Intelligent Optimizer Statistic Generation for Relational Databases

Gabriel Sammut
University of Malta
gabriel.sammut.13@um.edu.m

ABSTRACT

Modern relational databases operate through usage of a cost based optimizer (CBO), whose task is to generate efficient access paths for executed structured query language. Access paths chosen by the CBO are influenced by optimizer statistics, generated and renewed through regular maintenance windows, to ensure that any decisions established by the CBO are based on updated statistics. These maintenance windows are scheduled in a fixed manner, with rule defined parameters as to when and which statistics ought to be gathered or renewed. This approach is often not enough to satisfy the optimizer's needs, leading to skewed access plans and performance degradation. This research tackles identification of underlying transactional activity patterns through clustering and regression of computational resources, ensuring a clear distinction between high and low activity. Furthermore, classification techniques will be used to highlight potential access path skews, attempting to accurately suggest which optimizer statistics might have a bigger performance impact upon renewal.

1 INTRODUCTION

Amongst the key processes to achieve fast and 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 [14, 19, 38]. Informally referred to as an execution plan, an access plan is a set of steps that constitutes safe, correct and efficient access and retrieval of data 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 queries were initially built, is an NP-hard problem [17, 22]. Query optimization, that has to cater for multiple and varied categories of database schema architectures, uses statistics related measures to measure the cost of data retrieval methods [6]. These statistics are also referred to as optimizer statistics [32, 38], and are utilized in the process to help translate a query into a reasonable access plan. For the query optimizer component of a RDBMS 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 with the maintenance and upkeep of optimizer statistics, creating a dependency on maintaining accurate statistics pertaining to the overall system and application performance [36].

Optimizer statistics do not necessarily make query optimization tractable - attributing SQL performance largely due to the way it is written. In fact, it is useless to have accurate statistical generation policies if badly written SQL transactions and queries are a common cause. It should also be stated that optimizer statistics are oblivious to user-defined, SQL filter predicates, where in varying ranges of data may be retrieved and consequently even demanding more resources to compute. These statistics also come at a cost; i.e. to generate and store. The use of statistics in query optimization can generate or update suboptimal plans and sub par performance in the case of no, insufficient or stale (non-updated) statistics (also referred to as stats here onward)[1, 32, 38]. This often happens due to underlying schema changes, data upheaval (in the form of new data being inserted, or existing data being removed/changed) and infrastructure changes (environment uplifts) [36]. This makes constant upkeep of optimizer statistics detrimental to application and database performance. Traditional and common practices to ensure statistical upkeep resort to hard set, predefined maintenance windows, often initiated by manual database administrator (DBA) intervention. The decisions pertaining to which schema objects are eligible for renewal are determined through static database values which determine the threshold limit until a database object is eligible for stats renewal. This technique presents a rigid approach towards maintaining updated optimizer statistics, consequently having an adverse effect on execution performance of business dictated transactional workloads. This reliance on human intervention, and rigidness to the CBO's requirements for statistical upkeep [4], exposes a potential pitfall - one which can be potentially addressed through automated machine learned techniques, making the RDBMS reliant on itself to better and more accurately address this problem.

This research paper will establish background related literature to the proposed problem in section 2, followed by objective establishment in section 3. Section 4 covers the methodology as to tackling the proposed objectives, followed by section 5, which makes mention of the techniques applied for evaluation of the proposed methodology.

2 BACKGROUND AND LITERATURE REVIEW

As the demands for data sizing have increased drastically in modern day, so has the demand for more efficient data retrieval techniques [34]. With the relational database a well established back bone of innumerable data housing infrastructures, the need for efficient, fast data manipulation (read/write) has become a crucial component of any RDBMS [17, 22, 36]. This section covers pertaining literature behind the proposed challenges, drawing similarities with other works in the same field so as to further establish the concerned domain.

2.1 Background Domain

At the core of every DBMS is an optimizer, whose job is to generate reliable cost access plans for submitted SQL transactions. An access plan is constituted as a set of instructions describing how data is retrieved from the underlying schema, where cost is a measure of invoked computational resources (CPU Cycles, IO Read/Write) and time required for an SQL's execution (response time) [43]. It should be noted that there exists two types of optimizer variations in commercial DBMS:

- **Rule-based optimizers (RBO)** are a set of self contained rules which inhibit guidelines for the underlying database engine - in an effort to deliver data in the least possible time [12]. These rules function through usage of query representations (typically represented through query algebras) [7]. A shift towards cost based optimizers has become common - the latter cost based version considered to be more elastic than their rule-based counterpart [34].
- **Cost-based optimizers (CBO)** use acquired statistics so as to determine the most efficient access path on the underlying database schema. Through usage of this acquired 'meta-data', the CBO attempts to estimate different variations of SQL access paths, and base decisions on whichever is least expensive [10].

The nature of the proposed solution and reviewed literature concerns itself respectively towards cost-based optimizers, particularly due to the heavy shift towards cost based evaluation methods, and also due to the fact that rule base optimizers are not reliant on optimizer statistics. Therefore, RBO access plan formulation and over all accuracy when it comes to rule based decisions, fall outside the remit of the proposed research.

