Rhodes University

Department of Computer Science

Computer Science Honours Literature Review

Benchmarking Databases

By Samy Kabangu

Supervisor: Mr. John Ebden

Abstract

As most applications in the area of computer science and information technology industry seek for high performance, the database computing arena is not left behind. New database products are being developed with much concern about the speed of loading, modifying and retrieving stored data from the database. Furthermore, the database wars still continue unabated between vendors making spectacular claims in favour of their respective products in terms of performance. Database performance benchmarks provide means by which different database applications can be compared. An investigation into the critical benchmarking techniques and factors in tuning database management systems will constitute the focal point around which this literature review will be based. The transaction Processing Performance Council benchmarks will be investigated (TPC). As a case study, a commercial database product Microsoft SQL Server 2008 will be used on which to run the TPC-H benchmarks.

1. **Introduction**

Benchmarking a database is the process of performing well defined tests on that particular database for the purpose of evaluating its performance [Scalzo et al. 2008]. The Response time and the throughput are the two main criteria on which the performance of a database can be measured [Petkovic 2006]. Specific parameters and settings external as well as internal to the database management system need to be taken into consideration. These parameters include the hardware used to test the system, the internal configuration of the database engine, the operating system configuration as well as the database design and implementation [Burgess, 2009][Petkovic 2006][Watts et al., 2005]. All the parameters mentioned above play an important role in the overall performance of a database management system (DBMS). As such, particular attention will be given to them throughout this literature review.

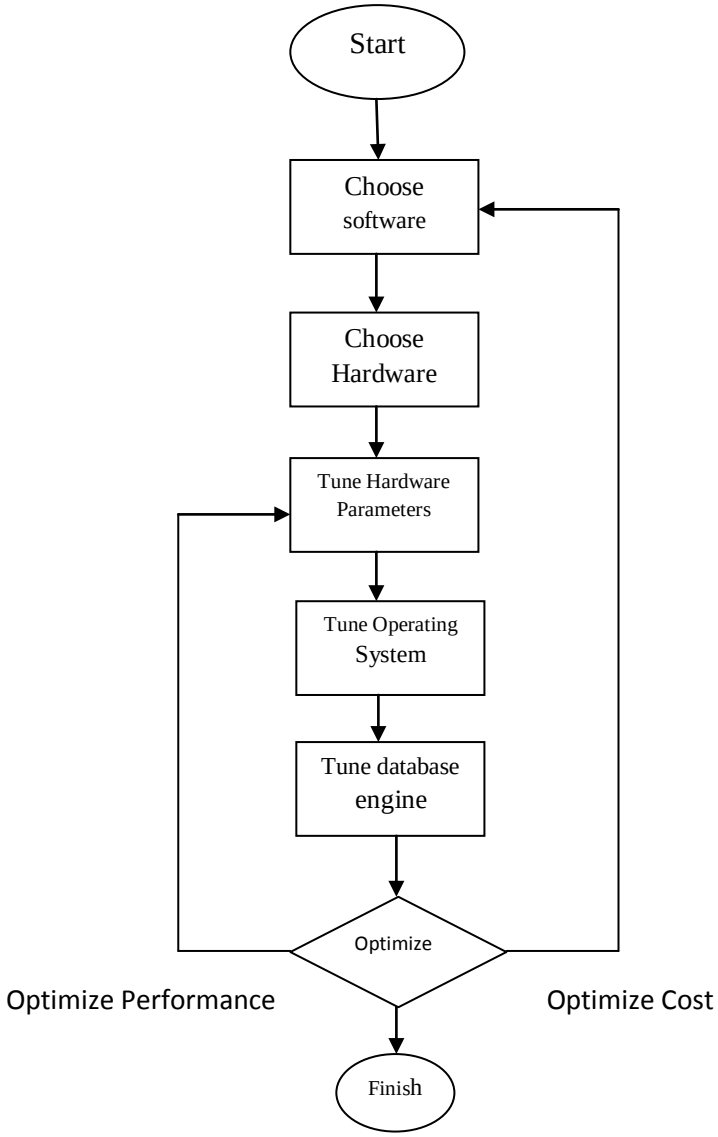
2. Significance of database benchmark tests

