# Suggestive Local Engine for SQL Developer: SLED

**Shahid Zaman Barbhuiya[1], Biplab Kumar Ray[2], Zenith Azim[3], Yumnam Jayanta Singh[4]**
Dept. of Computer Science and Engineering and IT, School of Technology,
Assam Don Bosco University, Guwahati - 781017, Assam. INDIA.
*shahidpt0982219[@]gmail.com[1], bip.ray11[@]gmail.com[2],*
*zenithazim[@]gmail.com[3], jayanta[@]dbuniversity.ac.in[4]*

**Abstract:** ***Information Technology (IT) industry recruits junior staff on regular basis. Most of the applications use databases to store or access the data. Structure Query Language (SQL) is used to communicate with database middleware. An expensive SQL statement may engage the data centers for longer time forcing the organizations to sellout high cost for data storage and maintenance. A tool is required for training the junior developers. This study proposes a Suggestive Local Engine for SQL Developer (SLED). It develops a warehouse using the optimized SQL statements collected from reputed software firms or expert team. This study uses the concept of data marts to grouped the data and frequent pattern search algorithm to calculate frequencies and support of patterns of SQLstatements. This system suggests the developers based on the common patterns of SQL statements used by those experts. It also warns the developers if their writing pattern maps to the outlier statement. This system helps all the junior developers in an organization and graduates in colleges or universities to practice with suggestions.***

**Keywords:** *Suggestive engine optimized SQL, Data Warehouse*.

## 1. Introduction

Most of the multinational projects spend a huge amount of money for computerization of their projects. Commonly they have two types of expenses; one given the developer group; another amount is given to the owner of the data server or data center for preserving the data. The first expense may be almost fixed for some years. However the second amount is variable. The data center charges the parent company as per the usages of data and also for the time that it engaged their server. The amount will be increased if the developer wrote SQL queries that take a longer time to execute and extra memory for data usages. This is very much applicable to the cloud computing environment also.

For every new IT project the firms recruit several untrained graduates from colleges or universities. All junior SQL developers are required to train themselves before writing and executing their queries. They need to write less expensive queries to reduce the cost (time, space and complexity). Normally the developers write the codes such as SQL statements in a Local or test environment and then it is moved to live environment for live uses. Several hit and trail methods are practice to provide an optimized one (engaging the local server). Still there are high possibilities that the developers write expensive queries. The firms may also provide them accessibility to all the old queries written by their seniors; however, it is time-consuming to follow several similar queries from a bank of a large dataset.

There exists an exponential number of ways in which a query can be executed. Let us try to understand with the help of an example why and how the manners in which SQL statements are written affect the query execution cost. Suppose, we want to write a query to find the name and address of all employees whose salary is more than Rs. 50,000 (say). The corresponding SQL query is as follows

*SELECT Name, Location FROM employee WHERE Salary > 50000;*

The query can be divided into two executable parts viz. **Selection** and **projection**. The selection part is responsible for the execution of the predicate i.e. select all employee tuples which satisfy the condition. The projection part is responsible for selecting the values of the 'Name' and 'Location' attributes. The query can be executed in two different ways or plans either by first carrying out the section part and then projection or vice-versa. Considering there are 1000 records in the employee table and 50 records that satisfy the predicate, the former plan of execution results in first selecting 50 records and then projecting the names and locations from this 50 records. The latter plan of execution, however, results in first projecting 1000 names and locations and then selecting records satisfying the predicate. Clearly, the former plan of query execution is much efficient than the latter one. The above query was one of the simplest and yet had different ways of execution. Queries containing joins of 5 tables can have as many as 120 different ways of execution and these increases exponentially with the number of tables involved and also as the complexity of query increases. Query optimization is the process of finding the best execution plan for the search space of execution plans. However, to overcome the overhead of the optimization process itself, most query optimization methods select an optimal plan that is quite close to the best one but the best one. However, more the size of the search space increases, the more the optimal plan gets away from the best plan. So, to increase the efficiency of the optimization process, it is necessary to write the optimal SQL statement in the first place.

