Conducting a Data Warehousing Benchmark

An Oracle White Paper
August 2006
Conducting a Data Warehousing Benchmark

Executive Overview .......................................................................................... 3
Introduction ....................................................................................................... 3
Decide whether you really need to do a benchmark and resource it appropriately ................................................................. 3
Have a clear goal in mind for the benchmark .................................................. 4
Avoid Comparing apples to oranges ............................................................... 5
Use real data ..................................................................................................... 6
Testing the right scalability ........................................................................... 7
Benchmark all key aspects of your workload ................................................. 8
Cost matters ..................................................................................................... 9
Define ground rules but be flexible ............................................................... 9
Learn from the benchmark ............................................................................ 11
Plan ahead ....................................................................................................... 11
Conclusion ..................................................................................................... 12
EXECUTIVE OVERVIEW

This paper describes how best to design and run a benchmark for a data warehouse environment. It covers what should be the motivating factors for doing a benchmark and outlines the main concepts or ideas to bear in mind when designing the benchmark test specifications. Finally it explains what you can expect to learn from the benchmark test results. By following the simple steps outlined in this paper you can avoid the obvious pitfalls that most benchmarks run into and have a successful and knowledge gaining evaluation experience.

INTRODUCTION

When your company starts a new data warehouse project, an evaluation of data warehouse vendors may be appropriate. As part of this evaluation, you may choose to do a ‘benchmark’: a technical test where two or more vendors are asked to run a test that you design, for the purposes of comparing the performance, scalability, and/or functionality of the vendors.

This paper is designed to help guide you in the process of designed and managing a data warehouse benchmark. Many of these statements may seem fairly obvious, yet there are unfortunately too many examples where even the most obvious guidelines have been overlooked.

DECIDE WHETHER YOU REALLY NEED TO DO A BENCHMARK AND RESOURCE IT APPROPRIATELY

The most fundamental question here is: how will the results of a benchmark influence your decision-making process for selecting a data-warehouse vendor?

If the decision-makers for the data warehouse platform are not going to take the benchmark into consideration, then the benchmark is not worth the cost and expense. This seems almost self-evident, but there is a common misconception is that "if I am doing a data warehouse evaluation, then I should run a benchmark". There are many, many valid reasons for doing a data warehouse benchmark, but you must ensure that you can identify exactly why you want to run a benchmark and what you expect to learn (see below).

It is also a good idea to determine, before the benchmark, how much influence you believe the benchmark will have on the decision process.
If you determine that a benchmark should be part of your data warehouse evaluation process, then you need to allocate resources to the benchmark effort like any other IT project. You should plan on 3-12 person-months to create a reasonable benchmark specification for an enterprise data warehouse (the low-end timeframe would typically be a company that already has a data warehouse and thus has a good idea of requirements and workloads, while the longer time frame would be to define a benchmark for a new project, where there is not as much experience that can be leveraged). If you do not have the appropriate resources to build a robust benchmark specification, then you should not run a benchmark. A poorly-defined benchmark yields little useful information in a data warehouse decision process, but produces a huge amount of frustration.

HAVE A CLEAR GOAL IN MIND FOR THE BENCHMARK

Every benchmark should have a well-defined, quantitative success criteria. Before creating a detailed benchmark specification, you need to decide what are the most crucial technical requirements of your data warehouse, and focus your benchmark on those areas.

If you can't describe what you hope to learn in the benchmark in a few simple sentences, then you are probably not going to be able to build a benchmark specification which delivers useful information. Here are some examples of the scenarios that should drive benchmark specification:

- "We are acquiring a company, and our data warehouse is going to grow by 50% over the next year. We need to see a solution that can handle our current workload, with the same performance, but with 50% more data. And we need to know that we can handle similar growth for future acquisition."

- "Our legacy mainframe data warehouse is being decommissioned. We are planning to roll out a new data warehouse, with a new enterprise data model. We need to understand whether our new data warehouse platform will support the current mainframe user's service levels, while simultaneously delivering support for three new applications"

An attitude of "We want to see what each data warehouse solution look like ... we'll know a good solution when we see it" is a recipe for disaster in a benchmark. The clearer that the goals can be quantitatively expressed, the more likely the benchmark results will address your core needs. Let's look at an example of how this might get expressed in a benchmark specification. A company determines the key criteria for the data warehouse is query performance (a typical scenario), but they also expect significant growth and want to ensure that their availability and load SLA's will continue to be met. They define a benchmark that covers all of those areas, and provide the following weightings a part of their benchmark specification:
In the absence of this type of quantitative success criteria, the vendors participating in the benchmark will choose to run the benchmark to focus on the areas that they feel most improves their chance to win a benchmark. However, this may or may not be in your best interest. If you do not define the benchmark criteria carefully, you may get to the end of the benchmark and discover that you don't have the information that you expected or needed from the benchmark. The benchmark results from different vendors may not be completely comparable since one vendor has focused more on scalability and loading, while another focused on performance.

