IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

Big Data: A Transition from Conventional Databases

Sahil Gupta
Student Department of Information Technology
Dronacharya College of Engineering, Gurgaon, Haryana, India

Abstract: *The primary programming language created to manage data contained in database systems is called Structured Query Language (SQL). SQL was initially solely used with relational database management systems (RDBMS), but as additional categories of database systems have emerged, its use has substantially expanded. When processing Big Data, or datasets with great volume, velocity, and variety, highly distributed and scalable systems have been proven to be particularly effective at using SQL as a query language. Although conventional relational databases currently only make up a small portion of the database systems landscape, the majority of database courses that cover SQL only take traditional relational systems into account when using SQL. In this study, we suggest that SQL be taught as a general language, which can be applied to a variety of database systems, ranging from conventional RDBMSs to Big Data systems. In the context of emerging database systems like MapReduce, NoSQL, and NewSQL, this paper offers well-organized suggestions for introducing SQL. The description of a variety of course tools, such as virtual machines, sample projects, and in-class exercises, to provide hands-on experience with SQL across a wide range of contemporary database systems is a fundamental addition of this work. Descriptors for Categories and Subjects General terms for computer and information science education in K.3.2 [Computers and Education] Creating and experimenting Keywords Curriculums for databases; SQL; Query Language; Big Data 1.*

Keywords: Big Data

I. INTRODUCTION

The most used database language is SQL, or Structured Query Language. Data definition language (DDL) of SQL allows for the specification of database schemas. Data manipulation language (DML) of SQL supports operations to retrieve, store, modify, and delete data. Data control language (DCL) of SQL allows for the configuration of security access to databases by database administrators. One of the main factors for SQL's widespread use is that it is primarily a declarative language, meaning that it specifies a program's logic (what needs to be done) rather than control flow (how to do it); that it is declarative and uses English statements, making it relatively simple to learn and understand; and that it is a standard of the industry. It is somewhat portable between various database systems and is recognised by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). Although SQL was first developed for conventional relational databases, it has now been discovered to be a useful language in a number of novel database systems, particularly Big Data management systems (BDMSs) that process enormous amounts of data quickly and diversely. The majority of books and database courses solely discuss SQL in the context of RDBMs, despite the fact that BDMSs have drastically altered the database landscape and classic RDBMs now make up a relatively minor portion of it. This article suggests studying SQL, a potent language that may be used to query a variety of database systems, from conventional relational databases to contemporary Big Data systems. The description of the new database environment and the identification of broad categories of Big Data systems where SQL can be effectively employed are the key contributions of this article. • The provision of well-organized instructions for creating course units that concentrate on learning SQL on novel database systems like MapReduce, NoSQL, and NewSQL. • A thorough explanation of the course materials for each subject to enable students to practise using SQL on a variety of contemporary database systems. These tools include data generators, example programmes, virtual machines, projects, and in-class activities. • Files including all the materials provided to educators [1]. To give educators the flexibility to use and expand these resources in accordance with their unique needs.The remainder of the essay is structured as follows. The new database landscape and the many BDMSs that support SQL are described in Section 2. The specifics of how SQL can be used in the context of the new types of database systems are presented in Sections 3, 4, and 5. (MapReduce, NoSQL, and NewSQL). Numerous

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

class resources that will be made accessible in [1] are described in these sections. The integration of the suggested units into database courses is covered in Section 6. The paper is concluded in Section 7.