2.2 Optimizer Statistics

On the highest abstract level, optimizer statistics can be described as a collection of data which describes the database and it's respective underlying objects [32]. The type of information gathered by particular vendor specific RDBMS varies, however it is staple of such technologies to take into account a number of statistical types. Often, these statistical categories revolve around the base calculation of table cardinality estimates, counting of tuple sizes and amounts, number of underlying data blocks constituting schema objects, data skewness representations, clustering factors, and more [5, 32, 35]. Oracle RDBMS distributions categorize statistics into four groupings, which adequately describe the resources and underlying data distribution available which are visible to the optimizer upon SQL compilation, for example [32]:

- **Table Statistics** Includes data pertaining to table number of rows, underlying table data blocks and average row length. These statistics are kept on a per table basis, across different schemas.
- **Column Statistics** These type of statistics pertain to column data, incorporating cardinality on a per column basis, min/max column values and otherwise. An additional type of statistics can be added in the form of histograms, so as to better represent data skewness [9].
- **Index Statistics** This variety of statistics provides data pertaining to schema held indexes, particularly denoting counts of index distinct values, index depth, index clustering factors and leaf block counts.
- **System Statistics** These type of optimizer statistics contain information pertaining to available CPU speed, IO seek time, number of available cores, and more [32].

Further to different statistical types, varied DBMS offer a number of modes with which optimizer statistics can be gathered or renewed. These stat collection modes dictate how thorough such jobs should be in terms of total vs partial scans to underlying database objects [31, 32].

2.3 Problem Domain

It should be noted that the act of statistics gathering and query optimization is a decoupled process [10], with limited insight to what transactional load is present on the underlying database. Daily queries and database jobs are not considered when it comes to optimizer statistic upkeep, taking into consideration only a percentile delta worth of data changes.

Additionally, a problem stems from the requirement of statistics upkeep, a process which is traditionally overseen by the system's DBA, and which requires adaptation to schema changes (e.g. new attributes or indexes) and new data, so as to ensure optimum and refreshed statistics for the ensuing database workloads[38]. Varied DBMS have introduced automated processes for this task [9, 18, 35], however these tend to approach the problem by introducing fixed automated jobs of optimizer statistic gathering, which are set to run at predefined points in time as defined by the DBA. Although this alleviates the administrator's requirement to manually define these schedules, it still does not incorporate the day to day shifting demands of varied business work flows, a constant and real issue for many on-line transaction processing (OLTP) and on-line analytical processing (OLAP) systems [36, 38] whose work schedules are driven by service level agreements (SLA) which must be honored and results delivered in time. Another facet of this challenge is that due to demanding time schedules and stringent scheduling patterns of varied database workloads, it is often a problematic task to obtain an ideal time window which can be allocated for database/application maintenance and optimizer statistic generation, a period of time in the day to day activity schedule, where database activity is at its minimum so as to diminish any negative impact on OLTP and OLAP schedules.

Finally, it should also be mentioned that optimizer statistics are gathered based on a rule defined basis, a percentile factor which dictates how stale respectively gathered statistics are on the database schema [32]. Statistics found to be old and invalidated, are considered to be 'stale', and are generally a common culprit to skewed access plans generated by the cost optimizer [1, 8]. The deciding factor dictating which schema objects are entitled for statistical renewal, varies between one DBMS vendor to the next, often taking the form of a quantitative measure which measures how much data has changed within a schema object. Due to the different varieties of data volumes and data structures in a schema, and the vast gaps between one particular schema design and another, it is an impossible task to satisfy this rule with a fit all threshold value. This challenge makes it an ideal scenario for a machine learning approach [37, 40, 43], which can tailor this value to the schema's needs, on a per schema basis.

2.4 Machine Learning Applicability

Involved in a number of domains, knowledge discovery through data mining and machine learning techniques has seen a number of uses amongst different domains, most prevalent those of a marketing, investment, manufacture, fraud detection roles [11]. With the widespread use and reliance upon RDBMS technologies, and the growing complexities of such systems, new techniques have been proposed in the past to combine the two fields together, concerning aspects of data cleaning, data mining, and most particular to the research at hand, efficient data access. A number of literature attempts to marry machine learning techniques with optimization centric problems exist, each of which proposing to tackle such challenges through different approaches [37, 40, 43]. It is relevant to mention that utilization of machine learned based techniques as a self integrated tool in commercial RDBMS is not new, and have gained traction over the last few years. Oracle Management Cloud [30] employs several machine learning techniques suites in order to achieve a level of self automation, but nothing geared towards better optimization statistic handling. Similarly, Microsoft Azure and DB2 have also been employing similar techniques respectively [15, 26]. However, none of the above mentions tackle the proposed problem, in an effort to more efficiently and more accurately maintain regular updated stats.