**AVOID COMPARING APPLES TO ORANGES**

A 'benchmark', as we have defined it in this paper, is a test to help evaluate potential new data warehouse solutions. Almost by definition, the term ‘benchmark’ is a test that is constructed to provide accurate (and thus fair) comparisons. Typically, benchmarks provide an opportunity for every vendor to demonstrate their capabilities in the best possible light. Assuming that your goal is to understand each vendor's potential capabilities, then you want to strive to make the benchmark fair.

The most common apples-to-oranges mistake is to compare a proof-of-concept conducted in a performance lab by one vendor, to a seat-of-the-pants project resourced by internal staff conducted on an internal test/development platform. The outside vendor will win the benchmark every time in those scenarios (and it shouldn't be surprising that many vendors make it a common sales practice across the industry to propose such "benchmarks"). Very little can be learned from such a "benchmark" other than the well-known fact that performance experts, running new software on hardware with faster CPU's and more IO bandwidth, will beat the pants off of your company's in-house development machine staffed on a part-time basis by your DBA's.

If you want to validate whether an alternative solution is faster than your current production system, then that is a valid test to conduct. But you should recognize that such a test is a very different exercise than a competitive benchmark: you are validating how well a solution can run your current workload, but meanwhile, you have essentially declined to evaluate your current solution, since it is likely that, by running in a performance lab on newer software and new hardware, your current solution would be an order of magnitude faster running a test workload.
Another apples-to-oranges situation, which commonly arises, is when a benchmark workload has not been precisely defined. Each vendor will try to take advantage of this situation, to run a workload which best show cases their strengths and will hide their weaknesses. If a benchmark specification says, “We would like to test DML and end-user queries in this benchmark”, it is likely that each vendor will interpret this statement differently. For example the results could be presented as:

- A single user runs one DML job on an otherwise idle system followed by a multiple user query test.
- A DML job run concurrently with multiple user queries however the DML job does not operate on any of the data used to satisfy the end user queries.
- A DML job run concurrently with multiple user queries however the user query results are not guaranteed to be correct as dirty reads were allowed.
- A DML job run concurrently with multiple user queries that guarantees the query results will be correct due to read consistency.

You need to explicitly state exactly which of these tests you want conducted so each vendor cannot misinterpret it for their benefit.

**USE REAL DATA**

Artificially generated data should never be used for a benchmark. If you want to learn about how a data warehouse system is going to perform on your workload, you must use your own business data and the size of the data set you use should be comparable to what you will expect in production. However, such an approach can lead to a couple of potential complications, both business and technical.

The technical complication is typically, if you want to conduct a multi-TB data warehouse benchmark, it’s infeasible to extract and ship TB’s of data to vendors (and you may not even have TB’s of historical data available). The most common approach is to create an extract of a slice of real data for the largest tables (typically one month), and replicate that data (possibly with some minor transformations) to the desired data volume, so that 100GB of data representing one month is expanded to 3.6TB of data representing three years. Smaller tables, and in particular tables containing reference data such as customers or products are typically provided in full. This approach is widely used for large-scale DW benchmarks.

The business complication is generally around the sensitivity of the data. Almost by definition, a data warehouse contains sensitive data about customers, about financial information, and about internal metrics. All benchmarks are conducted under non-disclosure agreements, but even then there is data that cannot be utilized for a benchmark. The typical approach is to remove or scramble individual columns. Sensitive data (like customer names or financial measures) are never join
keys, so many of the key performance characteristics of complex queries can be maintained if the sensitive columns are replaced with appropriate dummy data.

While the packaging of the data set may be one of the most time-consuming components of creating a benchmark specification, it is also one of the most crucial steps in creating a workload which will accurately model your data warehouse workload and must not omitted.

**TESTING THE RIGHT SCALABILITY**

Data warehouses grow. Over time, they have more users and more data. Scalability thus should be included as part of any comprehensive data warehouse benchmark.

There are three inter-dependent dimensions of scalability that you should consider: the number of users, the amount of raw data, and the size of hardware configuration. Your benchmark test should be based upon how you anticipate your workload to change in the future.

The simplest area to include in a benchmark is user scalability. A benchmark's query workload should never solely consist of running queries as a single-user. Data warehouse benchmarks should measure user concurrency: for example, let's suppose that you have defined 20 SQL queries for your benchmark. You should plan to have for example, 15 concurrent users, each running all 20 of those queries one after the other. In order to test scalability, you could separately measure performance for 30 concurrent users, and compare that to the performance of 15 concurrent users. This simple test will give you some idea how each data warehouse platform will behave as the number of users increases.

