EasyChair Preprint

№ 4190

Implementation of TIER : Table Index Evaluator and Recommender

Shefali Naik

September 15, 2020

Implementation of TIER : Table Index Evaluator and Recommender

(Demo Paper)

Shefali Naik
School of Engineering and Applied Science, Ahmedabad University
Ahmedabad Gujarat India
shefali.naik@ahduni.edu.in

ABSTRACT

Most of the relational database management systems have built-in tuning tools that recommend indexes to be created on tables. These tools consider queries of the single database. They do not support queries that are based on tables of multiple databases within the same relational DBMS or tables of multiple heterogeneous relational DBMSs. In this demonstration, the system "Table Index Evaluator and Recommender (TIER)" is presented which analyzes and recommends table indexes for queries that retrieve data from multiple heterogeneous distributed databases. The system takes set of queries as input, parses queries, analyzes and evaluates existing indexes and recommends new set of indexes.

CCS CONCEPTS

• **Information systems~Relational parallel and distributed DBMSs** • **Information systems~Database utilities and tools** • Information systems~Autonomous database administration • Information systems~Recommender systems

KEYWORDS

Distributed Databases, Index Evaluator, Index Recommender, Transaction Performance, TIER

1 INTRODUCTION

The applications or transactions that access data from multiple distributed databases have issues related to distributed concurrency control, distributed query processing, transparency at different levels, designing of physical components [5][6], etc. Physical components occupy space in memory and hence affect the performance of transaction [4]. Index is one of these components that require attention. Though indexes do search faster, improper designing of indexes slow down data manipulation and retrieval. Most of the RDBMSs provide tuning tools[9][10][11][12][13][14][15][16][17][18] that recommend indexes for inputted set of queries. One such tool is Oracle's SQL Analyzer that recommend indexes based on inputted workload. The system demonstrated in this paper evaluates and recommends indexes for the inputted set of queries that are based on remote tables. This system is developed in Java language. Data is generated from Benchmark Factory software and loaded into multiple databases. Overview of TIER is given in section 2.

2 OVERVIEW OF TIER

TIER [1][3] takes set of queries as input and displays recommended indexes as output. The recommendation is given based on analysis of current indexes, number of records in the tables and values in each field. To analyze the existing indexes, statistics are collected from the system catalog. The inputted queries are parsed to generate statistics based on fields referred in WHERE and HAVING clauses. The statistics is generated based on: no. of times the field is referred in all the queries and in individual query. From the inputted queries, details about the foreign key fields is also displayed. Bulk of data is loaded from Benchmark Factory tables. To check the correctness of suggested indexes, separate system is developed, which apply recommended indexes on test database. The same query is fired on both original and test database to record the response time before and after new indexes are applied. The model of TIER is given in Figure 1 [3].

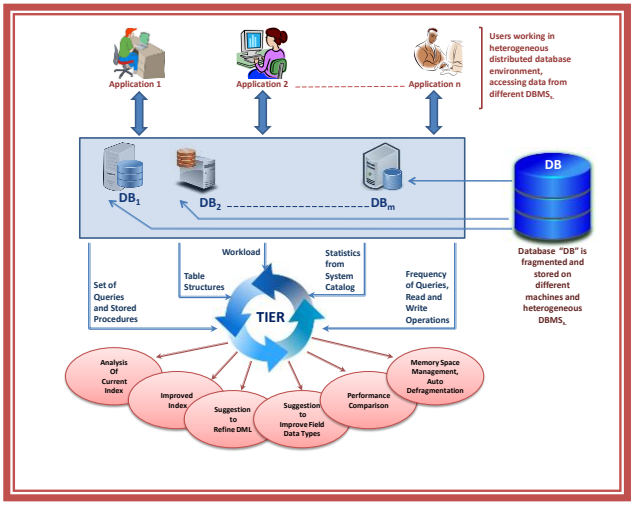

Figure 1 : Model - "TIER: Table Index Evaluator and Recommender"

Apart from this, the performance of recommended indexes in terms of response time is measured in Benchmark Factory [20] for various job runs, different sets of queries and for concurrent user loads - 1, 2, 4 and 8. To cross check relevance between fields

referred in queries, Apriori algorithm [19] is implemented, which is optional. Demonstration of TIER is discussed in Section 3.

3 DEMONSTRATION

Before TIER is executed, it requires distributed environment. The procedure to do this is given in Section 3.1.

