Comparing Object-Oriented Database Systems Benchmark Methods

Jia-Lang Seng, Ph.D.
Associate Professor
Dept. of Management Information Systems
National Cheng-Chi University
Mu-Cha, 116, Taipei, Taiwan, ROC
(O) +886-2-9387692
jljan@cc.nccu.edu.tw

Abstract

With the increasing use of the object-oriented database software, system performance has become an important issue in the system procurements and evaluation process. It is vital to understand and predict the functionality and performance characteristics of the systems to be procured in the real world settings so as to justify the use of the object-oriented systems over the traditional systems. Benchmarks are considered the most common approach to test and measure the quantitative performance of database systems. We have seen a set of de facto standard benchmarks such as the OO1, HyperModel, ACOB, and OO7 benchmarks receiving more and more attention and acceptance. It is hence the intent of this paper to provide an elaborate version of analysis of these benchmarks and to give a systematic comparison of these standards with a multi-set of comparison criteria. Through the comparison and contrast, we point out the essence of features of a desirable database benchmark.

1. Introduction

A benchmark is a standard by which something can be measured or judged. A database benchmark is defined as a standard set of executable instructions which are used to measure and compare the relative and quantitative performance of two or more database systems through the execution of controlled experiments. Benchmarking is therefore a process of evaluating different database software systems on the same or different hardware platforms. Each experiment is made up of two kinds of variables. One is the set of independent variables which will affect the performance of database systems and are called the experimental factors. The other is the set of dependent variables which represent the quantitative measurements collected from the benchmarking process. They are called the performance metrics. Common performance metrics include the throughput metric which is the ratio of work volume over certain time period and the response time metric which is the ratio of time spent over certain work volume. Benchmark results depend on the workload, specific application requirements, and system design and implementation. A workload is the amount of work assigned to or performed by a worker or unit of workers in a given time period. The workload of a database benchmark is the amount of work assigned to or performed by a database system in a given period of time. The scope and scale of benchmark hence rely on the workload defined

Current database benchmarks such as the Wisconsin, AS^{3}AP, TPC-C, TPC-D, and TPC-E benchmarks are used to measure the performance of the relational database management systems (RDBMS). These benchmarks cannot model the features and functions of the object-oriented database management systems (OODBMS). There is a new set of de facto standard benchmarks which are designed to measure the functionality and performance of OODBMS. These methods have been used in academia and industry and been received as more OODBMS are installed. They are used to study the quantum improvement of OODBMS performance. These standard benchmarks are the OO1, HyperModel, ACOB, and OO7 benchmarks.

In this paper, we describe each of the benchmarks and compare their features. The objective is to give us an opportunity to understand the set of tests and controls a benchmark should have to measure against OODBMS. We compare and contrast these methods to show the movement and improvement of OODBMS benchmarks. A multi-set of comparison criteria are used to distinguish each of the benchmarks. These criteria relate to the experimental nature of a database benchmark. One set of criteria is based on the performance metric, another set on the experimental factors, and the other on the control characteristics. The paper is organized into four sections. The first section is the introduction. The second section details each of the database benchmarks. The third section gives the comparison and discussion. The fourth section concludes the paper.

1060-3425/98 $10.00 (c) 1998 IEEE

2. OODBMS Benchmark Methods

2.1 HyperModel Benchmark

The HyperModel benchmark described in [1] is an early OODBMS benchmark method which is designed to test the hypertext and hyperlink applications. A hypertext is a graph of nodes and links to be traversed and retrieved. The HyperModel benchmark extends the graph and adds new relationships between nodes. The benchmark database is a graph of interconnected nodes which are text nodes or form nodes. The new relationships between nodes are the parent-child relationship, refer-to and refer-from relationship, and partof relationship between nodes which are considered as one hierarchical (1:N) relationship and two network (M:N) relationships. The database begins with the root node and forms a tree with a five-way fanout which gives k-1 levels of non-leaf nodes plus a level of leaf nodes as shown in Figure 1.

The HyperModel benchmark consist of six types of tests. The first group measures the single object lookup response time. The second group tests objects with range lookup. The third group measures the response time of the lookup over the new relationships. The fourth group uses indices to test the speed when traversing the three relationships. The fifth group is a sequential scan of the entire object hierarchy. The last group measures the response time of the closure lookup with different node relationships. Each of the closure operations performs a reachability traversal, starting at a randomly chosen node at level three of the 1:N hierarchy. These tests are stated in Table 1.

