Review Article

International Journal of Advanced Computer Research, Vol 11(55)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2021.1152017

GU_DB : a database management system prototype for academia

Bhumika Shah* **and Jyoti Pareek**
Department of Computer Science, Gujarat University, Ahmedabad, India

Received: 15-May-2021; Revised: 15-July-2021; Accepted: 17-July-2021

Abstract
There is hardly any open source database system having Academic or Research considerations. Moreover, the open-source databases do not provide developers, ease of implementing and testing their research work. We could not get full flexibility to analyze and implement our research proposal in any of the database system we approached. All the database systems approached, required the proposed work to be tested somewhere, before it could be implemented. This motivated us to develop our own Database management system that provides complete flexibility over the system, starting from the phase of the compilation till execution. GU_DB integrates our own Lexical and Syntax analysis engine which help in fetching the exact errors/messages for the query passed by the user. The ability of the database to plugin easily helps smooth connectivity with any front-end. The multi-threading nature of the database helps to seamlessly connect multiple web users. Our development has given wings to the new Database Management System Prototype named GU_DB, which is equipped with the basic features of a DBMS and has all academic and research considerations. The GU_DB is now built into a complete learning management system (LMS) in form of Virtual laboratory and is approved by Virtual Labs (a project under MHRD NMEICT) and the content is hosted at the Vlabs-Dev portal by IIT-Bombay. The Virtual laboratory hosted exist as a front-end and as a back-end of the same system, GU_DB exists in form of a database system.

Keywords
Database management system (DBMS), Open-source database systems, Learning management system (LMS), DBMS prototype, Academic DBMS, Custom DBMS development, Virtual laboratory.

1.Introduction
Database Management System (DBMS) is the basic course in any undergraduate / postgraduate courses for Computer science/computer engineering degrees. All the computer science courses demand practical hands-on in the subject, which is performed in laboratory hours. Database management systems are mainly classified as divided into two categories Commercial Database Systems and Open Source Database systems. Proprietary database systems come with a huge license cost, and hence nowadays almost all the educational/research bodies are moving towards Open Source Database systems. However, all these open source database systems are designed with enterprise or business considerations. There is hardly any open source database system which has Academic or Research considerations. Peters and Sikorski [1] mention in their article that the researchers who wish to test new concepts are forced to build the entire system from scratch.

*Author for correspondence

And as (Pavlo, n.d.) mentions that building a DBMS for academia is hard and finding resources for the same is even harder.

The paper focuses on to the issue of populating databases which are reliable for hidden queries, and hence presents the hydra generator to showcase metadata statistics and optimizer estimates [1].

Moreover, the open-source databases do not provide the developers the ease of implementing and testing their research work. We could not get full flexibility to analyze and implement our research proposal in any of the database system we approached. This issues like lack of flexibility and no academic considerations motivated authors to develop our own Database management system, which can provide us complete flexibility over the system, starting from the phase of the compilation till execution.

Our development has given wings to the new Database Management System prototype named GU_DB, which is equipped with all the basic features of

Database management system and has all academic and research considerations. The objective of the proposed system is to build our own database prototype which can give us full flexibility over the back end and helps us to make database learning easier for students by integrating it with the LMS (Learning Management System). GU_DB has been completely integrated with Virtual laboratory for Databases which is hosted at IIT-Bombay and is used by student community world-wide.

2.Literature review

The database system is one such area where there are varied research interests. There has been a lot of research in areas like Query Optimization, Large Databases, Big Data, Multi-Lingual Database systems, Data mining, to name a few. However, new Database development platforms dedicated for academia is one such research category, which has been almost untouched.

Peters and Sikorski [1] mention in their article that the researchers who wish to test new concepts are forced to build the entire system from scratch. Moreover, anybody who wants to develop the database prototype has a tough time finding resources. COBRA: The program transformation technique is used in developing a framework which can generate different alternatives of a program and selecting the least cost for optimizing relational algebra expressions is proposed for real life applications. The paper focuses on cost based model for optimizing programs with regions [2].

This research focuses on novel research approaches for robust query processing and characterizes their strength and limitations to describe the open challenges for query processing problems and discusses about various robust query processing techniques considering the novel areas in databases like big data and some machine learning approaches namely query based and data-based implementations [3].

There hardly exists any research for the development of database management systems for academia. As (Pavlo, n.d.) points out that building DBMS is hard, but building DBMS for Academia is even harder.

This was one of the core motivation for the authors to develop a database system dedicated for Academia, which can help researchers to explore many other possibilities and areas of research in the area of database systems in academia. Variety of Database systems are accessible for the student/learner community to implement the database knowledge they have acquired, but there exists hardly any complete guided Learning Management System.

Limitations of existing DBMSs:

- Most of the existing DBMS have commercial purposes none of them are solely for academia
- Most of the DBMS provide a syntax error on an incorrect SQL Statement, which is usually not self-explanatory and student struggles to interpret the error
- None of the DBMS mentioned have LMS (Learning Management System) considerations

There hardly exists any research for the development of database management systems for academia. As (Pavlo, n.d.) points out that building DBMS is hard, but building DBMS for Academia is even harder [4].

2.1Need for database prototype for academia

This was one of the core motivation for the authors to develop a database system dedicated for Academia, which can help researchers to explore many other possibilities and areas of research in the area of database systems for academia and help students in understanding the concepts thoroughly by integrating the concepts of the subject in the system.

2.2Commercial databases and open source databases

The Database management systems are divided into two broad categories viz.

For Commercial Database systems, licensing fees are to be paid to get the stable version and hence most of the institutions have moved towards Open-Source Database systems [5]. However, all the mentioned database systems have a commercial purpose and none of them is dedicated for learning or for the academic purposes [6].

Apart from commercial considerations, Open source databases will restrict the user in the following ways:

- Researchers are not able to propose their idea, without proper testing and implementation. However to propose your idea, you need the same environment, which is not available.
- Open source is not completely open source , there are many considerations to be applied before you actually try to reuse the code
- The error messages in any of these open source systems are not self-explanatory.

The students struggle to understand the message, hence end up spending more time on searching for the error, than solving the error. This generates the need for Database management system which is dedicated for learning and hence a DBMS for Academia is the need of the current learning environment (*Table 1*).

Table 1 Commercial databases Vs open source databases

Open source	Commercial
Available without paying fees in open Licensing scheme	Huge Licensing fees are to be paid
Need to rely on community support	Dedicated support by the vendor is provided
Lesser features in comparison with proprietary database	Feature Rich Database
Less support for advanced features	Support for information replication, Backup and retrieval of data
Dependability based on community support	Dependability and Stability for all systems
Simple implementation	Domain Expertise required
Abolishes single point failure	This systems have dedicated support system to hold responsible for any kinds of bugs
Open source Communities can propose new features	For the new features, you need to depend on next release from the vendor.

2.3Motivation

The above reasons and many other, which are discussed in the later sections of chapter, motivated the authors to come up with the prototype of Database Management Systems, known as GU_DB.

GU_DB is the prototype created and implemented by the authors to incorporate features required for guided learning. The Database management system prototype (GU_DB) integrates basic features of any open source database system. Moreover, it also serves as the guided learning system for the students which is incorporated in the Virtual laboratory.
The virtual laboratory for Database Systems has support for following SQL Queries

- Data Definition Language Commands (DDL-Create, Alter, Drop)
- Data Manipulation Language Commands (DML-Insert, Update, Delete)
- Data Query Language Commands (DQL-Select)
- Transaction Control Language Commands (TCL-Commit, Rollback)

As the Database system developed by authors is dedicated for Academia it has many distinguishing **features** dedicated for **learning** like Hints, pitfalls etc. Moreover, the capability of GU_DB to easily plug-in enables easy connectivity with the front-end system.