A particular well established problem is the aspect of query resource prediction - the ability to predict with some degree of accuracy the quantification of resources used, where in resources can be a measure of computational resources and/or time taken. Hasan and Gandon [37] proposes a model capable of accurately predicting RDF (Resource Description Framework) queries (SPARQL) through usage of past, already executed query logs. Execution metrics are treated as a feature vector, x=(x1, x2,...x(n)), a measure of expended resource statistics, in an effort to predict y, time taken to compute. A number of machine learned techniques are applied in the concerned literature, including usage of k-NN based regression techniques and SVM (Support Vector Machine) regression, for accurate time predictions. Evaluation of the proposed techniques is achieved through coefficient of determination [27] and root mean squared error measures.

Another subset of literature takes a more frontal role when it comes to optimization tuning, where in clustering and regression based techniques are utilized in an effort to fine tune database parameters so as to achieve maximal performance [40]. Although not directly related to the proposed research domain, some useful techniques can be applied for the proposed challenge of workload scheduling and statistics prioritization. Categorized under a single name, ‘OtterTune’ [2], is an automated tool which leverages past experienced workloads on the database and collects new information to further tune DBMS ‘knobs’ so as to better cater for future database workloads, varying in size and type. Through usage of supervised (Lasso Regression) and unsupervised (K-Means Clustering) techniques, Ottertune was tested against Postgress and MySQL database implementations so as to accurately characterize past workloads, and build upon this by identification of the most prevalent metrics during such workloads, so as to fine tune dependent database parameters.

The usage of machine learning regarding the influencing and ‘learning’ from past access plans is also a topic of considerable importance. Marcus and Papaemmanouil present Re-JOIN [25], an automatic table join enumerator, which utilizes information from past processed queries with the sole aim of learning and improving future executions. The novel approach is based on deep reinforcement learning (DRL), where by query execution metrics and optimizer generated access plans are used to train an artificial neural network - allowing Re-JOIN to flag inconsistent join order configurations, and attempts to provide better query plans using less optimization time.

2.5 Statistical Timeliness

The initial challenge for effective statistic gathering is to determine when during the day this generation will have the most positive impact upon SQL performance, taking into consideration the database workload schedule. A common approach [1] suggests the use of a throttling mechanism, where by the optimizer statistic generation job is offloaded as a background process which is constantly active. This alleviates personnel intervention, allowing the database to gather statistics at leisure, regardless of the underlying transactional activity and job schedule. This however risks long running executions of stats gathering, due to the limited resource pool allocated for such jobs. Another point which can be made in criticism of such a technique, is that concurrent generation of new access plans will be influenced by older/stale optimizer statistics (leading to incorrectly based assumptions and possible access plan skew), in turn leading to process timing degradation and potential missing of SLA cutoffs. In addition, such a technique proposes a resource cap, since adequate computational resources would need to be reserved for statistical analysis and generation, depending on the granularity of the job - an assumption that does not always hold true, depending on the underlying hardware infrastructure.

In contrast to this approach, particular RDBMS technologies allow the usage of ‘just-in-time’ techniques [16, 32], utilized to influence the optimizer’s behavior during compilation of an SQL transaction. Particularly Oracle RDBMS [32] utilize the concept of ‘Dynamic Sampling’ [31, 32], allowing the collection of additional transactional query schema object statistics during the optimization phase of an SQL compilation. Different levels of granularity can be opted for during this step, each requiring a significant resource and time overhead for this information to be gathered during run time. Dynamic sampling serves to support and enhance already established statistics, so it should be stated that this technique is not an alternative to traditional optimizer statistics [32]. The statistics gathered in this case are not of high quality or as holistic as the statistics gathered using Oracle’s DBMS_STATS package. Although this type of statistics gathering is incurred at run time and therefore has a better opportunity of handling and dictating what computational resources are required to be aligned for a particular transactional computation, dynamic sampling still requires a solid foundation of underlying optimizer statistics to be truly effective, due to inherently still being reliant upon optimizer statistics.

2.6 Statistical Eligibility

It is also highlighted [1, 5, 23] 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 [6], 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 [1], where in it is recognized the need to offload the task of maintenance upkeep from the DBA, and have this work to be done automatically by the database itself. Through usage of DB2's RUNSTATS statistics collection utility [21] and its constant optimizer statistic monitoring, this technique prioritizes which schema tables are eligible for statistics upkeep. Whilst this technique approximates an automated routine which caters for statistics upkeep, it does not take into consideration two important factors:

- The routine for statistics generation is executed in a predefined maintenance window, specifically set by the DBA. In addition, the aforementioned literature [1] 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.
- 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 to the proposed optimization problem.

Bruno and Chaudhuri [5] proposes the usage of additional statistical techniques to achieve this, which aim to utilize an additional small subset of optimizer statistics through simple maintenance of basic table stats. Introducing the concept of SITs (Statistics on Intermediate Tables), this type of meta data is used to model tuple distribution over intermediate nodes in a query access plan. The specific literature [5] states that this type of gathered statistics improves optimizer accuracy when it comes to efficient generation of SQL access plans, claiming the potential of improving execution timings ten-fold.