2.2 OO1 Benchmark

The OO1 benchmark described in [6] improves upon the SUN benchmark specified in [17]. This benchmark models the common requirements of engineering applications. It defines the common workload characteristic of computer-aided software engineering (CASE) and computer-aided design (CAD) applications. The benchmark assesses the performance of OODBMS, RDBMS, network database systems, and hierarchic database systems. The OO1 benchmark database is based upon part objects and connections between them. Each part has exactly three out-going (to) connections to other part objects plus a variable number of incoming (from) connections. The database size can range from 4 MB (small), 40 MB (large, and 400 MB (huge). Table 2 gives the table fields.

2.3 ACOB Benchmark

The ACOB (Altair Complex Object Benchmark) is a study of three workstation-server or client-server OODBMS architectures described in [8]. These architectures ar the object-server, page-server, and file-server systems. Object-server system transfers data in the unit of object, page-server in the unit of page, and file-server in the unit of file. The benchmark is designed to understand the system behavior of each architecture in the execution of the object operations. They found that advantages and disadvantages exist in these architectures. In short, when the size of main memory and the buffer space increase, the system performance of the page-server and file-server systems outperform the object-server system due to the higher hit rate on the cached data and the successful swizzling scheme, otherwise, object-server system does better in concurrency control and recovery. Like the HyperModel benchmark, the ACOB database imitates the hypertext database and has complex objects in the tree structure as shown in Figure 2. The ACOB tests include (1) sequential scan of objects, (2) random reads, and (3) random updates.

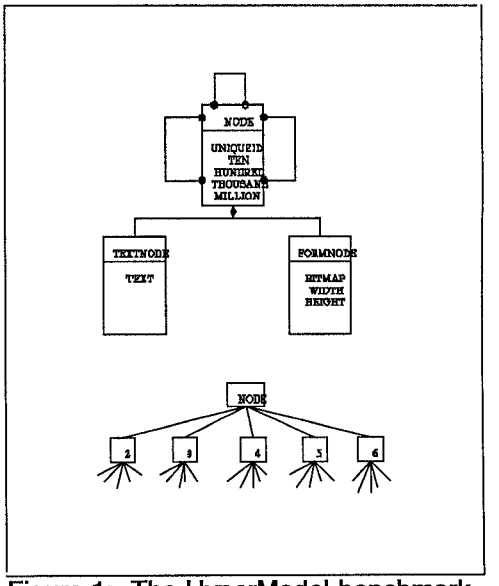

Figure 1: The HyperModel benchmark database

Table 1: The HyperModel benchmark tests

HyperModel Tests	Test Description
Name Lookup Operations	namelookup(), nameoidlookup()
Range Lookup Operations	rangelookuphundred(), rangelookupmillion()
Group Lookup Operations	grouplookupin(), grouplookupmn(), grouplookupmnatt()
Reference Lookup Operations	reflookupin(), reflookupmn(), reflookmnatt()
Sequential Scan	seqscan()
Course Traversal Operations	closurein(), closuremn(), closuremnatt(), closureinattsum(), closureinattset(), closureinpred(), closuremnattlinksum()

Table 2: The OO1 Benchmark Database

record (id int; type char[10]; x, y int; build date)
record (from part-id; to part-id; type char[10]; length int)

Table 3: The OO1 Benchmark Tests

OO1 Tests	Test Description
Lookup	Generate 1000 random part id's and fetch the corresponding parts from the database. For each part, call a null procedure written in any host programming language, passing the x, y position and type of the part.
Traversal	Find all parts connected to randomly selected part, or to a part connected to it, and so on, up to seven hops. For each part, call a null programming language procedure with the value of the x and y fields and the part type. Also, measure time for reverse traversal, swapping from to to direction to compare the results obtained.
Insert	Enter 100 parts and three connections from each to other randomly selected parts. Time must be included to update indices or other access structures used in the execution of Lookup and Traversal. Call a null programming language procedure to obtain the x, y position for each insert. Commit the changes to the disk.