3.1 Multiple Heterogeneous Distributed Database Setup

Set of queries are inputted manually or from file in TIER. When TIER is loaded, it displays all the tables created by the oracle user and tables of other databases for which oracle user is given access through database links. Before database links are created, it is required to setup distributed database environment. This environment is created using oracle's heterogeneous gateway services [7]. Four databases - Oracle, MySQL, PostgreSQL and MS Access are connected to fetch data into Oracle using ODBC data sources. Initialization parameter files for non-oracle databases are created and, tnsnames.ora and listener.ora files are updated with the details of data source names. Figure 2 [1] shows oracle user account "shefali" in which data from table "emp" of PostgreSQL is fetched using database link "topg" [1][2].

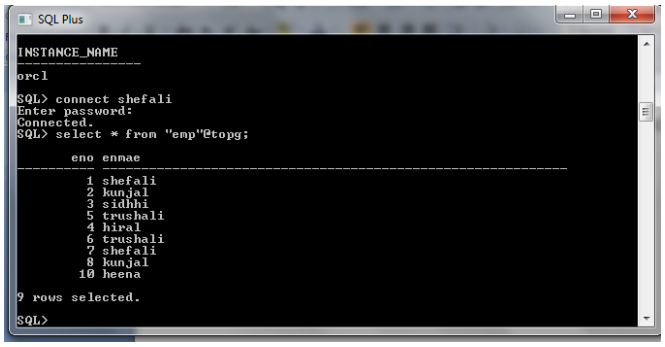

Figure 2 : Fetching data from PostgreSQL in Oracle

3.2 Interface of TIER

Figure 3 [1] shows interface of TIER which list all the table of current oracle user with non-oracle tables displayed with *** before table name to make user understand that these are the tables which are distributed and stored on remote databases. These tables are displayed in the top left section (text area) "Tables (List of Fields)" of TIER [1].

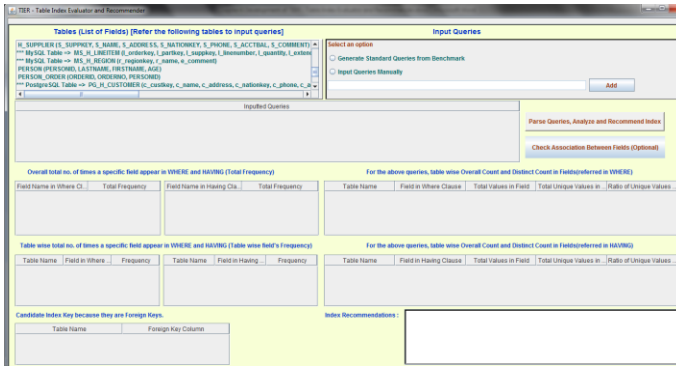

Figure 3 : Interface of TIER

3.3 How it works

Set of queries are inputted through the top right most section "Input Queries" of TIER. It is used to input set of queries either manually or automatically from the file. Once the queries are inputted, by clicking on "Parse Queries, Analyze and Recommend Index", the SQL Parser [8] will parse queries, separate fields used in where and having clause along with the frequency of fields and display intermediate result with the final index recommendations in different text area after analyzing them. Figure 4 [1]shows the index recommendation generated by TIER [1].

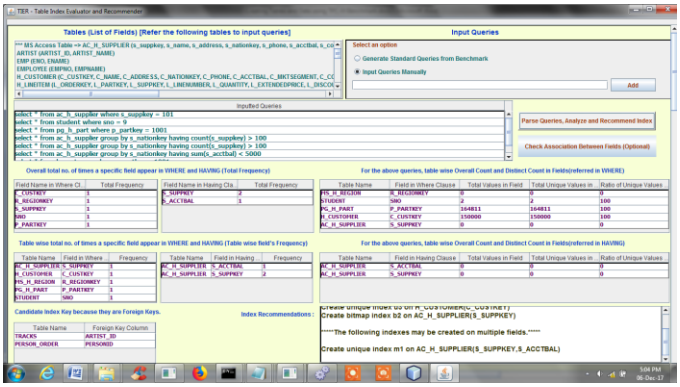

Figure 4 : Index Recommendation

The supportive module which implements Apriori algorithm as shown in Figure 5 [1], is also provided to find association between fields of the tables which are used in where and having clauses.

