International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 3, March - 2013

Survey on SQL Query Progress Indicator

Prof. Patil L.V.
SKNCOE, Pune.

Mane Urmila P.
SKNCOE,Pune.

Abstract

Nowadays, widely used Business Intelligence(BI) and Data Warehousing(DW) technologies are mostly based on long-running and complex queries. So for this purpose it is important for users to have information about progress of query execution. Recently interest in the development of percent-done progress indicators has been increased. We surveyed the literature of development of progress indicator in database management. In this paper, we summarized some existing methodologies along with its advantages and limitations. Finally we propose a method that constructs model of a percent-done progress indicators based on optimizer-based approach.

1. Introduction

Progress indicators have been studied in various contexts (typical example is file transfer or file download) but there exists very limited work on this topic in case of data management context. In day to day life a typical progress indicator is used to estimate how much of the task has been completed and when the task will finish. Figure 1 shows an example of progress indicator which actually we are trying to develop for database queries.

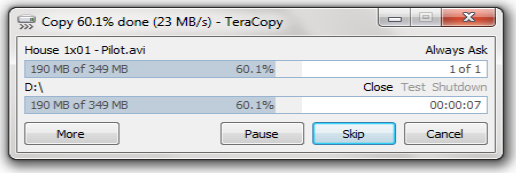

Fig.1. Typical File Transfer using TeraCopy.

In recent years, there has been increasing interest regarding development of progress indicators for SQL queries. A progress indicator in case of database queries is used to estimate precisely the value of a function that is related to the progress towards completion of a running query. For this purpose availability of such indicators can be of great help both to database administrators and end users. Given the complexity of any query in decision support or data warehousing applications, it is common for queries to take hours or days to terminate. During such cases, these indicators can greatly aid a user's understanding of the progress of a query towards completion and allow the user to plan accordingly for example, terminate the query and/or change the query parameters. Also from the point of view of administrators, unsatisfactory progress of queries may point to bad plans, poor tuning or inadequate access paths.

Many modern software systems nowadays provide progress indicators for long-running tasks. These progress indicators aim to make systems more user-friendly by helping the user quickly estimate how much of the task has been completed and when the task will finish. But already existing commercial RDBMSs provide progress indicator for long running queries which were not easy to prove.

Percent-done progress indicators basically used as a technique that graphically shows query execution time that means total and remaining or degree of completion. Also the progress indicator in proposed technique is based on postgerSQL database engine. PostgreSQL is a powerful, open source object-relational database system. Currently postgreSQL doesn't have SQL query progress indicator for long-running queries. With the help of user-system interaction (interface) the progress indicator show the progress of SQL queries through various phases like parsing, analyzing, rewrite, execution. The graphical user interface show all the queries running on system and their estimated time completion. The execution phase of query is critical phase and also the cost of query varies depending disk read time, type of join used, distribution or broadcast of table, order in which tables are joined, statistics information available.

Why use postgreSQL?

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored

IJERT

procedures (in multiple languages). It includes most SQL: 2008 data types. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC.
PostgreSQL prides itself in standards compliance. Its SQL implementation strongly conforms to the ANSI-SQL:2008 standard. It has full support for subqueries (including subselects in the FROM clause), read-committed and serializable transaction isolation levels. And while PostgreSQL has a fully relational system catalog which itself supports multiple schemas per database, its catalog is also accessible through the Information Schema as defined in the SQL standard.
The rest of the paper is organized as follows: Section 2 describes related work regarding the topic. Section 3 discusses the existing methodology along with its advantages and limitations in tabular format. Section 4 describes our proposed system or model for optimizer-based query progress indicator along with its architecture. Section 5 models some of the features of proposed system. Section 6 focuses on conclusion part of this paper and finally section 7 deals with the future enhancement.

2. Related Work

This paper [14] proposed technique sufficient for implementing progress indicator for a large subset of RDBMS queries. They consider select-project-join queries, assume that the available join algorithms are hash-join, nested loops join, and sort-merge join, and those base relations can be accessed by either table-scans or index-scans. They collect statistics at some selected points of a query plan and use that improved and precise information to continuously refine the estimated cost of given query. Thus they estimate remaining execution time of query but don't deal with the percentage of work that has been completed. Also they do not provide estimates for some SQL queries which are non-trivial.