Benchmark test results facilitate means for cross platform comparisons of various database management systems by providing valuable information to database professionals on whether to utilise a particular database product. Within an organisation, the workload supported by a database system might increase as the business expands; Proactive benchmark scalability testing can be beneficial in preventing bottlenecks. [Scalzo, 2009] Furthermore, there might be a need of migrating from one hardware platform, or system software or database product to a newer version or release. Database benchmarks tests can be valuable by providing a proof of concept that facilitate the job the DBA to make an apple to apple comparison between different software releases [Oracle, 2009] [Scalzo, 2009]. Finally, Benchmarks tests promote innovation due to competition between hardware manufacturer, operating system developer and database vendors [Watts et al., 2005].

3. Benchmarking Process

Benchmarking is a difficult and never ending process that requires a lot of patience and discipline [Dietrich et al 1992]. As the benchmarking process is being executed, measurements about the overall performance of the system have to be collected as various key configurations parameters specific to the hardware, operating system and database management system are being altered if necessary in order to improve the performance of the system under test [Dietrich et al 1992]. We would like to mention that the TPC benchmarks are used to test the performance without any attempt to modify the configuration parameters of the system under test [TPC-H, 2008][Darmont et al., 2007]. Any benchmark test performed using the TPC benchmark suites with the purpose of improving the performance of the system under test is qualified as “special” [TPC-H, 2008].

The flow chart below illustrates the steps involved in the benchmarking process [Dietrich et al., 1992]:



4. Types of database Benchmarks

Benchmarking processes can be classified into three types: Industry-standard benchmark, vendor and customer-application benchmark [Dietrich et al 1992].

4.1 Customer-application benchmarks

Customer application benchmark is the benchmark performed within an organisation. They have the advantage of meeting the specific requirement of the organisation in terms of workload and hardware implementation but they are costly and time consuming endeavours usually difficult to undertake [Watts et al, 2005].

4.2 Industry standard benchmarks

Industry standard database benchmarks were developed to provide a cross platform comparison among various database products in terms of performance and prices. The published performance test results are measured depending on specific database workload types.

The most common workload types are:

- Online Transaction Processing (OLTP),
- Online Analytical Processing (OLAP); eg: An online business intelligence system workload against which users submit queries to answer complex business questions [Wasserman et al., 2004].
- Decision system support

The decision support workload type will be of particular interest in this literature review since it is implemented under the Transaction Processing Performance Council (TPC) benchmark suite "TPC-H". It is also the only TPC benchmark suite that has no clients and no network components [Watts, 2005]. So the TPC-H benchmark suite appears to be the most suitable one for this research due to the limited resources that we possess at hand.

4.2.1 Reservations of industry standard benchmarks

Though they provide means of knowing the performance of database products, their adoption might be somewhat difficult in that:

- They simulate real world workload which might not reflect the actual workload of a particular application of interest [Hoste et al 2007] [Oracle, 2009].
- They are performed on specific hardware and operating system which makes their duplication not feasible especially for custom applications that do not match the platform requirements on which the benchmark tests were run.
- It is also difficult to compare database systems run on different hardware platforms because of the different machine architecture under which they are manufactured [Oracle, 2009].
- Database vendors use techniques such as preloading data and the SQL execution plans into memory (RAM) in order to avoid disk I/O access overhead so as to improve the benchmark performance [Burleson, 2002].

4.2.2 Transaction Processing Performance Council (TPC)

The TPC is a non–profit corporation with the mission of defining transaction processing and database benchmarks and publishing to the industry verifiable TPC performance data [TPC 2009]. The term transaction viewed from the business perspective is regarded by the TPC as a commercial exchange of goods, services or money. As a computer function, "transaction" refers to a set of operations comprising disk read/writes, operating system calls, or some data transfer from one subsystem to another [TPC 2009].

