CUBRID 2008 R4.0 Beta QA Completion Report

This document is a verification report of CUBRID 2008 R4.0 Beta in terms of functionality, performance, stability.

.Table of Contents

1.Test Overview

1.1 Test Objectives

The objectives of this test are to perform functionality, performance and stability tests for the final release candidate build of CUBRID 2008 R4.0 Beta (hereinafter referred to as R4.0 Beta), which is under development for release in April 2011 and to determine its release based on the test results. To test the stability of CUBRID, a test environment was configured as described below. Based on a comparison between the performance test result of CUBRID 2008 R4.0 Beta and that of CUBRID 2008 R3.1 (hereinafter referred to as R3.1), we tested to determine whether the performance of R4.0 Beta was regressed or improved.

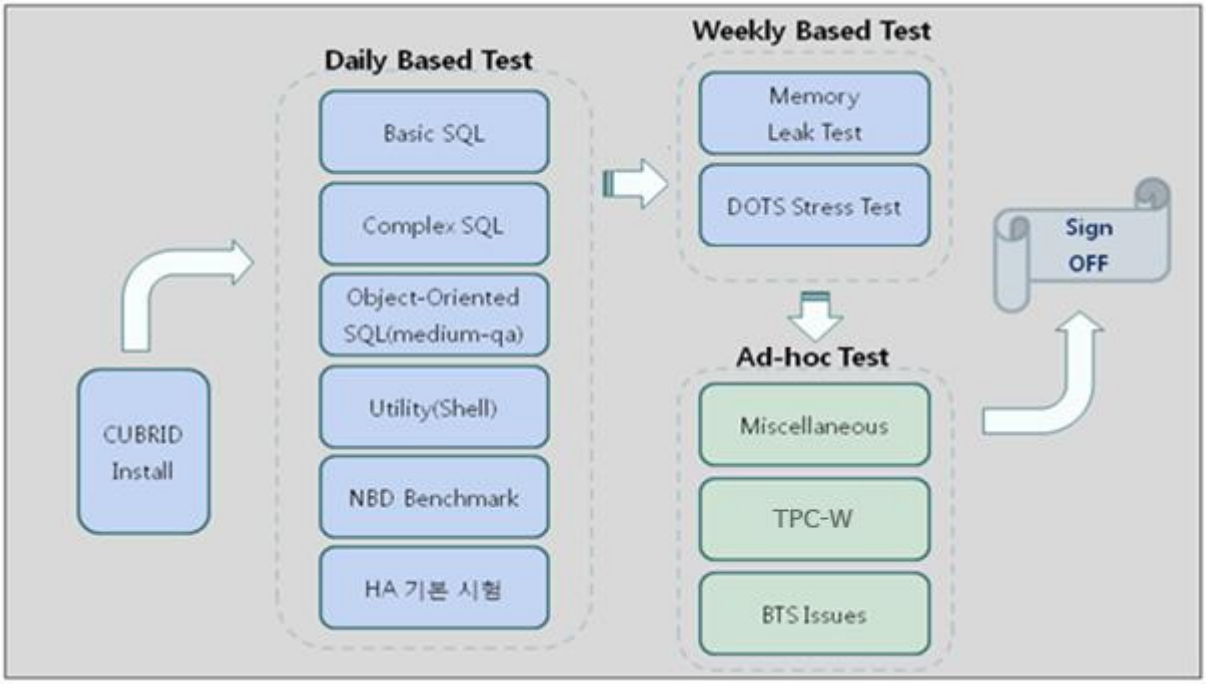
- CentOS 4.6 (32/64-bit) or compatible
- Windows 2003 (32/64-bit) or compatible
- Final test build: 8.4.0.0195 (Linux 64bit/32bit, Windows 64bit/32bit)

1.2 Test Environment

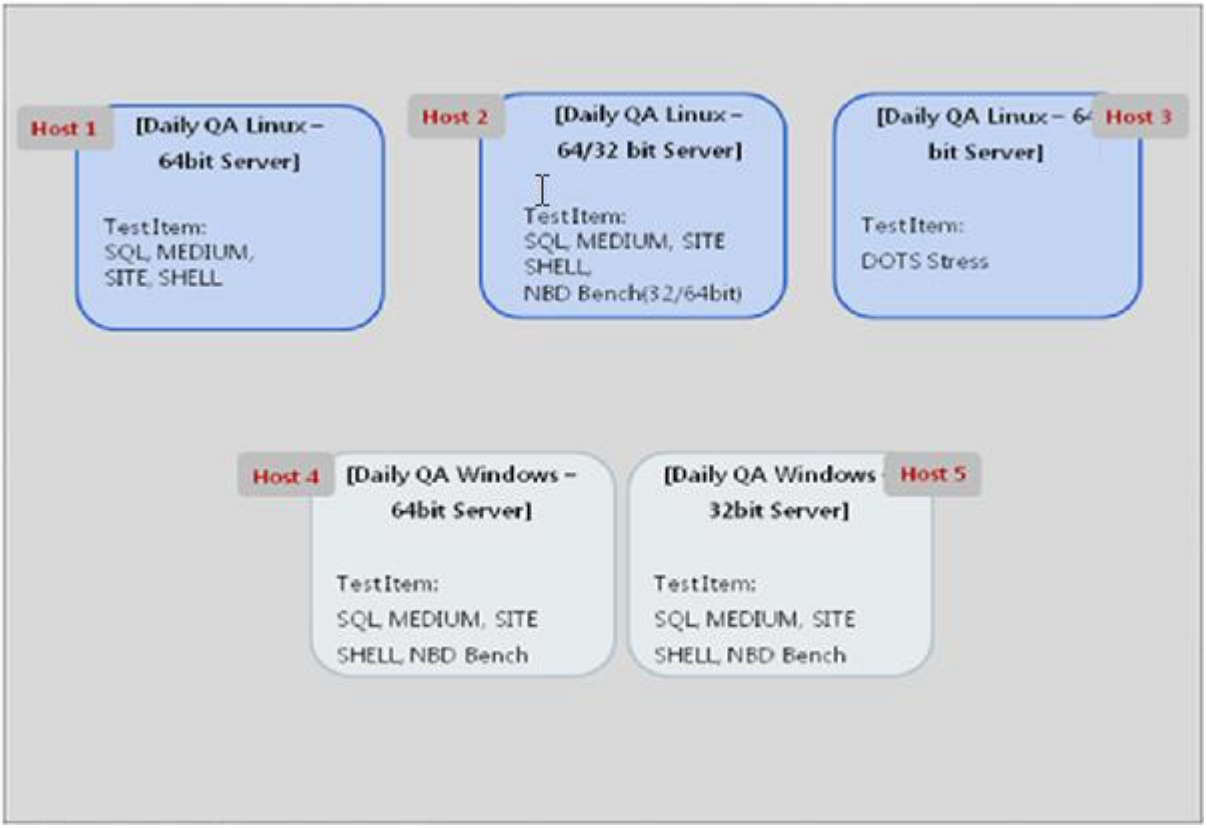
1.2.1 Test Procedures

Tests to verify the CUBRID product are shown below. The test sequence used may differ from the one described here. To verify product stability and functionality, performance, functionality, stability and other tests were performed for 4 types of builds as shown in the figure below. The details of each test are described in the appendix of this report.

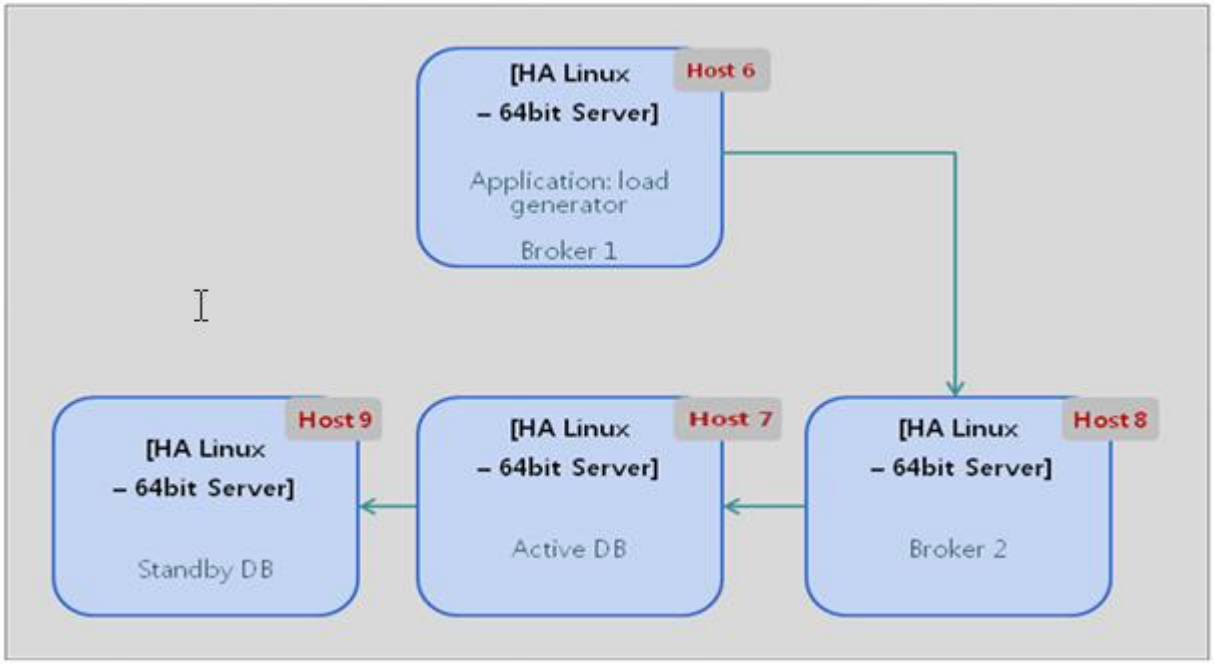
- CUBRID test procedures



- System diagram for basic test



- System diagram for HA test



1.2.2 Hardware Test Environment

Servers for the CUBRID test and their usage are listed in the table below.