II. NEW DATABASE LANDSCAPE

One of the most important recent technological advancements in computer science is the creation and widespread application of highly distributed and scalable systems to process Big Data [3, 4, 5, 6]. Current application scenarios include social media data mining, scientific data analysis, recommendation systems, and online service analysis. These systems have considerably expanded the database landscape. BDMSs are frequently dynamically scalable, meaning nodes can be added or withdrawn as needed. They are typically made up of clusters of common machines. There are currently dozens of BDMS-related open-source and for-profit products as a result of the swiftly growing field of BDMSs.Many of these systems have supported SQL as a query language, despite the fact that some of them have suggested their own query languages or application programme interfaces. Many more systems will transition to support SQL, according to some leading database researchers, as the use of SQL in DBMSs continues to grow [2]. In addition to typical RDBMSs, the authors suggest that the following BDMS types should also be covered while teaching SQL: • MapReduce. It is regarded as one of the primary frameworks for processing big data. It makes it possible to create highly distributed applications that work on scalable, fault-tolerant clusters. The most extensively used MapReduce implementation is Apache Hadoop [7].Examples of systems that support SQL to query data in Hadoop are: Hive [8] and Spark (Spark SQL) [9]. • NoSQL. These data stores have been designed to provide higher scalability and availability than conventional relational databases while supporting a simplified transaction and consistency model. Some examples of NoSQL data stores are: MongoDB [10], Apache HBase [11], Google's BigTable [12], and Apache Cassandra [13]. While many NoSQL systems do not support SQL natively, several systems have been proposed to enable SQL querying on these systems. Examples of such systems are: Impala [14], Presto [15] and SlamData [16]. • NewSQL. These systems aim to have the same levels of scalability and availability of NoSQL systems while maintaining the ACID properties (Atomicity, Consistency, Isolation and Durability), relational schema, and SQL query language of traditional relational databases. Examples of NewSQL systems are: VoltDB [17], MemSQL [18], NuoDB [19], and Clustrix [20].

III. SQL IN MAPREDUCE SYSTEMS

A popular programming framework for processing very big datasets is MapReduce. The most widely used implementation of it is Apache Hadoop. An extensive dataset is split up into discrete chunks by a MapReduce algorithm so that it can be processed in parallel across moving computer clusters. Map and Reduce are the two main processing phases, and each requires the framework user to supply the corresponding code. The following is the form of the map and reduce functions: (K1,V1) map to list (k2,v2) reduce: list(v2) + (k2, list) (k3,v3) The input dataset is often kept on a distributed file system underneath, and various portions of this information are analysed concurrently by several map jobs. Each map task analyses a single record or line of data at a time.Each time the map function is called, a key-value pair (k1,v1) is received, and a list of (k2,v2) pairs is produced. A reduction node receives each generated key-value pair delivered over the network (shuffle phase). The architecture ensures that every intermediate pair with the same key (k2) will be delivered to the same reduce node, where they will group together as a single unit. Each reduction call processes a group of items from the (k2,list(v2)) and produces a list of (k3,v3) pairs, which is the MapReduce job's final output. While powerful for creating highly distributed and scalable programmes, MapReduce is both complex and challenging to learn. In actuality, even straightforward data tasks like combining two datasets or figuring out the top K records demand moderately sophisticated MapReduce applications.This is true because MapReduce demands that users create programmes in procedural languages that require a thorough description of how a processing task should be carried out. Given this restriction, a number of systems have been proposed to: (1) make it possible to utilise SQL-like languages on top of MapReduce-based systems, such as Apache Hive [8] and Apache Pig [22], and (2) integrate SQL with MapReduce-based computations, such as Spark SQL [9].

3.1 Hive: SQL Queries on Hadoop

A framework called Apache Hive [8] facilitates the processing and analysis of data kept in Hadoop. Hive enables searching the data using HiveQL, a query language similar to SQL, and projecting structure onto this data. The transparent conversion of queries defined in HiveQL to MapReduce scripts is one of Hive's core advantages. As a result, the user is