The TPC provides different benchmark suites designed according to specific workload type and applications requirements. The TPC benchmark suites currently valid are given in the table below [TPC 2009]:

TPC Benchmarks	Workload and applications Types
TPC-App	• An application server and web services benchmark • Focuses on the performance capabilities of application server systems
TPC-C	• Simulates an applications where a population of users executes transactions against a database (OLTP)
TPC-E	• The new On-Line Transaction Processing • Is scalable, the workload generated can be varied to represent the workload of different-size businesses
TPC-H	• Decision Support benchmark consisting of ad hoc queries and concurrent data modifications.

4.2.2.1 TPC-H

The TPC Benchmark™H (TPC-H) is a decision support benchmark consisting of a suite of business oriented queries and concurrent data modifications [TPC-H 2008]. The queries and the data populating the database have been selected to have a broad industry-wide relevance [TPC-H, 2008].

The TPC-H Benchmarks simulates decision support system that [TPC-H, 2008]:

- Examine large amount of data;
- Execute queries with a certain degree of complexity;

- Give answers to critical business questions (operation).

Due to the ad hoc nature of the TPC-H queries, their execution time can be very long. That makes it difficult for the database administrator to optimize the database system as opposed to applications such as OLTP where the nature of queries as well as that of the workload is known in advance [Floyd, Meikel, 2000]. The properties of the TPC-H as reported by the TPC Benchmark™H standard specification Revision 2.8.0 are given as follows [TPC-H 2009]:

- Give answers to real-world business questions;
- Simulate generated ad-hoc queries
- Are far more complex than most OLTP transactions;
- Include a rich variety of operators and selectivity constraints;
- Are executed against a database complying to specific population and scaling requirements;
- Generate intensive activity on the part of the database server component of the system under test;
- Are implemented with constraints derived from staying closely synchronized with an on-line production database.

The TPC-H operations are modelled as follows [TPC-H, 2008]:

- The database is continuously available 24 hours a day, 7 days a week, for ad-hoc queries from multiple and users and data modifications against all the tables, except possibly during infrequent maintenance;
- The TPC-H database tracks, possibly with some delay, the state of the OLTP database through on-going refresh functions which batch together a number of modifications impacting some part of the decision support database;
- Due to the world-wide nature of the business, data stored in the TPC-H database, the queries and the refresh functions may be executed against the database at any time, especially in relation to each other.
- To achieve the optimal compromise between performance and operational requirements, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and refresh functions.

Since multiple end users may queries and modify the data concurrently against all tables, the TPC-H emphasises on the applicability of the ACID proprieties of a transaction. The ACID proprieties as follows [Coronel, 2002]:

- Atomicity: The System under test must guarantee that transactions are treated as single indivisible unit of work in that the results due to the execution of a given transaction must either be committed or rolled back. As an example, a given credit sale may require a minimum of three of the database operations given below to execute simultaneously or neither of them to execute:
    1. An invoice is created for the sold product;
    2. The product's inventory quantity is reduced;
    3. The customer accounts payable balance is increased by the amount the amount written on the invoice.
- Consistency: Requires that any execution of a transaction to transform the database from one valid state to another. A transaction can only be legal if it observes user defined integrity. Consistency is guaranteed by a database system in which referential and entity integrity checking is implemented. In the example below, given two relations:

    StudentsDetails (Name (Primary Key), Id, Address)
    StudentsMarks (Name (Foreign Key), ExamMark)

    A user defined integrity constraint could be: A student record can only exits in StudentsMarks table if there is already an instance of the same student record in the StudentsDetails table (A student can only be allowed to write an exam if he/she has registered with the university).

- Isolation: The system under test (database) must guarantee the invisibility of a given transaction from other transactions in that the data used during the execution of a transaction must not be used by an another transaction until the first one is completed. Locks are used in database system to prevent any breach of isolation such as the "lost update" problem.
- Durability: Assures that once committed the results of a transaction to be permanent even in case of a system failure. Logs and temporary files implemented on disk can be used to assure durability.