3 OBJECTIVES

This research task will address the following challenges by providing justifiable alternative resolutions to them, particularly through usage of machine learned techniques rather than the usage of rule defined parameters. This builds towards the concept of a self-tuning and self-reliant RDBMS [40], allowing the system to automatically and more accurately address the needs for optimizer statistic upkeep, whilst avoiding resource usage overload. This research will evaluate the following objectives so as to measure their effectiveness compared to current used techniques.

- Prediction of optimum, low-activity maintenance windows in the database day to day schedule of low job activity, where maintenance windows can be established to generate optimizer statistics without compromising the day to day batch execution, as established in [38].
- Upon establishment of an optimum maintenance time window, the following challenges can be highlighted:
  - Prioritization of which optimizer stats to gather. With multiple types of statistics at the DBMS disposal, each possessing varied importance to an access plan's formulation, the decision making for statistics generation must be intelligent enough to prioritize certain types of statistics over others [31], whilst keeping within the constraints of the day to day schedule [38].
  - Prioritization of which schema objects (e.g. database tables, columns, indexes, partitions) to gather stats upon, as highlighted in [38]. Due to daily execution work flows affecting and depending on multiple schema objects for efficient execution of operations, it is imperative to gather optimizer stats for the correct database structures as required by the impending schedule. This identification of which specific objects to generate stats upon, depends on the flow of execution predefined by an environment's schedule, during which such a system must be aware of what is going to be executed next in an effort to be preemptive with the optimizer statistic generation process.

4 METHODOLOGY

This section addresses the means as to tackle the proposed challenges and outlined objectives. Techniques established in the literature section will be applied to the proposed challenges, soon after coverage of the data acquisition process required for the research at hand.

4.1 Data Acquisition

To address the aforementioned objectives and effectively produce a number of solutions worthy of satisfying the proposed challenges, a representative dataset is required to base any testing and evaluation upon. A decision support TPC Benchmark schema [39] has been chosen, TPC-DS, particularly for its varied SQL transactions and capability of generating large quantities of data, and its wide usage in the RDBMS research field [33]. TPC related benchmarks are well established in a degree of RDBMS performance literature [40, 42], making them capable candidates for the proposed research domain. The TPC schema will be installed upon an Oracle12c [28] instance, which will serve as the foundation for testing. The decision to base the study upon an Oracle instance in contrast to other versions of Oracle DBMSs, or other vendors entirely, takes the following into consideration:

- The final proposed solution is capable of generalizing, irrespective of underlying relational database vendor, or database version.
- The level of information captured by the technology in relation to runtime execution. Oracle offers a number of internal detailed views [9], detailing the resource usage pertaining to underlying database activity.
- The latest version of Oracle RDBMS technologies, Oracle18c, was unreleased during time of consideration. No changes are

present which pertain to the proposed objectives, between Oracle 12c and Oracle 18c versions [28, 29].
- Familiarity with Oracle relational database usage, in contrast to other relational database technologies.

As a test bed for future evaluation, a total of three schemas are created and loaded with the TPC-DS benchmark suite, sized as 1G, 10G, and 100G (Gigabytes) respectively. The selection of three varied volumes allows the proposed solution to be evaluated across size variations of the underlying dataset - a requirement due to changing optimizer access paths dependent on underlying data distribution [14]. Following this, a thorough benchmark of all TPC-DS provided SQL transactions will be carried out (with / without optimizer statistics presence), so as to identify the top consumers and measure the performance degradation with respect to optimizer stats presence per schema. The usage of flashback functionality [20] will allow efficient reversal of the transactional workloads so as to ensure that each workload process is based upon the same baseline.

To simulate an active working database schedule, a number of simulated workloads are executed and monitored upon the already established volumes, built using the provided TPC-DS transactions. Each schedule would require to execute with varying degrees of transaction loads, underlying data volumes and rapidness of transaction execution. During each schedule run, a number of metrics pertaining to the infrastructure behavior and overall transaction load will be extracted, for which will be used for the next phase of the experiment, as highlighted below. The type of metrics eligible for extraction will pertain to each respective SQL behavior and respective resource consumption. A brief, summarized outlook of the type of metrics which will be captured are as follows, supported by past literature usage [3, 40, 43]:

- Plan Hash Value
- Cost
- Cardinality
- Operation
- Byte Usage

Further to the above captured information, Active Workload Repository (AWR) and Active Session History (ASH) based reports are also being considered, due to their data mining potential in retrieving data pertaining to query performance [9].

4.2 Industry Collaboration

Effort has also been made so as to collaborate with industry partners and acquire a similar workload/schedule dataset, which would portray a real-live working environment in contrast to a TPC synthesized schema. Due to respective company data policies and University of Malta submission regulations, this option is being considered second after the publicly available TPC benchmark suite, prioritizing testing and evaluation on the synthesized dataset.