2.4Problems with existing database systems with respect to Academia

- Commercial considerations
- Lack of guided support to students with reference to learning
- No flexibility over the database system, user needs to rely on error messages provided by the third party database (e.g: MySQL, PostgreSQL, etc.)
- No Research scope by academicians

2.5Objective

The reasons like LMS considerations, Academia support, customized messages in form of hints and the development of learner centric system motivated the authors to come up with the prototype of the Database Management System, which is named as GU_DB.

The proposed DATABASE SYSTEM PROTOTYPE: GU_DB has academic considerations which help the student to understand and interpret the error message in a self-explanatory manner. The error messages are provided in form of hints which help the student to try on his own and reach the goal which leads the student towards a higher cognitive level considering all the aspects of a complete LMS (Learning Management System are considered.

2.5.1Features of GU_DB

Authors have developed their own database management system (GU_DB) to integrate various features needed for guided learning and to have complete control and flexibility over the system.

GU_DB has support for following SQL Commands

- Data Definition Language Commands (DDL-Create, Alter, Drop)
- Data Manipulation Language Commands (DML-Insert, Update, Delete)
- Data Query Language Commands (DQL-Select)
- Transaction Control Language Commands (TCL-Commit, Rollback) & Describe Statement

2.5.2Advantages of the proposed system: GU_DB

The proposed DBMS (GU_DB) has academic considerations which help the student to understand and interpret the error message in a self-explanatory

manner. The error messages are provided in form of hints which help the student to try on his own and reach the goal which leads the student towards a higher cognitive level considering all the aspects of a complete LMS (Learning Management System.

The proposed system helps in achieving the following objectives

- Complete flexibility over the database system. As the back-end comprises of our own database prototype(GU_DB) we have full control over the system
- Complete guided support for students, which helps students in learning as they get guided message in terms of Hints/pitfalls
- It has been observed that there is very little/no research available with reference to new Database system development. Hence, authors plan to share the database system as open access to academicians to increase the research scope in database community.

3.Methods

A Database in simplest terms is the collection of Data. However, the simpler it looks, the more complex it is to implement. For any Database Management System to be implemented, following are the core functionalities required.

- Pre-Processor
- Parser
- Query Executor
- Disk Management/Memory
- Hash Table
- Transactions
- Index Handling

As the database system prototype (GU_DB) created by authors is dedicated for Academia, it has variety of eminent features dedicated for learning like Hints, pitfalls etc. At the highest level of abstraction, the database systems are viewed as Garlan and Shaw layered architecture consisting of three broad components.

3.1Components of Garlan and Shaw layered architecture

3.1.1Application layer

The clients interact with the database in the Application layer which communicates with the users directly (*Figure 1*). The users could be the people managing the system (Admins) or the end-users of the system. The end-users of the system access the database through some front end editor. The front-end editor of our system is "Query Editor." Query Editor helps the users directly communicate to the database engine "GU_DB." Due to the pluggable nature of the database, there can be one more category of users: "Clients," who access the system through APIs.

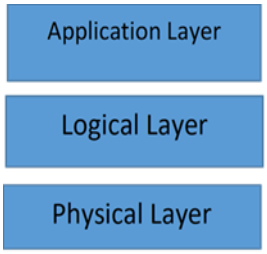

Figure 1 Garlan & Shaw architecture

3.1.2 Logical layer

The logical layer will take the data from the application layer and will process it. The task of processing the query, storage, and management of data, are the responsibility of the Logical layer.

3.1.3Logical architecture

In the logical architecture, the first component is the Thread Handling. Thread handling refers to managing the multi-user thread. Especially in the web-based environment, Thread handling has a considerable impact in the efficiency of the system. A new thread needs to be allocated for each client. Moreover, the combined threads are nothing but a process, so each client is allocated a process. Hence when the client fires a query, it technically initiates an individual thread and then goes to the Query Processor. The detailed discussion of all the components of logical architecture (*Figure 2*) is discussed in the later sections.

3.1.4Physical layer

In the database architecture, the physical layer is lowest layer. It has to take care of how the data will be stored on the disk. It manages the storage engine. The storage engine consists of the file system, different types of files, the directories created for the schema.

The physical layer consists of different type of files like:

- Data Files: Files related to users
- Data Dictionary: It stores the metadata and other structural information
- Index information: It stores the information about the indexes for each table
- Log information: keeps track of executed queries and default table space details.

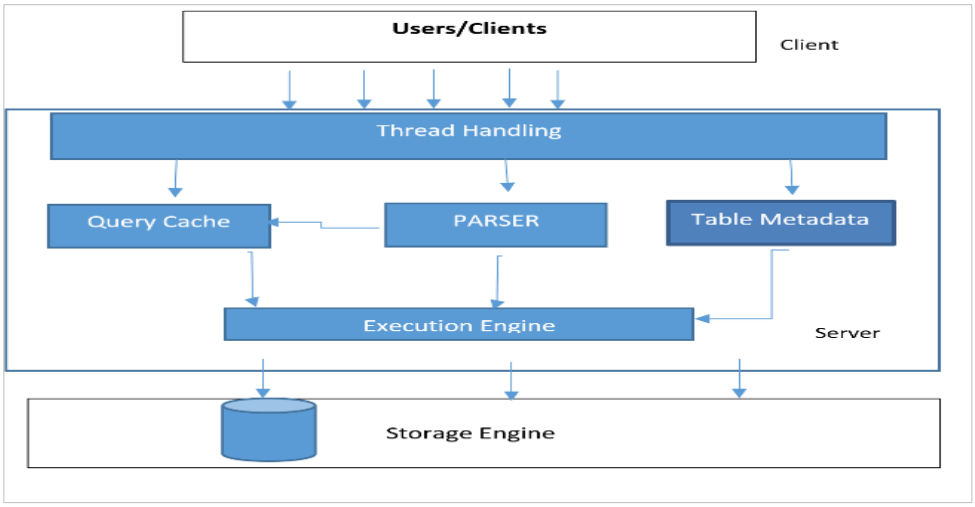

Figure 2 Logical architecture

Different database systems maintain different types of storage engines. The storage engines are the blueprint of how the data is stored. The storage engines could be Transactional or Non-Transactional. They differ on how locks are handled, what is the thread pool etc. Our System is equipped with the Transactional storage engine which manages all the memory requirements (including Thread pool considerations) of the Database.

Hence, the overall architecture of GU_DB is as follows (*Figure 3*):

3.2 Architecture of GU_DB

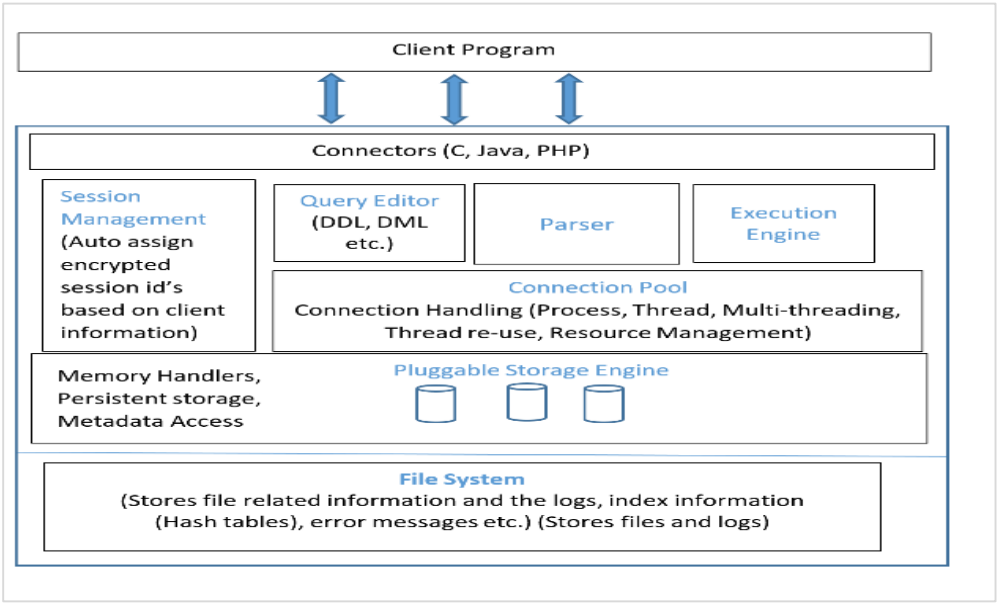

Figure 3 Architecture of GU_DB

3.2.1 Query processing

The query given by the user through query editor is parsed to check the syntactic correctness of the query.