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

free to concentrate on defining the data that the query should return rather than creating a Java procedural MapReduce application. Hive was created specifically to enable data warehouse applications, which call for the analysis of sizable read-only datasets. Hive uses indexing structures to speed up the execution of queries (data that does not change over time).Hive's primary query language is HiveQL, but it also supports the usage of customised map and reduce functions when doing so is more practical or effective for a particular query's logic. Although HiveQL broadly adheres to a SQL standard, it does not cover all of its features. Numerous DDL and DML commands, including CREATE TABLE, SELECT, INSERT, UPDATE, and DELETE, are supported by Hive. Additionally, starting with Hive 0.13, transactions with complete ACID semantics can be supported at the row (record) level. The following in-class exercise is our suggestion for introducing HiveQL. making use of virtual machines. A computer cluster is required to enable direct contact with these technologies because many Big Data systems rely on distributed architectures.However, such clusters are not often available for teaching at many institutions. The authors suggest using virtual machines (VM) that come pre-configured with all the necessary software packages as a solution. The authors advise using Cloudera's VM [23], which comes with CentOS Linux as the operating system, Hadoop, Hive, and (5) Hue, a web-based tool that can be used to create and execute HiveQL searches, in the case of Hive. Datasets and Data Generators. The queries in this section make use of MStation2, an artificial collection of meteorological station data that the authors created. Station (stationID, zipcode, latitude, and longitude) and WeatherReport are the two tables in the dataset (stationID, temperature, precipitation, humidity, year, month).You may find the data generator and a sample dataset in [1]. For new tasks or projects, the generator can be altered to create datasets with various sizes and data distributions. Data loading and querying. Fig. 1 lists the commands for this activity. We can launch the Hive console using command C1 to begin interacting with Hive by opening a terminal window in Cloudera's VM. Then, using the DDL commands C3 and C4, we can create the databases MStation (C2) and tables stationData and weatherReport, respectively. The data produced by the MStation2 data generator can then be loaded into both tables using commands C5 and C6. Next, you can use the Hue application or the Hive console to run SQL.We can create queries like (Q1-Q4 in Fig. 1) using the available tables. Calculate the average precipitation level for each zipcode. • Q2: Sort the data by station, and then produce the average relative humidity at each station. • Question 3: Identify the minimum and maximum temperatures each station has reported for years past 2000. Calculate the tenth-highest temperature that has ever been recorded. Include the temperature, zip code, year, and month. Figure 2 displays the outcomes of queries made using Hue and the Hive console.

```
C1: sudo hive

C2: CREATE database MStation;

C3: CREATE TABLE stationData (stationed int, zipcode int,
latitude double, longitude double, stationname string);

C4: CREATE TABLE weatherReport (stationid int, temp double,
humi double, precip double, year int, month string);

C5: LOAD DATA LOCAL INPATH '/home/cloudera/datastation/
stationData.txt' into table stationData;

C6: LOAD DATA LOCAL INPATH '/home/cloudera/datastation/
weatherReport.txt' into table weatherReport;

Q1: SELECT S.zipcode, AVG(W.precip) FROM stationData S JOIN
weatherReport W ON S.stationid = W.stationid GROUP BY
S.zipcode;

Q2: SELECT stationid, AVG(humi) FROM weatherReport GROUP BY
stationid;

Q3: SELECT stationid, MIN(temp), MAX(temp) FROM weatherReport
WHERE year > 2000 GROUP BY stationid;

Q4: SELECT S.zipcode, W.temp, W.month, W.year FROM
stationData S JOIN weatherReport W ON S.stationid =
W.stationid ORDER BY W.temp DESC LIMIT 10;
```

Figure 1. Hive Commands and Queries

3.2 Spark: Integrating SQL and MapReduce

A framework for highly distributed data processing is Apache Spark [9]. Spark operates by loading the data into a cluster's memory and querying it using a wider range of processing primitives than Hadoop's two-stage disk-based MapReduce technique (which also include operations similar to Map and Reduce). For some instances, Spark's performance has

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

reportedly been up to 100 times faster than Hadoop's [9]. Spark can operate with Hadoop Distributed File System (HDFS), OpenStack, Amazon S3, and other distributed storage systems. Resilient Distributed Datasets is a concept that is used by Spark (RDD). An RDD is an immutable group of objects spread among a number of physical nodes in a cluster, allowing for parallel processing of the data.After creating an RDD, the user can perform two different sorts of operations: transformations and actions. While actions provide outputs, transformations return new RDDs. One of the main tools offered by Spark is Spark SQL. The Spark module for structured data processing, known as Spark SQL, relies heavily on DataFrames, which are relational database tables' equivalents. Named columns are used to organise the data in a DataFrame. RDDs, Hive tables, or JSON files that are already in existence can all be used to construct DataFrames. Spark SQL is a potent tool for learning SQL while dealing with large datasets because it supports many of the characteristics of SQL.

```
C1: wget http://real-chart.finance.yahoo.com/table.csv?
s=AAPL&d=6&e=4&f=2015&g=d&a=11&b=12&c=1980&ignore=.csv

C2: mv table.csv?s=AAPL table.csv

C3: hadoop fs -put ./table.csv /data/

C4: spark-shell --master yarn-client --driver-memory 512m --
executor-memory 512m

C5: import org.apache.spark.sql._

C6: val base_data = sc.textFile("hdfs://sandbox.hortonworks.
com:8020/data/table.csv")

C7: val attributes = base_data.first

C8: val data = apple_stocks.filter(_(0) != attributes(0))

C9: case class AppleStockRecord(date: String, open: Float,
high: Float, low: Float, close: Float, volume: Integer,
adjClose: Float)

C10: val applestock = data.map(_.split(",")).map(row =>
AppleStockRecord(row(0), row(1).trim.toFloat,
row(2).trim.toFloat, row(3).trim.toFloat, row(4).trim.toFloat,
row(5).trim.toInt, row(6).trim.toFloat,
row(0).trim.substring(0,4).toInt)).toDF()

C11: applestock.registerTempTable("applestock")

C12: applestock.show

C13: applestock.count

C14: output.map(t => "Record: " + t.toString).collect().
foreach(println)

Q1: val output = sql("SELECT * FROM applestock WHERE close >=
open")

Q2: val output = sql("SELECT MAX(close-open) FROM applestock")

Q3: val output = sql("SELECT date, high FROM applestock ORDER
BY high DESC LIMIT 10")

Q4: val output = sql("SELECT year, AVG(Volume) FROM applestock
WHERE year > 1999 GROUP BY year")
```

