American Journal of Computer Architecture 2022, 9(1): 1-7
DOI: 10.5923/j.ajca.20220901.01

Implementation of High Available and Scalable Syslog Server with NoSQL Cassandra Database and Message Queue

Rajeshkumar Sasidharan

Systems Engineering, Scientific Games, Illinois, Chicago, USA

Abstract Today's IT infrastructure can be very demanding in terms of event logs storage and management. Thousands of devices, applications, operating systems, and appliances produce vast event log messages. Enterprise and service providers use third-party commercial log management software to collect data from the servers, storage, and network for alerting, auditing, and forensic purposes. This proprietary software is very costly and has limitations in terms of storage and clients' licensing restrictions. This project aims to develop new log management software using the open-source tools Syslog and the Cassandra database. This project will reduce companies' expenses and overcome the storage and client licensing limitations; using Syslog, it collects all the alerts, system events, and logs from the servers, network devices, and storage components, which are then stored in the database for future reference. Additionally, the team is alerted whenever a rule matches an event or an alert is received from the client machines. The proposed software will help the team proactively act on an incident before it becomes a problem. The Cassandra database can hold multiple terabytes of data and provide high availability by replicating the database on multiple servers. Finally, this project saves millions for the service provider by successfully managing banks, corporate accounts, and retail customers' logs and events messages for auditing and forensic purposes.

Keywords Open-source log management software, Syslog, AMQP, NoSQL, Syslog server, Distributed database Syslog server, Cassandra database Syslog application, NXLog, Apache, PHP

1. Introduction

Our client is an Internet Service Provider (ISP), Enterprise Service Provider (ESP), and cloud service provider, so they manage thousands of server network devices for hundreds of enterprise and retail customers, including their infrastructure and environments. Therefore, maintaining uptime commitment and Service Level Agreement (SLA) is the primary criteria in terms of providing better service to their customers. Uptime and SLA primarily depend on the way in which events, incidents, and problem management are addressed. Event management helps quickly resolve incidents before they require problem management mode. Therefore, monitoring event management is a critical task that requires the right software to address the issue. Most of the problems can be addressed when the event is properly assessed by the operation team. Capturing events becomes more important as it initiates the next step of monitoring and alerts the relevant team.

* Corresponding author:
sr.rajeshkumar@gmail.com (Rajeshkumar Sasidharan)
Received: Mar. 15, 2022; Accepted: Mar. 30, 2022; Published: Apr. 15, 2022
Published online at http://journal.sapub.org/ajca

Since the relevant team is responsible for customer infrastructure and setup, they need to protect the system from external intruders. Any attempt made by intruders must be clearly captured through Syslog and addressed through necessary actions. Events captured through Syslog will be analyzed and ameliorated in the OS and applications services. Therefore, events captured through Syslog will help improve the performance of the system and alert the operation team of any malfunction of the OS and its applications or unwanted attempts by external intruders. In addition, it is useful for auditing and forensic purposes.

2. Syslog – An Overview

Syslog is a standard for computer data logging. The Syslog protocol utilizes a layered architecture, which allows the use of any number of transport protocols for the transmission of Syslog messages [1].

Syslog messages should be captured and stored centrally; this practice is now an unavoidable task for enterprise providers who support customers' servers, appliances, and all network and storage components [1].

Syslog captures all of the OS and its related messages, including warnings, errors, and critical events, which is required for compliance during an audit. It will be widely

used for forensic purposes and to analyze and trap all the events during unavoidable incidents [1].

Syslog Facility Level Values:

Table 1. Syslog 24 Facility Level Value

Facility Number	Keyword	Facility Description
0	Kern	kernel messages
1	User	user-level messages
2	Mail	mail system
3	Daemon	system daemons
4	Auth	security/authorization messages
5	Syslog	messages generated internally by syslogd
6	Lpr	line printer subsystem
7	News	network news subsystem
8	Uucp	UUCP subsystem
9		clock daemon
10	Authpriv	security/authorization messages
11	ftp	FTP daemon
12	-	NTP subsystem
13	-	log audit
14	-	log alert
15	Cron	clock daemon
16	local0	local use 0 (local0)
17	local1	local use 1 (local1)
18	local2	local use 2 (local2)
19	local3	local use 3 (local3)
20	local4	local use 4 (local4)
21	local5	local use 5 (local5)
22	local6	local use 6 (local6)
23	local7	local use 7 (local7)

Syslog Severity Level Values:

Table 2. Syslog Severity Level Value

Code	Severity	Keyword	Description
0	Emergency	emerg (panic)	System is unusable.
1	Alert	Alert	Action must be taken immediately.
2	Critical	Crit	Critical conditions.
3	Error	err (error)	Error conditions.
4	Warning	warning (warn)	Warning conditions.
5	Notice	Notice	Normal but significant conditions.
6	Informational	Info	Informational messages.
7	Debug	Debug	Debug-level messages.