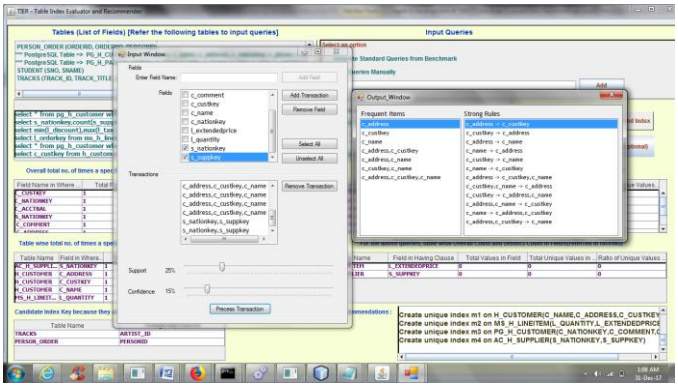

Figure 5 : Implementation of Apriori Algorithm

3.4 Comparison with Oracle's SQL Analyzer

Oracle's SQL Analyzer also recommends indexes for the inputted set of queries, but it does not support queries that are based on remote tables. The comparison of index recommendation generated by TIER and SQL Analyzer is displayed in Table 1 [1] for the following set of inputted queries [1].

1. select substr(p_comment,1,5) from pg_h_part where p_type = "PROMO PLATED NICKEL" or p_type = "LARGE BRUSHED TIN" or p_type = "STANDARD ANODIZED COPPER";
2. select l_shipmode,count(l_partkey) from ms_h_lineitem group by l_shipmode having count(l_partkey)>10

3. select count(*) from ms_h_lineitem where l_shipmode = "RAIL" or l_shipmode = "MAIL";
4. select distinct * from ac_h_region where r_name = "ASIA" or r_name = "AMERICA" and r_comment is not null and r_regionkey = 2;
5. select * from pg_h_nation where n_nationkey = 1 or n_nationkey = 2;
6. select count(distinct s_name) from ms_h_supplier where s_suppkey in between 1 and 100;
7. Select l_returnflag,l_linestatus,sum(l_quantity), sum(l_extendedprice), sum(l_extendedprice * (1-l_discount)), sum(l_extendedprice * (1-l_discount) * (1+l_tax)), avg(l_quantity), avg(l_extendedprice),avg(l_discount), count(*) from ms_h_lineitem group by l_returnflag, l_linestatus having sum(l_discount) > 10;

Table 1 : Comparison of output generated from TIER and SQL Analyzer

Recommendation from TIER : Create bitmap index b1 on AC_H_REGION(R_NAME) Create bitmap index b2 on PG_H_PART(P_TYPE) Create bitmap index b3 on PG_H_NATION (N_NATIONKEY) Create bitmap index b4 on MS_H_LINEITEM (L_SHIPMODE) Create bitmap index b5 on AC_H_REGION (R_REGIONKEY) Create bitmap index b6 on AC_H_REGION(R_COMMENT) Create bitmap index b7 on MS_H_SUPPLIER (S_SUPPKEY) *****The following indexes may be created on multiple fields.***** Create unique index m1 on PG_H_PART(P_TYPE) Create unique index m2 on MS_H_LINEITEM (L_SHIPMODE,L_PARTKEY,L_DISCOUNT) Create unique index m3 on PG_H_NATION (N_NATIONKEY) Create unique index m4 on MS_H_SUPPLIER (S_SUPPKEY) Create unique index m5 on AC_H_REGION (R_REGIONKEY, R_NAME,R_COMMENT)
Recommendation from SQL Analyzer is not generated due to invalid input of statement : ERROR at line 1: ORA-13600: error encountered in Advisor QSM-00775: the specified SQL statement cannot be stored in the workload due to invalid table references ORA-06512: at "SYS.PRVT_ACCESS_ADVISOR", line 1808 ORA-06512: at "SYS.WRI$_ADV_SQLACCESS_ADV", line 180 ORA-06512: at "SYS.PRVT_ADVISOR", line 3636 ORA-06512: at "SYS.DBMS_ADVISOR", line 711 ORA-06512: at line 1

SQL Analyzer recommends indexes for all types of queries based on single database, but it does not recommend indexes for queries that are based on remote tables [1].

