Technology Review

HP: Big Data Analytics
Group 13

Oregon State University
CS 461
2016-2017

Prepared By:
Nic Desilets, James Stallkamp,
and Nathaniel Whitlock

November 13, 2016

Abstract

Three pieces were identified for this project: SQL development environment; Parallel and Partitioning; and Performance Reporting. From each of those integral pieces, three or more technology options were evaluated and compared. The resulting decisions will act as the initial path forward until there is a need to consider one of the other options.

Add more content here....

CONTENTS

1 Introduction 2

2 Integrated Development Environment 2

2.1 Purpose . . . 2

2.2 Existing Technologies . . . 2

2.2.1 SQL Plus . . . 2

2.2.2 Oracle SQL Developer . . . 2

2.2.3 Toad . . . 2

2.3 Conclusion . . . 2

3 Parallelism and Partitioning 2

3.1 Purpose . . . 2

3.2 Database Technologies . . . 3

3.2.1 Oracle . . . 3

3.2.2 PostgreSQL . . . 3

3.2.3 Redshift . . . 4

3.3 Conclusion . . . 4

4 Toolkits 4

4.1 Purpose of a Toolkit . . . 4

4.2 Existing Toolkits . . . 5

4.2.1 Oracle Enterprise Manager . . . 5

4.2.2 SQLd360 . . . 5

4.2.3 Snapper . . . 5

4.3 Creating a Toolkit . . . 5

4.4 Conclusion . . . 6

References 6

1 Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin et suscipit eros. Etiam arcu orci, maximus ac gravida varius, pulvinar vitae sapien. Ut malesuada pulvinar nisl. Nullam sollicitudin neque eros, at semper velit feugiat quis. Nam vulputate metus sit amet leo auctor, eu tempus sem finibus. Cras blandit dui a porta pellentesque. Integer et sollicitudin massa. Ut blandit egestas viverra.

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Donec tristique vestibulum ullamcorper. Vivamus pellentesque felis id erat auctor sagittis. Nulla tincidunt vestibulum vehicula. Duis ac blandit neque, eu commodo dui. Maecenas lacus mi, tristique eget ultrices a, maximus a tortor. Pellentesque molestie lobortis lacus.

Phasellus rhoncus tincidunt quam sit amet cursus. Pellentesque dictum ac ex ut rhoncus. Sed eu enim vitae urna elementum condimentum id sit amet augue. Aenean sodales volutpat odio sit amet suscipit. Vivamus dictum dolor a sapien scelerisque, eu euismod est suscipit. Donec suscipit non massa id semper. Sed vitae nulla non lorem faucibus vestibulum. Mauris vestibulum diam arcu, in ullamcorper metus blandit vel. Cras consectetur, magna ut tempor luctus, nulla arcu accumsan felis, quis egestas diam turpis vitae lorem. Ut sodales lacinia malesuada. Morbi molestie, risus eget tincidunt mattis, augue enim condimentum mi, a porta quam justo quis sapien. Aliquam ac pellentesque arcu.

Nunc scelerisque velit at odio elementum iaculis sed luctus risus. Integer fringilla lacinia convallis. Curabitur convallis est ac dui semper fringilla ut ut lorem. Fusce orci ante, varius quis fringilla et, placerat sed dui. Integer porttitor pharetra justo eget vulputate. Vestibulum dui justo, placerat in interdum ac, tempus ac felis. Pellentesque gravida porttitor lectus vitae dictum. Fusce pulvinar rhoncus odio eget porta.

2 Integrated Development Environment

2.1 Purpose

2.2 Existing Technologies

2.2.1 SQL Plus

2.2.2 Oracle SQL Developer

2.2.3 Toad

2.3 Conclusion

3 Parallelism and Partitioning

3.1 Purpose

When datasets become large or the aggregate functions used to extrapolate statistics from a table are too complex the database can take a long time to return anything useful. Techniques such as parallelism and partitioning can be implemented in order to increase computational efficiency and reduce the observed time it takes to return a query.

Parallelism refers to the concept of dividing a large task, such as the processing of a complex query, and breaking it up into specified chunks in order to distribute the work among additional nodes or threads. During the optimization process, the steps of a query are ranked by estimated cost in order to generate an efficient execution plan. Logically grouped steps from the execution plan can then be processed on different nodes in order to reduce the time it takes to

complete. The degree of parallelism refers to the number of processors which are utilized during program execution. As the number of processors splitting the workload increases, the time needed for computation decreases.

Partitioning refers to the logical separation of ranges in a table, index, or indexed table. The general idea is to make subdivisions that would assist in the processing of a query. If you had sales data for several years and wanted to calculate the average sales per month, the table of data could be logically partitioned by month and processed on separate nodes. By dividing the dataset among many nodes, each table partition can be processed in concert leading to a quicker return of target data.