3. Problem Statement

Presently, our client is using commercial log management software that is licensed based on the number of devices and storage space. However, our client manages thousands of devices, and that number is growing every day; hence, this license model does not work. The log management appliance is a non-customizable and locked product; also, it does not have high availability or scalability. In addition, it has a flat-file mechanism for storing logs, which degrades the performance of queries. It supports only limited events per second; therefore, the possibility of dropping event messages is very high. To increase the events per second acceptance, the client must buy the higher-end devices. Their current software has many graphical interface screens and report formats available, which is good, but our client is using only a limited number of the features.

a. Syslog events are stored in a flat file, which is cumbersome to process, and it is difficult to query reports based on the day, week, or month.
b. Storing the logs in the MySQL database is sufficient, and it does support millions of records. However, when the database reaches billions of records, it degrades the system's performance. Writing data into the MySQL database creates a bottleneck when the number of transactions is so high.
c. Therefore, considering the database used to store the logs for this project, events and alerts should be highly available, scalable, and high-performance.
d. Regarding the Syslog software, we must consider high availability, scalability, performance, and zero log loss capability, as the existing commercial software is dropping events when it reaches its threshold.
e. Our client has multiple data centers split across multiple geographic areas and thousands of devices. Therefore, designing the product should be foolproof and stable, and it should ensure the logs are transferred from one location to another location using a secure and reliable method.
f. Copying the logs as files from one datacenter to a central datacenter server can be reliable, but it does not help in retrieving the run time report from the central database server.

In summary, the Syslog server, database server, and log transfer application should not only be open-source but should also be high performance, scalable and reliable, and easily manageable.

4. Requirement Analysis & Decision

We conducted several meetings and discussions with all of our client's stakeholders, carefully captured all the requirements, prepared the final draft, shared it with everyone, and obtained approval to proceed to select the software from the open-source market.

We finally decided to use log management software that meets the following minimum core components.

Table 3. Core Components and their Specification

Core Components	Description
Log receiver / Syslog server	To receive logs from all the client machines. It should receive at least 10,000 messages per second.
Log parser / parser engine	To parse the logs received by the log receiver and store them in the database.
Messaging engine	Messaging system that can be used to store all the logs received by the log receiver for parsing and alerting.
Alert engine	To alert the user based on the criticality of the logs as defined by the user.
Database server	To store the log events. It should be capable of holding billions of records and terra bytes of data. It should support at least Availability and Partition Tolerance
Web server	To serve the web reports and query the logs from the database online.

Table 4. Comparative Analysis for Software and Selection

Components	open-source software comparison & selection
Log receiver	Nxlog vs Syslog While Syslog is exclusively available on Unix, Nxlog is available on both Unix and Windows, it furthermore supports Windows event logs. As a result, Nxlog is considered.
Messaging	Apache QPID vs Active MQ (MQ- Message Queuing) Both use the AMQP protocol and have the same functionality. However, we considered Apache QPID, because our client is already using it and familiar with it.
NoSQL Database	Cassandra vs MongoDB MongoDB offers a master-slave design, in which a single master is in charge of a group of slave nodes. However, because Cassandra allows many masters, we considered it as our NoSQL database.
Database	MySQL vs PostgreSQL Both are nearly identical in terms of performance and functionality. However, because our Client is already familiar with MySQL, we contemplated using it.
Webserver	Apache vs Nginx Both are popular open-source webservers, But the Client is very comfortable with the Apache style of configuration files, hence we considered Apache Webserver.
Log parser	C/C++ vs Python Considering the requirements and speed of processing for a large volume of logs, we have chosen C/C++
Alerting engine	C/C++ vs Python Considering the processing needs and speed, as well as making judgments based on a massive volume of log data. We have chosen C/C++.