Figure 3. Spark Commands and SQL Queries

The following section describes a Spark SQL in-class exercise. A great virtualization environment for giving we practical Spark SQL experience is Hortonworks Sandbox [24]. This virtual machine allows for the execution of commands from a terminal and has Apache Spark fully installed. The dataset to be utilised in this activity was taken from the Yahoo! Finance website [25] and includes the historical values of the Apple stock (AAPL, 1980–2015). The dataset is available as a comma-separated values (CSV) file for download. Each record comprises the following attributes: date, open price, high price (highest stock value), low price (lowest stock value), close price, volume, and adjClose. Each record provides the stock values for a specific date (close price adjusted for dividends and splits).In order to facilitate the derivation of some intriguing queries, a year-specific additional attribute (extracted from the date) is added. The commands used to load and query the data using the Hortonworks VM are shown in Fig. 3. The student initially downloads the dataset (C1) and changes its name to table. C2 opens the csv file, C3 places it in Hadoop's HDFS, and C4 launches the Spark shell (C4). After that, C5 is run to import a library required for using Spark SQL. The dataset located in the HDFS is used by command C6 to generate the RDD base data.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

With the exception of the record with the headers, C7 and C8 build a new RDD (data) containing all of the records in base data. The class AppleStockRecord, which defines the properties and data types of the dataset for Apple stock, is created by C9. The data records are parsed by C10, which then generates an instance of AppleStockRecord for each record and adds the new instances to the RDD applestock. Applestock is registered as a DataFrame by C11. Now, you may utilise this DataFrame as a relational table. The top 20 records and total number of records for applestock are displayed in C12 and C13, respectively. Now, teachers can assign we to create SQL queries like (Q1–Q4 in Fig. 3):

- Q1: Report the records where the close price is greater or equal than the open price.
- Q2: Identify the record with the largest difference between the close and the open prices.
- Q3: Report the ten records with the highest high prices.
- Q4: Report the average stock volume per year, considering only years greater than 1999. A key feature of Spark is that the output of SQL queries can be used as the input of MapReduce-like computations. This is, in fact, done in C14 to print the results of the SQL queries.

IV. SQL IN NOSQL SYSTEMS

NoSQL (Not Only SQL) is a broad category of data storage with the goal of offering classic relational databases more scalability and availability. The following fundamental characteristics define NoSQL systems: Data is divided up among various computers, data is replicated to provide fault tolerance, they typically use a distributed and fault-tolerant architecture that allows for the easy addition of more computers, and they frequently support a streamlined transaction/consistency model, such as eventual consistency (given a sufficiently long period of time in which no changes are made, all the previous updates are expected to eventually propagate through the system).There are several types of NoSQL, including: (1) key-value stores, like Cassandra and Riak, which use key-value pairs to store data in a schema-less manner; (2) document stores, like MongoDB and CouchDB, which store documents in a specific format (XML, JSON, binary, etc.); and (3) tabular data stores, like HBase and BigTable, which store tabular data with multiple attributes and records. One of the most significant recent developments in this field has been the realisation that a declarative, highly expressive, and standardised query language like SQL can be a useful way to query NoSQL systems, despite the fact that the majority of NoSQL systems initially did not adhere to the relational database model and did not support SQL for data manipulation [2].Numerous systems that allow SQL queries on popular NoSQL data stores, such as Impala [14], SlamData [16], and Presto [15], reflect this. For example, these systems support SQL queries on HBase, MongoDB, and Cassandra, respectively.

