/mysql/sql-bench/test-connect
-- Up-to-date: /usr/local/mysql/man/man1/mysqlman.1
[root@localhost mysql-5.5]# make install
[  0%] Built target INFO_BIN
[  0%] Built target INFO_SRC
[  0%] Built target abi_check
[  1%] Built target zlib
[  4%] Built target edit
[  5%] Built target mysqlservices
[  9%] Built target strings
[ 19%] Built target mysys
[ 20%] Built target dbug
[ 20%] Built target comp_err
[ 20%] Built target GenError
[ 21%] Built target blackhole
[ 21%] Built target csv
[ 21%] Built target csv_embedded
[ 23%] Built target myisammrg
[ 25%] Built target myisammrg_embedded
[ 28%] Built target heap
[ 30%] Built target heap_embedded
[ 30%] Built target hp_test1
```

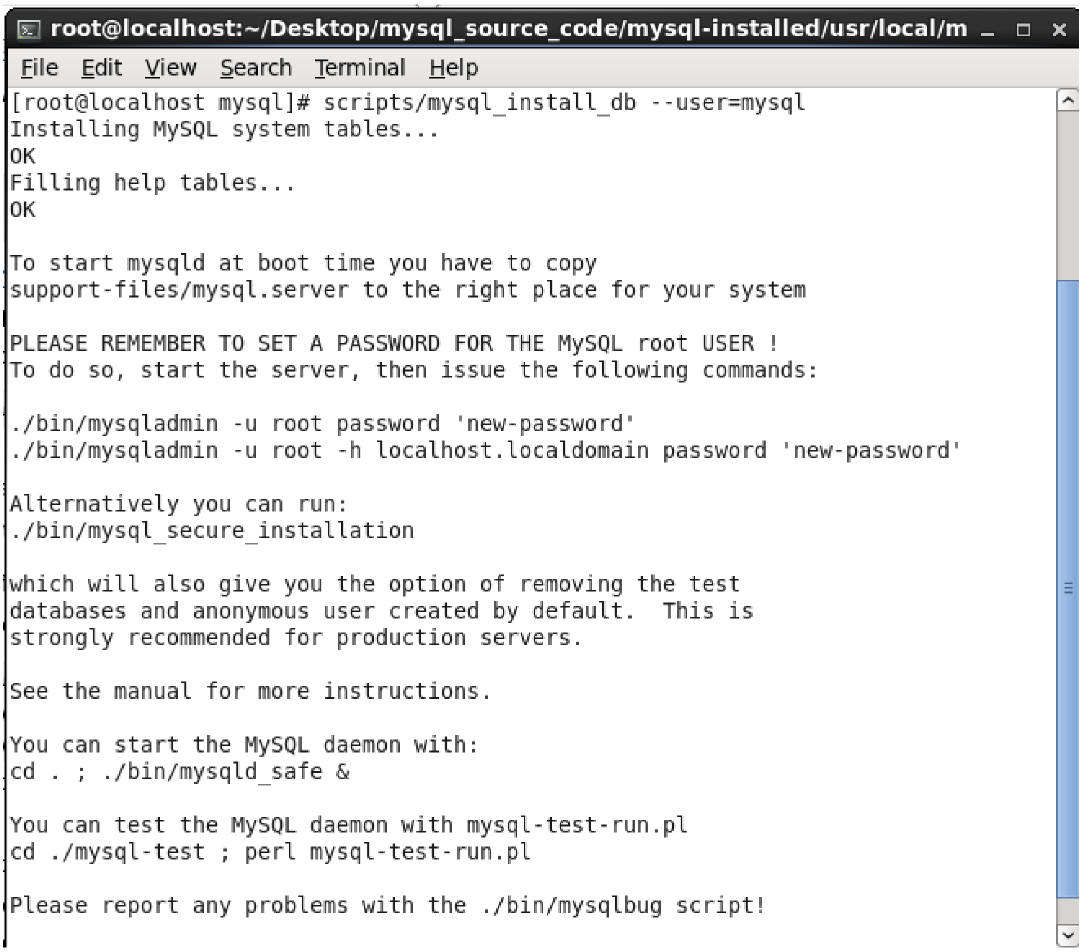
B.4: How to Start the Server

The first step is to initialize the grant tables. They are set up by the mysql_install_db program. To be able to create these tables, the user must be root linux user. Navigate to the scripts folder in the mysql installation folder (i.e. /usr/local/mysql) and execute the mysql_install_db program.

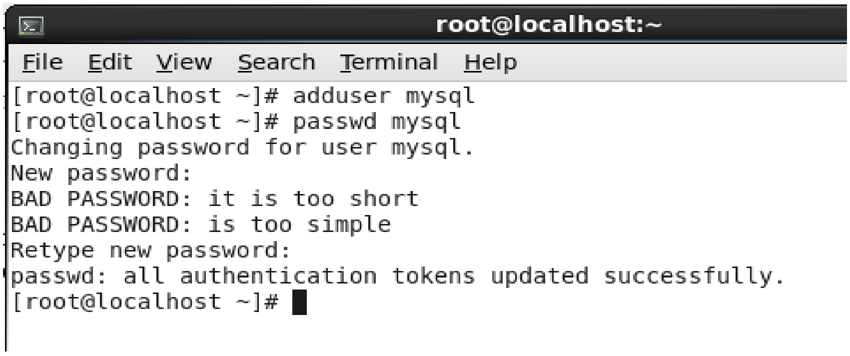
```
mysql_install_db --user=mysql
```

An option can be added to create a new database data directory in the install folder (i.e. /usr/local/mysql) as shown below.