This study proposes a data warehouse named as Suggestive Local Engine for SQL Developer (SLED) to train the junior developers. This warehouse uses optimized SQL statements as input to a system which as extracted from some World-class

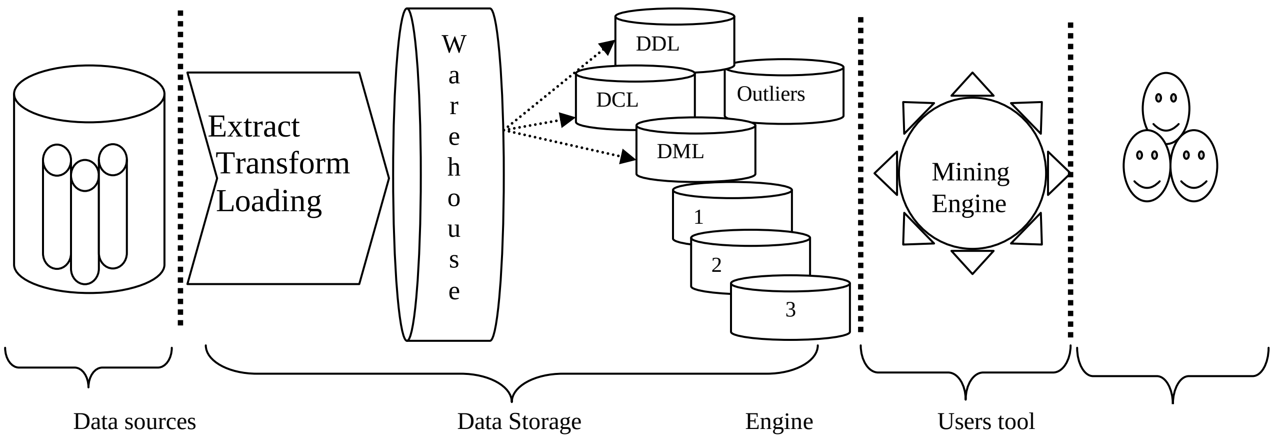


**Fig.1.**Sample of warehouse setup

Software development organization like Oracle, Microsoft or IBM etc. This warehouse has the facility to suggest the developer about the frequency or less expensive queries before they write and execute queries on either local or live server. An interface is proposed to facilitate this activity. Possible Data marts are formed to group the similar queries. This interface is linked to a warehouse and also to data marts which have statistics of patterns of SQL statements. Such warehouse system can be provided to all the academic institutions, so that the college or university graduate practices SQL based on data provided by some World-class Software firms. This may help to them to be industry ready.

The remaining portion of the paper is organized as follow: Section 2 presents some of the related works in this direction. In section 3, we propose the study model and in section 4, the outcomes of the study are forwarded. We conclude the paper with section 5.

## 2. Related Works

SQL tuning is the process of re-writing a poorly written SQL query such that it performs better.SQL tuning and optimization have always remained a hot topic in the area of database research. Such automatic SQL tuning capabilities have been developed for Oracle 10g and Oracle 11g respectively [1][2][3]. Microsoft and its database management solution, SQL Server, are also not lagging behind in this area of SQL tuning and optimization. Automatic SQL tuning capabilities for SQL Server 2005 have been discussed in [4]. The advents of such SQL Tuning capabilities by these major database vendors have aided in improving the execution plan selected by the query optimizer and also restructure and simplify the text of badly written SQL query. However, such SQL Tuners are mainly tailored to suit the needs of their products but the need for such a system that uses the concept of SQL Tuning to educate novice developers about the good practices to follow while writing SQL queries is urgent. Educational Institutions and hobbyist database developers can benefit much from such a system. SQL tuning goals are discussed in [9]. [10] Proposes an SQL tuner based on guidelines presented in [9]. A Query Optimizer that is SQL tuning aware has been discussed in [11].

The novel frequent pattern tree (FP-tree) structure is proposed to allow the mining frequent patterns in transaction databases, time-series databases etc [5][6]. An algorithm is presented to find all occurrences of one given string within another [7].The faults in PL/SQL are predicted using SQL metrics. Based on actual project defect data, the SQL metrics are validated based on fault detection across PL/SQL files [8].

## 3. The Proposed Study Model

We developed a warehouse to help the developers. This warehouse consists of a huge SQL dataset (around 20,000 SQL statements) collected from a reputed software firm. The following steps are performed to develop the system.

A. Design of the warehouse system
B. Extraction of data from the data set
C. Formation of data marts
D. Development of suggestive engine

A. Design of the warehouse system

The sample of development stages of the study is shown in the following thediagram.

1. Data sources:

   The SQL statements are collected from different sources and stored. Error data are separated from the data sources.

2. Data Storages:

   The data are extracted and stored in the main data warehouse. Several data marts are formed based on the requirement.