Name	OS	CPU	MEMORY	DISK
Host 1	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (dual-core) * 1	4 GB	SATA 500G * 2 (No Raid)
Host 2	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 3	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 4	Windows 2003 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 5	Windows 2003 (32-bit)	Xeon 2.10 GHz (quadcore) * 1	4 GB	SATA 500G * 2 (No Raid)
Host 6	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 7	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 8	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 1	8 GB	SATA 500G * 2 (No Raid)
Host 9	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 1	8 GB	SATA 500G * 2 (No Raid)

1.3 Test Category

The following tests were performed to determine whether CUBRID can be released. The details of each test are described in the appendix of this report.

- Functionality test
  - SQL query test
  - MEDIUM query test
  - SITE query test
  - Utility (Shell) test
  - Basic HA feature test
  - CCI/PHP/JDBC Interface test
- Performance test
  - NBD Benchmark
  - Performance test for basic DBMS functions
  - Performance test for index volume size
- Stability test
  - DOTS stress test
- HA Enhancement
  - TPC-W test
  - SQL/MEDIUM with valgrind test
  - Dots on HA test

    - Porting replication scenario test
- Other tests
    - Test for checking CUBRID 2008 R4.0 Beta functionalities/bug fixes
    - Memory check by Valgrind

2.Test Results

2.1 Functionality Test Results

2.1.1 Basic Query Test

This test was performed to verify the basic DBMS functionalities by using SQL statements. SQL statements stored in 8943 files were tested to verify DBMS conformity. We executed the stored SQL statements in a JDBC-based application and compared the results to the stored reference file for verification.

Test Category	Total Number of Scenarios	Number of Successful Tests	Success Rate
SQL query test	8582	8582	100%
MEDIUM query test	970	970	100%
SITE query test	1213	1213	100%

2.1.2 Basic Utility and Other Scenario Tests

This test was performed to verify the basic DBMS functionalities by using shell scripts. In particular, this test was also performed to verify CUBRID utilities that could not be tested by using SQL statements. We ran scenarios written by 472 shell scripts to verify DBMS conformity.

Test Category	Total Number of Scenarios	Number of Successful Tests	Success Rate
Utility	207	207	100%
Bug regression	175	175	100%
Environment variable	7	7	100%
Other	83	83	100%

2.1.3 HA Feature Test

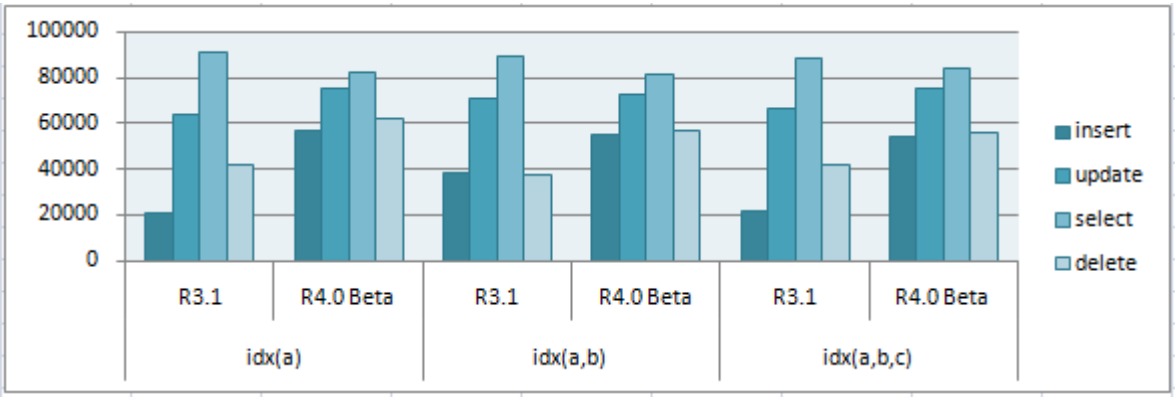
Test Category	Total Number of Scenarios	Number of Successful Tests	Success Rate
Data replication test	5	5	100%
Node fault test	16	16	100%
Process fault test	8	8	100%
Broker fault test	8	8	100%
Run replication test scenarios	115	115	100%

2.2 Performance Test Results

2.2.1 CUBRID Basic Performance Test

This test was performed to check the performance of the CUBRID DBMS basic operations, which are select, insert, update and delete. For more information about test scenarios, see the appendix. For all environment variables, except for SQL_LOG=OFF in cubrid_broker.conf, default configuration values were used. As shown in the table below, we have found that the overall basic performance of CUBRID 2008 R4.0 Beta was significant better than that of CUBRID 2008 R3.1. Especially performance improvement of CUBRID 2008 R4.0 is significant in this test because the sequence of data in insert/update/delete/select is sequenced. Refer to appendix II for more information.

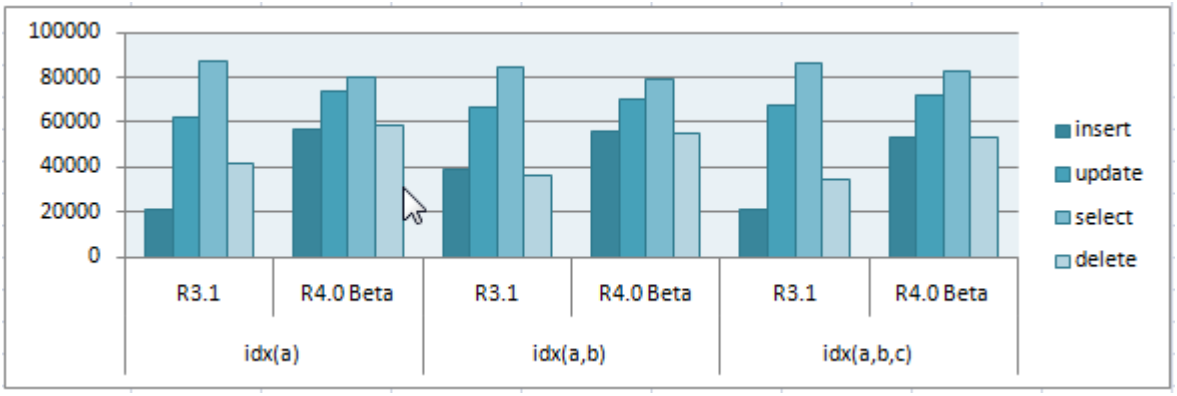
- Linux: Performance Comparison between CUBRID 2008 R3.1 and CUBRID 2008 R4.0 Beta (64-bit)



(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**
Insert	21241	56596	266%	37974	54978	145%	21323	54441	255%
Update	63531	75228	118%	70995	72467	102%	66133	75038	113%
Select	90941	82114	90%	89059	81777	92%	88248	83753	95%
Delete	41663	62419	150%	37176	56609	152%	41712	56027	134%
Total	217376	276357	127%	235204	265831	113%	217416	269259	124%

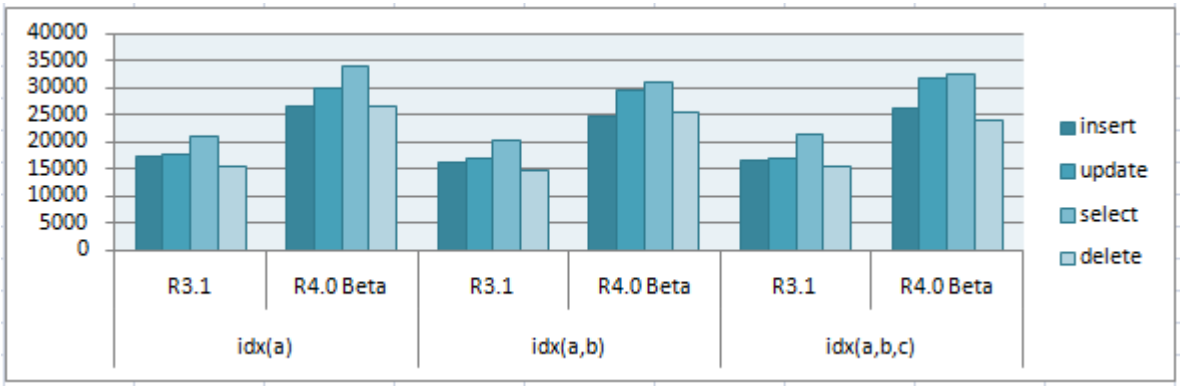
- Linux: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 Beta (32-bit)
  Similarly to the results for the 64-bit tests, we have found that there was significant change in performance between 32-bit CUBRID 2008 R3.1 and 32-bit CUBRID 2008 R4.0 Beta except select operation. Select performance of 32/64-bit R4.0 Beta is lower about 5% than 32/64-bit R4.0 Beta. It is needed to investigate it more in the future.