Lexical analysis:

The purpose of lexical analysis is to extract the individual words from an input stream and in return provide the tokens to the parser. The tokenization processes take input and pass this input for the classification of keywords, identifiers, number, string, etc. The goal of the lexical analyser is to classify the input in tokens and help to identify them. The lexical analyser gives the token to Syntax analyser for

validating the syntactic correctness of the query. Lexical Analysis is implemented using Automata. The sequence of tokens is returned as output to the parser for syntax analysis. The DFA for lexical analysis is as follows (*Figure 4*):

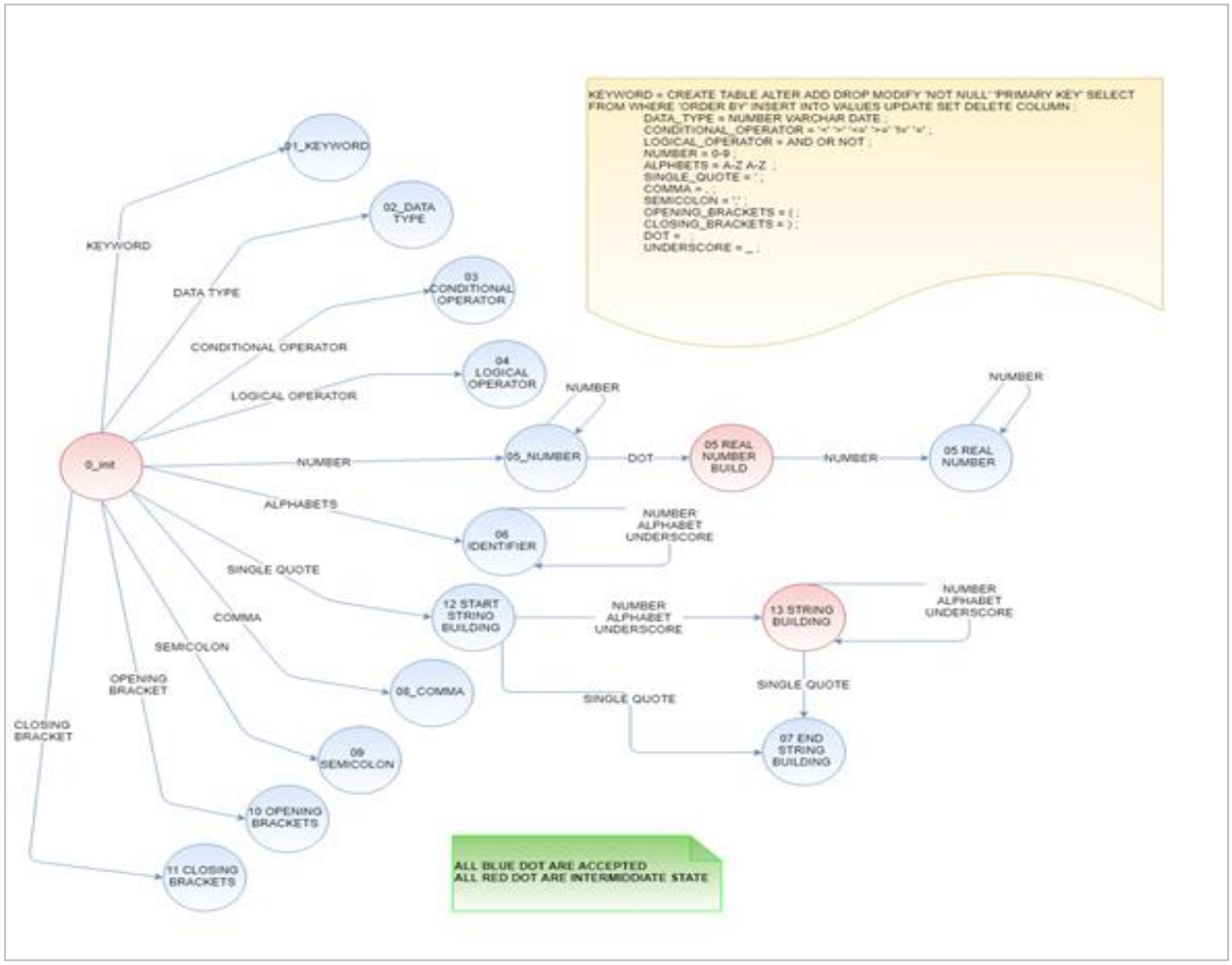

Figure 4 Finite automata for lexical analysis

Process of lexical analysis

Syntax analysis:

Syntax Analyser takes input from Lexical Analyser. Syntax Analysis (Parsing) is used to determine whether the tokens provided are valid for the given system. The parser analyses the token stream against the rules defined to detect any errors in the input string. However, all the valid tokens might not always be meaningful and appropriate as per the rules defined. Hence, Semantic analysis is applied after syntax analysis.

The goal of syntax analyser is:

- Check that the input string is well-formed
- Helps to detect all type of Syntax errors
- Give the exact position where the error occurred

Hence, to check syntactic correctness of queries, a state diagram has been designed (*Figure 5*). The goal of the parser is to find all syntax errors: for each error, produce an appropriate message and recover quickly. To analyse the syntax, we have used finite automata. Deterministic Finite Automata (DFA) has a strong mathematical background and there are already some real-world systems which used DFA's [7]. DFA's are used for text parsing, protocol analysis, natural language processing, speech recognition, and video games. Following diagram depicts the Finite automata for Create table statement, which is part of DDL statements.

The DFA here represents various states which are defined for a Create Table statement. The initial state

starts at 0, and all the accepted states like input, keywords, datatypes; identifiers are defined. The final state is 25, which depicts that the valid end of statement is ";" The states of the machine correspond to the user input, which changes according to various events. Hence, the complete "Create Table" statement has 25 valid states. It processes the words in the ordered state (Create, table, identifier (Tablename), bracket, and so on.). It also has a resemblance with the Mealy machine state which processes the output based on the input being processed.

Though state machines are not the most sophisticated means of processing SQL Queries, but due to the simplicity in nature and non-Turing compliance of SQL queries, it is easy to depict the state-based behaviours effectively using state machines (*Figure 6*).