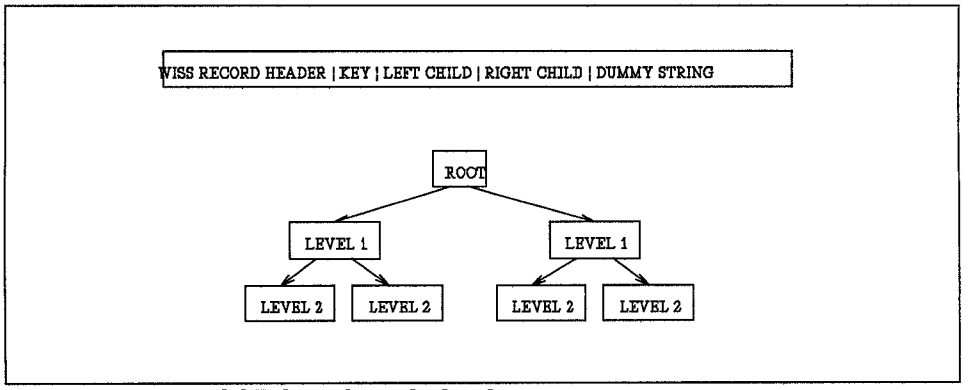

Figure 2: The ACOB benchmark database

2.3 ACOB Benchmark

The ACOB (Altair Complex Object Benchmark) is a study of three workstation-server or client-server OODBMS architectures described in [8]. These architectures ar the object-server, page-server, and file-server systems. Object-server system transfers data in the unit of object, page-server in the unit of page, and file-server in the unit of file. The benchmark is designed to understand the system behavior of each architecture in the execution of the object operations. They found that advantages and disadvantages exist in these architectures. In short, when the size of main memory and the buffer space increase, the system performance of the page-server and file-server systems outperform the object-server system due to the higher hit rate on the cached data and the successful swizzling scheme, otherwise, object-server system does better in concurrency control and recovery. Like the HyperModel benchmark, the ACOB database imitates the hypertext database and has complex objects in the tree structure as shown in Figure 2. The ACOB tests include (1) sequential scan of objects, (2) random reads, and (3) random updates.

2.4 OO7 Benchmark

The OO7 benchmark described in [4] is the key standard OODBMS benchmark method which is considered an important extension of the OO1 benchmark where the benchmark database and test sets are expanded to include more complex objects and operations. The complex objects refer to the atomic parts, composite parts, base assembly, and complex assembly in the OO7 benchmark. The complex operations consist of the raw and sparse traversals, make and build tests, bulk updates, and ad-hoc joins. The main performance metric is still the elapsed time. The set of experimental factors are broader and include the factors of the types of complex objects, the size of complex objects, the relationships of complex objects, the types of tests, the complexity of tests, the dense and sparse traversals, single and range lookups, bulk and indexed updates, ad-hoc joins, bulk insertions and deletions, and the number of users.

The OO7 benchmark database imitates the CASE/CAD database which centers on the composite part that can be a procedure in a software program. Each composite part comprises a number of atomic parts which connects to each other with the fanout number of three, six, or nine links. Assembly hierarchy is created to give levels of object classes to traverse. Composite parts form the first level of complex objects in the assembly hierarchy which are called the base assemblies. Base assemblies, in turn, form the complex assemblies for the second level and the other higher levels of objects in the hierarchy. Due to the fanout size of the connections between the atomic parts and the number of atomic parts in a composite part, the OO7 benchmark database can range from small, medium, and large database bigger than that in the OO1 benchmark. Table 4 gives the OO7 benchmark database parameters and Figure 3 shows the entity relationship diagram (ERD) of the OO7 benchmark database. The OO7 benchmark tests have the raw traversal of the atomic parts for each composite part, the sparse traversal of the root of each composite parts, the cached data traversal, the updates in the complete and sparse traversals, the updates on the indexed and unindexed object fields, exact-match lookups, range lookups, manual searches, bulk insertion and deletion. Table 5 is the list of the OO7 benchmark tests.