The TPC-H benchmark defines different sizes of the database according to specific scale factor as given below:

1GB, 10GB, 30GB,100GB, 300GB, 1000GB, 3000GB, 10000GB, 30000GB

The scale factor of 1(1GB) is the minimum required for a test database. Any database size not mentioned is not permitted by TPC. This requirement is meant to encourage comparability of results at low end and to ensure a substantial actual difference in test database sizes [TPC-H 2008]. The TPC-H Benchmark models the analysis end of the business environment where trends are computed to support decision making of sound business decisions [TPC-H, 2008].

4.2.2.2 Database design and implementation

The TPC-H benchmark database has been designed to be in the third normal form [Floyd, Meikel, 2000]. That is, it has the following properties [Coronel, 2002]:

- All the key attributes are defined;
- There are no repeating group in the tables;
- All the attributes are dependent on the primary key;
- No attribute is dependent on only a portion of the primary key;
- It contains no transitive dependencies

The entity relationship model is given by the schema below:

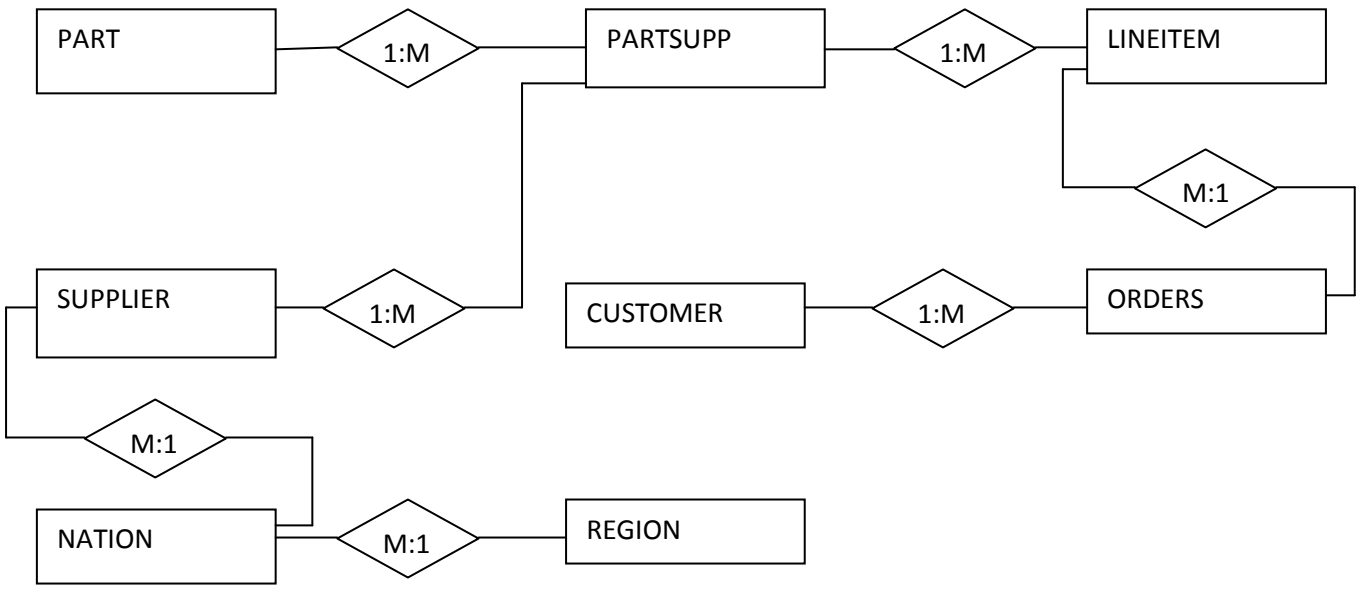


Figure 1: TPC-H benchmark database schema

The database consisting of eight tables is populated using a designed program that comes with the TPC-H benchmark suite called DBGEN. The maximum cardinality supported by each table is dependent on the scale factor used. The tables “SUPPLIER” and “LINEITEM” contain about 83% of the total data stored in all the tables [Floyd, Meikel, 2000].