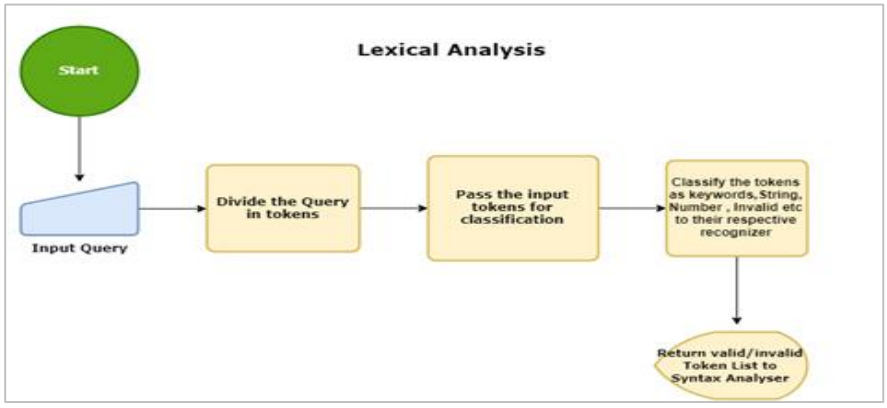

Figure 5 Process of lexical analysis

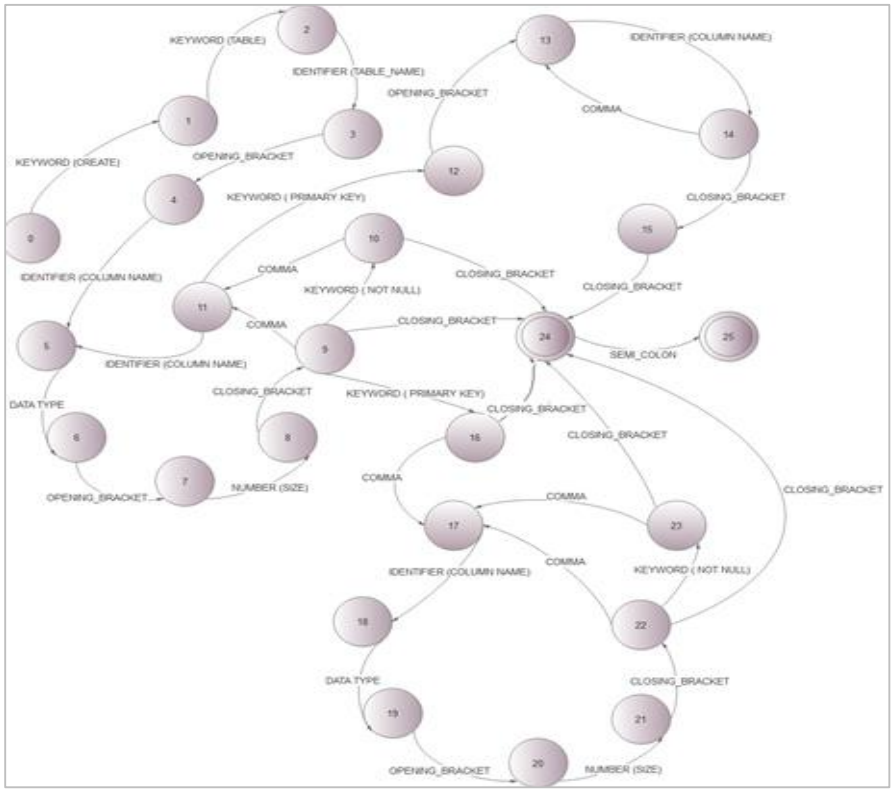

Figure 6 State diagram of create

3.2.2 Pluggable storage engine

With the advent of Web-based systems, it has become imperative to provide Inter-process and Inter-application communication. The Database engine also needs to communicate with the front-end systems (GUI) which directly communicate with the clients and then process the data on the server. To provide such a facility, there is a need for the database engine to be pluggable.

We had explored various approaches to make the Database engine pluggable and establish intercommunication of PHP and Java:

Various approaches explored for development of Pluggable Database engine are as follows

- Build Native Library
- Write our own Library and compile the Source code
- Use Some Inter-communication tools like JSon or any other.

Hence, we developed a C++ utility (which uses all the three approaches) to establish a connection between the Client and the server.

The following diagram depicts the approach followed by us for development of the Pluggable Database engine (*Figure 7*).

Pluggable DB:

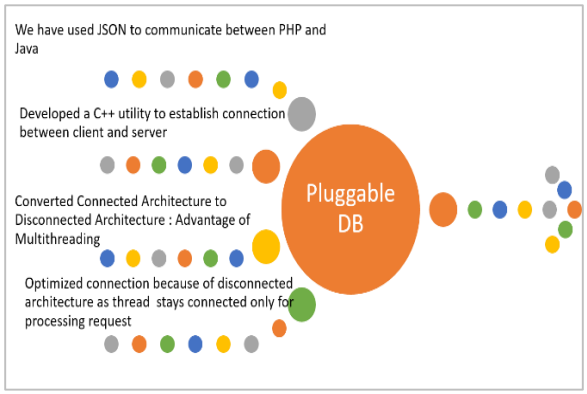

Figure 7 Pluggable database diagram

3.2.3 How GU_DB executes queries

The user interacts with the system with the help of the GUI Simulator: "Query Editor". The most essential and important part of the entire system is Query Editor. It enables the communication between the front-end system (DB Vlab) and the DBMS prototype (GU_DB). The query is entered in the editor by the user and it is parsed to the database engine by the Query processing engine. The syntactical correctness of the query is checked and the output in tabular form is generated else hint is provided to the user which guides the user on the exact mistake made by them. The advantage of being able to provide the flexibility in hints or the message is core advantage of GU_DB which plays a crucial role in enhancing the pedagogical perspective in student learning. All other commercial systems need to depend on message provided by the database whether it is MySQL or any other commercial or open source database [8].

In order to communicate with the back-end, Query editor needs to maintain the Session.

3.2.4 Session management of front-end with back-end (GU_DB)

Web-based and implicit login system:

The Database which resides on the back-end is accessed without any specific user-id. The authentication details are managed implicitly. Hence, to maintain the session with the Database, the front end takes care of managing the session. However, the back-end also needs to maintain the User log for the schema.

Each user is associated with unique id in the form of login, and that is to be referenced every time a user tries to access the database. The schema is to be attached with the user-id created. Our system handles such types of requirements implicitly.

In any database system, no login means no customized user experience or no user-specific schema. However, that means, without login, there is only a single default user that is accessed by all the users in the system. This gives rise to the Transaction management issues. Even in the multi-user environment, it is imperative to give full support for ACID (Atomicity, Consistency, Isolation, and Durability) properties. Though modern-day database systems are turning to Transaction-less systems wherein, they are not bound to support Transactions safe tables, which implies that database is not supposed to worry about rolling back changes. MySQL maintains different storage engines and allows the user to choose Transaction less engine to get a faster response to the queries. The transaction safe environment also exists for GU_DB, however, since it was integrated with the web based system, transaction less model is followed to provide faster retrieval. Our Web based system: Virtual laboratory provides implicit login facility and hence it is not possible to Rollback sessions [9]. Therefore, the

transaction-less environment is used and gives the advantage of Multi user and Mobil systems.

Moreover, the database systems need to maintain the user schema or should be given a default schema to work on. Hence to maintain the user schema, identification is mandatory, which can associate the user id to the database schema.

Implicit login architecture

The database system needs to maintain the schema of the user. Each user is associated with his/her own database schema. In order to give user a personalised schema experience, there was a need to maintain user login in the web based system. Hence, the following features were added to add the login implicitly in front end (Virtual Laboratory).

- Generate unique userid based on the user session in the browser
- Display the ID to the user in the system
- Attach the ID to the Database(GU_DB)
- Allow the user to download the login details(credentials) if the user wishes to download
- Next time, when the user visits the system, fetch the login details implicitly if he access the lab from the same system.
- If the same user tries to access the system from another environment/system/browser, provide the user the option to upload the login details (After fetching from GU_DB) he had downloaded in the previous session. Hence, the user gets his own schema he had used in the previous environment.

Our system (Front-end Side: Virtual laboratory) uses client-side cookies to achieve customized session experience. The requirement for the system was to detect whether it is the new user or existing user. The system then looks for the session entry in the database metadata. If the entry is found, the user is assigned the previous session, else the user is granted with the new session. The user will get his/her session in encrypted form, and they can access the session back by uploading that encrypted credential to the Query Editor.