3. Suggestive engine:

   The suggestive local engine is developed to provide the suggestions while users are using the system.

4. Users tools

   Interfaces are developed for user interaction with the data marts available for the purpose.

B. Extraction of data from the data set

   Data are extracted from the sources as per the requirement. Only the SQL statements are extracted. It transformed and shorted as per required by the data following data marts

C. Formation of data marts

The data mart is a subset or small slices of the data warehouse that is usually oriented to a specific subject. This prepares data marts based on types of SQL viz. DDL, DML and DCL. The DDL and the DCL queries are quite compact and hence do not require further optimization. Most of DDL and DCL queries are expressed in a specified format and do not exhibit much variation in the format. However, it is the DML queries that draw greater concern. DML queries can be expressed in many different forms. All different forms of the same query return the same output but they differ significantly in the execution cost involved. The study focuses on the DML statements where the junior developers need training for connections to any data stores.

So, to achieve the goal and for optimization reasons, the DML data mart is further divided into some data marts. Now, each data mart consists of queries that are similar to one another regarding the functionalities that they perform such as select, update, insert or delete. By convention the functionality of an SQL query can be determined from the first keyword in the SQLText of the query. For example, the query ‘SELECT * FROM Sample’ indicates that the functionality of the query is to retrieve data from a table because the first keyword in the SQLText is SELECT. Using this convention, we assign SQL queries to each data mart and the data martsare identified by the functionality of the queries that it contains such as SELECT, INSERT, UPDATE and so on. The algorithm to set up the data marts can be summarized as follows-

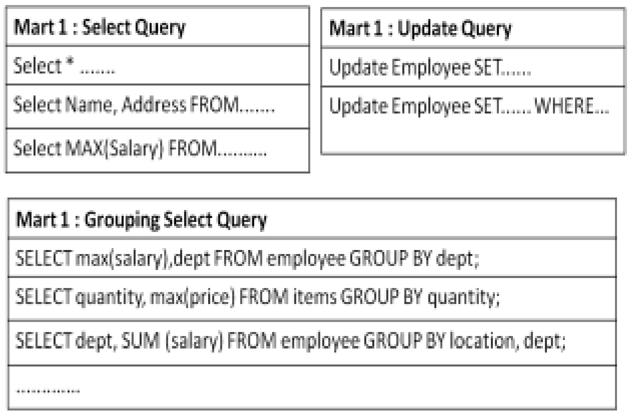
```
Input - The dataset of SQL queries

Output - Various data marts based on the type of
queries
begin
for each SQL query in the dataset
        Parse SQL Query
        Identify the first keyword in query
        if data mart present
                Assign SQL to mart
        else
                Create new mart
                Assign SQL to mart
end for
end
```

The outlier data are analyzed based on the economy of the SQL and gathered in the separate data mart. The considered economy of SQL are cost (time), space associated are also recorded (if it is provided in the dataset). We have prepared the data marts based on the functional similarity of the SQLs. Example SQL queries having 'select...', 'Update' or 'group by', 'having', 'union...' etc. in their SQL statements.

D. Development of suggestive engine

We utilize data mining/pattern recognition techniques to identify hidden patterns from the expert data set of SQL queries. The patterns will emerge in the DML query data mart and more specifically the selection queries. Now, to search for hidden patterns in data, there are two novel approaches: the Apriori approach and the FP-Growth approach. The drawback of the Apriori approach is that it requires multiple scans of the dataset to generate the hidden patterns. This poses an extra overhead to our suggestive engine. So, to overcome the overhead involved, we chose to use the FP-Growth approach for pattern generation. In this approach, an FP-tree is generated from the SQL dataset and all the different kinds of patterns with their respective frequencies can be generated from the tree. The FP-Growth approach allows for pattern generation at various levels.



| Mart 1 : Select Query |
|---|
| Select * ....... |
| Select Name, Address FROM....... |
| Select MAX(Salary) FROM.......... |

| Mart 1 : Update Query |
|---|
| Update Employee SET...... |
| Update Employee SET...... WHERE... |

| Mart 1 : Grouping Select Query |
|---|
| SELECT max(salary),dept FROM employee GROUP BY dept; |
| SELECT quantity, max(price) FROM items GROUP BY quantity; |
| SELECT dept, SUM (salary) FROM employee GROUP BY location, dept; |
| ............ |

**Fig 2 (a)**: Generated Data Marts

The suggestive engine comprises of these frequent patterns in SQL queries as well as an SQL editor for writing SQL queries. The editor works in an online fashion and compares the user-supplied SQL query with the frequent ones and suggested the user of the frequency of the pattern in the dataset thus suggesting how popular the specific pattern of the query is. It also informs the user of the cost involved in executing this query. The final decision is vested on the user whether s/he goes with the suggestion or wants to continue with his/her query. The next section describes the working of the SLED with an example.