The amount of time required for complete execution of query would be reported to the user at any point during the query's execution. But any existing method which will provide such a measure will subject to the uncertainty arising from concurrent execution of other queries. Hence, due to this difficulty [13] focus on this problem of estimating the percentage remaining or equivalently completed of the given query, at any point during its execution. This paper also deals with the problem of reporting a "progress bar" for query execution. The follow-up work [11] proves that it is impossible for this proposed progress indicator to provide robust guarantees for the problem of progress estimation in the worst case. They provide estimates which are imprecise in certain cases.

This paper [10] considers the problem of supporting the progress indicators for a wider class of SQL queries with more precise estimates. They also discuss and deal with the need of such a progress indicator which is not easy to prove. This paper aims to increase the coverage of progress indicator to large set of queries. They propose techniques to improve the accuracy of the estimates and also to provide new functionality that was not covered in previous work.

Before this all the previously proposed query progress indicators mainly consider each and every query in isolation and thus they ignore the impact of simultaneously running queries on each other's performance. For this purpose [8] proposes technique to extend the single-query progress estimation to enable progress estimation for multiple queries. They explore a multi-query progress indicator, which deals with concurrently running queries and also queries predicted to arrive in the future at the time of producing its estimates. Also they extend the use of progress indicators beyond just being a GUI tool by showing how to apply that multi-query progress indicator to workload management.

This paper [1] implements a cost-based approach for query progress indicator with the help of two proposals which were proposed simultaneously and independently in [12, 13]. They summarize some common cases in which both are accurate and also some cases in which they fail to provide accurate and reliable estimates. This proposed query progress indicator is similar to these early progress indicators but without the uniform speed assumption. The previously proposed progress indicators make a common simplifying uniform future speed assumption. Also the developers of these progress indicators were aware that this assumption could cause errors but they did not explore how large those errors might be as well as they did not investigate the feasibility of removing that assumption.

IJERT

3. Existing Methodology

Table 1. Advantages and disadvantages of Existing Work.

Tool	Purpose	Implementation Technique	Advantages	Disadvantages/Limitations
Toward a Progress Indicator For Database Queries (referred as WiscPI) proposed in [14]	For implementing simple but useful progress indicator for large subset of RDBMSs queries.	-Collect statistics at selected points of query plan. -Monitor continuously query execution speed. -Unit of Work is one byte processed.	-Gives continuous accurate estimated query execution time. -Monitors progress of rollback operation. -From time to time, it presents latest estimates to user.	-For long-running aggregate queries, online aggregation provides no estimate of the remaining query execution time. - Also, no estimate of the remaining query execution time or the percentage completed is provided in dynamic query optimization, as the refining the query cost is not continues.
Estimating Progress of Execution For SQL queries (referred as MSRPI) proposed in [13]	Estimating percentage remaining (or equivalently completed) of query at any point during its execution. -Reporting a "Progress bar" for query execution.	-The GetNext() model of work (MSRPI calculates % of GetNext() calls finished as an estimation of current query progress). -Progress estimation based on GetNext model. -Unit of Work is one GetNext() call.	-Such an estimator is simpler than estimating time remaining since it is independent of other queries (i.e., MSRPI is simpler than WiscPI in case of implementation.).	This estimator does not deal with remaining time while dealing with percentage remaining or completed.
Increasing The Accuracy And Coverage Of SQL Progress Indicators, proposed in [10].	Consider problem of supporting non-trivial progress indicator for a wider class of SQL queries with precise estimates.	-Technique to improve accuracy of estimates. -Technique to provide new functionality.	-Progress indicator can profit from defining segments at a finer granularity. -Simple approach of using the optimizer's estimate of whether segment is CPU or I/O bound can substantially increase the accuracy of progress indicator.	-It is a non-trivial task to make hybrid method for handling correlated sub-queries work at a reasonable overhead. -This approach doesn't deal with supporting progress indicator for SQL queries in ORDBMSs. -This approach doesn't investigate how to support progress indicator for SQL queries in parallel DBMSs. -Fails to prove handling of skew on different data server nodes.
Multi-query SQL Progress Indicator, proposed in [8].	Consider concurrently running queries and even queries predicted to arrive in future when producing its estimates.	-The RDBMSs processes work units at constant rate C (work units per second) that is independent of number of running queries. -The progress indicator has perfect knowledge about remaining cost C_i of each running query Q_i. -Queries execute at speed proportional to weights associated with their priorities.	-Extends use of progress indicator beyond being a GUI tool. -Shows how to apply multi-query progress indicator to workload management. -Provides more accurate estimates than single-query progress indicator. -Considers impact queries have on each other's progress and eventual termination.	In workload management environment one does not want to sacrifice resource utilization ratio in any RDBMS. Queries may incur substantial I/Os and run for long time.
GSLPI: a Cost-based Query Progress Indicator, proposed in [1].	Implement MSRPI and WiscPI both progress indicator in same RDBMS and propose new progress indicator without uniform speed assumption.	- Decomposition of execution plan into set of speed-independent pipelines. -Utilization of wall-clock pipeline cost to represent cost of pipeline. -Estimation of speed of each future pipeline based on its wall-clock pipeline cost.	-They present deeper insight into query's execution. -Due to this it directly affects prediction accuracy. -Lays down foundation for further development of progress estimation.	-GSLPI doesn't deal with parallel database systems regarding some additional challenges like data skew, new operators. -GSLPI doesn't focus on multiple concurrently running queries and also regarding utilization of information provided by progress indicator for better workload and resource management.