Thread pool in GU_DB

Connections and sessions are closely maintained in the database instance. Communication is instantiated once the connection is established (*Figure 8*). Our system maintains the sessions with each unique user session id implicitly assigned.

Clients: Clients are the different users accessing the system from the front-end

Connection request: The Query Editor (Web interface) sends requests to the GU_DB Server. It is established through socket communication.

Receiver thread: Incoming connections are queued and processed one by one. The receiver thread instantiates the user thread and the user thread does further processing

Thread cache: Our system uses the thread cache maintained by the OS, wherein either free existing thread is been used or new thread is instantiated.

Database point of view, Buckets are an integral part of efficient storage. And Bucket is one element of hashmap array which is used to store nodes. The bucket utilization depends on how optimal is your hashcode().

Motivation for using hashmap data structure

Hashmap data structures efficiently utilize the multi-threading concept, which means that only one thread can modify a hash table at one point of time, which implies that threads are synchronized. Hashmap uses array in background. Each element is another data structure (like binary tree or linked list etc.). Hashmap solves the problems of searching objects from a large data set. Hash code is an integer value given to each object it identifies. Every object has Hashcode() method that identifies the object, which primarily helps to make searching faster[10].

The technicalities of GU_DB were discussed in the previous sections. The following sections highlight the flow of execution of any query in GU_DB with the help of the example. The example provides clarity on each of the stage discussed in the previous sections and how GU_DB traverses to each of the stages for execution of any of the query. The example is taken of a Data Definition Language (DDL) command-Create Table.

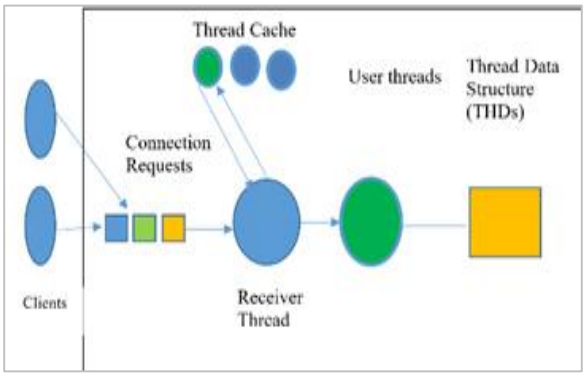

Figure 8 Thread pool in GU_DB

3.2.5 Flow of execution for create table query

For creation of the table, the user would fire a query similar to the query displayed:

Create table emp (empid number (5), ename varchar (10), salary number(5,2));

Once user writes the query, the query passes through lexical and syntax analysis phase as discussed in the previous sections. Once the query is syntactically correct, the value of Table name is checked in metadata, if that table does not exist in the database, further processing is to be done. In further processing, list of columns, primary key availability, any not null constraints and size and type of data are read and a column list is built in Usercolumn Add this information in two tables (Tableinfo and Table_Metadata):

1. Add new table row in Tableinfo
2. Generate new table index to be stored in table meta data
3. Create key to store in hashmap (key;table name;value;index in table meta data)
4. Put key in Hash map data structure
5. Add new table entry in Table Metadata
6. Set Numberofrows = Size of row size of Table info
7. Assign the starting row to metadata
8. Add records to metadata (tablename, startrow, numberofRow)
9. Add Metadata and table_info into a file

The above steps explain the flow of Create table statement, however other DDL statement like Drop table and Alter table have similar type of flow, with only change of modification in metadata by locating the Table info with help of the index key as discussed in the previous section. [Flow Charts for Query flow execution added in appendix]

The alter table and drop table statements first need to locate the metadata entry with the help of the index key and changes are made accordingly in the respective associated tables. Similarly for the DML statements like Insert, Update and Delete, the metadata information of the table is located and accordingly the data is inserted or modified in the said table and other metadata tables are also updated at the same instance. The DQL Statement Select fetches the details from the underlying table based on the user query. However, the compiler needs to locate the metadata entry, match it with the table requested, check for the filters if any and then display the data. Each command cannot be explained in detail due to content restictions.

To understand the technicality in detail, let us have a look at the algorithm:

3.2.6 Create table algorithm

The create table algorithm is employed in the database engine main thread, which is used to Create the new table and give user the output about success or failure of table creation.

Algorithm

This section describes the algorithm to create the table. The Finite Automata (FA) will check the validity of the create table statement and return the status as success or failure of the statement. Depending on the correctness status of the statement, further processing is done.

The overview of the algorithm is presented as follows

i)Check if Directory exists for the table/user. If it does not exist create the same

ii)The scanner examines the query submitted and extracts the tokens (call Tokenization function: (A: Tokenisation)) If the scanner cannot ascertain an appropriate token, the user is displayed a message (Hint) with the wrong token and guided on how to convert it into a valid token (reserved word, keyword, new identifier)

A: Tokenisation

iii)The syntax analyser checks the validity of query with the help of FA using reserved words, keywords and symbol table

B: Syntax checker, and if query syntactically correct query execution starts.

iv)The Metadata information is checked for the existence of the table, the parser checks if the table exists For creating a new table

The new table entry along with the table name, number of attributes, primary key information and index information if any, is done in metadata table.

The User gets the message "**Table Created.**"

Tokenisation Process: (Tokenisation ())

v)Pre-processing (Lexical analyser)

Iterate for all the tags defined in lexical_analysis_scanner Return with the Keywords, accepted states, dump states, columnlist, value list, special input values etc.

Syntax analysis: (Syntax_checker())

Send list of tokens to appropriate Syntaxcheckers like CreateSyntaxchecker, alter, drop, etc. based on the token fetched The Create table Query is passed for valid query checking, There are total 25 valid states

defined for create table finite automata. The last state ";" is considered the end of the statement.

Return with message (True/False)
Once the query passes through all the above stages, the user gets the message "Table Created" and new table information is updated in metadata.

4.Results

4.1 Implementation

GU_DB is implemented as back-end in the Database Virtual Laboratory. The laboratory is approved by Virtual Labs (a project under MHRD NMEICT) and the content is hosted at the Vlabs-Dev portal by IIT-Bombay.

Following are some of the screen-shots of the implementation of GU_DB with Virtual Laboratory (*Figure 9*).

Figure 9 First Screen to enter the Query after clicking on the simulator

To create a new table, user will issue the following statement. *Figure 10* shows that the user receives exact hint of his mistake of the mistake made and the correction to be done of adding datatype and size. Hence, the user gets self-explanatory message and is able to correct own mistake.

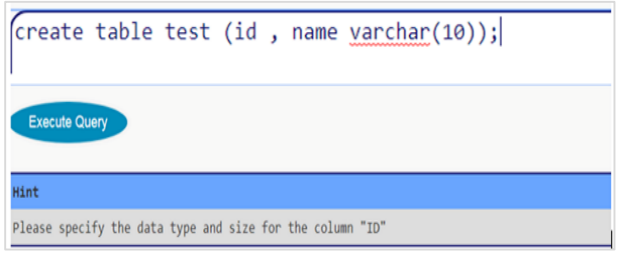

Figure 10 Hint screenshot

In *Figure 11, 12* above, the user has entered the correct query, and hence, the output is displayed on the same screen

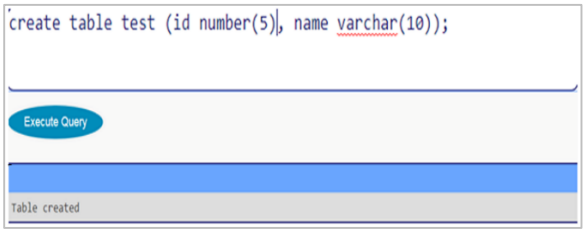

Figure 11 Screenshot displaying the corrected and the message table created

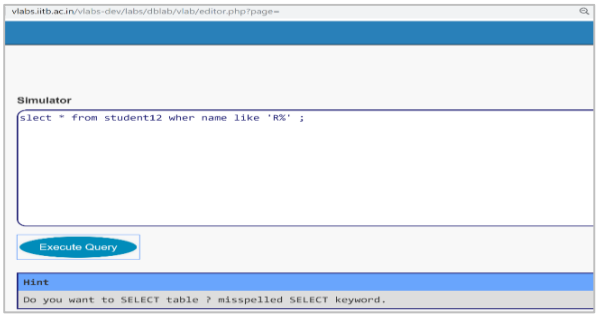

Figure 12 Error in select parameter

The system asks the user, whether he/she meant Select?
This clearly displays that system is also correcting the user on wrong spelling, and it also shows that system is able to read each character and process it. The algorithm developed here reads each character and compares it with the valid keyword, identifies the incorrect character(s) and provides suggestions accordingly (*Figure 13*).