It is relevant for us to mention that according to Shao et al [2005] “TPC benchmark kits for most state-of-the-art database systems are not readily available” and writing and tuning the benchmarks to meet the required specifications of the workload of interest on a given database system may require over six months of experimentation even by trained database system managers.

4.2.2.3 Workload

The TPC-H benchmark workload consists of a database load, the execution of 22 queries read only query running on single as well as in multiple users mode and two batch update statements (RF1 and RF2) [Floyd, Meikel, 2000] [Shao et al, 2005].

The Database load consists of constructing the test database. It includes the process of loading the data into the database test, the creation of tables, indices, the definition and validation of constraints, the collection of statistics and the configuration of the system so that it can meet the ACID requirements. Synchronizing the loaded data on RAID (Redundant array of independent disks) devices is also taken into consideration. The refresh RF1 inserts new rows into the tables LINEITEM and ORDERS while RF2 removes the same number of rows from those tables.

4.2.2.4 Queries

The program used in the TPC-H benchmark to generate queries against the test database is the QGEN program. It is written in ANSI’C’ and has been ported to a large number of platforms [TPC-H 2008]. Minor syntactic modifications of the TPC-H queries are permissible so that they can be run on specific commercial database application and all the tables created during the execution of query must meet the ACID proprieties [TPC-H, 2008].

Each and every TPC-H benchmark query is defined by the following components [TPC-H, 2008]:

- The business question, which illustrates the business context in which the query could be used;

- The functional query definition, which defines, using the SQL-92 language, the function to be performed by the query;
- The substitution parameters, which describe how to generate the values needed to complete the query syntax;
- The query validation, which describes how to validate the query against the current database.

According to Zaharioudakis et al, [2000] modern decision-support queries involve joins, arithmetic operations, complex aggregations and nested subqueries which make them to be complex. TPC-H benchmark queries also involve the characteristic stated by Zaharioudakis et al, [2000]. The ad hoc nature of the TPC-H benchmark queries combined with their complexity make their execution time to be long. They may take hours or even days of execution runs. Graefe, [1993] defines a complex query as one requiring a number of query processing algorithms to work together and a large database, as one using files with sizes ranging from several megabytes to many terabytes.

Some works have been done with the aim of optimizing decision support database applications that are mainly characterised by their database size being large. Much detail will be given in the Application tuning section of this literature review.

5. Application Tuning (Database engine)

In order to obtain an optimum performance from a database system, the design and implementation model of the database must be well performed [Coronel, 2002]. That alone, does not guarantee a complete solution to the performance issues. Other techniques such as query optimization need to be taken into consideration as well. Key configuration variables specific to a particular database product affecting the query optimization process might need to be altered.

Graefe, [1993] defines the query processing as the component filling the gap between the database query languages and the data storage systems in a database system. It comprises the query optimizer that translates queries written in a high level query language into a series of operations that are implemented in the query execution engine. The query optimizer has the responsibility of finding a query evaluation plan that minimizes the performance cost measures of a database [Graefe, 1993]. These performance measures include:

- The database user's wait for the first or last result item;
- CPU cycles;
- Memory costs(as maximum allocation or as time-space product);
- I/O (Input/Output) data transfer;
- Network time and effort;
- Total resource usage

5.1 Query execution

A query plan as states by Shao et al., [2005] is made of a cooperating tree of operators. Since a logical operator may be executed using a combination of various physical operators, a single query can be executed using one of the many possible query execution plans that can be produced by the query optimizer though any of the valid query plans calculate the same answer to the query. [Shao et al, 2005]. The difference between the logical operators and physical operators as stated by Graefe, [1993] resides on the fact that physical operators implement logical operators. A logical operator defines how the query can be expressed in the data model where as a physical operator is more specific to the query processing system. The illustration below describes it better:

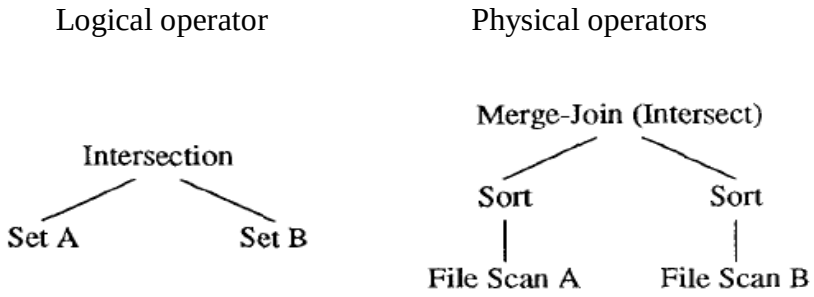


Figure 2: Logical and physical operators [Graefe, 1993].

The most used physical operators are [Shao et al 2005]:

- A table scan: Read through an entire table and generates a stream of records that satisfy the condition (predicate) part of the query statement; An index scan operator provides the same results as the table scan by using an index to access only records that meet the predicate;
- Table joins: Match rows from two tables based on an equality or other condition on common fields. Joins may be implemented using the nested loop, sort-merge or hash join algorithms.

- Order-by clauses are implemented using the sort operator that sorts records in the input table with respect to a subset of fields.
- Group-by: categorises the input records into groups based on a subset of fields and outputs of the groups. It can be implemented using sorting or hashing.
- Aggregate: Refers to a function such as sum, max, min, count, avg etc. that can be applied on the input records to output a single value.

After running the TPC-H benchmarks on IBM DB2 UDB V.7.2 using a 4 way 733MHz Intel Pentium III server, Shao et al., [2005] observed that most of the TPC-H queries execute basic query operations such as sequential scan or join. The query optimizer's suggested plans for the TPC-H queries were found as follows:

- 50% of the queries were dominated by tables scan(Over 95% of their execution time is estimated to be due to table scans);
- 25% of the queries spent more than 95% of the time executing nested-loop joins;
- 25% of the remaining queries executed table scans for about 75% of the time on average and nested-loop joins for about 25% of the time on average

The obtained results were said to be counterintuitive considering the complexity and depth of a TPC-H query plan [Shao et al., 2005]. Such results could be explained by the filtering being done at the lowest levels of the operator tree and the size of the result being reduced as the execution continues to the upper levels of the tree. Shao et al., [2005] concluded by suggesting the scaling down of the TPC-H workload by constructing representative queries that execute the dominant operators (Sort and join) and the use of small datasets that fit in the research testbed. The primary characteristics of interest for measuring performance of the system under test were [Shao et al., 2005]:

- Query execution time breakdown;
- Memory stall time breakdown in terms of cycles lost at various cache levels and TLBs;
- Data and instruction cache misses per instruction at each level branch;
- Branch misprediction per instruction

Shao et al., [2005] noted that on average the processor remained idle more than 80% of the time when executing the TPC-H queries.

Wasserman et al., [2004] after analysing the TPC-H queries run on DB2 UDB, grouped them into 4 classes based on their processing, I/O and n-way table joins characteristics as described by the table below:

Classes	Query Number	Characteristics
Class 1	Q11, Q14, Q5, Q12, Q8, Q7, Q1, Q3, Q4, Q10	• Medium-complexity Query • High Response times • Moderate CPU and I/O usage
Class 2	Q2, Q20, Q17 (Q19 and Q6 are borderline)	• Simple queries which are I/O-bound and join small number of tables
Class 3	Q9, Q18, Q21	• Large and complex queries which are long-running • Have large number of tables joined • Exhibit high sequential and random I/O usage
Class 4	Q13, Q22, Q15, Q16	• Trivial queries • Short run times • Small number of tables joined • Exhibit high CPU utilization

6. Conclusion

The process of benchmarking involves various different procedures, factors and key configuration parameters to be taken into consideration concerning the hardware platform, system software and application design for the chosen database product. The aim is to evaluate the performance of the test database under the workload of interest and optimize its performance if possible. The variety of hardware, operating system and database products in the market combined with various workloads that can be simulated them, makes the benchmarking process complex to perform. Nevertheless researches are still being undertaken towards finding optimum ways of testing the performance databases.

