**PROJECT PROGRESS REPORT 2**

1. **Title**: Implementing an auto-indexing algorithm as an extension on PostgreSQL’s platform.
2. **Authors**: Charlie Liu (charlie.liu-1@ou.edu), Dragon Tran (dragon.tran@ou.edu), Emily Black (emily@ou.edu), Charles Johnson (charles.t.johnson@ou.edu)
3. **Project Objectives and Motivations**:

The objective for our project is to implement an auto-indexing algorithm within an extension on PostgreSQL’s platform and compare how efficiently the algorithm indexes tables. We will be branching off of an extension for PostgreSQL called Dexter, rewriting our own extension, and then implementing the auto-indexing algorithm within that extension. We decided to change up our topic from the previous report’s topic because after extensive research, we came to the conclusion that PostgreSQL indeed does not have an auto-indexing algorithm in it’s source files, nor does it do any sort of auto-indexing techniques. With this in mind, we still want to implement an auto-indexing algorithm within PostgreSQL, but we will need to re-implement and create an extension that attaches to PostgreSQL to read log files and determine what should be indexed based on the query workload. We chose PostgreSQL because unlike Oracle, PostgreSQL is open source and modifiable. We will be benchmarking our results with TPC-C. According to the TPC website[7], TPC-C is an online transaction processing benchmark (OLTP) that “involves a mix of five concurrent transactions of different types and complexity either executed online or queued for deferred execution. The database is comprised of nine types of tables with a wide range of record and population sizes.”. Using TPC-C against our algorithm, once implemented, is one of our project objectives, because it provides what we need as far as testing standards go.  
 The overall objective of our project is incredibly important when it comes to lessening the daily workload of a Database Administrator, while improving the overall performance of the database itself.

1. **Literature Review**:

(Old literature review is the same as progress report 1)

When new indices are generated, we would like to evaluate and assess its performance accurately and to know that they are valid and consistent metrics of performance. Self-designing a test database, designing queries, and designing comprehensive test that simulate transactions and varying workloads is challenging which might not result in a valid or comparable metrics. When database vendors need to test their database performance on a new product, they use industry standard database benchmarking frameworks, such as TPC, AS3AP, etc. [7,8]. Benchmarking is the process of testing a database system with a widely accepted set of data and tools to determine its relative performance. These benchmarking frameworks are designed to simulate an Online Transaction Processing (OLTP) application and workloads. An OLTP application is a system where the database processes the transaction (i.e. user action such as depositing into bank account) immediately, transactions can occur whenever a user performs an action, and multiple transactions can occur.

AS3AP, known as the ANSI SQL Standard Scalable and Portable, is a “benchmark designed to provide meaningful measure of database processing power. [8]” As described in the name, this benchmark framework is intended to test and gather performance metrics and intended to be portable and compatible with different systems. Turbyfill states that this benchmark introduces a metric called the *equivalent database ratio* which is intended to provide a “non-ambiguous interpretation of benchmark results.” [8] The benchmark test consists of 3 parts: single user test, multi-user test, and special user tests. It determines an equivalent database size for the AS3AP database, where the system will execute a set (of max size = equivalent database size) of tests consisting of the (some or all) three parts for a total run time of less than 12 hours [8].

The TPC, also known as the Transaction Processing Performance Council is a non-profit organization whose mission is to define database benchmarking and transaction processing methods. The TPC offers many different types of benchmarks for a variety of different database applications. TPC-C (Type-C) is an OLTP benchmark with the specific guidelines/approach where it states that “results are expected to be accurate representations of system performance” and “is an accepted standard” [7] The benchmark is designed to simulate the workload of a database application of a wholesale supplier. [7] The metric that this benchmark uses is the *tmp-C* metric, which is a measure of transactions per minute or business throughput. It consists of a test database of 9 tables with varying records amounts and several types of transaction (including concurrent transactions) that simulates wide range of complexity and types of business activity performed on the database. [7]

1. **Work completed**:

As mentioned earlier, we discovered that postgreSQL does not have any built in auto-indexing functionality. This causes issues with what we originally intended to research. We would not be able to compare algorithms if there was no way to implement them. Because of this we decided to refocus our goal of this project. This set back has caused us to fall behind on our project. Alternatively, it has given us a clear objective to pursue instead of continuing to find a way to make postgreSQL work the way we need it to, in a way that postgreSQL clearly cannot do. Going forward the new goal of our project is to develop an extension that will allow users to implement auto-indexing components.

Given that we are currently behind due to the refocus of our project, we’ve established a plan for the remaining time we have to accomplish our goal. We’ve divided up the project work into two sub teams. Along with dividing project work, we’ve set up a timeline of objectives to complete the necessary work.