## 4. The Outcomes Of The Study

The sample of new SQL editor layout is given in figure (2). This SQL editor will work as a suggestive local engine to suggest the developers. It will dynamically suggestion options to the developer. An enhanced association mining algorithm is applied to find the frequently used SQL statement within a data mart.

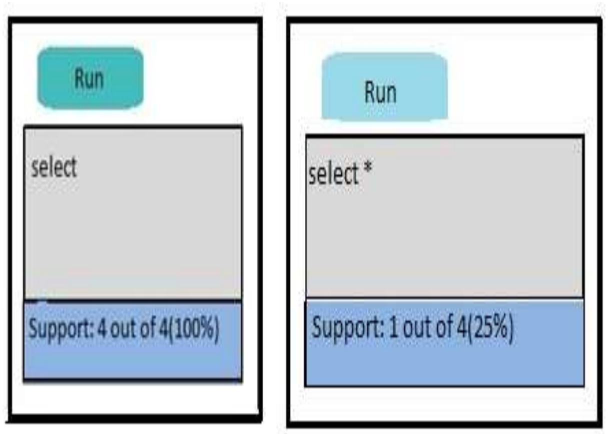
For example a developer is writing a 'Select' statement, the SQL editor will display a count of the total query (say 4) which is a total number of the items in data mart 1. If the developer further continues completing the query 'Select *', the 'Local suggestive Engine' starts working. It uses the association mining algorithm to calculate the frequency of the SQL query pattern written by this new developer until that point of time.

It shows the support of the SQL pattern provided by user developer (say 1) per total number of occurrence (say 4) in that data mart. The support is given by the frequency of pattern in the data mart divided by the total number of patterns in the data mart. Then the developer will come to know that s/he is adapting a frequently used technique or using an unpopular or an expensive SQL statement. Unpopular SQL statements can be considered as outlier data.

The goal of the system is to find the frequency of a user-supplied SQL query pattern in the dataset. After the data marts have been created and all the queries have been assigned to the marts, the system is now ready to start providing suggestions to the novice developer. This is achieved by FP-Growth pattern matching algorithm which searches for the frequency (or support) of the SQLText pattern provided by the user. As the user starts writing his/her query and enters the first keyword, the system recognizes the functionality that the user wants to execute and accordingly selects the data mart to use for providing suggestions. The system functions in an online fashion i.e. it starts searching for patterns and provide suggestions as the user continues writing his query. As the user continues writing his query, he keeps on generating a new pattern. The system continuously takes in the generated pattern and searches for its frequency of occurrence in the data mart. A pattern is said to exist in an SQLText if it is a substring of the SQLText. The system searches for all such substring matches and gives the user information about the support of the pattern. The final decision whether to go with the query or change it if it seems to be unpopular is vested on the user. The generic algorithm can be summarized as follows

```
Input- User supplied SQL query
Output- Frequency/support of the query pattern in SQL
        dataset
begin
 Select data mart based on query type
Calculate the support of query pattern
from FP-tree
Provide Suggestion(s)
end
```

As mentioned earlier, the economy of an SQL query corresponds to the cost and space requirements of the query. Now, this section of the system comes into play after the user has completed writing his query and has submitted it for execution. However, before actual execution of the query, the system provides the user with statistics about the cost and space that will be requiredto the query and the average time and space requirements of the queries contained in the data marts. The economy statistics of the query is achieved from the statistics about the relations maintained by the database.



select

Support: 4 out of 4(100%)

select *

Support: 1 out of 4(25%)

**Fig2(b)**. Sample of suggestive alters by the new SQL editor

An example of such process is like explaining the query execution plan of SQL query generated by the user. The decision again rests with the user whether to continue with the present query or not depending on the comparison between the average space and time requirements of all queries in the data marts and the user generated a query. The generic algorithm is as follows

1. Explain the user generated query execution plan based on statistics maintained by the database.
2. Evaluate the average time and space requirements of all queries in the mart.
3. Compare and decide whether to follow with present query execution plan or change the query with better economy.