One important point which was mentioned and captured in the requirements is performance, scalability, reliability, and high availability should be considered when we select and design the components for the application development.

To Design the log management system, considering all the above factors, we have analyzed most of the software in the open-source market and checked their features, stability, and continuous support by the communities.

We evaluated the bunch of software, and no doubt most of them are good with a lot of features after carefully selecting the following software. We ensured that it would fulfill our requirements based on the business and end-user requirements.

4.1. NXLog and Its Features

NXLog can work in a heterogeneous environment to collect event logs from thousands of different sources in many formats. NXLog can accept event logs from Transmission Control Protocol (TCP), User Datagram Protocol (UDP), files, databases, and various other sources in different formats such as Syslog and windows event log, among others [3].

We do consider the NXLog as our Syslog server because it has high security and high-performance I/O can receive messages from thousands of devices, can process 100,000 EPS (events per second), and supports Transport Layer Security (TLS) / Secure Sockets Layer (SSL), which ensures that the logs are transferred across the network in a secure mode [3].

Therefore, NXLog has met the performance, security, reliability, and high-availability parameter requirements, and after conducting a basic level of testing and sharing the

output with all the stakeholders, it was accepted as the Syslog server for this project.

4.2. Cassandra Database

Apache Cassandra is an open-source, distributed, decentralized, elastically scalable, highly available, fault-tolerant, tunable, consistent, column-oriented database that bases its distribution design on Amazon's Dynamo and its data model on Google's Bigtable [2].

We can replace failed nodes in the cluster with no downtime, and data can be replicated to multiple data centers to offer improved local performance and prevent downtime if one data center experiences a catastrophe such as a fire or a flood.

Based on Brewer's CAP theorem, which states that "within a large-scale distributed data system, there are three requirements that have a relationship of sliding dependency: Consistency, Availability and Partition Tolerance" [5].

Consistency: All the database clients will read the same value for the same query, even given concurrent updates [5].

Availability: All database clients will always be able to read and write data [5].

Partition Tolerance: The database can be split into multiple machines; it can continue functioning in the face of network segmentation breaks [5].

Brewer's theorem states that only two of the three can be strongly supported.

Cassandra chooses Availability and Partition Tolerance from the CAP guarantees, compromising on data Consistency to some extent. Cassandra's design and its label are "eventually consistent" [2].

4.3. Messaging System

4.3.1. Advanced Message Queuing Protocol

Advanced Message Queuing Protocol (AMQP) is an open internet protocol standard for message-queuing communications and is also called the wired protocol [2].

The defining features of AMQP are message orientation, queueing, routing (point-to-point and publish-and-subscribe), reliability, and security [2].

4.3.2. Apache QPID

Apache QPID is a cross-platform enterprise messaging system that implements the AMQP, providing message brokers written in C++ and Java along with clients for C++, Java JMS, Net, Python, and Ruby. It also transfers messages from one system to another without any issue. It is a reliable and secure messaging system, and it supports high availability and ensures no data loss. Apache QPID implements the latest AMQP specification, and it is a 100% AMQP-enabled framework. It allows users to create multiple queues in the server and it also supports creating queues during runtime. It supports transaction management, queuing, distribution, security, management, clustering, federation, heterogeneous multi-platform support, and much more [6].