The second area of scalability is data scalability. Most data warehouses grow over time, by storing more historical data. Thus, you could consider running query tests for 3 years of data and 5 years of data. Data scalability could also be measured by maintaining 3 years of data, but growing each year's worth of a data by 50%; this is not as easy an operation to set up and is thus not often used.

Finally, you might want to consider how the addition of hardware resources might impact your data warehouse. Typically, adding more hardware involves the addition of more CPU's (and nodes), in conjunction the addition of further IO bandwidth. There are three basic tests:

- add more hardware and measure the impact on performance on the same data set and workload.

- add more hardware to accommodate additional users. Measure the performance for larger numbers of concurrent users after adding more hardware.

- add more hardware to accommodate extra data. Measure the performance on the same workload over a larger data volume.
In general, the latter two scenarios are more realistic.

Scalability should be tested in a data warehouse benchmark, and the key is that you should make sure to test the scalability scenarios that match your anticipated data-warehouse growth.

**BENCHMARK ALL KEY ASPECTS OF YOUR WORKLOAD**

It is unfortunately not unusual to see a ‘benchmark’ that consists of a small number of queries, without any measurements of the performance for data-loading, or for concurrent/mixed workloads. These benchmarks are rarely an accurate way to evaluate vendors, as inevitably there will be some critical aspect of your day to day operations, which was not tested, or the select vendor can not perform or performs badly. These benchmarks are also often the easiest for a single vendor to manipulate as they can ensure their weakest area is not tested.

Although it means both more work on your side to prepare the benchmark specification and more time to run the actual benchmark it is important that you test three key areas:

- **ETL** – It is critical you do a data load and transformation test. The test should be as close to your expected production scenario as possible, for example if you load 50, 1GB files into staging tables during a 4 hour window then you need to test this exact scenario. You won’t need to supply 50 distinct files, just one file will do, the vendor can duplicate the file to build the required 50 files.

- **Summarization of data for analysis** – Most data warehouses leverage some form of aggregation, be it summary tables or OLAP cubes. The time it takes to build and maintain these aggregations is a vital part of any data warehouse system. If these aggregations are not kept current end-users will either get out of date data in response to their queries or have to calculate the aggregations on the fly for each query, which can be expensive both in terms of time and processing power. As with the ETL, the best way to test the summary creation and maintenance is to mimic what you expect to do in production.

- **End user queries** – Your data warehouse will have some number of end-users who submit SQL queries to the system by either running predefined or ad-hoc reports of differing complexity. You should spend time analyzing the number of concurrent queries you expect on the system and the complexity of these queries. It is quite common to break the query mix into three types of queries; simple, medium, and complex. These categories should be used to distinguish how much system resources a query will require and NOT the simplicity of the SQL statement. A SQL statement such as `select * from main_fact_table;` may appear to be a simple but can be very resource intensive as it requires a full table scan of
the largest fact table which could be multiple terabytes in size. It is also important to ensure the queries you choose don’t just make the CPUs and disks busy, they should emulate your business users requirements. For example, if your users will be doing a lot of complex analytic queries you should test for those to ensure the vendors have the required functionality and can cope with multiple concurrent requests of this nature. You should also prepare a set of surprise queries. This set of queries will not be given to the vendors at the beginning of the benchmark. You should only give the queries to the vendors after all other testing has been completed and no further system tuning is allowed. These queries should represent the ad-hoc queries you expect to see on your data warehouse. Having the vendors run them without tuning for them will give you the best possible view of how each system will handle unexpected queries.

By testing all key aspects of your data warehousing operations you will get the most accurate evaluation.

**COST MATTERS**

You should benchmark configurations that fit within your budget, and you should benchmark configurations of similar costs between vendors.

This is one of the obvious statements that, sometimes, is easier said that done. For example, you may be partially constrained by the configurations available in each vendor’s performance labs in the timeframe of your benchmark. Or, it may be difficult to accurately gauge the end-cost of each solution at the beginning of the benchmark process, before any pricing negotiations have taken place.

However, benchmarks should still strive for having comparable configurations. If one vendor runs on a solution that is twice as expensive as another solution, it can be difficult to interpret the performance results (intuitively the more expensive vendor should be twice as fast in that scenario ... but that assumes that the cheaper vendor, if given twice as much budget, would only be twice as fast, which is not necessarily true).

Moreover, there’s not much reason in benchmarking using the $2M configuration that one vendor happens to conveniently have available, when at the end of the day your budget will only be $1.5M.

All of this seems like another obvious statement, yet it somehow gets overlooked. When your team has a deadline for delivering a data warehouse benchmark, tradeoffs will necessarily be made. Just make sure that those tradeoffs aren’t going to devalue the benchmark.