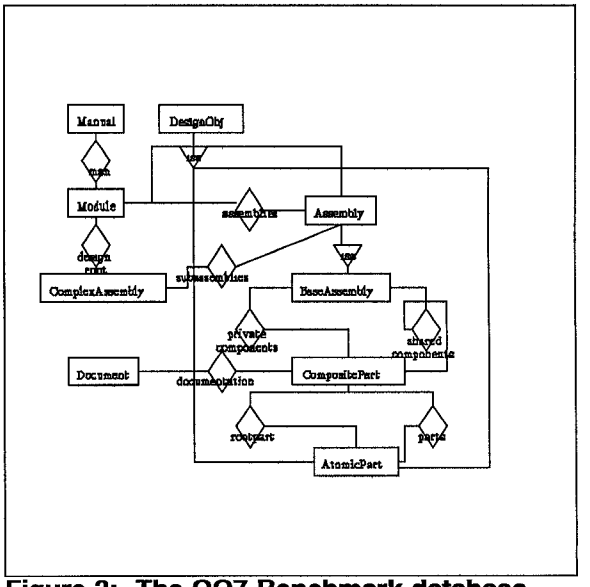

Figure 3: The OO7 Benchmark database schema

Table 4: The OO7 Benchmark database parameters

Parameter	Small	Medium	Large
NumAtomicPerComp	20	200	200
NumConnPerAtomic	3,6,9	3,6,9	3,6,9
DocumentSize	2000	20000	20000
ManualSize	100K	1M	1M
NumCompPerModule	500	500	500
NumAssmPerAssm	3	3	3
NumAssmLevels	7	7	7
NumCompPerAssm	3	3	3
NumModules	1	1	10

Table 5: The OO7 benchmark tests

OO7 Tests	Test Description
Raw Traversal Speed: Traverse the assembly hierarchy.	As each base assembly is visited, visit each of its referenced unshared composite parts. As each composite part is visited, perform a depth first search on the graph of atomic parts. Return a count of the number of atomic parts visited when done.
Sparse Traversal Speed: Traverse the assembly hierarchy.	As each base assembly is visited, visit each of its referenced unshared composite parts. As each composite part is visited, visit the root atomic part. Return a count of the number of atomic parts visited when done.
Traversal with Update: Repeat raw traversal speed test, but update objects during the traversal.	There are three types of update patterns in this traversal. In each, a single update to an atomic part consists of swapping its (x,y) attributes.
Traversal with Indexed Field Update: Repeat sparse traversal speed test, except that now the update is on the date field, which is indexed.	The specific update is to increment the date if it is odd, and decrement the date if it is even.
Operations on Manual	Scan the manual object, counting the number of occurrences of the character "I".
Operations on Manual	Checks to see if the first and last character in the manual object are the same.
Cached Update	Repeat raw traversal speed test and traversal with update test in a single transaction. Report the total time minus the raw traversal speed test hot time and minus the raw traversal speed cold time.

OO7 Queries	Query Description
Exact Match Lookup	Generate 10 random atomic part id's, for each part id generated, lookup the atomic part with that id. Return the number of atomic parts processed when done.
1% Range Query	Choose a range for dates that will contain the last 1% of the dates found in the database's atomic parts. Retrieve the atomic parts that satisfy this range predicate.
10% Range Query	Choose a range for dates that will contain the last 10% of the dates found in the database's atomic part. Retrieve the atomic parts that satisfy this range predicate.
Scan	Scan all atomic parts.
Path Lookup	Generate 100 random document titles. For each title generated, find all base assemblies that use the composite part corresponding to the document. Also, count the total number of base assemblies that qualify.
Single-Level Make	Find all base assemblies that use a composite part with a build date later than the build date of the base assembly. Also report the number of qualifying base assemblies found.
Ad-Hoc Join	Find all pairs of documents and atomic parts where the document id in the atomic part matches the id of the document. Also return a count of the number of such pairs encountered.
Insert	Create five new composite parts, which includes creating a number of new atomic parts (100 in the small configuration, 1000 in the large, and five new document objects) and insert them into the database by installing references to these composite parts into 10 randomly chosen base assembly objects.
Delete	Delete the five newly created composite parts and all of their associated atomic parts and document objects.

3. A Comparative Study

A comparative study is conducted to compare and contrast these standard benchmark methods. The study is based on a multi-set of comparison criteria. One set is the performance metrics the benchmark uses. Another set comprises the experimental factors the method includes. The other set cover the control characteristics the method has.