To compare the performance of the system, there is few suggestive training system which works in a similar manner. This system will take less time to execute, less space because it works on the data (clean SQL statements) stored on in local host. Even this system can be embedded to low-cost PDAs for quick accessing or referencing the SQLs.

## 5. Conclusion

A local suggestive engine for SQL developer is proposed. A warehouse is developed using the marked data provided from a reputed software development organization. The data marts are prepared to group the SQLs based on functional similarities. The association algorithms are used for calculating the statistics of the 'uses of the patterns' of those experts in their earlier projects. A dynamic SQL editor is proposed which help to suggest the developers based on data from the warehouse by inspecting the 'keyword' those they provided on this editor.

They system the warehouse and dynamic suggestive SQL editor will be very useful to the junior developer or future developer who are studying in college or universities. In future, we intend to enhance the system for academic purposes. Such system will help students learn the best practices to write SQL queries.

## References

[1] B Dageville, D Das, K Dias, K Ragout, M Zait, M. Ziauddin.Automatic SQL tuning in oracle 10g.In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 1098-1109).VLDB Endowment, 2004.

[2] G Harrison. Oracle SQL High-Performance Tuning.In *Prentice Hall Professional Technical Reference*,2000.

[3] P Belknap, B Dageville, K Dias & K Yagoub. Self-tuning for SQL performance in Oracle database 11g.In *Data Engineering, 2009.ICDE'09. IEEE 25th International Conference on* (pp. 1694-1700). IEEE, 2009.

[4] S Agrawal, S Chaudhuri, L Kollar, A Marathe, V Narasayya, M Syamala. Database tuning advisor for microsoft SQL server 2005: demo. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 930-932).ACM, 2005.

[5] J Han, J Pei, Y Yin. Mining frequent patterns without candidate generation.In *ACM SIGMOD Record* (Vol. 29, No. 2, pp. 1-12). ACM,2000.

[6] J Han ,J Pei, Y Yin, R Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, *8*(1), 53-87, 2004.

[7] DE Knuth, JH Morris, Jr., VR Pratt. Fast pattern matching in strings. *SIAM journal on computing*, *6*(2), 323-350, 1977.

[8] TS Quah, MMT Thewin. Prediction of Software Development Faults In *PL/SQL Files using Genetic Nets, WSEAS Trans. on Commn.*, Issue1, Vol. 3, pp228-233,2004.

[9] S Chaudhuri, R Krishnamurthy, S Potamianos, K Shim.Optimizing queries with materialized views. In *icde* (p. 190). IEEE,1995.

[10] Karthik, PG Thippa Reddy,E K Vanan. Tuningthe SQL Query in order to Reduce Time Consumption. *IJCSI International Journal of Computer Science* , 2012.

[11] H Herodotou,S Babu. Xplus: A sql-tuning-aware query optimizer.In *Proceedings of the VLDB Endowment*, *3*(1-2), 1149-1160, 2010.

**Author Profile**

**Shahid Zaman Barbhuiya,**is MTech (CSE) student of School of Technology, Assam Don Bosco University. He completed his Bachelors in Engineering from APIIT SD India in 2013 and currently pursuing his Masters in Technology in Computer Science and Engineering. His specialization is Data Mining.

**Biplab Kumar Ray,** MTech (CSE) student of School of Technology, Assam Don Bosco University. He completed his Bachelors in Engineering in 2012. Currently pursuing his Masters in Technology in Computer Science and Engineering. His specialization is Data Mining.

**Zenith Azim,** is MTech (CSE) student of School of Technology, Assam Don Bosco University. She completed her Bachelors in Engineering from same University in 2012. Currently pursuing his Masters in Technology in Computer Science and Engineering with specialization is Data Mining.

**Yumnam Jayanta,** is working as Professor and Head of Dept of Computer Science & Engineering and IT, School of Technology, Assam Don Bosco University, Guwahati. He has received his PhD from Dr. B.A Marathwada University in 2004. He has worked with Swinburne University of Technology (AUS) at Malaysia campus, Misurata University, Keane (India and Canada), Skyline University College (UAE) etc. His research areas are ETL, Data Warehouse and Mining, Real-time Database system, and Image processing. He has produced several papers in International and National Journals and Conferences. He also holds memberships of IACSIT, IETE, IASTED, EUROSIS and IAENG etc.