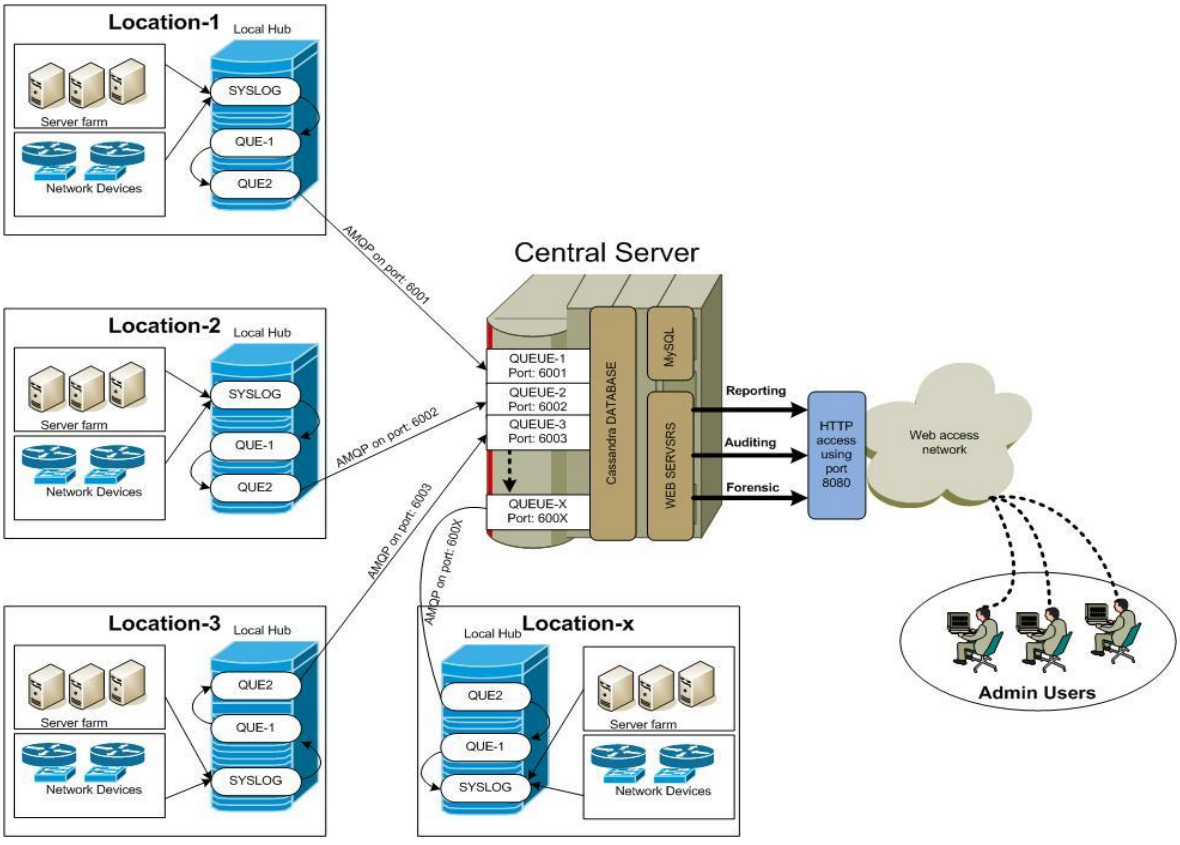

Figure 1. High-level architecture

4.4. Web Module Components

4.4.1. Apache Server

Apache is one of the best open-source web servers on the internet, and it is a powerful, flexible HTTP/1.1 compliant web server that is actively being developed [8].

4.4.2. PHP

PHP is an open-source general-purpose server-side scripting language originally designed to produce dynamic web pages. It has improved considerably, and many features are available since it has begun trying to compete with major server-side web application languages [7].

4.4.3. MYSQL

MySQL is the world's most popular open-source database, and it helps to deliver high-performance and scalable database applications. It supports multiple storage engines such as InnoDB and MyISAM. It also supports replication and partitioning [9]. We decided to use MySQL as a backend for storing customer information such as usernames and passwords, and we also use it to keep the filtering rules for logs to alert users.

5. Solution Design

5.1. High-Level Architecture

The high-level architecture encompasses the complete flow of the application, the client's data centers in multiple locations, and the thousands of devices that are installed in each data center.

The events and logs will be accepted by the location hub server, and all the logs will be routed to the central server queue on a defined port using the Apache QPID application through AMQP. From the central server, all the logs will be stored in the Cassandra database, where the administrative users will connect through the web to access the logs and reports.

5.2. Central Server Architecture

In the central server, Apache QPID, the Cassandra database, MySQL, and Apache will be installed.

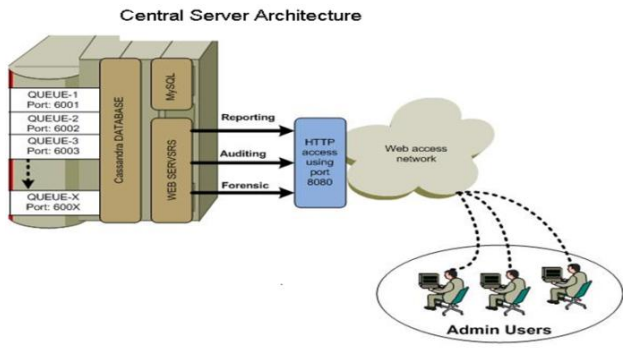

Figure 2. Central server architecture

Apache QPID uses AMQP as a protocol to accept messages from the location hub server queue. We will create multiple queues in different port numbers to accept events and logs as a message from hub server queues in multiple locations. The port numbers will differentiate, segregate, and accept the messages from different hub servers.

