**Abstract**

**Introduction**

Amongst the key processes to ensure efficient execution of queries in a relational database, is the writing of well-crafted query text and the execution of a compiled access plan that aligns with available run time computational resources. Informally referred to as an execution plan, an access plan is a set of steps that constitutes safe, correct and efficient access and retrieval from an underlying database. Efficiency is a measure expressed in terms of computational resources parameters (e.g. disk bandwidth, number of CPU cycles) and performance characteristics (e.g. response time). In general, finding the most efficient access plan for a general purpose relational calculus query, on which SQL’s queries were initially built, is a NP-hard problem [1, 2]. Query optimization, that has to cater for multiple and varied categories of database schemas, use statistics related measures [3]. They are also referred to as optimizer statistics, utilized to translate a query into a reasonable access plan. For the query optimizer component of a DBMS to make more informed decisions, statistics are generated based on the volume and distribution of the underlying data. Therefore this relationship ties the effectiveness of an SQL execution plan generation upon the maintenance and upkeep of optimizer statistics, creating a dependence on maintaining accurate statistics pertaining to the overall system and application performance [4]. Optimizer statistics do not necessarily make query optimization tractable and also comes at a cost; i.e. to generate and hold. The use of statistics in query optimization can generate suboptimal plans and subpar performance in the case of no, insufficient or stale (non-updated) statistics (also referred to as stats here onward), which is often the case when new data is introduced in the underlying schema, as well as drastic and abrupt modifications of existing data, followed by no updating of optimizer stats [4].

**Objectives**

Therefore this research will attempt to tackle the following challenges:

1. Identification of optimum maintenance windows in the database day to day schedule of lowest database activity, where maintenance windows can be established to generate optimizer statistics without compromising the day to day batch execution.

2. Once a maintenance time window has been established, three challenges can be highlighted:

1. Prioritization of which schema objects (e.g. database tables, columns, indexes, partitions) to gather stats upon. Due to the day to day execution workflow affecting and depending on multiple schema objects for efficient execution of operations, it is detrimental to gather optimizer stats for the correct database structures as required by the impending schedule. This pinpointing of specific objects to generate stats upon depends on the flow of execution, in which such a proposed system must be aware of what is going to be executed next so as to be pre-emptive with the stats generation process.
2. Prioritization of which optimizer statistics to gather. With multiple types of statistics, each possessing varied importance subject to the task at hand, the decision making for statistics generation must be intelligent enough to prioritize certain types of statistics over others [15], whilst keeping within the constraints of the day to day schedule.
3. Finally, those prioritized statistics found to be stale, must be generated within the predicted maintenance time window, to avoid exceeding the allocated time for stats generation and risk conflicting with a batch schedule. These stats generation jobs need to be weighted in order of time taken to complete, and otherwise dropped in favor of other maintenance jobs which are predicted to finish in the allocated time.

**Background and Literature Review**

A problem stems from the constant requirement of statistics upkeep, a process which is traditionally overseen by the system’s database administrator (DBA), a task which requires adaptation to schema changes (e.g. new attributes or indexes) and new data to ensure optimum statistics generation for the ensuing database workloads. Many DBMSs have introduced automated processes for the task at hand [5], however these tend to tackle the problem by introducing fixed automated jobs of stats gathering, which are set to run at predetermined points in time as defined by the DBA. Although this alleviates the DBA’s responsibility, it still does not factor in the day to day shifting demands of varied business work flows, a constant and real issue for many online transaction processing (OLTP) and online analytical processing (OLAP) systems [4] whose work schedules are driven by service level agreements (SLA) which must be met and delivered in time. Another facet of this problem is that due to very stringent time schedules of varied database workloads, it is often a problematic task to obtain an optimal time window which can be allocated for database maintenance and optimizer statistic generation, a slot of time where database activity is at its lowest so as to reduce any negative impact on the critical batch and online processing. Past literature [6] suggests ‘just in time’ approaches, where in relevant statistics are gathered prior to query execution. Such a task is considered too workload specific to solve by traditional automatic approaches, and too costly to solve by manual intervention; consequently an ideal scenario for a machine learning approach.