(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**
Insert	21376	56818	266%	38592	56146	145%	20873	52891	253%
Update	62092	73858	119%	66882	70160	105%	67751	71885	106%
Select	87359	79608	91%	84773	78876	93%	86248	82267	95%
Delete	41346	58292	141%	36262	54763	151%	34899	52802	151%
Total	212173	268576	127%	226509	259945	115%	209771	259845	124%

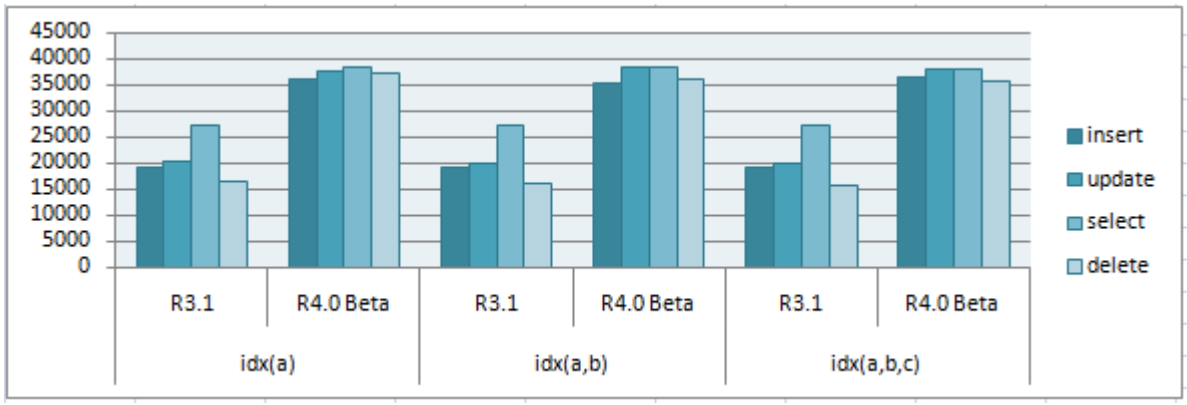
- Windows: Performance Comparison between CUBRID 2008 R3.1 (64-bit) and CUBRID 2008 R4.0 Beta (64-bit)



(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**
Insert	17385	26580	153%	16334	24519	150%	16566	26002	157%
Update	17576	29956	170%	17106	29383	172%	16977	31574	186%
Select	20837	33984	163%	20404	30958	152%	21279	32409	152%
Delete	15278	26526	174%	14533	25530	176%	15569	23931	154%
Total	71076	117046	165%	68377	110390	161%	70391	113916	162%

- Windows: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 Beta (32-bit)
  Similarly to the results for the Windows 64-bit tests, we have found that there was significant change in performance between 32-bit CUBRID 2008 R3.1 and 32-bit CUBRID 2008 R4.0 Beta.



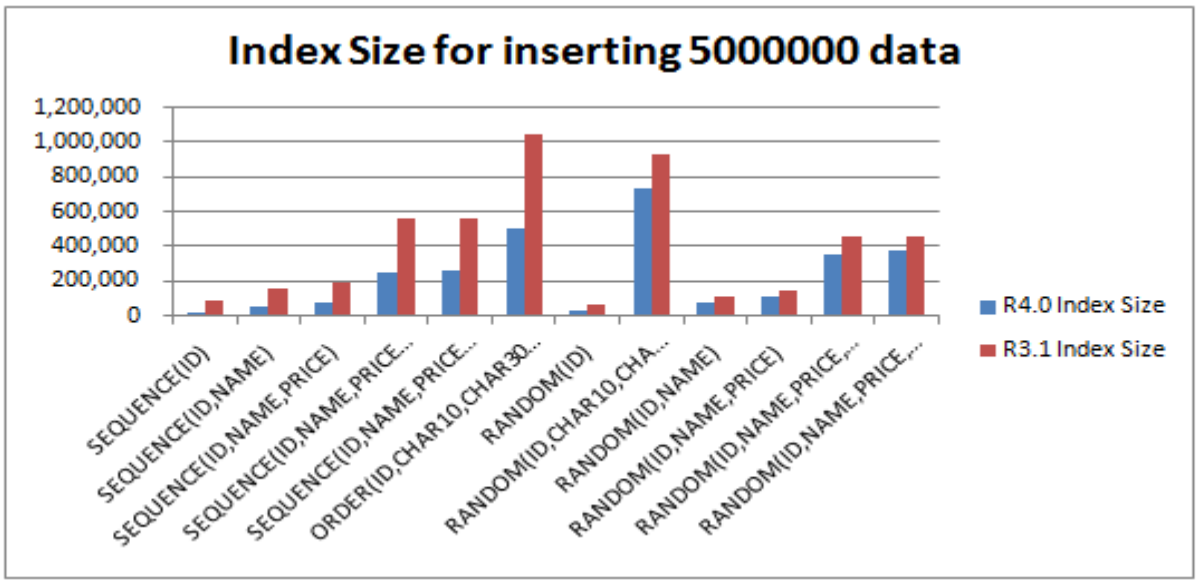
(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**	**R3.1**	**R4.0 Beta**	**Performance Ratio**
Insert	19208	36179	188%	19173	35458	185%	18942	36563	193%
Update	20131	37688	187%	20029	38192	191%	20054	38099	190%
Select	27244	38499	141%	27162	38356	141%	27139	38014	140%
Delete	16287	37062	228%	15972	36027	226%	15641	35830	229%
Total	82870	149428	180%	82336	148033	180%	81776	148506	182%

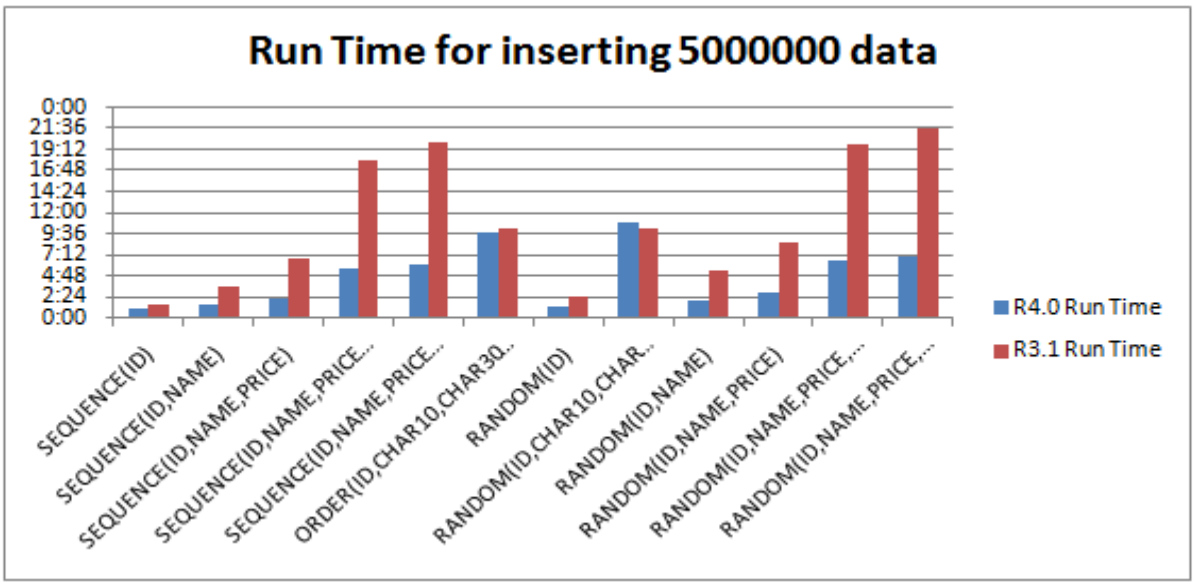
2.2.2 CUBRID Index Volume Performance Test