4.1 Impala: SQL to Query HBase Tables

An open-source query engine called Impala [14] enables the use of SQL queries on top of HBase tables (Impala also supports HDFS as storage sub-system). Large volumes of sparse tabular data can be stored and processed fault-tolerantly using the open-source NoSQL database HBase [11], which runs on top of HDFS. On top of HDFS, HBase offers effective random reads and writes (HDFS does not directly support random writes). Impala was created by utilising and extending essential Hive elements including the SQL syntax, metadata, and schema.Despite the fact that both systems accept SQL, Impala offers superior support for dynamic analytic queries while Hive is better suited for long-running batch processing. The following class exercise is our suggestion for introducing SQL in Impala. We can utilise the Cloudera VM [14] for this assignment, which already has Impala, HBase, and Hive installed. In Fig. 4, the instructions list is displayed. The dataset used for this activity includes statistics on American vocations. The dataset (sample 007.csv) is located in the VM's dataset folder.Code (occupation ID), description, total emp (number of employees), and salary are its four attributes (combined income). The data file was renamed to occupations.csv. We should launch the HBase shell in the VM's terminal window first (C1). Next, command C2 is used to create the HBase table "occupationsData."

We should now move to Hive (C3 and C4), create the Occupations table (an external table), link it to OccupationsData (C5), and import the data (C6). We can then launch the Impala shell (C7) to begin writing and running SQL queries. The table can be queried in Impala after it has been built in Hive because Impala and Hive use the same metadata database.Impala needs to be informed about the newly created table using command C8. Now, we can create and run queries like (Q1–Q4 in Fig. 4): Q1: Display the ten jobs with the greatest total salary. Q2: List the ten professions with

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

the highest earnings per employee. Q3: Display the average annual salary for each position in the computer industry. Q4: Display the profession with the highest total annual salary.

```
C1: hbase shell

C2: create 'OccupationsData', 'code', 'description',
'total_emp', 'salary'

C3: exit

C4: hive

C5: CREATE EXTERNAL TABLE Occupations (key STRING, description
STRING, total_emp int, salary int) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED BY 'org.apache.hadoop.hive.hbase.HBa
seStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mappin
g" = ":key, description:description,total_emp:total_emp,salary
:salary") TBLPROPERTIES("hbase.table.name"="OccupationsData");

C6: hbase.org.apache.hadoop.hbase.mapreduce.ImportTsv '-
Dimporttsv.separator=,' -Dimporttsv.columns=HBASE_ROW_KEY,
details:code,details:description,details:total_emp,details:sal
ary OccupationsData /home/cloudera/occupations.csv;

C7: impala-shell

C8: invalidate metadata [[default.]Occupations]

Q1: SELECT description, salary FROM Occupations ORDER BY
salary DESC LIMIT 10;

Q2: SELECT description, salary/total_emp AS PerPersonSalary
FROM Occupations ORDER BY PerPersonSalary DESC LIMIT 10;

Q3: SELECT SUM(salary)/SUM(total_emp) AS AvgSalary FROM
occupationsIncomeHBase WHERE description LIKE '%computer%';

Q4: SELECT code, description, salary FROM Occupations WHERE
Salary IN (SELECT MAX(salary) FROM Occupations);
```