```
mysql_install_db --user=mysql --datadir=/usr/local/mysql/new_datadir
```



To start the server, switch to mysql linux user. Create mysql linux user account if there is none. The figure shows the process of adding a new adding account from the terminal.



```
[root@localhost ~]# adduser mysql
[root@localhost ~]# passwd mysql
Changing password for user mysql.
New password:
BAD PASSWORD: it is too short
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
```

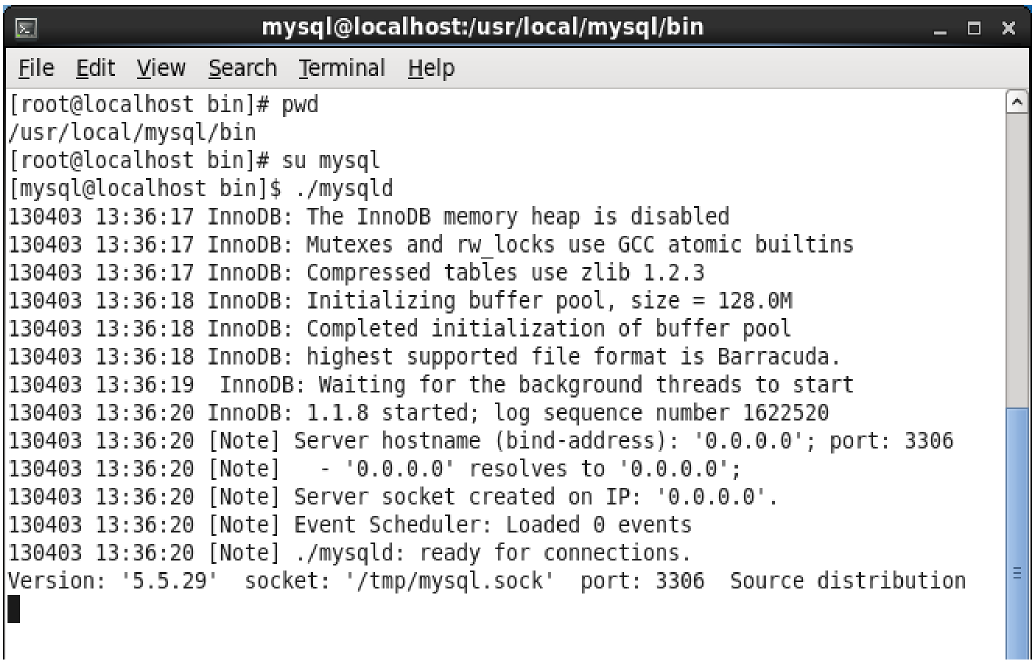
Navigate to the bin folder in the installation folder and execute mysqld program. That is:

```
/usr/local/mysql/bin/mysqld
```

The data directory to be used and the port number can be specified when starting the server. That is:

```
/usr/local/mysql/bin/mysqld --datadir=/usr/local/mysql/new_datadir --
port=3306
```

The image below shows the process of starting the server and the output from the terminal.



```
[root@localhost bin]# pwd
/usr/local/mysql/bin
[root@localhost bin]# su mysql
[mysql@localhost bin]$ ./mysqld
130403 13:36:17 InnoDB: The InnoDB memory heap is disabled
130403 13:36:17 InnoDB: Mutexes and rw_locks use GCC atomic builtins
130403 13:36:17 InnoDB: Compressed tables use zlib 1.2.3
130403 13:36:18 InnoDB: Initializing buffer pool, size = 128.0M
130403 13:36:18 InnoDB: Completed initialization of buffer pool
130403 13:36:18 InnoDB: highest supported file format is Barracuda.
130403 13:36:19  InnoDB: Waiting for the background threads to start
130403 13:36:20 InnoDB: 1.1.8 started; log sequence number 1622520
130403 13:36:20 [Note] Server hostname (bind-address): '0.0.0.0'; port: 3306
130403 13:36:20 [Note]   - '0.0.0.0' resolves to '0.0.0.0';
130403 13:36:20 [Note] Server socket created on IP: '0.0.0.0'.
130403 13:36:20 [Note] Event Scheduler: Loaded 0 events
130403 13:36:20 [Note] ./mysqld: ready for connections.
Version: '5.5.29'  socket: '/tmp/mysql.sock'  port: 3306  Source distribution
```

B.5: How to Start the MySQL Client

From the current terminal click on File and then click on Open Terminal To launch another terminal. The present working directory is the directory from which the server was started from. That is /usr/local/mysql/bin. Execute mysql program to start the client. The image below shows this process and the output from the terminal.