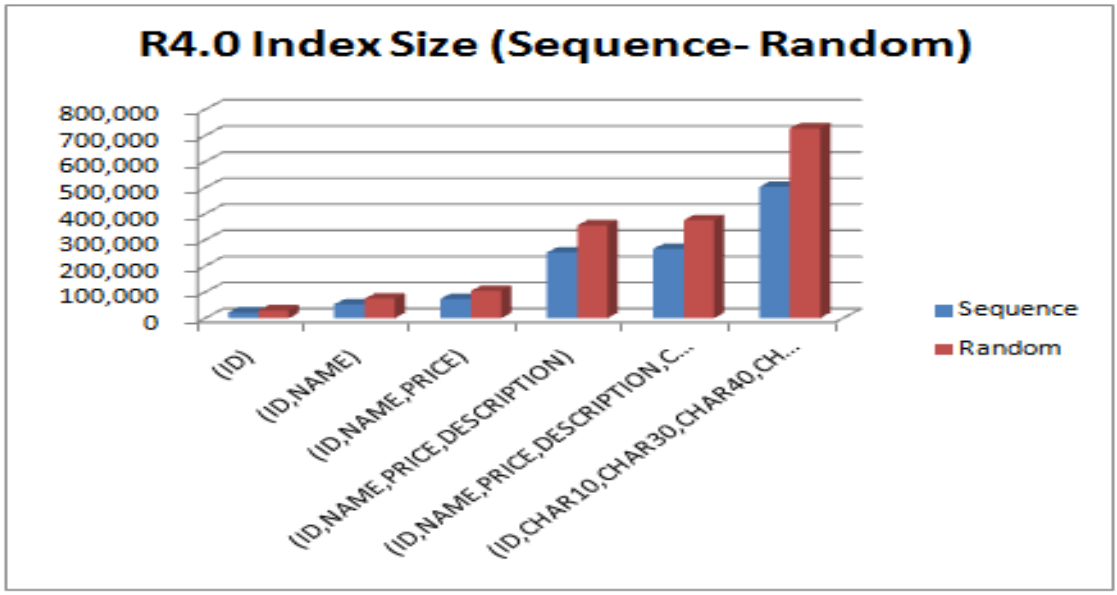
This test was performed to check index volume reducement amount in case of various index types and data generation types. Expecially we have focused on inserted data sequence, random and ordered. For more information about test scenarios, see the appendix. As shown in the table below, we have found that there was significant promotion in performance between CUBRID 2008 R4.0 Beta and CUBRID 2008 R3.1.

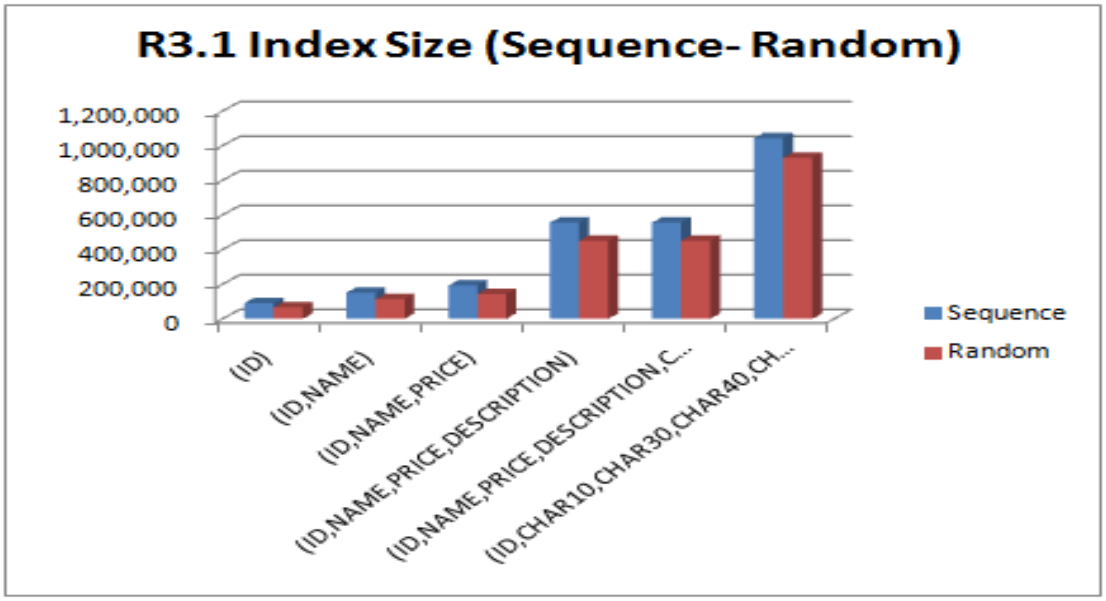
- Linux: Performance Comparison between CUBRID 2008 R3.1 and CUBRID 2008 R4.0 Beta (64-bit)
  On Linux 64-bit, the overall test results has significant change. Not only index size of CUBRID R4.0 Beta has significant save (about 50%) than that of CUBRID 2008 R3.1, but also run time has corresponding significant decrease (about 56%).
  In addition, as shown in last two table below, we have found that there was significant change between CUBRID 2008 R4.0 and CUBRID 2008 R3.1. For CUBRID 2008 R4.0, sequence mode has better performance of index size than random mode. But for CUBRID 2008 R3.1, random mode has better performance of index size than sequence mode.



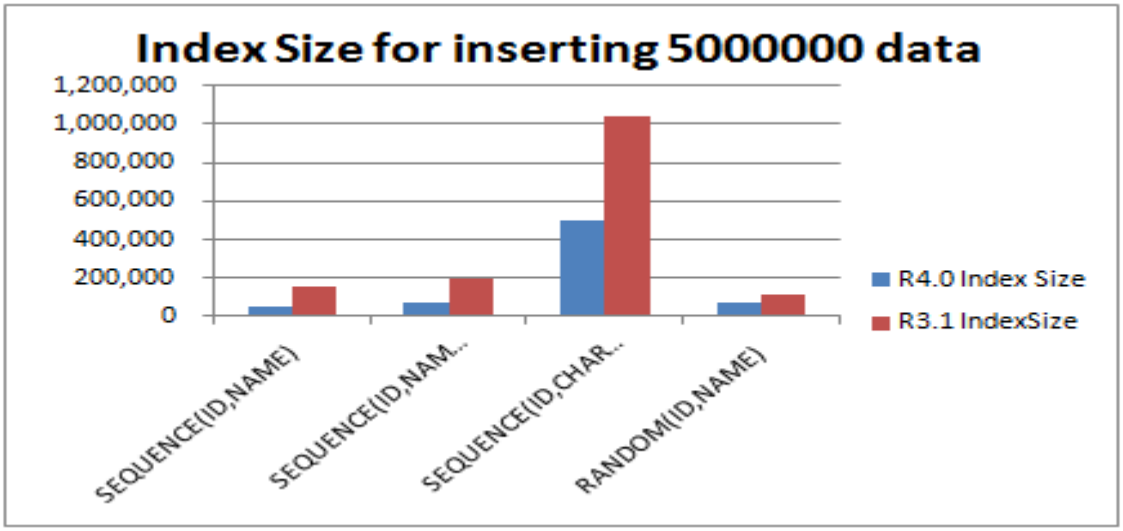
For above table, SEQUENCE(ID) means that data in column ID is inserted by sequence and ascending order. RANDOM(ID, NAME) means that data in column ID is inserted by random order and data in column NAME is inserted by sequence and ascending order. For more information about test scenarios, see the appendix.