On the performance metric, the query elapsed time is used in each of the four benchmarks. On the experimental factors, we classify them into the factors of benchmark database and the factors of object operations. Database schema include the definitions of object types (simple object, complex object, and object hierarchy), object sizes, index types, and object relationships (direct relationship, hierarchical relationship, and network relationship). Object operations consist of dense traversal, sparse traversal, exact-match lookup, range lookup, join, character search, string search, bulk update, bulk insertion, and bulk deletion. Control characteristics mean the number of users and the system architectures (standalone and client/server settings) the benchmark measures on. We mark under the benchmark method for each criterion that the method qualifies in Table 6.

In essence, the OO7 benchmark ranks high as the more comprehensive and complete benchmark method in our study. The OO1 benchmark is known for the measures of the simple navigational and update tasks. Though the benchmark covers both OODBMS and RDBMS, the benchmark does not support complex object definition and semantic traversals. Many more complex and extensive tests are found in the HyperModel benchmark and the OO7 benchmark. A minor point is the definition of object locality which is considered to rough and may mislead the performance readings. The HyperModel benchmark gives more than 17 tests to measure the system performance. Though it is extensive, the method is difficult to implement. There is a doubt if the additional tests give insights of the system performance to be worthy of the efforts. However, to test the three types of object relationships are inventive in performance evaluation. This gives the insight of performance of closure operations. The ACOB benchmark distinguishes itself from the others by its contribution to the analysis of three different client/server architecture for OODBMS. The benchmark database and object operations do not give inventive designs. But the test results give an in-depth understanding of the advantages and disadvantages of each of the client/server structures.

4. Summary

In this paper, we have compared four de facto standard OODBMS benchmark methods. We use the essence of the design of a database benchmark as the comparison base. A benchmark is executed as a series of controlled experiments. Each experiment is formed by two sets of variables. They are the set of independent variables and the set of dependent variables. These give a natural distinguishing base for the standard benchmarks. We add a new dimension of distinguishment of control characteristics for benchmarking. In the execution of benchmark, the duration, replication, input, steady state, single user versus multi-user mode give the common distinguishing points. Our future research is based on this study to further the essence of these benchmarks and develop a open-end custom object-oriented database benchmark.

References

[1] Ahrens, J., and Song, I. Y., "EER Data Modeling Aids for Novice Database Designer," Proceedings of the 2nd International Conference of the Information Resources Management, Memphis, TN, May 19-22, 1991, pp. 99-114.

[2] Anderson, T. L., Berre, A. J., Mallison, M., Porter, H. H., and Schneider, B., "The HyperModel Benchmark," Proceedings of the Second International Conference on Extending Database Technology, March, 1990, pp. 317-331.

[3] Astrahan, M., "System R: Relational Approach to Database Management," ACM Transactions on Database Systems, 1(1), 1976.

[4] Bitton, D. and C. Turbyfill, "Design and Analysis of Multiuser Benchmarks for Database System," Proceedings of the HICSS-18 Conference, 1985.

[5] Boral, H. and D. J. DeWitt, "A Methodology for Database System Performance Evaluation,"

Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, May 1984, pp. 176-185.

[6] Carey, M. J., D. J. DeWitt, and J. F. Naughton, "The 007 Benchmark," Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, May 1993, pp. 12-21.

[7] Carey, M., DeWitt, D., Kant, C., Naughton, J., "A Status Report on the OO7 OODBMS Benchmarking Effort," In Proceedings of the ACM OOPSLA Conference, pp 414-426, Portland, OR, Oct. 1994.

[8] Cattell, R. G. G. and J. Skeen, "Engineering Database Benchmark," ACM Transactions on Database Systems, 17(1): 1-31, March 1992.

[9] Cattell, R. G. G., "An Engineering Database Benchmark," appears in The Benchmark Handbook for Database and Transaction Processing Systems, Ed. by Jim Gray, Morgan Kaufmann, Inc., 1993, pp. 397-434.

[10] Chen, P. P., "The Entity Relationship Model - Toward a Unified View of Data," ACM Transactions on Database Systems, 1:1, 1976, pp. 9-36.

[11] DeWitt, D. J., S. Ghandeharizadeh, and D. Schneider, "A Performance Analysis of the Gamma Database Machine," Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, May 1988, pp. 350-360.