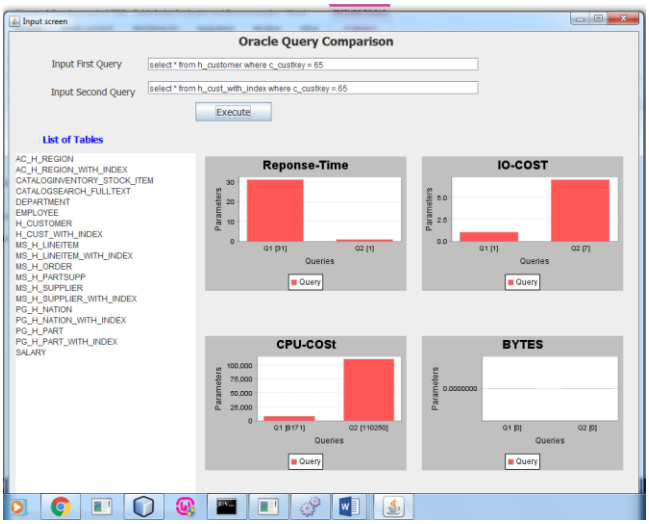

Figure 6 : Output generated from the module Performance Comparer

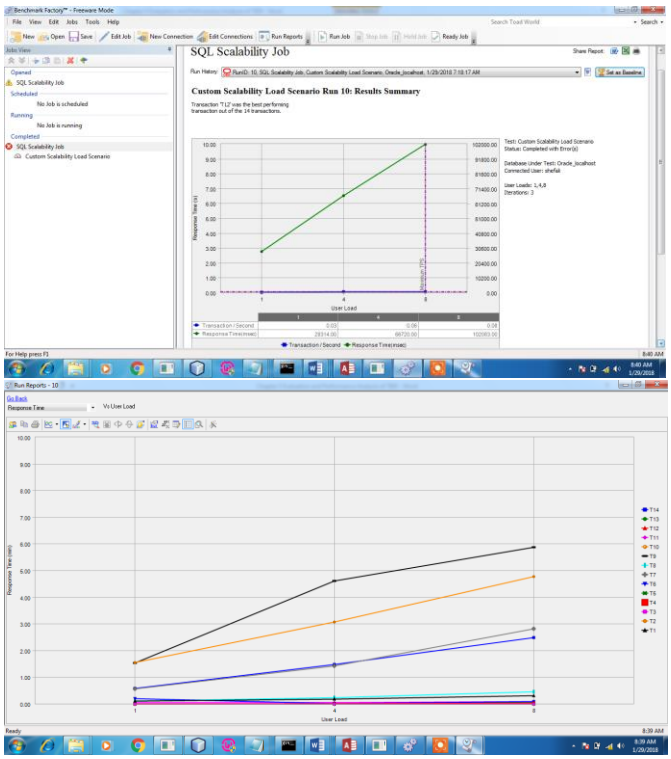

Figure 7 Graphs generated after Run 3 for 14 transactions and 8 user loads for Set of queries specified in Section-3.4

3.5 Performance Evaluation

The separate module, which will compare performance of the query before and after, applied the recommended indexes. This module uses oracle's cost based optimizer to generate cost, response time and I/O cost. Figure 6 [1] shows the comparison generated by performance comparer for the same queries where one is executed on database before index recommendation and the other is executed after index recommendation [1].

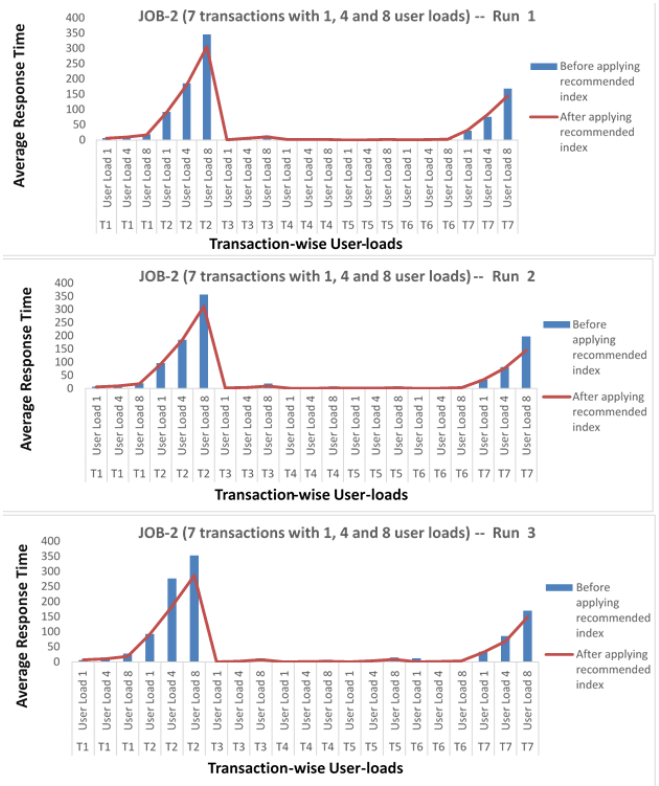

Figure 8 Figure 8 Job-2 for given set of queries for User load 1, 4 and 8 – run 1, run 2 and Run 3 (7 transactions = 14 queries)

Besides this, the performance after index recommendation is also measured in benchmark factory for different jobs, different runs and different user loads [1]. Sample test run done in benchmark factory is shown in Figure 7 [1].