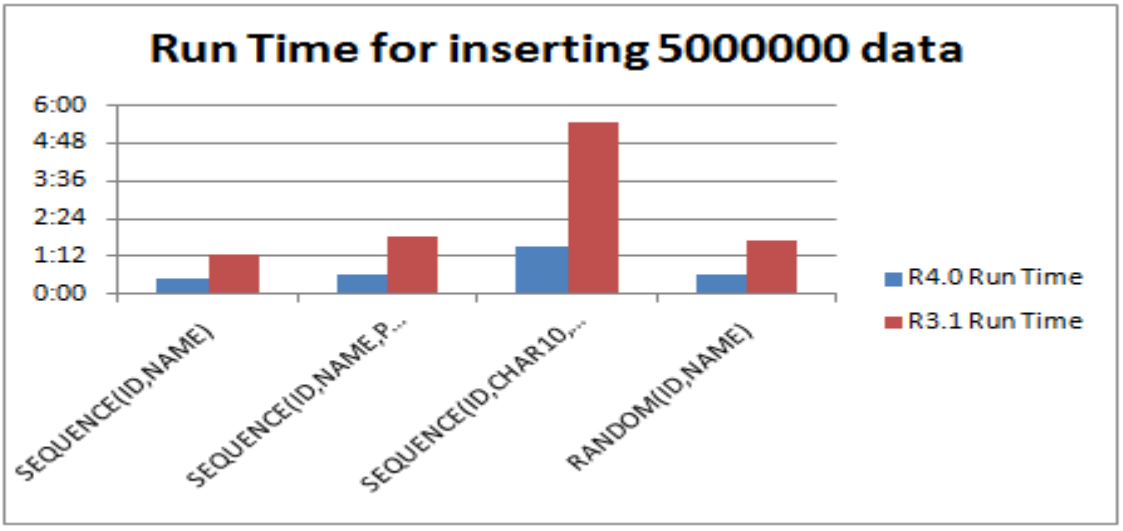




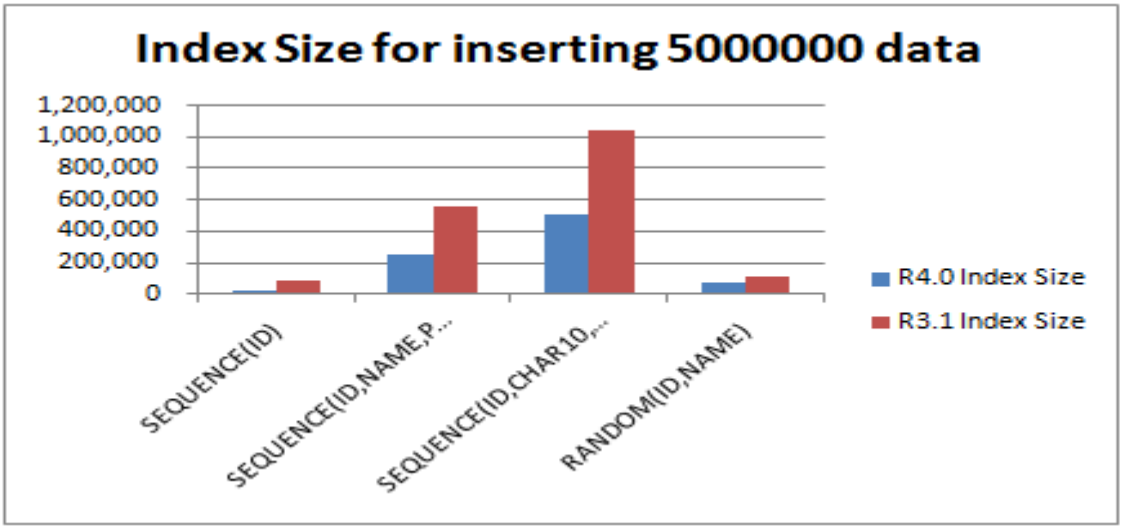


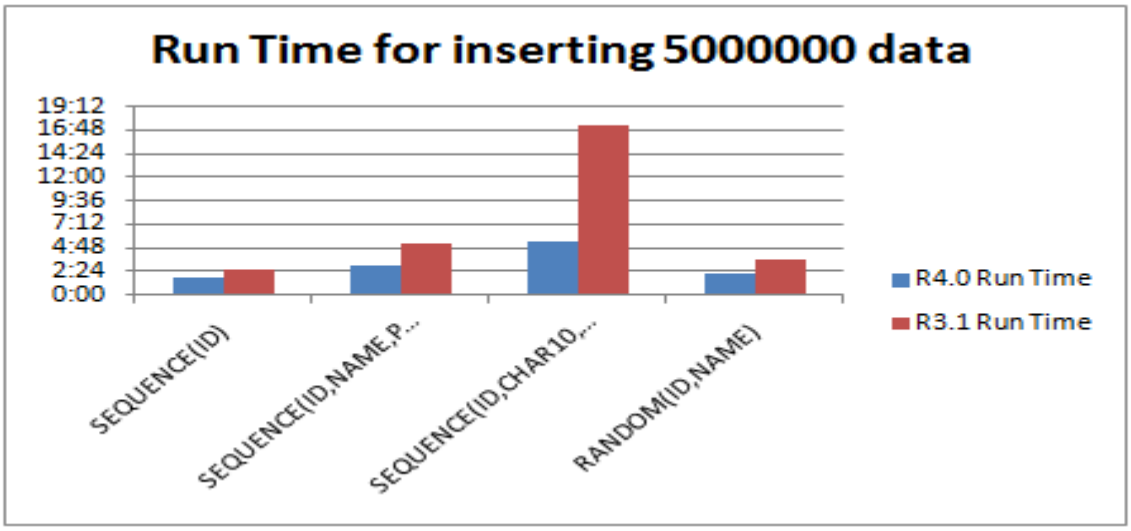
- Linux: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 Beta (32-bit)
  On Linux 32-bit, the overall test results were the same as the Linux 64-bit results, and no significant difference was found.



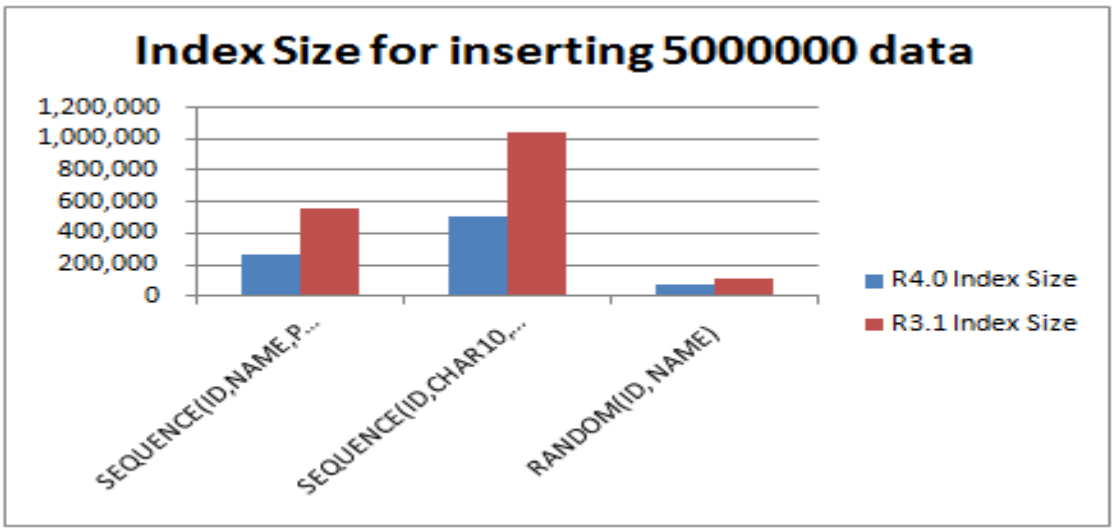


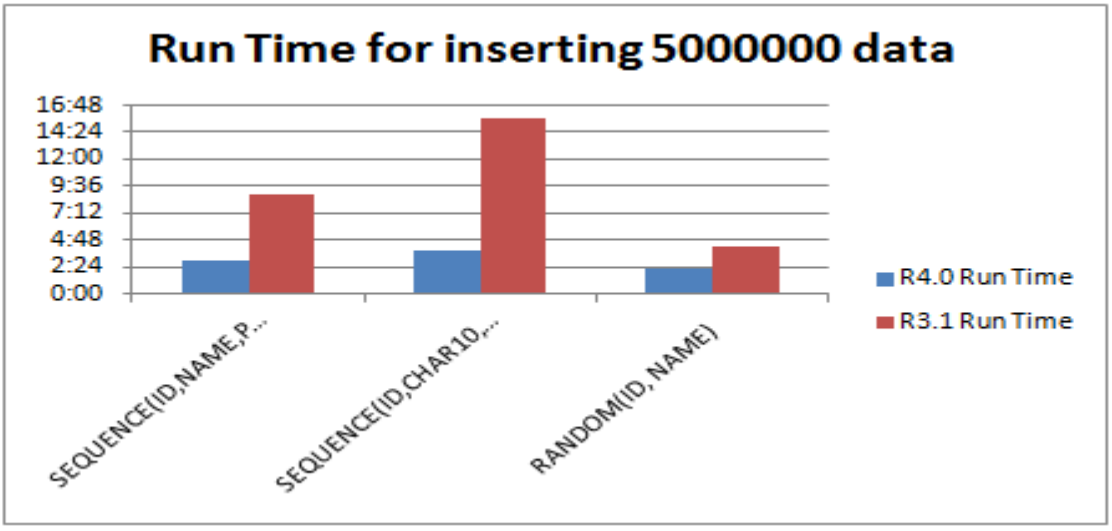
- Windows: Performance Comparison between CUBRID 2008 R3.1 (64-bit) and CUBRID 2008 R4.0 Beta (64-bit)
  On Windows 64-bit, the overall test results were the same as the Linux 64-bit results, and no significant difference was found.





- Windows: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 Beta (32-bit)
  On Windows 32-bit, the overall test results were the same as the Linux 64-bit results, and no significant difference was found.