4.3 Machine Learning Approach

Using the pre-stored scheduling metrics, attempts will be made to incorporate a series of machine learning heuristics upon the acquired and already stored scheduling metrics:

- Incorporation of a number of regression and clustering techniques (inspired from work carried in [40]) which can be applied to determine when during the day, a particular database instance is at its lowest, and highest activity, allowing better accuracy pertaining to ideal time slots when optimizer statistics processes can be carried out. SVM and Lasso Based Regression techniques will serve as a start off point due to already established work in the field, with potential research and utilization of Hidden Markov Model applicability to this time-series problem.
- Identification of which transactions are scheduled for upcoming time slots, and identification which are the top SQL consumers amongst those scheduled - achieved through methods as already covered in Hasan and Gandon and other related work [37, 42]. Furthermore, it is detrimental to identify whether upcoming transactions should be eligible for optimizer statistic gathering, to avoid unnecessary overheads of stats renewal. Suitable means as to detecting access plan changes for respective transactional statements in advance, can be achieved through training heuristic models upon data acquired from Oracle's plan table [34]. It is being proposed to maintain heuristic models trained upon top consumer access plans, and compare new access plans with the already established models, so as to flag any inconsistencies in predicted resource usage, cardinality skewness, and access predicates, similar to work done in Zahir and El Qadi [43] and Marcus and Papaemmanouil [25].
- The choice of which optimizer statistics eligible for generation / updating should be incorporated in these artificially learned heuristics, enabling the relational database to take better informed decisions as to which type of optimizer stats are opted for, a possibility for application of supervised classification based techniques, as suggested in [3]. The choice of which schema objects are most susceptible to improve from optimizer stats, should also be considered.

5 EVALUATION

The evaluation process is categorized and distributed over a number of categories. It should be noted that it is not the impact of optimizer statistics effectiveness which will be measured, a research domain for which vast literature already exists. Instead, due to the nature of the proposed research topic, effort will be taken to gauge the accuracy as to when optimizer statistics are generated upon the database. It is the timeliness and accuracy of the scheduling time slot which will be evaluated, gauging how self reliant the proposed solution is when it comes to basing decisions based off already established transactional execution timings. Factors of time taken to generate new stats, dependency upon upcoming schedule workloads, and relevance of which optimizer related statistics to be gathered, will be considered.

Further more, the discussed evaluation techniques require exhaustion against varied types and lengths of scheduling work flows. The presented tests will be applied to workloads of varied, artificially generated schedules, composed of underlying TPC-DS [39] transactions. In addition, tests require to be performed against schedules concerning transactions of varied resource usage (as quantified in **TPC-DS Top Consumer Identification**) so as to effectively demonstrate the applicability of the proposed techniques on a real life working RDBMS schedule.

5.1 TPC-DS Top Consumer Identification

Initially, each TPC-DS transaction is executed across varied volumes, with and without optimizer statistics, to establish the detrimental performance of optimization stats upon TPC-DS schema. This will also serve to highlight which transactions are most heavy on the underlying data volumes, allowing future work and evaluation to cater for particular attention on these use cases. It should be highlighted at this stage, that due to particular complex permutations of data access retrieval methods, particularly at the level of limited or no relevant optimizer statistics available, it is expected that particular transactional runs will take a measure of time to complete, orders of magnitude timed across hours or days. For such scenarios, such timings will be time capped - timed out ahead of time and flagged as not being able to complete within formal expectation.

Further to the above, it is being considered the validity of using the above acquired metrics so as to build a testing pipeline of scheduled activity, similar to techniques established in [13, 41]. Instead of executing TPC-DS transactions at every stage of the pipeline (some of which require orders of magnitude time to execute), the alternative of attributing the above metrics to transactional activity is being proposed, in an effort to save time on evaluation runs. To simulate metric variation for the transactional executions, the quantified consumed resources could be skewed according to standard deviation about a mean so as to better simulate a realistic executing testing pipeline.

5.2 Scheduling Timeliness

After executing a number of varied workload schedules against the pre-established TPC-DS schemas, the before mentioned clustering and regression techniques [40] need to be evaluated for with which low activity database windows are correctly identified. One of the proposed techniques of evaluation at this phase is to split the 24x7 time window into five (an arbitrary value) minute pockets. Each pocket of time will be flagged with the level of activity established/predicted to be on the database. This technique allows a clear and well understood representation, with which time based scheduling allocation can be verified, allowing the upcoming evaluation techniques to work upon the established assumption that stat generation will be taking place in time slots of low database activity. The effectiveness of these techniques can be measured through Recall scorings of low database activity pockets in the case of clustering evaluation, which measures how accurately a prediction is made, in terms of choosing low database activity measured as percentile usage. Alternatively in the case of regression based predictions, an F-score measure is particularly useful due to incorporation of both precision and recall metrics, in predicting future time line regressions. Regression to the mean techniques are already standardized in Hasan and Gandon's work [37], another means of evaluation which is under consideration for the task at hand.

5.3 Detecting Access Plan Inadequacy

Upon establishment of when optimizer statistics are to be run, the proposed solution should be capable of giving recommendations as to which transactional workloads should be analyzed for optimizer statistics renewal. This phase of the evaluation requires a number of the individually trained heuristic models (trained upon SQL explain plans) to correctly classify whether SQL access plans are varied enough to satisfy optimizer renewal. Self annotated test data for this subset of the problem will be provided through a number of TPCDS transactions, some of which will be altered through optimizer hint injection [24]. Each transaction will be annotated before hand whether it is eligible to have statistics gathered or not, allowing each transaction to be evaluated through measures of the scored recall metric.