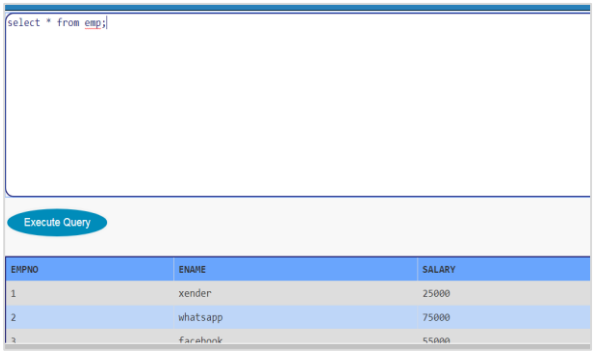

Figure 13 Correct query – output displayed

4.2 Evaluation

The Database virtual laboratory which integrates GU_DB, was submitted to IIT Bombay Virtual Labs. IIT Bombay got the submitted Database lab evaluated by various experts. With final approval from MHRD,

the lab has been hosted at IIT Bombay Virtual Labs development portal in approved labs section.

VLab is a project initiated by the Ministry of Human Resource Development, Government of India, under the National Mission on Education through Information and Communication Technology. The project aims to provide remote-access to Laboratories in various disciplines of science and engineering for students at all levels from under-graduate to research [11].

The Virtual Laboratory for Database is accessed by the student's nation-wide to learn basic database concepts. Moreover, it proved to be of great help during the lockdown, when students could not go to their laboratories for practical's, they were able to perform DBMS practical in our laboratory simulator. The simulator was accessible from desktop/laptop or even mobile, hence there was no need for students to download any DBMS software.

4.2.1Results

To understand the utility of the Database virtual laboratory, the students of Department of computer science, Gujarat university were asked to perform the experiments in the vlab. They performed some of the basic queries of SQL in the Virtual Laboratory.

Figure 14 displays the ratings provided by the students in a questionnaire after the end of evaluation:

The feedback received from students are quiet promising. Following *Table 2* displays the average of all the ratings received.

Limitations of GU_DB

GU_DB is introduced with basic features of SQL like Create, Alter, Drop and Select. Some advanced features like subqueries, Joins and views will be incorporated in future.

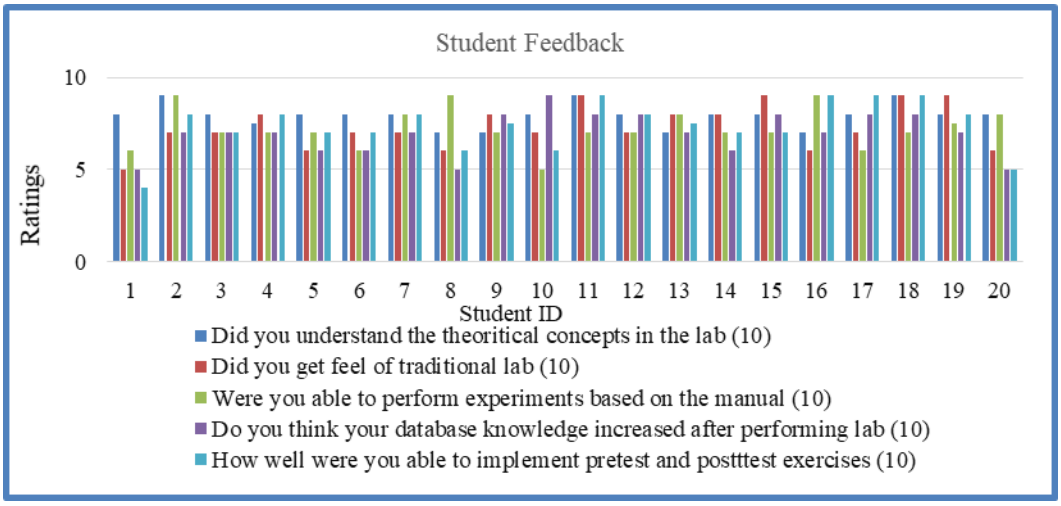

Figure 14 Student-Feedback ratings

Table 2 Average ratings of the responses received

Average of the ratings				
Theoretical concepts clarity (10)	**Did you get a feel of the traditional lab (10)**	**Experiments performed from manual(10)**	**Database knowledge gained (10)**	**How well were you able to implement pretest and posttest exercises (10)**
7.925	**7.3**	**7.225**	**6.95**	**7.35**

4.3 Other Utilities Of GU_DB

Multilingual Support for storage and retrieval in GU_DB

GU_DB allows data to be stored in Gujarati and would be enhanced for efficient retrieval for multi-lingual data. A lot of research exists on many other languages except Gujarati language. The repository of the multilingual data must be as inherent as those in the default database character set [12].

Novel Approach to Optimize Sub Queries

Subquery optimization tool proposes the algorithm for subquery optimization. There might be various

alternative plans which could be selected for a given query. The crucial aspect for such plans is usually the total time required for such a query to execute. The authors have evaluated the existing algorithms on how the sub queries are joined for different relations and proposed an innovative sorting based algorithm for optimally processing these queries. The testing of the algorithm is done in the simulator developed and the results obtained are compared with the existing strategies of MySQL. Authors tried working on and overcoming the limitations of the existing algorithms by proposing the novel enhanced sorting based join algorithm using sorting and merging techniques. The algorithm was tested and implemented in the web based join optimization tool having our algorithm at its depth [13].

5.Discussion

There is hardly any open source database system which has Academic or Research considerations. There hardly exists any research for the development of database management systems for academia. As (Pavlo, n.d.) points out that building DBMS is hard, but building DBMS for Academia is even harder [4]. (Peters & Sikorski, 2003) mention in their article that the researchers who wish to test new concepts are forced to build the entire system from scratch [3]. Moreover, anybody who wants to develop the database prototype has a tough time finding resources. All the database systems approached by us required the proposed work to be tested somewhere, before they could allow us to implement our research work. This motivated us to develop our own Database management system, which can provide us complete flexibility over the system, starting from the phase of the compilation till execution.

The GU_DB which originated as just a Database prototype has now ended up into a complete learning management system (LMS) which can provide a complete learning environment to the learners. The LMS is equipped with the Tutorial of the subject, Procedure to implement the concepts learnt, the simulator for implementing the queries, and the assessment system. The learner is not required to go to any other system, and he can learn, implement, and validate the concepts learned of SQL in one system only at his own pace, time, and location. The research related to our LMS development is published at [8, 14].