IJERT

4. Proposed System

The architecture shown below, describes how the different components of the system interact and there working collaboratively to achieve the desired functionality of the system. The system mainly consists of user/dba, postgresql database, and the GUI which shows the progress of the query and all these components interact with each other.

When user/DBA fires a query then it passes through different phases i.e. parsing, analyze, rewrite, planning, execution of postgresql and at every phase it gives the feedback to the user/dba through the GUI .The feedback is about how much percent of query is completed , how long it will take for query to run to its execution. Also the user/DBA can interact with the GUI during execution by aborting the query in between and the DBA can see at what percentage of the query it is aborted. Aborting the query in between will not harm the data as the kill signal is sent which cause the shutdown of query execution i.e. data integrity is maintained. Effect is only reflected into the database

when the execution of the query is complete. GUI also handles the history of committed queries.

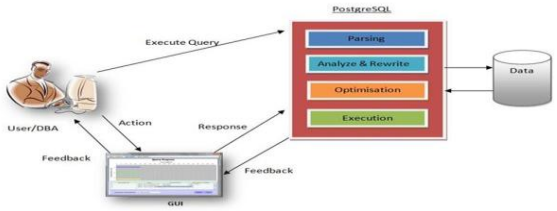

Fig.2.General Outline of system.

A Progress indicator for postgresql database will provide feedback to the user/DBA on.

- How much percentage of query is completed.
- How much percentage and time is required by the query to run to its completion.
- Current phase of executing query.
- Control over the query execution i.e. either allow the query to run to its completion or to abort the query.

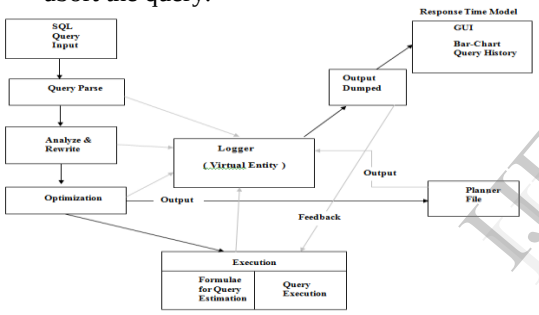

Fig.3. Working of Proposed System

5. Features of Proposed Progress Indicator

The proposed system is having the following features To provide enhanced feedback to the user/DBA on how much of a SQL query execution has been completed i.e. phase of the query and how long it will take for query execution.

- **Multiple Query Progress Graph Display**: The system is designed to handle and display multiple queries progress in form of graphs. The graphs can be disguised by the distinct transaction-id and XY-Line color. The transaction-id is unique local transaction-id given by postgresql for every query.
- **Estimated Time for Query Completion**: The system is gives the estimated time for query completion. The estimated time is dynamic i.e. it varies depending on the system load, resource etc.
- **History of Committed Queries**: The system is also featured with query history. It shows both last committed query and the list of committed queries.
- **Dynamic Variation of Y-Axis**: The Y axis is the time axis and is dynamic in nature as query completion time for different queries is different i.e. one query may commit early and other may long time to complete.
- **Client-Server Implementation**: The system is implemented in 2 tier architecture i.e. client-server .From client side user can fire the query and GUI of query progress will be at client side. At server side query execution is done by the database.