Further to the above, it is highlighted [6, 7, 10] that the issue is not simply a task of determining the occurrence of a maintenance window where in optimizer statistics can be gathered. It is also the quality of the gathered stats which determines the efficiency of an optimizer stats gathering processes, both in terms of the stats gathering job itself, and also the degree of accuracy with which it can anticipate subsequent reliance of stats in the future. Attempts have been made to tackle this fundamental aspect of statistics upkeep through automated means [3], particularly through use of novel techniques which aim to reduce the varied amounts of stats being gathered, and pinpointing those that are needed. Other techniques involve the usage of rule defined solutions, as demonstrated on DB2 [7], where in it is recognized the need to offload the task of maintenance upkeep from the DBA, and offload this work to be done automatically by the database itself. Through usage of DB2’s RUNSTATS statistics collection utility [8, 9] and constant statistic monitoring, this technique prioritizes which schema tables are eligible for statistics upkeep. Whilst this technique approximates an automated routine which caters of statistics upkeep, it does not take into consideration two important factors:

1. The routine for statistics generation is executed in a predefined maintenance window, specifically set by the DBA. In addition, the aforementioned literature [7] proposes a throttling mechanism of stats generation during the day, risking the potentiality that stats upkeep becomes a lengthy process, whilst putting a tamper on available hardware resources due to the constant background stats process.
2. Although the proposed solution attempts to achieve schema object priority for statistic generation, it does so only from the perspective of database tables, giving no attention and purpose to index and column statistics, which are equally relevant.

**Methodology**

Therefore, to address the afore mentioned objectives and effectively produce a number of proposals worthy of tackling the proposed challenges, a representative dataset is required to base any testing and evaluations upon. A decision support TPC Benchmark Schema has been chosen, particularly for its varied, sizeable SQL transactions it is capable of imposing upon the instance under test.

**References**

[1] A. Kumar and M. Stonebraker, "The effect of join selectives on optimal nesting order", ACM SIGMOD Record, vol. 16, no. 1, pp. 28-41, 1987.

[2] T. Ibaraki and T. Kameda, "On the optimal nesting order for computing N-relational joins", ACM Transactions on Database Systems, vol. 9, no. 3, pp. 482-502, 1984.

[3] S. Chaudhuri and V. Narasayya, "Automating statistics management for query optimizers", IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 1, pp. 7-20, 2001.

[4] R. Borovica-Gajic, S. Idreos, A. ilamaki, M. Zukowski and C. Fraser, "Smooth Scan: Statistics -Oblivious Access Paths", IEEE, Seoul, South Korea, 2015

[5] D. K. Burleson, Oracle Tuning, The Definitive Reference Second Edition, 2nd ed. Ora, 2008, pp. 241 -280.

[6] N. Bruno and S. Chaudhuri, "Exploiting Statistics on Query Expressions for Optimization", SIGMOD '02 Proceedings of the 2002 ACM SIGMOD international conference on Management of data , vol. 1, pp. 263-274, 2002.

[7] A. Aboulnaga, P. Haas, M. Kandil, S. Lightstone, G. Lohman, V. Markl, I. Popivanov and V. Raman, "Automated statistics collection in DB2 UDB", Proceedings of the Thirtieth international conference on Very large data bases, vol. 30, no. 04, pp. 1158-1169, 2004.

[8] International Business Machines Corporation, Armonk, N.Y., "SYSTEM FOR DATASTRUCTURE LOADING WITH CONCURRENT STATISTICAL ANALYSIS", 5,873,091, 1997.

[9] [3]"IBM Knowledge Center", Ibm.com, 2018. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSEPEK\_10.0.0/ugref/src/tpc/db2z\_utl\_runstats.html. [Accessed: 17- Aug- 2018].

[10] M. Ziauddin and F. Calif, "METHODS FOR COLLECTING QUERY Attorney, Agent, or Firm Wagner, Murabito & Hao LLP WORKLOAD BASED STATISTICS ON COLUMN GROUPS IDENTIFIED BY RDBMS OPTIMIZER", 6,029,163, 2000.