Figure 4. Impala Commands and SQL Queries

4.2 SlamData: SQL on MongoDB

SlamData [16] is a platform that enables SQL commands to be executed on MongoDB [10], a widely used document-oriented open-source NoSQL database. MongoDB uses documents as its primary form of data storage and does not natively support SQL or multi-object transactions. Documents are analogous to rows in relational databases and have a structure similar to JSON (BSON). A collection is made up of several documents (table). Each document is made up of pairs of fields and values that resemble JSON objects. Other documents, arrays, and arrays of documents are all possible values for fields.Joins are no longer necessary thanks to these embedded documents and arrays. To introduce the use of SQL in the context of MongoDB and SlamData, we suggest the following in-class activity. Figure 5 lists the commands for this experiment using Microsoft Windows. The collection consists of the IRS's 2012 U.S. Income Tax Statistics by ZIP Code (CSV file) [26]. We apply the following qualities: STATE, ZIPCODE, N1 (number of returns, which approximately represents the number of households), N2 (number of exemptions, which approximately represents the population), A00100 (AGI), and A00200 are the variables. 99999 is the sum of all ZIP codes with fewer than 100 returns, while 0 represents the entire state (salaries and wages, in thousands of dollars).The initial step is to download and set up MongoDB [10]. (we installed it in C:\mongodb). C1 creates a necessary data directory. C2 starts MongoDB and establishes a connection with it. In C3, an exe shell is established. The databases that are accessible can be listed using C4 (initially, there should be one called local). SlamData needs to be downloaded [16] and installed at this stage. We must mount a MongoDB cluster in SlamData and host it elsewhere in order to execute queries. We'll use MongoLab's free tier to accomplish this [27]. We should establish a new MongoDB deployment (choose single-node and standard line) and enter the database name after creating a MongoLab account (project).We can set the database user name and password by clicking on the new deployment after it has been created. The CSV file will then be imported into the MongoDB deployment by the we. One of the pre-filled commands offered by MongoLab under the Tools/Import-Export tab can be used to accomplish this. This tab's CSV section offers a command formatted as C5. This command must be executed by we in a Microsoft Command Prompt window. The MongoDB database will then be mounted in the GUI-based SlamData programme by the we.We must use a connection URI (formatted as C6) that they can find in the recently

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

built MongoDB deployment in MongoLab and click on the Mount icon in SlamData to do this. The SlamData application now allows we to create SQL queries like (Q1–Q4 in Fig. 5): • Question 1: How many households are there in your ZIP code? • Question 2: Calculate the population of each New York ZIP code. • Question 3: List the ten zip codes in New York with the highest total gross revenue. Calculate the average wage and salary amount for each state in question 4.

V. SQL IN NEWSQL SYSTEMS

NewSQL is a class of database systems that not only offers the same levels of scalability and availability of NoSQL systems but also preserves the ACID guarantees, relational data model, and SQL query language of traditional relational databases. NewSQL systems are divided into three groups: (1) New architectures, e.g., VoltDB and NuoDB, these are new systems designed to work on a distributed cluster of shared-nothing nodes that can dynamically scale; (2) Optimized SQL Engines, e.g., MySQL Cluster and Infobright, these systems support the same programming interface of traditional systems like MySQL but have better scalability; and (3) Transparent sharding, e.g., dbShards and ScaleBase, these systems provide a middleware layer to automatically fragment and distribute databases over multiple nodes. Most NewSQL databases can be used to learn SQL considering the application scenarios commonly used to teach traditional RDBMs, e.g., university, employee, or inventory databases. A better approach, however, is to consider application scenarios that require the capabilities of NewSQL systems, e.g., the need of processing large or fast data in the stock market, social media networks and online games. Moreover, educators should consider applications that take advantage of the features of the NewSQL system used in class, e.g., an application that requires processing a high number of short transactions per second would be a good fit for VoltDB.

5.1 Learning SQL with VoltDB

A shared-nothing architecture, partitioning, and replication techniques are used by VoltDB [17], an in-memory and ACID-compliant NewSQL database system, to achieve high transaction throughputs. The technology was created by renowned database researchers and is open-source. VoltDB seeks to provide answers to real-time analytical issues by gleaning knowledge from rapidly flowing data.VoltDB offers client libraries for Java, Python, PHP, C++, C#, and other programming languages and may be installed on Linux and Mac OS X machines. Several topics of SQL can be learned with VoltDB through practical class exercises. It can be used, for instance, to learn SQL's data definition and manipulation languages. Another efficient method for enabling direct contact with NewSQL platforms like VoltDB is by using a virtual machine. In this situation, educators can make use of the

```
C1: md \data\db

C2: C:\mongodb\bin\mongod.exe

C3: C:\mongodb\bin\mongo.exe

C4: show dbs

C5: C:\mongodb\bin\mongoimport.exe -h <host:port> -d <your
database name> -c <collection> -u <dbuser> -p <dbpassword> --
type csv --file <path to .csv file> -headerline

C6: mongodb://<dbuser>:<dbpassword>@<host:port>/<dbname>

Q1: SELECT N1 FROM "/project/taxes" WHERE ZIPCODE=85304

Q2: SELECT N2, ZIPCODE FROM "/project/taxes" WHERE STATE='NY'

Q3: SELECT ZIPCODE, A00100 FROM "/project/taxes" WHERE
STATE='NY' AND ZIPCODE!=0 ORDER BY (A00100) DESC LIMIT 10

Q4: SELECT DISTINCT STATE, AVG(A00200) FROM "/project/taxes"
WHERE ZIPCODE!=0 AND ZIPCODE!=99999 GROUP BY STATE
```