Many jobs are created for the given set of queries (transactions) which refers remote tables. Figure 8 [1] shows the graphical representation of of Run 1, Run 2 and Run 3 for one of these jobs with User loads 1, 4 and 8 for each run.

4 CONCLUSION

The system parses, evaluates and recommends indexes for simple distributed queries, which could be enhanced for complicated queries. The modules to analyze and recommend other physical components for distributed database such as Partitions and Materialized Views could also be developed. The demonstration of the TIER is given on the following links :

https://www.youtube.com/watch?v=52Z1HyLUdOA

https://www.youtube.com/watch?v=6fO26sF2pRY&t=160s

REFERENCES

[1] http://hdl.handle.net/10603/213859
[2] Naik, S. (2017, August). Ensuring Database and Location Transparency in Multiple Heterogeneous Distributed Databases. In International Conference on Future Internet Technologies and Trends (pp. 157-163). Springer, Cham.
[3] Naik, S. (2017, December). TIER: Table index evaluator and recommender—A proposed model to improve transaction performance in distributed heterogeneous database. In Soft Computing and its Engineering Applications (icSoftComp), 2017 International Conference on (pp. 1-8). IEEE.
[4] Naik, S. (2014). Concepts of database management system. Dorling Kindersley.
[5] Özsu, M. T., & Valduriez, P. (2011). Principles of distributed database systems. Springer Science & Business Media.
[6] Shefali, N., & Samrat, K. (2015). Revisited performance issues in concurrent transaction execution in distributed database management system. Int. J. Curr. Eng. Sci. Res, 2(4), 23-26.
[7] https://docs.oracle.com/cd/E11882_01/server.112/ e11050.pd
[8] Parsing Set of Queries to Obtain Parse Matrix for Table Index Evaluator and Recommender, In Press
[9] Bruno, Nicolas & Chaudhuri, Surajit. (2007). An Online Approach to Physical Design Tuning. 826-835. 10.1109/ICDE.2007.367928.
[10] Trung Tran, Quoc & Jimenez, Ivo & Wang, Rui & Polyzotis, Neoklis & Ailamaki, Anastasia. (2015). RITA: an index-tuning advisor for replicated databases. 1-12. 10.1145/2791347.2791376.
[11] Dash, Debabrata & Ailamaki, Anastasia. (2018). CoPhy: Automated Physical Design with Quality Guarantees.
[12] Agrawal, S., Chaudhuri, S., Kollar, L., Marathe, A., Narasayya, V., & Syamala, M. (2005, June). Database tuning advisor for microsoft sql server 2005. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data(pp. 930-932). ACM.
[13] Valentin, G., Zuliani, M., Zilio, D. C., Lohman, G., & Skelley, A. (2000). DB2 advisor: An optimizer smart enough to recommend its own indexes. In Data Engineering, 2000. Proceedings. 16th International Conference on (pp. 101-110). IEEE.
[14] Maier, C., Dash, D., Alagiannis, I., Ailamaki, A., & Heinis, T. (2010, March). Parinda: an interactive physical designer for postgresql. In Proceedings of the 13th International Conference on Extending Database Technology (pp. 701-704). ACM.
[15] Schwartz, B., Zaitsev, P., & Tkachenko, V. (2012). High performance MySQL: optimization, backups, and replication. " O'Reilly Media, Inc.".
[16] Lahdenmaki, T., & Leach, M. (2005). Relational Database Index Design and the Optimizers: DB2, Oracle, SQL Server, et al. John Wiley & Sons.
[17] Gurry, M. (2002). Oracle SQL Tuning Pocket Reference: Write Efficient SQL. " O'Reilly Media, Inc.".
[18] Pedrozo, Wendel & Gomes Vaz, Maria Salete. (2014). A Tool for Automatic Index Selection in Database Management Systems. Proceedings - 2014 International Symposium on Computer, Consumer and Control, IS3C 2014. 1061-1064. 10.1109/IS3C.2014.277.
[19] Wasilewska, A. (2007). Apriori algorithm. Lecture Notes, http://www. cs. sunysb. edu/~ cse634/lecture_notes/07apriori. pdf, accessed, 10.
[20] https://www.quest.com/products/benchmark-factory/