The Cassandra database will be installed in the central server to store all log events for auditing and forensic purposes. The parsing engine is the program that runs in the central server and will read the messages from the messaging queue and write them to the Cassandra database server. Once they are written to the Database, they will be removed from the messaging queue.

The alerting engine will not be implemented in the central server since the location hub server will be responsible for alerts.

Initially, based on the architecture, only one central server will be installed, but according to the business demand, multiple servers can be installed for high availability.

Even if the parsing engine is not running, the messaging queue will hold a large number of messages based on the disk space allotted, and no message loss will occur. The parsing engine will be monitored using the crontab script, and it will alert the central server administrator when it is down so that the administrators can restart the parsing engine.

Apache will run on the central server to serve the requests to administrative users online on port 8080; the server will have two static IP addresses and two Ethernet cards. One dedicated Ethernet card will have a dedicated IP address that will be allotted to the Apache web server, and one Ethernet card will be allocated to Apache QPID.

Users will be connected to an Apache server through the browser, and their details will be authenticated against the MySQL server database. To report and query the logs, Apache will connect to the Cassandra database.

Initially, web server access will be allowed only to internal system administrators, but the same access can be extended to internet users based on business demand. In the existing commercial system, the website cannot be opened to internet users due to user access limitations. In this new application, this limitation will be carefully considered and addressed.

As it is only for internal users and not open to the internet, the SSL option is not enabled in the Apache web server, but in the future, if required, the SSL can be enabled at any time in the Apache server.

5.3. Hub Server Architecture

In the location hub server, NXLog, Apache QPID and an alerting engine have been installed.

NXLog will be installed in the location hub server to accept logs events from the thousands of client machines and forward them to the Apache QPID AMQP server queue.

The alerting engine is the proposed program, which was developed in C++. It connects the central server to the MySQL database and fetches all the rulesets and stores them

in the in-memory cache for faster access. It then reads the message from the queue and checks the ruleset. If it matches, it triggers an email to the respective server logs administrator and moves the message from queue 1 to queue 2.

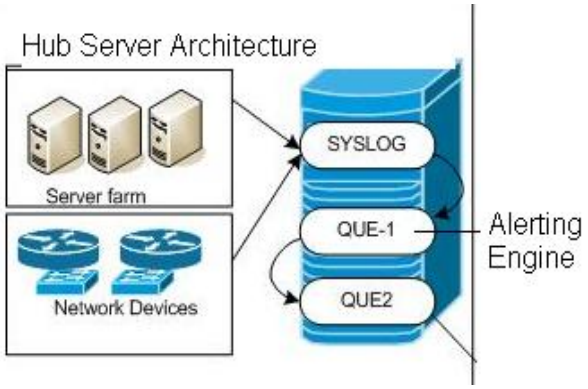

Figure 3. Hub server architecture

Even when the alerting engine is not running, the queue is capable of holding a large number of messages to ensure no data loss. The alerting engine's monitoring script will be created in crontab to check the alerting engine every 3 minutes and alert the respective administrative team if the alerting engine stops.

Once the message moves to queue 2, Apache QPID will forward the messages to the central server queue on the configured port. The respective routing and queue forward will be configured in the location hub server.

Transport from the location hub to the central server is reliable; even if there is any network congestion or issue, the message will be retransmitted with a defined time limit to ensure the messages have been delivered to the central server and to prevent data loss.

Apache QPID comes with many tools to configure and route the queue to a different queue without any complexity. It also supports high availability, senses the network reachability and ensures messages will be delivered without any issues.

Initially, we were planning to use file transfer for the logs from the location hub to a central server, but considering the agility and reliability of this mode of transfer, AMQP has been identified and chosen.

5.4. Parsing Engine

The parsing engine is a program developed in C++ and implemented only on a central server, as it is not required in the location hub server. It reads the messages from the messaging queue and writes them to the Cassandra database server. Once they are written to the database, the message will be removed from the messaging queue. The alerting engine will not be implemented in the central server the location hub server is already responsible for alerting. The parsing engine will run as multiple processes based on the number of queues, and it will map each process against the dedicated queue to read and delete the message once it is updated in the Cassandra database.