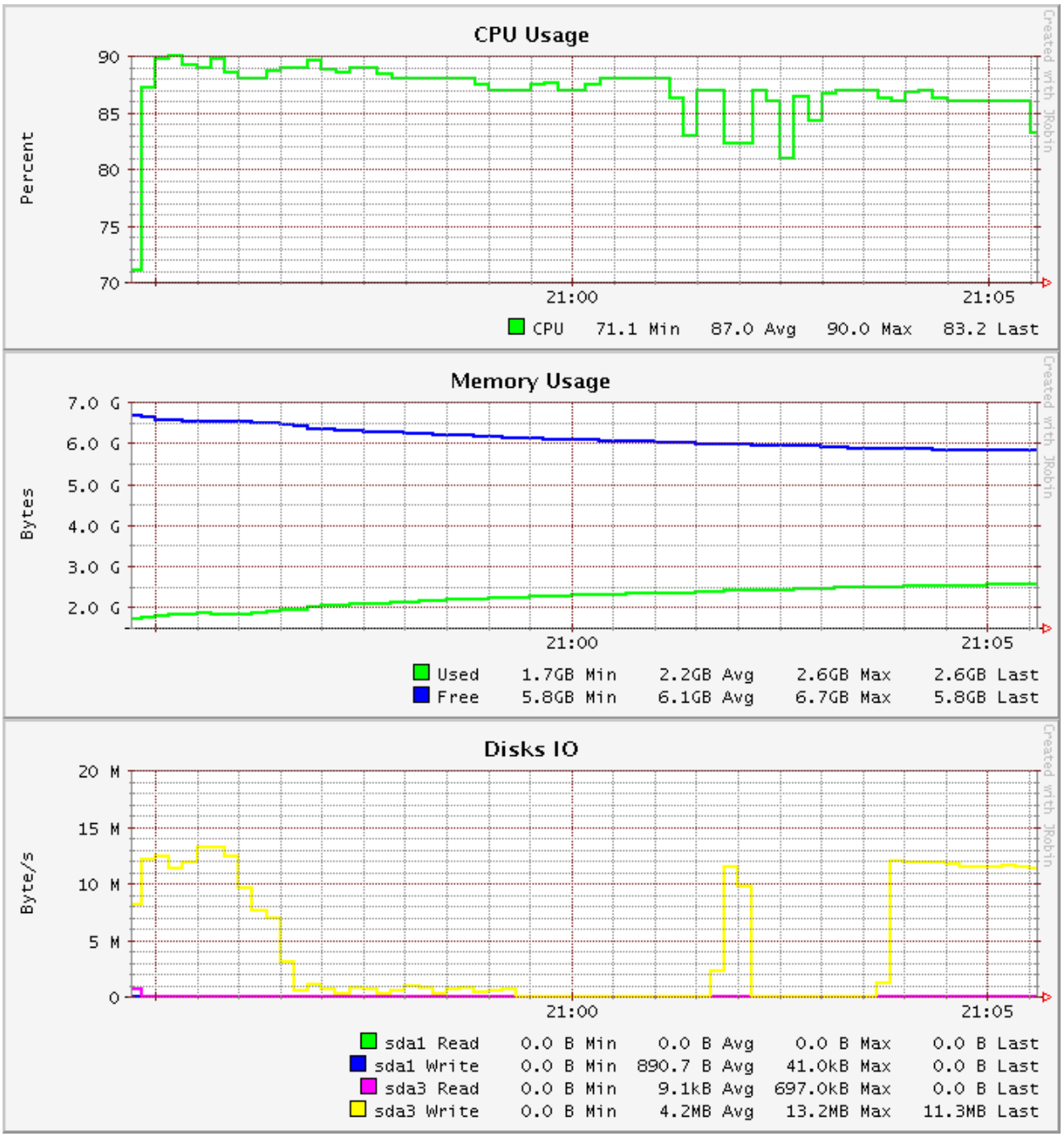


2.2.3 NBD Benchmark Performance Test

This test was performed to verify CUBRID performance by using the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. The scalability of the test DB was Level 1. As shown in the results below, there was no significant difference in NBD Benchmark test loads between R3.1 and R4.0 Beta.

Platform	Version	BIT	Page View
	CUBRID 2008 R3.1 (Default parameter configured)	64-bit	892
	CUBRID 2008 R3.1 (Default parameter configured)	32-bit	856
	CUBRID 2008 R4.0 Beta (Default parameter configured)	64-bit	876
	CUBRID 2008 R4.0 Beta (Default parameter configured)	32-bit	871

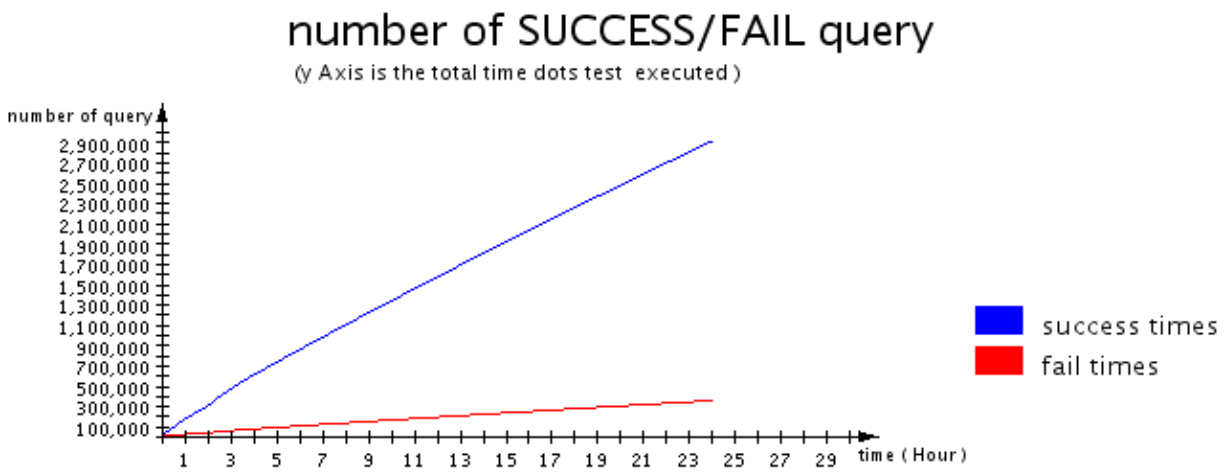
The following graphs represent the usage rate of each resource while processing the NBD benchmark test on Linux 64-bit.



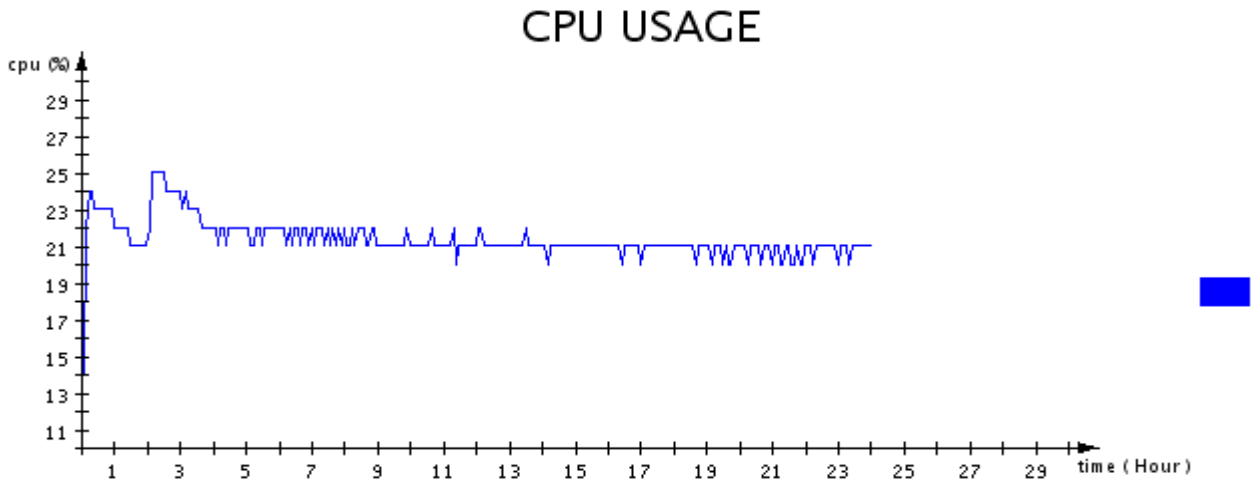
2.3 Stability Test Results

DOTS, a sub-project of an open project called "Linux Test Project," is an open test tool for testing the DBMS. For more information about DOTS, see the appendix. As shown in the test results below, the system operated stably without any abnormalities during the 24-hour load period. You can ignore the fails because they are unique violations due to the modification of duplicate data.