3.2 Database Technologies

3.2.1 Oracle

The Oracle Corporation has been around for over 39 years and is largely known for database software which got its name from a CIA-funded project that one of the founders had previously worked on. The current version of the Oracle database software is 12.1.0.2 and was released in July 2014.

According to Oracle documentation, Parallel queries and subqueries are handled by evaluating two components: the degree of parallelism (DOP) and the decision to parallelize. The degree of parallelism is determined most often by the table being modified by an INSERT, UPDATE, DELETE, or MERGE command. In order to use parallelization you must include a statement level hint to indicate the desire to parallel process, the objects referred to need a parallel declaration associated with them, and Auto DOP must be enabled.

Partition Methods

- Range Partitioning
  - Maps data based on ranges of values of the partitioning key, this is the most common type of partitioning and is often implemented with dates or times.
- Hash Partitioning
  - Maps data based on based on the Oracle hashing algorithm which is applied to the partitioning key. All partitions have even distribution of data making them similar in size.
- List Partitioning
  - Maps data by applying a list of specified partitioning key values that should go in each partition.
- Composite Partitioning
  - After applying one of the above partitioning schemes, another partition scheme is applied to each partition. The result is subpartitions which represent a logical subset of the data.

3.2.2 PostgreSQL

PostgreSQL is an open source object-relational database system that was developed by a team of volunteers. Although this database software is free it is feature rich and standards compliant. It is also highly customizable allowing stored procedures written in more than a dozen programming languages to be executed. Some potential drawbacks of the software include: installation and configuration difficulties; psql command line different than traditional commands; sheer number of features can take years to learn.

PostgreSQL has a feature called parallel query which evaluates an optimizer serial query plan in order to check if the query contains steps which can processed independently of each other. Though this is not true for small simpler queries, more complex queries have steps which can be split off, processed separately, and recombined. Queries that benefit from this feature the most occur when a query reaches out to a large amount of data for calculations and only returns a few rows. Within the PostgreSQL system each separate node or process used during parallelization are

referred to as workers. The number of workers the planner has available is determined by an internal variable called max_worker_processes, this variable is initialized at server start with the default value of eight but it can be manually set. It is possible to utilize less than the maximum number of workers and still have an optimized parallel query plan.

Basic table partitioning is supported by PostgreSQL. The partitioning is implemented by creating a master table, as well as multiple child tables that inherit from the master table. Table constraints are then set in order to define what data is placed in each partition. The data from the master table is then copied to each appropriate partition and given a partitioning key, the result is smaller tables which are subdivided by a declared expression.

3.2.3 Redshift

Redshift is a cloud based data warehousing product from Amazon Web Services (AWS) . Since it is cloud based, there are no upfront fees from hiring experienced people to set up servers or configure the system. According to the AWS site, Redshift is a petabyte scale data warehouse which is fault tolerant with built in encryption. They also claim that there is no need for tuning to maintain performance and speed since the system is self managed. **Redshift based on PostgreSQL 8.0.2**

Unlike traditional row-oriented databases, Redshift uses columnar or column-oriented storage. Columnar storage works by reading only the values from a row which are needed to complete the transaction. This means if a query is executed that only needs to evaluate the value of one out of four columns, then only the value from that column can be loaded thus drastically reducing the disk I/O. With row-oriented storage, each of the rows containing the values of interest would have to be read into memory.

Redshift is a classified as a shared nothing or Massively Parallel Processing system. Shared nothing refers to the fact that no single node has a complete view of the database, and therefore cannot communicate with each other by sharing data during processing. There are multiple options for distribution styles which work to redistribute rows to compute nodes when a join or aggregation is needed. Though there are settings for some system configuration, the distribution of work over the compute nodes is automatically handled in order to create a simple experience for the user. There is limited...

Redshift does not support the partitioning concept, rather it relies on the distribution style option selected by the user as well as distribution/sort keys. **Elaborate...**

3.3 Conclusion

Add content here...

4 TOOLKITS

4.1 Purpose of a Toolkit

The main purpose of a database toolkit is to assist a Database Administrator (DBA) with analyzing the performance of queries made to a database. The different types of toolkits that can be used with Oracle Database 12c vary widely from PL/SQL queries that can be run directly in the database to complex, fully featured web applications. Likewise, the use cases for these toolkits can range from quick and easy query statement diagnostics all the way to automating the analysis and optimization of parameters within the database to a particular SQL query or multiple queries.

When analyzing queries, being able to effectively quantify and visualize multiple aspects of a query statement is important to a DBA. Some of the information gathered might include the time it took for a query to run, how much memory was allocated to a process handling a query, and the amount of disk input and output operations. This information can then be combined and analyzed in order to identify problematic areas within the database with a particular query or multiple queries.