[12] DeWitt, D. J., P. Futtersack, D. Maier, and F. Velez, "A Study of Three Alternative Workstation Server Architectures for Object-Oriented Database Systems," Proceedings of the 16th International Conference on Very Large Data Bases, August 1990, pp. 107-121.

[13] Duhl, J., Damon, C., "A Performance Comparison of Object and Relational Databases Using the Sun Benchmark," In Proceedings of the ACM OOPSLA Conference, San Diego, CA, Sept. 1988.

[14] Gray, J. N. and F. Putzolu, "The Five Minute Rule for Trading Memory for Disk Accesses and the 10 Byte Rule for Trading Memory for CPU Time," Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, May 1987, pp. 395-398.

[15] Gray, J. N., The Benchmark Handbook for Database and Transaction Processing Systems,Ed., Morgan Kaufmann, Inc., 1993.

[16] Gray, J. N., P. Sundaresan, S. Englert, K. Baclawski, and P. J. Weinberger, "Quickly Generating Billion-Record Synthetic Databases," Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, May 1994, pp. 243-252.

[17] Hagmann, R. and D. Ferrari, "Performance Analysis of Several Back-End Database Architectures," ACM Transactions on Database Systems, 11(1): 1-26, March 1986.

[18] Hawthorn, P. B. and D. J. DeWitt, "Performance Analysis of Alternative Database Machine Architectures," IEEE Transactions on Software Engineering, SE-8(1): 61-75, January 1982.

[19] Heidelberger, P. and S. S. Lavenberg, "Computer Performance Evaluation Methodology," IEEE Transactions on Computers, C-33(12): 1195-1220, December 1984.

[20] Leutenegger, S. T., and D. Dias, "A Modeling Study of the TPC-C Benchmark," Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, May 1993, pp. 22-31.

[21] Park, S., Miller, K , "Random Number Generators: Good Ones Are Hard to Find. Communications of ACM, 31(10), 1988.

[22] Rabb, F., "Overview of the TPC Benchmark C: A Complex OLTP Benchmark," appears in The Benchmark Handbook for Database and Transaction Processing Systems, Ed. by Jim Gray, Morgan Kaufmann, Inc., 1993, pp. 131-144.

[23] Rubenstein, W. B., M. S. Kubicar, and R. G. G. Cattell, "Benchmarking Simple Database Operations," Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, May 1987, pp. 387-394.

[24] Song, X. and L. J. Osterweil, "Experience with an Approach to Comparing Software Design Methodologies," IEEE Transactions on Software Engineering, SE-20(5): 364-384, May 1994.

[25] Stonebraker, M., J. Frew, K. Gardels, and J. Meredith, "The Sequoia 2000 Storage Benchmark," Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, May 1993, pp. 2-11.

[26] Teorey, T. J., Yang, D., and Fry, J. P., "A Logical Design Methodology for Relational Databases Using the Extended Entity Relationship Model," Computing Surveys, 18:12, June 1986, pp. 197-222.

[27] Turbyfill, C., C. Orji, and D. Bitton, "AS^{3}AP - A Comparative Relational Database Benchmark," Proceedings of the IEEE COMPCON, 1989, pp. 560-564.

[28] White, S., DeWitt, D., "A Performance Study of Alternative Object Faulting and Pointer Swizzling Strategies," In Proceedings of VLDB Conference, Vancouver, Aug. 1992.

Table 6: The OODBMS benchmark methods comparison

Comparison Criteria		OO1	HyperModel	ACOB	OO7
Performance Metric: Elapsed Time		X	X	X	X
Experimental Factors: Database Schema	simple object	X	X	X	X
	complex object		X		X
	object hierarchy		X	X	X
	index		X		X
	direct relationship	X	X		X
	hierarchical relationship	X	X		X
	network relationship		X		
Experimental Factors: Object Operations	dense traversal	X	X	X	X
	sparse traversal		X	X	X
	exact-match lookup	X	X	X	X
	range lookup		X	X	X
	character search				X
	string search				X
	join		X		X
	bulk update		X	X	X
	bulk insertion		X	X	X
	bulk deletion		X	X	X
Control Characteristic	number of users	1	1	N	N
	client/server architecture			X	X