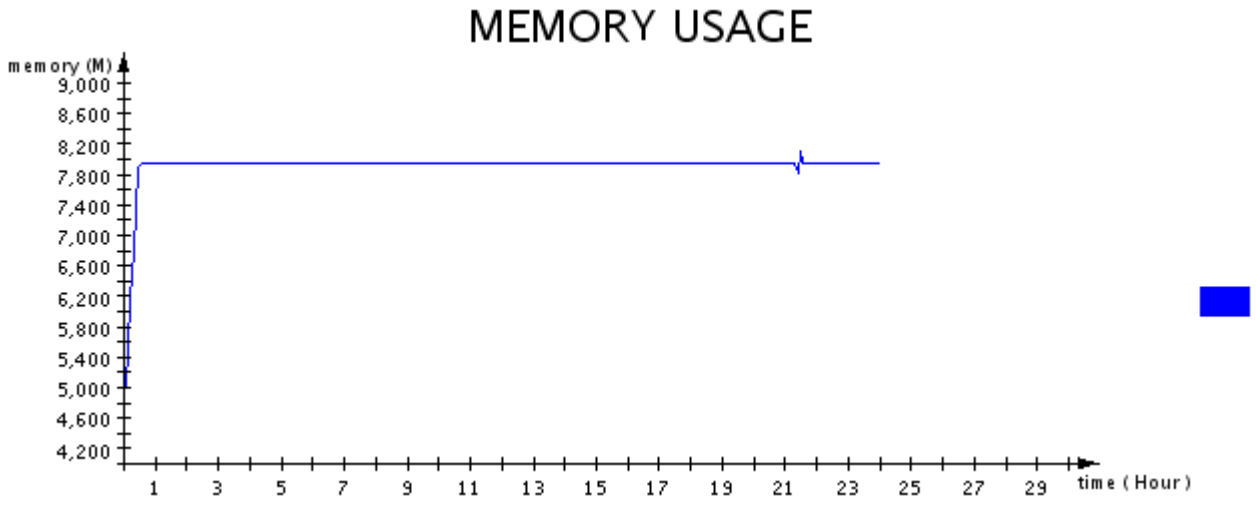
- Number of accumulated SQL queries



- CPU usage



- Memory usage



2.4 Other Test Results

All bug fixes resolved in CUBRID 2008 R4.0 Beta have been confirmed.

2.5 Quality Index

The standard quality index of CUBRID 2008 R4.0 Beta is listed below.

Quality Index Name	Project Quality Standard	Approved Quality Index during Implementation	Measurement Target	
Coding Standards Compliance Rate	100%	100%	Number of coding conventions observed in a project	56
			Number of coding conventions applied to each team	56
Code Review Execution Rate	100%	100%	Number of source code lines for which code review is performed.	868,083 LOC
			Total number of source code lines in the changed files	868,083 LOC
QA Scenario Code Coverage	75%	75.27%	Number of tested statements	181,448
			Total number of statements	241,072
Fault Density Detected by Static Analysis	4 /KLOC	2.4 /KLOC	Number of faults detected by static analysis (Level 1)	126
			Number of faults detected by static analysis (Level 2)	7
			Number of faults detected by static analysis (Level 3)	390
			Number of faults detected by static analysis (Level 4)	0
			Total number of source code lines	766,613 LOC
Cyclomatic Code Complexity	3.3%	2.65%	Number of modules whose complexity is over 30	599
			Total number of modules in a project	22,645
	12%	14.56%	Number of modules whose complexity is over 10	3,298
			Total number of modules in a project	22,645

3. Conclusions

As described in Chapters 1 and 2, CUBRID 2008 R4.0 Beta has been tested in terms of its functionality, performance, stability and other issues before its release.

The tests have been performed in the Linux 32-bit, Linux 64-bit, Windows 32-bit and Windows 64-bit environments. All tests were successful, and did not show any special issues for either platform type (32/64-bit) on Linux/Windows.

Based on the results obtained through the basic performance test, we have found that the overall basic performance of CUBRID 2008 R4.0 Beta was significant better than that of CUBRID 2008 R3.1. But select operation is lower about 5% than that of CUBRID 2008 R3.1 in only Linux 32/64-bit. It is needed to investigate it more in the future.

Based on the results obtained through the index volume performance test, we have found that the overall save of index volume size of CUBRID 2008 R4.0 Beta was better about 50% than that of CUBRID 2008 R3.1. Also, we have found that there was another significant change between CUBRID 2008 R4.0 and CUBRID 2008 R3.1. For CUBRID 2008 R4.0, sequence mode has better performance of index size than random mode.

Based on the results obtained through the NBD benchmark test, we have found that there was no significant change in performance between CUBRID 2008 R3.1 and CUBRID 2008 R4.0 Beta.

Appendix

I. Functionality Test Scenarios

This test was performed to verify the basic DBMS functionalities by using SQL statements. SQL statements stored in files were tested to verify DBMS conformity. We executed the stored SQL statements in a JDBC-based application, and compared the results to the stored reference file for verification. The scenario files included in the basic functionality test are stored in the SQL and MEDIUM directories of the CUBRID QA tool.

- SQL Query Test

Total: 8582		
Case Name	Path	**Description**
object	sql/_01_object	Performs functionality tests of objects supported by CUBRID, and has the largest number of scenarios (3200 scenarios).
user_authorization	sql/_02_user_authorization	Performs functionality tests of user and authorization management.
object_oriented	sql/_03_object_oriented	Performs tests for the object-oriented concept. CUBRID is an object-relational database management system (DBMS).
operator_function	sql/_04_operator_function	Performs functionality tests of basic functions and operators supported by CUBRID.
manipulation	sql/_06_manipulation	Performs tests of the insert, update, delete, and select statements, which are the most commonly used SQL statements in DML. Basic statements, subqueries and various join queries are tested.
misc	sql/_07_misc	Performs functionality tests of DCL (Data Control Language), including statistics update or other functionalities.
javasp	sql/_08_javasp	Performs functionality tests of Java stored procedures.
64-bit	sql/_09_64bit	Performs basic functionality test scenarios of the bigint and datetime types, which have been added in CUBRID 2008 R2.0.
Connect_by	sql/_10_connect_by	Performs a test of the hierarchical query feature, which has been added in CUBRID 2008 R3.1.
Codecoverage	sql/_11_codecoverage	Performs a test of uncovered codes based on the code coverage results, which has been newly added in CUBRID 2008 R4.0 Beta.
Syntax Extension	sql/_12_mysql_compatibility	Performs a test of the syntax extension, which has been added in CUBRID 2008 R3.1.
BTS issues	sql/_13_issues	Performs a test of known issues, which comes from issu e management system.
MySQL compatibility	sql/ _14_mysql_compatibility_2	Performs an unit test of the syntax extension 2, which has been added in CUBRID 2008 R4.0 Beta.
FBO	sql/ _15_fbo	Performs a test of the FBO feature, which has been added in CUBRID 2008 R3.1.
Index enhancement	sql/ _16_index_enhancement	Performs an unit test of the index enhancement, which has been added in CUBRID 2008 R4.0 Beta.
SQL Extension	sql/ _17_sql_extension2	Performs a test of the syntax extension 2, which has been added in CUBRID 2008 R4.0 Beta. Includes a test of syntax enhancements, system parameters, show statements, date/time functions, string functions,

		aggregate functions, other functions.
Index enhancement	sql/ _18_index_enhancement_qa	Performs a test of the index enhancement, which has been added in CUBRID 2008 R4.0 Beta. Includes a test of limit optimizing, using index clause enhancement, descending index scan, covering index, ordering index, optimizing group by clause, Index scan with like predicate, next key locking,etc.

- MEDIUM Query Test

Total: 970		
Case Name	**Path**	**Description**
01_fixed	medium/_01_fixed	Performs regression test scenarios for bug fixes that have been implemented since the initial version.
02_xtests	medium /_02_xtests	Performs test scenarios for functionalities supported by CUBRID, but not by other DBMSs.
03_full_mdb	medium /_03_full_mdb	Performs test scenarios for sequential/index scan queries with an index.
04_full	medium /_04_full	Performs test scenarios that include testing queries for limit values of CUBRID.
05_err_x	medium /_05_err_x	Performs negative test scenarios for functionalities that are supported by CUBRID, but not by other DBMSs.
06_fulltests	medium /_06_fulltests	Performs test scenarios for search queries with OIDs.
07_mc_dep	medium /_07_mc_dep	Includes a query that gives various conditions to a WHERE clause in the SELECT query, and tests whether or not a correct result has been selected.
08_mc_ind	medium/_08_mc_ind	Includes scenarios that test queries performing schema change.

- SITE Query Test

Total: 1215		
Case Name	**Path**	**Description**
k_count_q	site/k_count_q	Retrieves count (*) results of a query that is included in the kcc_q query.
k_merge_q	site/k_merge_q	Forces to give a hint to the kcc_q queries allowing merge joins.
k_q	site/k_q	Performs tests for OID reference, collection type, and path expression that are part of the object-oriented concept supported by CUBRID with different scalabilities. In addition, it performs functionality tests while increasing the number of join participating tables.
n_q	site/n_q	Performs tests for a complex query in which subqueries, outer/inner joins or group-by queries are combined, and checks whether correct results are retrieved.