5.4 Optimizer Statistic Prioritization

With a level of flagged SQL transactional consumers, the next evaluation step requires to gauge the effectiveness as to the decisions taken behind which optimizer statistics are considered eligible for execution. In addition to this and as already stated previously [1, 6] the quality of the optimizer statistics generation process does not only depend on the type of stats which are renewed, but also which schema objects will considered for this renewal process. Due to the varied permutations for which a cost based optimizer might propose different access plans, it is a difficult task to generate and compare the effectiveness of all possible query plans. In addition, the usage of flashback techniques [20] will be required here, to revert any statistic generation jobs back to an original state, for the purposes of a streamlined and equal based evaluation. In an effort to compare and evaluate the effectiveness of the proposed solutions, comparisons will be made to the traditional established method with which Oracle gathers statistics [32]. This enables a contrasting between the already established well used techniques and those proposed in this research paper - allowing to draw conclusions whether any benefits were noted between different techniques. To effectively compare the two, access plan metrics (cost, estimated cardinality, time taken to execute) per TPC-DS transaction will be compared using both techniques.

6 CONCLUSIONS

With a degree of coverage pertaining to the efficient gathering of optimizer statistics in relational databases, current literature leaves room to explore machine learned automated approaches in contrast to already established, rule-defined techniques opted for by a range of DBMS vendors. By carrying out work on the publicly recognized TPC synthesized benchmark tools, attempts will be made to mimic realistic transactional workloads, and propose a number of machine learning based methods as to how the optimizer statistics scheduling process can be enhanced. A particular focus will be allocated to intelligently automate this task, through analysis of past execution activity. This analysis will not only take into account the time activity of the schedule, but also which types of access plans are predicted to change in upcoming executions. This allows the proposed system to identify potential performance degradation ahead of time, and alleviate this through generation of optimizer statistics. These techniques together allow the RDBMS to intelligently schedule a task detrimental for performance upkeep, in a manner of least impact to transaction workload whilst ensuring a timeliness factor for optimizer statistic gathering.

REFERENCES

[1] A Aboulnaga, P Haas, M Kandil, S Lightstone, + G Lohman, V Markl, I Popivanov, and + V Raman. 2004. Automated Statistics Collection in DB2 UDB. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer (Eds.). VLDB Endowment ©2004, Toronto, Canada, 1146–1157. http://www.vldb.org/conf/2004/IND5P3.PDF

[2] Aken Dana Van, Gordon Geoff, and Pavlo Andy. 2017. Tuning Your DBMS Automatically with Machine Learning. https://aws.amazon.com/blogs/machine-learning/tuning-your-dbms-automatically-with-machine-learning/. https://doi.org/10.1145/3035918.3064029 Accessed: 2018-06-28.

[3] Matthias Boehm. 2015. Costing Generated Runtime Execution Plans for Large-Scale Machine Learning Programs. *CoRR* abs/1503.06384 (2015), 1–6. https://arxiv.org/pdf/1503.06384.pdf

[4] Wolfgang Breitling. 2003. *Fallacies of cost based optimizer statistics.* Technical Report. Centrex Consulting Corporation, Dallas, Texas. 1–16 pages. http://www.centrexcc.com/Fallacies%20of%20the%20Cost%20Based%20Optimizer.pdf

[5] Nicolas Bruno and Surajit Chaudhuri. 2002. Exploiting statistics on query expressions for optimization. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD '02*, J. Franklin Michael, Moon Bongki, and Ailamaki Anastassia (Eds.). ACM, New York, NY, USA, 263–274. https://doi.org/10.1145/564691.564722

[6] Surajit Chaudhuri and Vivek Narasayya. 2001. Automating statistics management for query optimizers. *IEEE Transactions on Knowledge and Data Engineering* 13, 1 (2001), 7–20. https://doi.org/10.1109/69.908978

[7] Mitch Cherniack and Stanley B Zdonik. 1996. Rule Languages and Internal Algebras for Rule-Based Optimizers. In *In Proc. ACM SIGMOD Int'l Conference on Management of Data*, CT Jennifer Widom Stanford Univ., Stanford (Ed.). ACM, Montreal, Quebec, Canada, 401–412. https://doi.org/10.1145/235968.233356

[8] Karl Dias, Mark Ramacher, Uri Shaft, Venkateshwaran Venkataramani, and Graham Wood. 2005. Automatic Performance Diagnosis and Tuning in Oracle. Automatic Performance Diagnosis and Tuning in Oracle. In *Second Biennial Conference on Innovative Data Systems Research, Asilomar, Online Proceedings*, (Ed.). www.cidrdb.org, CA,USA, 84–94. http://cidrdb.org/cidr2005/papers/P07.pdf

[9] Donald K. Burleson. 2010. *Oracle Tuning: The Definitive Reference Book 32 of Oracle in-Focus Series* (second edition ed.). Rampant TechPress, 2010, Kittrell, Nort Carolina. 1200 pages. https://books.google.com.mt/books?id=hiOhVSO-EFcC&pg=PA97&lpg=PA97&dq=artificial+intelligence+oracle+performance&source=bl&ots=56LDDEsPIN&sig=lpfGzeVo_0yMyW3v7BxLAxAsYAU&hl=en&sa=X&redir_esc=y#v=onepage&q&f=true