Figure 5. SlamData Commands and SQL Queries

VM on the VoltDB website [17]. The virtual machine (VM) has VoltDB loaded, along with a selection of sample apps and tools for app development.The sample apps are helpful learning tools for understanding VoltDB. They are: (1) a system that mimics a phone-based voting process; (2) a memcache-like implementation using VoltDB; (3) a Key-Value store supported by VoltDB; and (4) a system that makes use of a flexible schema and JSON. A portfolio of programmes that can be utilised to create educational activities was also made available by VoltDB [21]. We suggest using the Ad

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

Performance programme, which can be downloaded at [21] and set up in VoltDB's virtual machine.This programme replicates a fast-moving stream of advertising events (impressions, clickthroughs, and conversions), which are enhanced and saved in real time. The following in-class exercise using this application can be used by teachers. The provided scripts that launch the database server, build tables and views, and launch a client application that generates the stream of ad events are first examined and executed by the we. The statement (C1) in Fig. 6 establishes the table event data, which houses all of the events that have been received and processed.We use the offered web-based interface, depicted in Fig. 7, to monitor the stream of events and the costs and rates related to advertising campaigns while the client programme is running. The next step is for we to write and execute SQL queries using VoltDB's Management Center. Fig. 6 displays a number of queries the authors created to compute different real-time data.

- Q1: For each website that shows ads, compute the number of received ad events, impressions, clicks and conversions, and the total ad cost.
- Q2: For each advertiser, get the number of processed ad events, impressions, clicks and conversions, and the total ad cost.
- Q3: For each advertiser, compute the daily number of ad events, impressions, clicks and conversions, and the daily ad cost.
- Q4: Identify the 10 ads with the largest associated cost. This learning activity also provides a good opportunity to highlight some of the key properties of this type of NewSQL system, e.g., high transaction throughput (transactions/second) and low latency (time to process the events).

```
C1: CREATE TABLE event_data (utc_time TIMESTAMP NOT NULL,
creative_id INTEGER NOT NULL, cost DECIMAL, campaign_id
INTEGER NOT NULL, advertiser_id INTEGER NOT NULL, site_id
INTEGER NOT NULL, is_impression INTEGER NOT NULL,
is_clickthrough INTEGER NOT NULL, is_conversion INTEGER
NOT NULL);

Q1: SELECT site_id,  COUNT(*) AS records, SUM(is_impression)
AS impressions, SUM(is_clickthrough) AS clicks,
SUM(is_conversion) AS conversions, SUM(cost) AS cost FROM
event_data GROUP BY site_id;

Q2: SELECT advertiser_id, COUNT(*) AS records,
SUM(is_impression) AS impressions, SUM(is_clickthrough) AS
clicks, SUM(is_conversion) AS conversions, SUM(cost) AS cost
FROM event_data GROUP BY advertiser_id;

Q3: SELECT advertiser_id, TRUNCATE(DAY,utc_time) AS utc_day,
COUNT(*) AS records, SUM(is_impression) AS impressions,
SUM(is_clickthrough) AS clicks,  SUM(is_conversion) AS
conversions, SUM(cost) AS spent FROM event_data GROUP BY
advertiser_id, TRUNCATE(DAY,utc_time);

Q4: SELECT creative_id AS ad, SUM(cost) AS cost FROM
event_data GROUP BY creative_id ORDER BY cost DESC LIMIT 10;
```