7. References

Burgess G, What is the TPC Good For? Or, the Top Reasons in Favour of TPC Benchmarks, http://www.tpc.org/information/other/articles/TopTen.asp, 2009 [Access 17-06-2009]

Burleson, D., Database benchmarking, http://www.builderau.com.au/strategy/businessmanagement/soa/Database-benchmarking/0,339028271,320267276,00.htm, 2002, [Accessed 17-06-2009]

Coronel, C., *Database Systems: Design, Implementation & Management*, 5th Edition, Thomson Learning, Inc., 2002, Massachusetts, USA

Darmont, J., Bentayeb, F., Boussaid, O., 2007 "Benchmarking data warehouses", *Int. J. Bus. Intell. Data Min*, 2, 1:79-104, Inderscience Publishers

Dietrich, S., Brown, M., CORTES-RELLO, E., WUNDERLIN, S., "A Practitioner's Introduction to Database Performance Benchmarks and Measurements", *THE COMPUTER JOURNAL*, VOL.35, NO. 4, 1992

Floyd, C., Meikel, P., "New TPC benchmark for decision support and web commerce", *SIGMOD Record***,** Volume 29 Issue 4, ACM, December 2000

Graefe, G.,"Query Evaluation Techniques for Large Databases", *Computing Surveys (CSUR)*, ACM, 1993

Hoste, K., Eeckhout, L., Blockeel H., "Analyzing commercial processor performance numbers for predicting performance applications of interest", *SIGMETRICS international conference on Measurement and modelling of computer systems*, June 12–16, 2007, San Diego, California, USA

Oracle, Database Benchmarking, http://wiki.oracle.com/page/Database+Benchmarking, 2009, [Accessed 16-06-2009]

Petkovic D., Microsoft SQL Server 2005, *A Beginner's Guide*, The McGraw-Hill Companies, 2006, California, USA

Scalzo B., Ault M., Burleson D., Fernandez C., Klein K., *Database Benchmarking*, Practical Methods for Oracle & SQL Server, Rampant TechPress, USA, April 2007

Shao, M., Ailamaki, A., Falsafi, B., "DBmbench: Fast and Accurate Database Workload Representation on Modern Microarchitecture", *CASCON '05: Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research*, ACM, 2005

Scalzo, B., Benchmark Factory for Databases, http://www.quest.com/events/podcast/default.asp?path=/Quest_Site_Assets/podcasts/Quest_Software_-_Benchmark_Factory_-_Bert_Scalzo.mp3&title=Benefiting%20your%20IT%20Environment%20with%20Benchmark%20Factory, 2009, [Accessed 12-09-2009]

TPC, TPC Benchmarks, http://www.tpc.org/information/benchmarks.asp, 2009, [Accessed 15-06-2009]

TPC-H, http://www.tpc.org/tpch/default.asp, 2009, [Accessed 14-06-2009]

Wasserman, T.J., Martin, P., Rizvi, H., "Sizing DB2 UDB® Servers for Business Intelligence Workloads", ACM, 2004

Watts, D., Slavko, B., Watson, C., Understanding IBM eServer xSeries Benchmarks, http://www.redbooks.ibm.com/redpapers/pdfs/redp3957.pdf, 2005, [Accessed 3-06-09]

Zaharioudakis, M., Cochrane, R., Lapis, G., Pirahesh, H., Urata, M., "Answering complex SQL queries using automatic summary tables", *Proceedings of the 2000 ACM SIGMOD international conference on Management of data,* ACM, 2000