Once we have developed the framework and implemented the auto-indexing algorithm, we will need some way to determine if the auto-indexing component is an improvement to the DBMS. for this we’ve decided to use TPC-C, which is an OLTP benchmark.

In summary, we’ve been set back due to the lack of functionality in postgreSQL. Alternatively, we re-established our project goal, decided on developing an implementation framework in python, chose the benchmarking tool to use, divided the work to be done among two teams and set up a timeline to accomplish our project goals. While we are currently behind, if we adhere to the plan we’ve set, we should conclude our project in time.

1. **Work to be done**:

Since our plan changed from replacing the postgreSQL auto-indexing method with a different algorithms and compare their efficiency to implementing our own framework to add an auto-indexing algorithm as an extension to postgreSQL, we split the work into two groups: Emily and Charlie will be creating the algorithm and Dragon and Charles will be creating the framework that allow the integration between our algorithm and PostgreSQL.

For the algorithm team, Emily and Charlie will work together to create algorithm for automatic index selection. Based on our current list of references, our algorithm will be based on non-clustering techniques used for on-line tuning. Over the period of next week, we will work to consolidate what we know and start implementing our algorithm. When the algorithm is finished, Emily and Charlie will work together with Dragon and Charles to finish the implementation.

For the framework team, Dragon and Charles will work together to create a bridge between PostgreSQL and our algorithm. The language we’ve decided to work in for our implementation is Python due to its high readability and cross-platform compatibility, and everyone on the team has had prior experiences working with Python. We have discovered a host of useful resources made by the PostgreSQL community, and from these resources HypoPG (<https://hypopg.readthedocs.io/en/latest/index.html>) and psqlparse (<https://github.com/alculquicondor/psqlparse>) seems very relevant to our goals, and we’ll try to integrate them moving forward.

When the algorithm is finished and deployed on PostgreSQL we will conduct benchmark testing using Transaction Processing Council (TPC)’s TPC-C benchmarking database, which contain dataset and queries for an online transaction database. To assess if our algorithm is beneficial, we will compare the benchmark results between PostgreSQL without our extension and PostgreSQL with our extension.

Time Table (April)

|  |  |  |
| --- | --- | --- |
|  | Emily & Charlie | Dragon & Charles |
| Week of April 2nd | * Finalizing and start implementing algorithm | * Implementation of extension in Python |
| Week of April 9th | * Finish implementing algorithm. * Help Dragon & Charles with implementation | * Continue Implementation of Python code * HypoPG and psqlparse integration |
| Week of April 16th | * Finishing implementation * Implement TPC-C benchmark database * Debugging and preliminary testings | |
| Week of April 23rd | * Wrap-up benchmark testing * Final report | |

1. **References**:
2. Schnaitter, Karl & Abiteboul, Serge & Milo, Tova & Polyzotis, Neoklis. (2006). COLT: Continuous on-line tuning. Proceedings of the ACM SIGMOD International Conference on Management of Data. 793-795. 10.1145/1142473.1142592.
3. Schnaitter, Karl & Polyzotis, Neoklis. (2010). Semi-Automatic Index Tuning: Keeping DBAs in the Loop. Computing Research Repository - CORR. 5.
4. 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.
5. Nimkanjana, Klinsukon & Vanichayobon, Sirirut & Wettayaprasit, Wiphada. (2008). Auto-Indexing Selection Technique in Databases under Space Usage Constraint Using FP-Growth and Dynamic Programming. Computer and Electrical Engineering, International Conference on. 932-935. 10.1109/ICCEE.2008.132.
6. Ziani, B & Ouinten, Youcef. (2011). Improving Star Join Queries Performance: A Maximal Frequent Pattern Based Approach for Automatic Selection of Indexes in Relational Data Warehouses. Proceedings - 2011 International Conference on Internet Computing and Information Services, ICICIS 2011. 119 - 122. 10.1109/ICICIS.2011.36.
7. Junping Sun and Jiman Hong. 2013. Index selection: a query pattern mining based approach. In *Proceedings of the 2013 Research in Adaptive and Convergent Systems* (RACS '13). ACM, New York, NY, USA, 100-106. DOI: https://doi.org/10.1145/2513228.2513253
8. “TPC Benchmark C”. Transaction Processing Performance Council, 2010.

<http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf>

<http://www.tpc.org/tpcc/>

1. C. Turbyfill, C. Orji and D. Bitton, "AS3AP-a comparative relational database benchmark," *Digest of Papers. COMPCON Spring 89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage*, San Francisco, CA, USA, 1989, pp. 560-564.

doi: 10.1109/CMPCON.1989.301993