[10] Amr El-Helw, Ihab F. Ilyas, Wing Lau, Volker Markl, and Calisto Zuzarte. 2007. Collecting and maintaining just-in-time statistics. In *Proceedings - International Conference on Data Engineering*, (Ed.). {IEEE} Computer Society, Istanbul, Turkey, 516–525. https://doi.org/10.1109/ICDE.2007.367897

[11] Fayyad Usama, Piatetsky-Shapiro Gregory, and Smyth Padhraic. 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine* 17, 3 (1996), 1–18. https://doi.org/10.1609/aimag.v17i3.1230

[12] Johann Christoph Freytag. 1987. A Rule-Based View of Query Optimization. In *SIGMOD '87 Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, Umeshwar Dayal (Ed.). ACM New York, NY, USA ©1987, San Francisco, California, 173–180. https://doi.org/10.1145/38713.38735

[13] G. Sullivan David, I. Seltzer Margo, and Pfeffer Avi. 2004. Using probabilistic reasoning to automate software tuning. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, (Ed.), Vol. 4. ACM New York, NY, USA ©2004, New York, NY, USA, 404–405. https://doi.org/10.1145/1012888.1005739

[14] P Griffiths, Selinger M M Astrahan, D D Chamberlin, R A Lorie, and T G Price. 1979. *Access Path Selection in a Relational Database Management System.* Technical Report. ACM New York, NY, USA ©1979, Boston, Massachusetts. 23–34 pages. https://doi.org/10.1145/582095.582099

[15] Hechler Eberhard and Borrello Francesco. 2017. IBM Machine Learning for z/OS. , 46 pages. https://www.ibm.com/us-en/marketplace/machine-learning-for-zos Accessed: 2018-09-11.

[16] Chun Nan Hsu and Craig A. Knoblock. 2000. Semantic query optimization for query plans of heterogeneous multidatabase systems. *IEEE Transactions on Knowledge and Data Engineering* 12, 6 (2000), 959–978. https://doi.org/10.1109/69.895804

[17] Toshihide Ibaraki and Tiko Kameda. 1984. On the optimal nesting order for computing N-relational joins. *ACM Transactions on Database Systems* 9, 3 (1984), 482–502. https://doi.org/10.1145/1270.1498

[18] Ihab F Ilyas, Volker Markl, Peter J Haas, Paul G Brown, and Ashraf Aboulnaga. 2004. CORDS: Automatic Generation of Correlation Statistics in DB2. In *VLDB '04 Proceedings of the Thirtieth international conference on Very large data bases*, A. Nascimento Mario, M. Tamer Özsu, Kossmann Donald, J. Miller Renée, A. Blakeley José, and K. Bernhard Schiefer (Eds.). Morgan Kaufmann, Toronto, Canada, 1341–1344. https://doi.org/0-12-088469-0

[19] Yannis E Ioannidis. 1996. *Query Optimization.* Technical Report 1. ACM Computing Surveys (CSUR), ACM New York, NY, USA. 121–123 pages. https://doi.org/10.1145/234313.234367

[20] J. William Lee, Juan Loaiza, Michael J. Stewart, Wei-Ming Hu, and Jr. William H. Bridge. 2007. Flashback Database. , 8–11 pages. https://patents.google.com/patent/US7181476B2/en Accessed: 2018-06-24.

[21] John Marland Garth, Koshy John, James Alan Ruddy, David Ray Schwartz, and Bryan Frederick Smith. 1999. System for datastructure loading with concurrent statistical analysis. , 5–9 pages. https://patents.google.com/patent/US5873091A/en

[22] Akhil Kumar and Michael Stonebraker. 1984. On the Optimal Nesting Order for Computing N-Relational Joins. *ACM Transactions on Database Systems (TODS)* 9, 3 (1984), 482–502. https://doi.org/10.1145/1270.1498

[23] M. Ziauddin and F. Calif. 2000. Methods for collecting query workload based statistics on column groups identified by RDBMS optimizer. , 11–16 pages. https://patents.google.com/patent/US6029163A/en Accessed: 2018-07-12.

[24] Manhung Wong Daniel and Hei Lei Chon. 2004. Dynamic generation of optimizer hints. , 7 pages. https://doi.org/10/247323 Accessed: 2018-07-12.

[25] Ryan Marcus and Olga Papaemmanouil. 2018. Deep Reinforcement Learning for Join Order Enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD 2018, Houston,TX, USA, June 10, 2018*, NA (Ed.). ACM New York, NY, USA ©2018, Houston, TX, USA, 3:1–3:4. https://doi.org/10.1145/3211954.3211957

[26] Microsoft Azure. 2017. Azure Machine Learning Studio. https://azure.microsoft.com/en-us/services/machine-learning-studio/ Accessed: 2018-09-09.