Figure 6. VoltDB SQL Queries

VI. DISCUSSION

Numerous beginning and advanced database-related courses can incorporate the suggested modules. One strategy is to switch from the conventional relational databases now utilised in the majority of basic database courses to Big Data management systems. We advise utilising a NewSQL system for this method because it supports both SQL and the relational paradigm, such as VoltDB. Combining the use of a regular database with a Big Data system is an alternative strategy for basic database courses. In this situation, instructors can demonstrate how the same or related concepts can be applied to contemporary Big Data systems while teaching database fundamentals using conventional databases. Another strategy is to include the learning of SQL in a course on big data that goes beyond conventional databases. The use of SQL in MapReduce, NoSQL, and NewSQL systems can be studied in this course along with how it compares to other native query languages. The authors are currently integrating the study of SQL in a Big Data system as part of an undergraduate database course. They previously integrated the study of NewSQL systems in a Big Data course. In a subsequent paper, we want to report the findings from these encounters.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, July 2022
Impact Factor 6.252

VII. CONCLUSION

Different kinds of Big Data systems have been developed to meet the issue of processing very big datasets in a highly scalable and distributed manner, which is required for many application situations.Many of these systems are aware of SQL's advantages as a query language. However, the majority of database courses only cover conventional relational databases. In this essay, we suggest that the newer, more expansive database landscape be taken into account when teaching SQL. Through this experience, we will be able to better understand the data on a larger range of systems and applications. In order to connect the study of SQL with the three primary categories of Big Data systems—MapReduce, NoSQL, and NewSQL—this article offers a set of guidelines and a wide range of class resources.

REFERENCES

[1]. ASU. SQL: From Traditional Databases to Big Data - course resources. http://www.public.asu.edu/~ynsilva/iBigData.
[2]. Barron's. Michael Stonebraker Explains. http://blogs.barrons. com/techtraderdaily/2015/03/30/michael-stonebraker-describ es-oracles-obsolescence-facebooks-enormous-challenge/.
[3]. D. Kumar. Data science overtakes computer science? ACM Inroads, 3(3):18–19, Sept. 2012.
[4]. A. Cron, H. L. Nguyen, and A. Parameswaran. Big data. XRDS, 19(1):7–8, Sept. 2012.
[5]. A. Sattar, T. Lorenzen, and K. Nallamaddi. Incorporating nosql into a database course. ACM Inroads, 4(2):50–53, June 2013.
[6]. Y. N. Silva, S. W. Dietrich, J. Reed, L. Tsosie. Integrating Big Data into the Computing Curricula. In ICDE 2015.
[7]. Apache. Hadoop. http://hadoop.apache.org/.
[8]. Apache. Hive. https://hive.apache.org/.
[9]. Apache. Spark SQL. https://spark.apache.org/sql/.
[10]. 10gen. Mongodb. http://www.mongodb.org/.
[11]. Apache. Hbase. http://hbase.apache.org/.
[12]. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Trans. Comput. Syst., 26(2):1–26, 2008.
[13]. Apache. Cassandra. http://cassandra.apache.org/.
[14]. Cloudera. Impala. http://impala.io/.
[15]. Presto. https://prestodb.io/.
[16]. SlamData. SlamData. http://slamdata.com/.
[17]. VoltDB. Voltdb VM. https://voltdb.com/run-voltdb-vmware.
[18]. MemSQL. MemSQL. http://www.memsql.com/.
[19]. NuoDB. Nuodb. http://www.nuodb.com/.
[20]. Clustrix. Clustrix. http://www.clustrix.com/.
[21]. VoltDB. Application gallery. http://voltdb.com/community/applications.
[22]. Apache. Pig. https://pig.apache.org/.
[23]. Cloudera. Cloudera VM 4.7.0. http://www.cloudera.com/content/cloudera/en/downloads.html.
[24]. Hortonworks. Hortonworks Sandbox on a VM, HDP 2.3. http://hortonworks.com/products/hortonworks-sandbox.
[25]. Yahoo Finance. Historical Prices -AAPL.http://finance.yahoo.com/q/hp?s=AAPL+Historical+Prices.
[26]. IRS. Tax Stats. http://www.irs.gov/uac/SOI-Tax-Stats-Indivi dual-Income-Tax-Statistics-2012-ZIP-Code-Data-(SOI). [27] Mongolab. Mongolab. https://mongolab.com/.