5.5. Alerting Engine

The alerting engine we developed connects the central server MySQL database, fetches all the rulesets, and store them in the in-memory cache for faster access. It then reads the messages from the queue and checks the ruleset. If it matches, it triggers an email to the respective server log administrator and moves the message from queue 1 to queue 2.

The alerting engine connects to the central MySQL database server on a defined period such as once every 2 hours, fetches all the rulesets and stores them locally based on their location.

The users have been notified that their rule will start work only after the next fetch cycle. The same information has been circulated through the web interface during the time when alert rulesets are created.

Alerting the engine is a single process since each location hub server will have only a single queue to accept messages from the client servers and appliances.

The parsing engine will not be implemented in the location hub server as parsing is not required.

5.6. High Availability & Failover

Multiple Central servers will be placed in production to provide high availability. Thus, if a server goes down completely or an application becomes unavailable, the other servers in the cluster will take care of it. As discussed previously, the crontab script that runs on the servers monitors the application's availability and restarts it when it goes down. If the script is unable to restart the program automatically, it will notify the administrative team. The same is true for the location hub server. Additionally, the entire server and apps will be monitored through the Client existing NMS application and alert the respective team, if there is an issue.

6. Reports

The following reports have been developed based on the business and user requests:

a) IP address-wise
b) Type-wise
c) Description-wise
d) Event-wise
e) Device type-wise
f) Based on the Syslog severity level keyword

As we are capturing all the information from the events and storing it in the Cassandra database, the reports can be customized and altered at a later point in time.

7. Results Summary

a. Cassandra database testing:

During the testing phase, many Syslog files from the Unix

machines were uploaded to the Cassandra database using scripts. The activity time was < 1 second, which is very satisfactory compared to other Relational Database Management Systems (RDBMS) databases. We have confirmed that the Cassandra database is effective in writing operations.

b. NXLog performance testing

We tested NXLog by sending a large number of transactional logs to NXLog, and we confirmed that it supports 10,000 EPS. In theory, it supports 100,000 EPS, which is the number mentioned on their home page.

c. AMQP Queue testing

Using Apache QPID, we configured and routed the messages from one queue to another in the remote server and ensured that the messages were reached successfully. We also tried to send more than 10,000 messages using the QPID AMQP queue and confirmed that it works at lightning speed.

8. Summary

The new open-source log management software developed would save organizations money by reducing their present expenses. In addition, it can support more client machines in the future by adding extra disk space. It has been carefully designed to use only open-source software such as Apache, Cassandra, MySQL, Apache QPID and NXLog. The communities that support these products have a large developer base, and they have been on the internet for a long time. As all the components are open-source, we can change or customize the components and codes at any time based on our requirements.

The proposed product has fulfilled all our requirements, such as high availability, scalability, performance, and reliability. Each and every component selected and implemented in this project fully supports the above-mentioned requirements. In addition, we have demonstrated that we developed a fully open-source product to replace the existing fully commercial log management software.

REFERENCES

[1] Gerhards, R. (2009). The Syslog Protocol (5424) Internet Engineering Task Force IETF RFC 5424 (Proposed Standard).

[2] Lakshman, A.; Malik, P. (2008.) Apache Cassandra. [Computer software]. Available: https://cassandra.apache.org/_/index.html.

[3] Botyanszki, B. (2011). NXLog. [Computer software]. Available: https://nxlog.co/.

[4] O'Hara, J. (2003). Advanced Message Queuing Protocol. [Computer software]. Available: http://amqp.org.

[5] Simon, S. (2000). Brewer's cap theorem. CS341 Distributed Information Systems, University of Basel (HS2012).

[6] Apache. (2005). Apache QPID. [Computer software]. Available: http://qpid.apache.org/.

[7] Bakken, S. S., Suraski, Z., & Schmid, E. (2000). PHP Manual: Volume 1. iUniverse, Incorporated.

[8] McCool, R. (1995). Apache HTTP Server. [Computer software]. Available: https://httpd.apache.org/docs/.

[9] Letkowski, J. (2015). Doing database design with MySQL. Journal of Technology Research, 6, 1.