[27] N. J D Nagelkerke. 1991. A note on a general definition of the coefficient of determination. In *Biometrika*, Vol. 78. Biometrika Trust, Great Britain, 691–692. Issue 3. https://doi.org/10.1093/biomet/78.3.691

[28] Oracle. 2017. *Transforming Data Management With Oracle Database 12c Release 2.* Technical Report. Oracle Corporation, World Headquarters, Redwood Shores, CA 94065, USA. 1–10 pages. https://www.oracle.com/technetwork/database/plug-into-cloud-wp-12c-1896100.pdf

[29] Oracle. 2018. *Introducing Oracle Database 18c.* Technical Report. Oracle Corporation, World Headquarters, Redwood Shores, CA. 1–15 pages. https://www.oracle.com/technetwork/database/oracledatabase18c-wp-4392576.pdf

[30] Oracle. 2018. Oracle Management Cloud. https://docs.oracle.com/en/cloud/paas/management-cloud/index.html Accessed: 2018-08-05.

[31] Oracle Corporation. 2017. *Best Practices for Gathering Optimizer Statistics with Oracle Database 12c Release 2.* Technical Report. Oracle Corporation, World Headquarters, Redwood Shores, CA 94065, USA. 1–25 pages. https://www.oracle.com/technetwork/database/bi-datawarehousing/twp-bp-for-stats-gather-12c-1967354.pdf

[32] Oracle Corporation. 2017. *Understanding Optimizer Statistics With Oracle Database 12c Release 2.* Technical Report. Oracle Corporation, World Headquarters, Redwood Shores, CA 94065, USA. 1–29 pages. https://www.oracle.com/technetwork/database/bi-datawarehousing/twp-statistics-concepts-12c-1963871.pdf

[33] Raghunath Othayoth Nambiar and Meikel Poess. 2006. The Making of TPC-DS. In *VLDB '06 Proceedings of the 32nd international conference on Very large data bases*, Umeshwar Dayal, Khu-Yong Whang, David Lomet, Gustavo Alonso, Guy Lohman, Martin Kersten, Sang K. Cha, and Young-Kuk Kim (Eds.). VLDB Endowment, Seoul, Korea, 1049–1058. http://www.tpc.org/tpcds/presentations/the_making_of_tpcds.pdf

[34] Renata Pacholewicz. 2014. Assessment of impact of selected factors on the effectiveness of cost-based optimizer in database systems. *Zeszyty Naukowe Akademii Morskiej w Gdyni* 84, NR (2014), 92–105. https://www.infona.pl/resource/bwmeta1.element.baztech-3d3716f1-0a1d-4481-9f4d-f6475e5da6c2

[35] PostgreSQL. 2018. The Statistics Collector. https://www.postgresql.org/docs/9.6/static/monitoring-stats.html Accessed: 2018-07-30.

[36] R. Borovica-Gajic, S. Idreos, A. ilamaki, M.Zukowski, and C. Fraser. 2015. Smooth Scan Statistics-Oblivious Access Paths. *IEEE 31st International Conference on Data Engineering* 31, NA (2015), 2–5. https://doi.org/10.1109/ICDE.2015.7113294

[37] R. Hasan and F. Gandon. 2014. A machine learning approach to SPARQL query performance prediction. In *Proceedings - 2014 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2014*. IEEE, Warsaw, Poland, 266–273. https://doi.org/10.1109/WI-IAT.2014.43

[38] Sammut Gabriel. 2018. *Faculty of ICT Master of Science in Artificial Intelligence-(Part-Time) Proposal Form Title: Intelligent Optimizer Statistic Generation for Relational Databases.* Technical Report. University of Malta, Msida, Malta. 1–4 pages.

[39] Transaction Processing Performance Council. 2018. TPC BENCHMARK TM DS. , 136 pages. http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-ds_

v2.9.0.pdf Accessed: 2018-03-01.

[40] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. 2017. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17*. ACM New York, NY, USA ©2017, Chicago, Illinois, USA, 1009–1024. https://doi.org/10.1145/3035918.3064029

[41] Andre Van Hoorn, Matthias Rohr, and Wilhelm Hasselbring. 2008. Generating probabilistic and intensity-varying workload for web-based software systdems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Samuel Kounev, Ian Gorton, and Kai Sachs (Eds.). Springer-Verlag Berlin, Heidelberg © 2008, Darmstadt, Germany, 124âĂŞ143. https://doi.org/10.1007/978-3-540-69814-2_9

[42] Wentao Wu, Yun Chi, Shenghuo Zhu, Junichi Tatemura, Hakan Hacigümüş, and Jeffrey F. Naughton. 2013. Predicting query execution time: Are optimizer cost models really unusable?. In *Proceedings - International Conference on Data Engineering*. https://doi.org/10.1109/ICDE.2013.6544899

[43] Jihad Zahir and Abderrahim El Qadi. 2016. *A Recommendation System for Execution Plans Using Machine Learning*. Technical Report. Mohammed V University in Rabat, Moulay Ismail University in Meknes, Meknes, Rabat , Meknes. 1–2 pages. https://doi.org/10.3390/mca21020023