6. Conclusion

The SQL progress indicators for long-running queries are nowadays becoming a desirable user-interface tool to monitor progress of executing query in RDBMSs. But all the previously proposed techniques for supporting the construction of progress indicators for SQL queries are having very limited functionality and accuracy. In this paper, we have surveyed all the previously proposed techniques used for the development of query progress indicator. Also we propose a technique which can be used for the construction of optimizer-based query progress indicator. Also we have modeled some of the features of the proposed system along with its general architecture and principle working.

7. Future Enhancement

As we know that today's world is completely dependent on the internet and online tools. We can enhance our idea and can make our tool as web portal, so that anyone can use it at any time. We can also send the progress status of the query through email or the sms to the DBA. So that he can know the progress of the query without running the GUI and sitting in front of the machine. So like this possibilities are endless.

References

[1] Jiexing Li, Rimma V. Nehme , Jeffrey Naughton; "GSLPI: a Cost-based Query Progress Indicator"; 2012 IEEE 28th International Conference on Data Engineering.

[2] Basit Raza, Abdul Mateen, M M Awais and Muhammad Sher; "Survey on Autonomic Workload Management: Algorithms,Techniques and Models"; Journal of computing, volume 3,Issue 7,July 2011, ISSN 2151-9617.

[3] Kristi Morton, Abram Friesen, Magdalena Balazinska, Dan Grossman; "Estimating the Progress of MapReduce Pipelines";ICDE Conference 2010.

IJERT

[4] Elnaz Zafarani, Mohammad_Reza Feizi_Derakhshi, Hasan Asil, Amir Asil; "Presenting a New Method for Optimizing Join Queries Processing in Heterogeneous Distributed Databases"; 2010 Third International Conference on Knowledge Discovery and Data Mining.

[5] Mario Milicevic, Krunoslav Zubrinic, Ivona Zakarija; "Dynamic Approach to the Construction of Progress Indicator for a Long Running SQL Queries"; international journal of computers issue 4, volume 2, 2008.

[6] Mario Milicevic, Krunosla V Zubrinic, Ivona Zakarija; "Adaptive Progress Indicator for Long Running SQL Queries"; Proceedings of the 8th WSEAS International Conference on Applied Computer Science(ACS'08).

[7] Chaitanya Mishra, Nick Koudas; "A Lightweight Online Framework For Query Progress Indicators"; 2007 IEEE.

[8] Gang Luo , Jeffrey F. Naughton , and Philip S. Yu;" Multi-query SQL Progress Indicators"; Y. Ioannidis et al. (Eds.): EDBT 2006, LNCS 3896, pp. 921 – 941, 2006, Springer-Verlag Berlin Heidelberg 2006.

[9] Christian M. Garcia-Arellano, Sam S. Lightstone, Guy M. Lohman, Volker Markl, and Adam J. Storm; "Autonomic Features of the IBM DB2 Universal Database for Linux, Unix, and Windows"; IEEE Transactions on systems, MAN, And Cybernetics Part C:Applications And Reviews, Vol.36,No.3, May 2006.

[10] Gang Luo, Jeffrey F, Naughton, Curt J. Ellmann, Michael W. Watzke; "Increasing the Accuracy and Coverage of SQL Progress Indicators"; Proceedings of the 21st International Conference on Data Engineering (ICDE 2005).

[11] S. Chaudhuri, R. Kaushik, and R. Ramamurthy, "When can we trust progress estimators for SQL queries?" in SIGMOD, 2005.

[12] DB2, "IBM DB2 query monitor for z/OS," ftp://ftp.software.ibm.com/software/data/db2imstools/whitepapers/db2querymon-wp05.pdf, 2005.

[13] Suraji Chaudhuri, Vivek Narasayya, Ravishankar Ramamurthy; "Estimating Progress of Execution for SQL Queries"; *SIGMOD 2004*, June 13–18, 2004, Paris, France, 2004 ACM.

[14] Gang Luo , Jeffrey F. Naughton , Curt J. Ellmann , Michael W. Watzke; "Toward a Progress Indicator for Database Queries"; ACM SIGMOD 2004, June 13–18, 2004, Paris, France,2004 ACM.

[15] Chaitanya Mishra, Nick Koudas; "A Lightweight Online Framework For Query Progress Indicators"; 2002 ACM.

IJERT