**DEFINE GROUND RULES BUT BE FLEXIBLE**

You should define ground rules for your benchmark. Here are some quick examples of the type of rules that might appear in a benchmark specification:
- The nightly load must complete in less than 4 hours. All data must be loaded into the database, and the maintenance of all indexes and other secondary structures must be complete.

- The syntax of the SQL queries may not be changed.

- Since this test only consists of 20 queries (while the production data warehouse will support thousands of different SQL queries), summary tables are not allowed.

The purpose of these ground rules to help ensure that all vendors are running the workload in a similar (and thus comparable) manner. These rules, and the enforcement of these rules, are an important part of the benchmark.

However, at the same time, one of your main goals of a benchmark is to learn about the best practices for each potential data warehouse solution. The rigid adherence to these rules may prevent a vendor from demonstrating capabilities that you would find useful in your production system.

How can you be flexible in a benchmark, but remain fair? Here are some recommended approaches:

- Prior to the actual benchmark, send the benchmark specification to vendors for review before any vendor actually starts the benchmark. If vendors want to submit technical changes (for example, changes to the schema or the SQL queries), then mandate that all other vendors review the proposed changes. This review process will identify areas of the benchmark that may be unreasonable or unduly restrictive, while ensuring that suggestions proposed by each vendor are for the benefit of the benchmark, rather than solely for the benefit of the vendor.

- Schedule "demonstration" time in the benchmark to allow vendors to show you functionality either not covered or not allowed by your benchmark specification that they believe will be valuable to you.

- If a vendor proposes a change to the rules during the benchmark such as changing the syntax of query, then require the vendor to run and submit the performance for the original SQL syntax, and permit the vendor to submit the performance for the modified SQL syntax as an addendum to their performance results. Only your own team can determine whether that SQL modification would represent an undue tuning burden in your environment, but this approach allows you to see both the shortcoming and strength of the vendor you are evaluating.

- Allow for the possibility that your benchmark specification may have deficiencies. When a vendor complains vehemently about a detail of the benchmark specification, they are no doubt unhappy with their own performance on that aspect of the benchmark. In some cases, the vendor may be acutely aware that they are at a competitive performance
disadvantage for the workload, and are trying to change the benchmark in their favor. In other cases, however, the vendor may have some tangible suggestions for improving the applicability and relevance of the benchmark. If your benchmark is designed in a way that goes against a given vendor’s best practices, then you should consider whether your benchmark specification should be corrected.

**LEARN FROM THE BENCHMARK**

It seems obvious, but it's often overlooked. When you conduct a benchmark, the hardware and software vendors will apply some of their most competent resources to your benchmark. Your DBA’s and architects should be involved in the benchmark, not only to ensure that the benchmark meets your need for evaluating data warehouse solutions, but also to learn best practices for the solution that you subsequently choose to implement. A good benchmark workload will be realistic enough that the tuning techniques for the benchmark can be directly applied to your future production data warehouse, but you can only do this when your data warehouse team gains an understanding of how the benchmark was set up and run.

You should dedicate at least two members of your technical staff to be onsite for at least one week of the benchmark for each vendor. You should ask for the parameter setting and scripts to configure, create, and manage the database, and use the information gained from the benchmark in your production systems.

You shouldn't give the sole responsibility of the benchmark to an ‘architecture’ team, while leaving your ‘administration’ and ‘development’ teams out of the benchmark process. The people who are going to be designing and running your new data warehouse are the most important people who need to learn from the benchmark. Even though those people may have important jobs supporting current production systems, they need to be given dedicated time to be involved in the benchmark.

**PLAN AHEAD**

Benchmarks are not projects that can be put together at the last minute, nor can benchmarks be put together on the fly.

Some companies try to save time by defining a benchmark on the fly. They start with one vendor and a loose set of requirements, then try to build out their test specification as the first benchmark is running. This has two potential risks: first, the benchmark specification will likely reflect the capabilities and strengths of the first vendor, and second, since the benchmark is being developed on the fly, it is likely that subsequent iterations of the benchmark won’t exactly reproduce the steps run in the first test.

A successful benchmark requires project-planning. Benchmarks necessarily require the coordination of multiple teams within your company, as well as multiple vendors. A successful benchmark is one that has well-defined goals and success-
criteria, and a well-defined technical specification that covers the key areas of your company’s data warehouse requirement.

A well-executed benchmark can provide a wealth of information, both addressing your key questions and likely turning up new information that you may not have expected to learn. With these guidelines, you can ensure that any benchmark effort that you embark upon will be successful.

**CONCLUSION**

With careful planning and adequate resources a data warehouse benchmark can be a very valuable tool to help with the evaluation of data warehouse vendors. By following the simple steps outlined in this paper you can avoid the obvious pitfalls that most benchmarks run into and have a successful and knowledge gaining evaluation experience.