Flow charts for execution of create table query are shown in *Figure 15 to Figure 20.*

6.Conclusion and future work

The paper discusses the development of our Database Management System prototype (GU_DB). It focuses on the detailed implementation of the internal working of the development of the DBMS dedicated for Academia. The paper describes the complete architecture of the DBMS and explains various levels at its core, namely Application, Physical, and Logical. The layers are, in turn, elaborated with the type of information they store. The paper highlights the Architecture of GU_DB, detailing all the important components of our DBMS. The Query processor is elaborated by explaining how the lexical and syntax analysis works. The system has pluggable storage engine, implicit login feature and easy connectivity with the Web-based front-end system. GU_DB is used as back-end in Database virtual laboratory which is successfully hosted at IIT-Bombay portal.

GU_DB would be further enhanced with the advanced features like Joins, Subqueries, Views etc. Authors plan to provide open access of the database system to the academicians/researchers to take the research further in proposing new features in the database prototype. The system will be deployed on the global repository like GitHub or similar platform to take the research further. Moreover, the system will be further enhanced for Automated Evaluation of SQL Queries integrating GU_DB. Authors have done a detailed study on various automated assessment system, and one such system performs automated assessment of use-case diagrams [15].

Acknowledgment

The authors would like to extend special thanks to VLabs IIT-Bombay for their continuous support in Virtual Lab development and also congratulates them for various initiatives taken for Virtual labs project.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Peters R, Sikorski R. Building your own: a physician's guide to creating a Web site. JAMA. 1998; 280(15):1365-6.
[2] Venkatesh Emani K, Sudarshan S. Cobra: a framework for cost based rewriting of database applications. arXiv e-prints. 2018: arXiv-1801.
[3] Khurana K, Haritsa JR. UNMASQUE: a hidden SQL query extractor. Proceedings of the VLDB Endowment. 2020; 13(12):2809-12.
[4] https://www.cs.cmu.edu/~pavlo/blog/2017/03/building-a-new-database-management-system-in-academia.html. Accessed 20 March 2021.

[5] https://www.trustradius.com/open-source-database. Accessed 20 March 2021.
[6] Tan WC, Zhang M, Elmeleegy H, Srivastava D. REGAL+ reverse engineering SPJA queries. Proceedings of the VLDB Endowment. 2018; 11(12):1982-5.
[7] Gribkoff E. Applications of deterministic finite automata. UC Davis. 2013:1-9.
[8] Shah B, Pareek J, Patel S, Patel V. Database Virtual laboratory for guided learning.2019; 8(2):5418-25.
[9] Shah B, Pareek J. Virtual Laboratories in STEM courses: a critical review. 2019; 14(2):118-29.
[10] https://en.wikipedia.org/wiki/Hash_table. Accessed 20 March 2021.
[11] https://www.vlab.co.in/. Accessed 20 March 2021.
[12] Shah B, Pareek J. Query optimization for information retrieval in multilingual environment for e-governance resources. In international conference on ICT in business industry & government 2016 (pp. 1-4). IEEE.
[13] Shah B, Pareek J, Kanziya D. A novel approach to optimize subqueries for open source databases. In smart trends in systems, security and sustainability 2018 (pp. 331-346). Springer, Singapore.
[14] https://en.wikipedia.org/wiki/Learning_management_system. Accessed 20 March 2021.
[15] Vachharajani V, Pareek J. Framework to approximate label matching for automatic assessment of use-case diagram. International Journal of Distance Education Technologies. 2019; 17(3):75-95.

Dr. Bhumika Shah is Ph.D in Computer Science. She is working as Assistant Professor at Department of Computer Science, Gujarat University. She has more than 19 years of experience in Academics and Corporate. Her research interests are Database Systems, Multilingual systems, Virtual Laboratories, Application Frameworks, Open Source Systems, Artificial Intelligence, Gamification Big Data, and Cloud computing. She is a senior member at ACM and a life member of CSI.
Email: drbhumikashah@gmail.com

Dr. Jyoti Pareek is Ph. D. in Computer Science. She is working as Professor in Computer Science at Department of Computer Science, Gujarat University. She has more than 30 years of research and teaching experience. She has to her credit a Book and several published research papers. She has been the reviewer of the research papers and member of technical program committee at many International Conferences. Her area of interest are Machine Learning, Natural Language Processing, Information Retrival, Technology for Education, Object Oriented Paradigms. She is member of Board of studies and other statutory bodies at various Universities. She has delivered lectures as resourse person in various seminars and workshops. She is member of ACM, senior Member of IEEE and Life member of Computer Society of India.
Email: drjyotipareek@yahoo.com