For example, if you have a particular slow running query, you might use a toolkit to analyze what is happening when the query is being run. After analyzing the output of the toolkit you then find out that the way the database is allocating memory is suboptimal for the query thus resulting in a lot of swap space usage. With this knowledge you can then adjust parameters within the database to allocate more memory to the processes that handle queries.

4.2 Existing Toolkits

4.2.1 Oracle Enterprise Manager

Oracle Enterprise Manager (EM) is already included in the Enterprise edition of Oracle Database 12c. It is intended to be a one stop, all in one application that can be used for analyzing and optimizing query performance. Unlike other toolkits which are comprised of PL/SQL queries that can be executed in the database, EM is a full fledged web application that is packed with many features beyond analyzing query performance.

Some features that are included in EM are: the ability to manage and maintain several databases from one web application; automating repetitive tasks involved with maintaining databases; testing changes before rolling out to production; diagnosing performance problems and automated database tuning. Additional examples of what EM can do are it can determine if indexes will be helpful for performance of a particular table and analyzing sql query statement structure for potential performance issues.

One particularly useful feature of EM for analyzing query performance is a subcomponent called Active Session History (ASH). The ASH tool runs in the background and samples each active session within the database once per second. Each sample for each session contains detailed info about what resources are being used by the session and how the session is using it. This is especially useful for identifying bottlenecks and other performance issues of running SQL queries.

4.2.2 SQLd360

SQLd360 is a tool that is written by Mauro Pagano, an ex-Oracle Database Engineer of about ten years, which analyzes SQL queries. Unlike EM, it will only analyze a single specified query and will not perform any database optimization for you. Instead, it is only intended for performance analysis of a query. SQLd360 is very similar to EMs ASH tool as it operates under the same underlying principle of periodically sampling the session that the query is running in to gather performance metrics. This information can be specified to be output in either html, text, csv, or graphical charts. One important difference is the SQLd360 is provided as a completely free tool while EMs ASH requires additional licensing from Oracle. This makes it particularly useful for users who do not have the required licensing but still need to obtain key metrics from session data. After the tool is finished running, it will dump all of the collected information into a zip file for later analysis.

4.2.3 Snapper

Snapper is another toolkit written by Tanel Poder, an Oracle Certified Master DBA, which also analyzes SQL queries. Similar to SQLd360, it is also provided completely free of use and does not require additional licensing from Oracle. It also operates similarly to SQLd360 in that it uses the databases session information to extract metrics information from running queries. It will also not provide any recommendations about how to optimize queries or database parameters. In addition to sampling data from query sessions, it also gathers data from the V$ and X$ tables within the database which both house extra information regarding query performance. Snapper combines this information together in order to provide a data rich visual representation of how exactly a query is running in the database.

4.3 Creating a Toolkit

Of the three toolkits mentioned above, it might sound like Oracles EM is the obvious way to go since it is very feature rich and is even capable of automatically optimizing SQL queries and database parameters for you. However, in our

case we are still going to develop a toolkit of our own that will provide similar functionality. While avoiding reinventing the wheel is typically important when it comes to designing and creating software to prevent wasting time, in our case it makes sense for us to design a toolkit of our own that provides similar functionality to the three toolkits described above. The reason for this is, is that it will force us to become familiar with the inner workings of the database. While we could just use of the already existing toolkits to perform query performance analysis for us, we would probably not be as effective in modifying important database parameters when it comes to memory management, database table partition design, and parallelism. The expected outcome of developing our own toolkit is that not only will we have a custom made toolkit tailored to our needs but we should come out much more knowledgeable about Oracle 12c. This in turn should allow us to become more proficient in customizing the parameters of the database to increase query performance.

4.4 Conclusion

Add content here...

REFERENCES

[1] Oracle Company. *Query Optimizer Concepts*. [Online]. Available: https://docs.oracle.com/...
[2] Oracle Company. *How Parallel Execution Works*. [Online]. Available: https://docs.oracle.com/...
[3] Oracle Company. *Partitioning Concepts*. [Online]. Available: https://docs.oracle.com/...
[4] Postgresql Volunteers. *Documentation: 9.1: Partitioning*. [Online]. Available: https://www.postgresql.org/docs/...
[5] Postgresql Volunteers. *Parallel Query*. [Online]. Available: https://www.postgresql.org/docs/...
[6] Amazon. *Choosing A Data Distribution Style*. [Online]. Available: http://docs.aws.amazon.com/redshift/...
[7] Amazon. *Tuning Query Performance*. [Online]. Available: http://docs.aws.amazon.com/redshift/...
[8] Amazon. *Unsupported Postgresql Features*. [Online]. Available: http://docs.aws.amazon.com/redshift/...