Appendix

Flow charts for execution of create table query

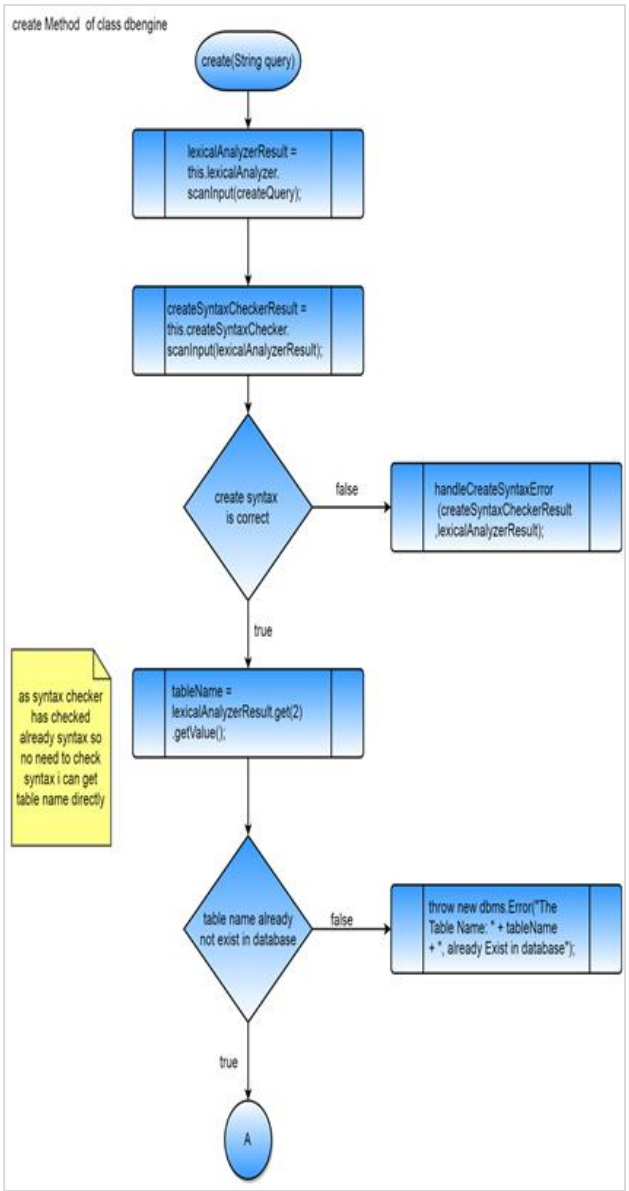

Figure 15 Flow diagram for "Create" – Part 1

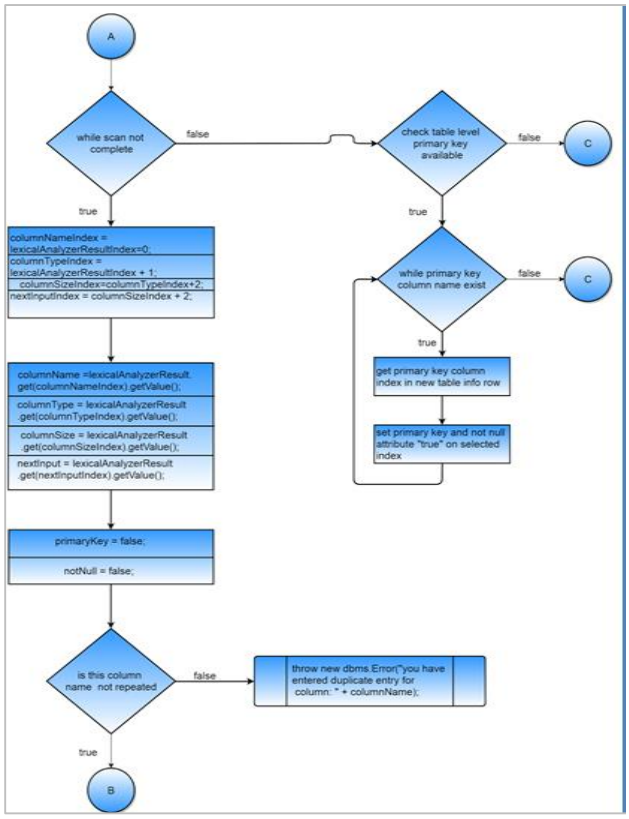

Figure 16 Flow diagram for "Create" – Part 2

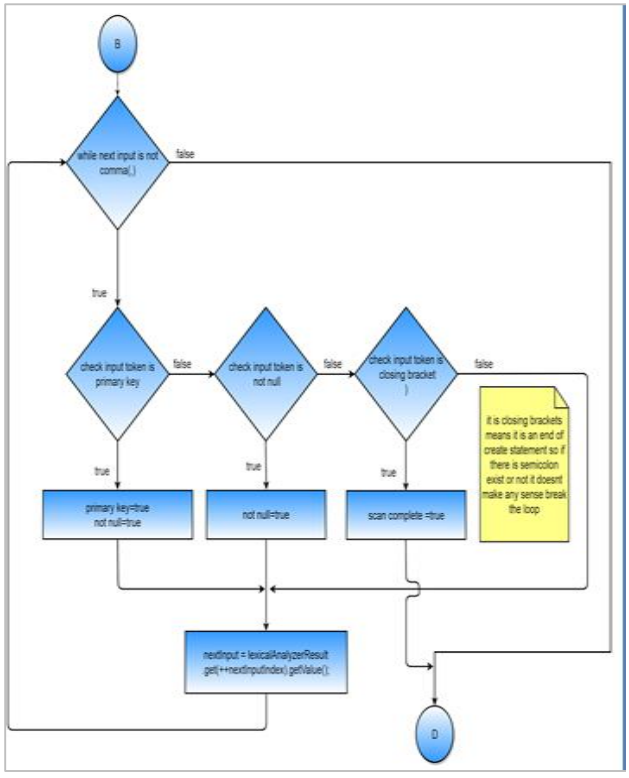

Figure 17 Flow diagram for "Create" – Part 3

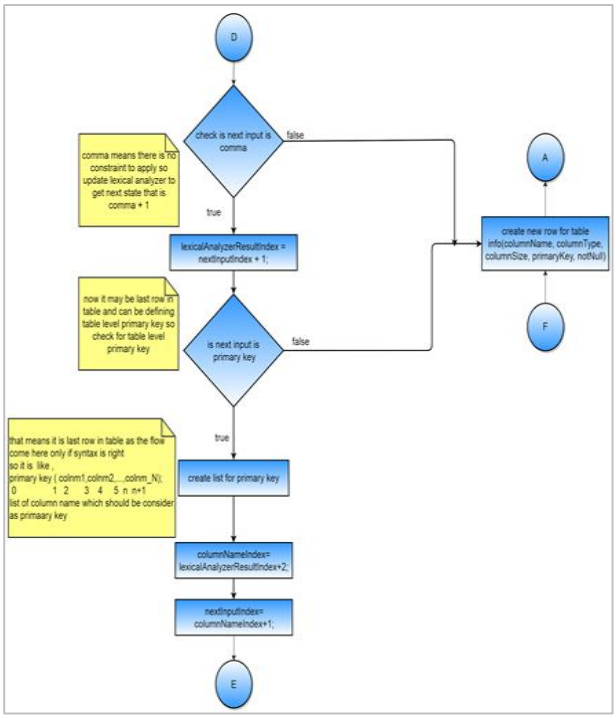

Figure 18 Flow diagram for "Create" – Part 4

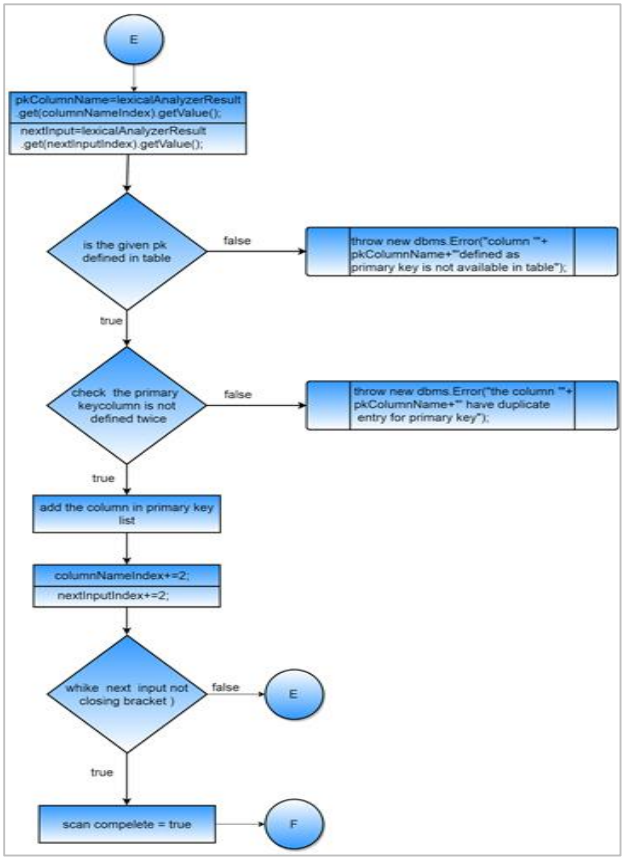

Figure 19 Flow diagram for "Create" – Part 5

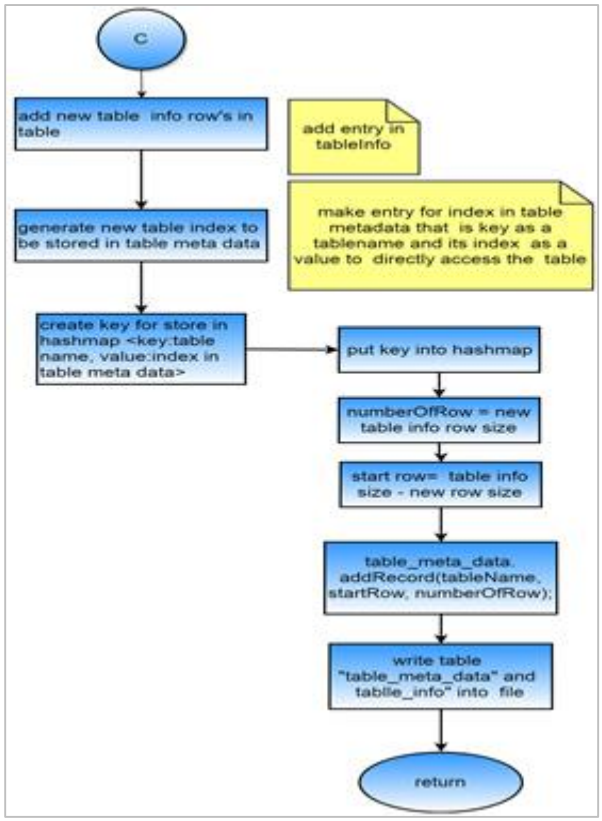

Figure 20 Flow diagram for "Create" – Part 6