- Utility (Shell) Test

  This test was performed to verify the basic DBMS functionalities by using shell scripts. In particular, this test was also performed to verify CUBRID utilities that cannot be tested by using SQL statements. We ran scenarios written using shell scripts to verify DBMS conformity.

Total: 472		
Case Name	**Path**	**Description**
utility	shell/_01_utility	Includes a script that tests the database management commands supported by CUBRID.
sqlx_init	shell/_02_sqlx_init	Includes scenarios that change the configuration of CUBRID DBMS parameters, and checks whether they are working correctly.
itrack	shell/_03_itrack	Includes scenarios that verify there is no regression by checking the bug fixes in CUBRID, and stores scenarios that cannot be tested by SQL.
addition	Shell/_05_addition	Includes scenarios added to improve code coverage and mainly tests the options of CUBRID utilities.
BTS issues	shell/_06_issues	Includes scenarios that verify there is no regression by checking the bug fixes in CUBRID, and stores scenarios that cannot be tested by SQL.
Index enhancement	shell/_07_index_enhancement	Includes scenarios that verify next key lock and change the configuration of CUBRID DBMS related to index enhancement, which has been added in CUBRID 2008 R4.0 Beta.
MySQL compatibility	shell/_23_mysql_compatibility	Includes scenarios that verify syntax extension, which has been added in CUBRID 2008 R3.1.

- HA Feature Test

Total: 37		
Case Name	**Path**	**Description**
Data replication test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed in a normal state with no fault.
Node fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a node fault occurs during insert/update/delete operations.
Process fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a process fault occurs that causes the database process to stop during insert/update/delete operations.
Broker fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a broker fault occurs during insert/update/delete operations.
Replication scenario	scripts/sql	Includes scenarios that test whether HA is working properly for each CUBRID transaction type, and has two sub directories: random_case and special_case

II. Performance Test Scenario

- CUBRID Basic Performance Test

  To evaluate the basic performance of DBMS, the following 5 variables were used. Database Server, Broker, and Load Generator were run on a single server.

  - Number of data (or number of program loops)
    - ✧ Total number of data: 900,000 items
    - ✧ Number of program loops: 100,000 loops/program (900,000 items)
      - ⬩ COMMIT Interval
        - After every execution
        - After 100 executions
        - After 1,000 executions
      - ⬩ Number of concurrent users
        - 5 users
        - 10 users
      - ⬩ Number of index attributes
        - create index idx1 on xoo(a)
        - create index idx2 on xoo(a,b)
        - create index idx3 on xoo(a,b,e)
      - ⬩ Interface
        - JDBC (Dynamic SQL): Prepared statements were used.

  - Test data
    - ✧ Test schema

```
CREATE TABLE xoo (
        a       int,
        b       int,
        c       int,
        d       int,
        e       char(10),
        f       char(20),
        g       char(30)
)
```

```
CREATE INDEX idx1 on xoo(a);
CREATE INDEX idx2 on xoo(a,b);
CREATE INDEX idx3 on xoo(a,b,e);
```

    - ✧ Test data

Enter data from 1 to 450,000; total number of data is 900,000.

✧ How to perform a test

- Insert/update/select/delete data from a specific number.
- For concurrent user tests, the start and end numbers are defined to prevent data from overlapping, in order to ensure that there is no competition between the concurrent clients.
- For concurrent user test programs, a JDBC test program is tested with a multi-threaded program, and a C program is tested with a multi-process program.
- If the number of loops is 10,000, a user repeats execution 10,000 times in the case of the 1-user test, and each user repeats execution 2,000 times in the case of the 5-user test. Similarly, if the number of loops is 100,000, a user repeats execution 100,000 times in the case of the 1-user test, and each user repeats execution 20,000 times in the case of the 5-user test.

✧ How to measure test results

- Measure the number of loops per second.
- For concurrent user tests, add the execution times of all users.

- CUBRID Index Volume Performance Test

To evaluate the index volume size and run time performance, below guide should be followed.

■ Purpose

- Compare the index size of CUBRID R4.0 Beta with CUBRID R3.1, to check whether the former's index size smaller to later.
- Compare the inserting speed of CUBRID R4.0 Beta with CUBRID R3.1, to check whether the former's inserting speed faster to later.
- After a larget amount of data inserting operation at CUBRID R4.0 Beta and R3.1, verify whether the index data is integrated and data don't lost or damaged

■ Test Requirements

- Modify CUBRID configuration info

```
data_buffer_pages=250000
temp_file_memory_size_in_pages=12
sort_buffer_pages=128
media_failure_support=no
```

- Create enough space for the index, the scripts are as below

```
cubrid addvoldb -p index testdb 1000000 -S (execute 6 times)
cubrid addvoldb -p data testdb 1000000 -S (execute 8 times)
cubrid addvoldb -p temp testdb 1000000 -S (execute 2 times)
```

- Test completion conditions
  - The index size of CUBRID R4.0 Beta will smaller than CUBRID R3.1 at same data size ,the system resource is normal.
  - After a larget amount of data inserting operation, the index data of CUBRID is integrated and data don't lost or damaged.
  - The inserting speed of CUBRID R4.0 Beta is faster than CUBRID R3.1.
  - For CUBRID R4.0 Beta, the index used space of order index larger than random index, For CUBRID R3.1 the index used space of order index smaller than random index.
  - In the frequently execute database operation (e.g. create table, insert, update,delete) ,execute "checkdb" & "compactdb" command, the result is ok.

- How to insert data for every index type
  - ✧ Overall

Build a dictionary data table, there are two column to decide the order of other column (ID,Random) ,if insert data for "sequence type index", I will select data order by ID colum, if insert data for "random type index", I will select data order by RANDOM column, the demo dictionary are as below. I only build 10 rows data for test report, but at real test environment, there are 5,000,000 row data.

ID	Random	name	price	createdate	description	char10	char30	char40	char100	char150
1	1	product-0-1	0	04/08/2011	[illegible]	ting two s	nse in which there is no oblig	pts the GPL v2.0 or later license, which	[illegible]	[illegible]
2	9	product-0-9	0.1	04/08/2011	[illegible]	want to c	cy Software Distribution licen	s no obligation of opening derivative wo	[illegible]	[illegible]
3	3	product-0-3	0.2	04/08/2011	[illegible]	al databas	oping and distributing various	are Distribution license in which there	[illegible]	[illegible]
4	4	product-0-4	0.3	04/08/2011	[illegible]	sive open	o obligation of opening deriva	e, which allows distribution, modificati	[illegible]	[illegible]
5	5	product-0-5	0.4	04/08/2011	[illegible]	anagement	ive works. The reason of adopt	and provide excellent cost savings to c	[illegible]	[illegible]
6	2	product-0-2	0.5	04/08/2011	[illegible]	e source c	various CUBRID based applicat	e the Berkeley Software Distribution lic	[illegible]	[illegible]
7	10	product-0-10	0.6	04/08/2011	[illegible]	dent softw	ter license, which allows dist	ensive open source relational database m	[illegible]	[illegible]
8	8	product-0-8	0.7	04/08/2011	[illegible]	services	BRID does not want to create a	ng two separate license systems is that	[illegible]	[illegible]
9	7	product-0-7	0.8	04/08/2011	[illegible]	ribute or	mized for Web applications, es	o different license policies for interfa	[illegible]	[illegible]
10	6	product-0-6	0.9	04/08/2011	[illegible]	ution lice	ns. The CUBRID license policy	CUBRID does not want to create any limit	[illegible]	[illegible]
SEQUENCE	RANDOM	RANDOM	SEQUENCE	SEQUENCE	SEQUENCE	RANDOM	RANDOM	RANDOM	RANDOM	RANDOM

How to generate test data for each case.

- SEQUENCE(ID)

ID
1
2
3
4
5
6
7
8
9
10
SEQUENCE

- SEQUENCE(ID,NAME)

ID	name
1	product-0-1
2	product-0-9
3	product-0-3
4	product-0-4
5	product-0-5
6	product-0-2
7	product-0-10
8	product-0-8
9	product-0-7
10	product-0-6
SEQUENCE	RANDOM

- SEQUENCE(ID,NAME,PRICE)

ID	name	price
1	product-0-1	0
2	product-0-9	0.1
3	product-0-3	0.2
4	product-0-4	0.3
5	product-0-5	0.4
6	product-0-2	0.5
7	product-0-10	0.6
8	product-0-8	0.7
9	product-0-7	0.8
10	product-0-6	0.9
SEQUENCE	RANDOM	SEQUENCE

- SEQUENCE(ID,NAME,PRICE,DESCRIPTION)

ID	name	price	description
1	product-0-1	0	----------
2	product-0-9	0.1	----------
3	product-0-3	0.2	----------
4	product-0-4	0.3	----------
5	product-0-5	0.4	----------
6	product-0-2	0.5	----------
7	product-0-10	0.6	----------
8	product-0-8	0.7	----------
9	product-0-7	0.8	----------
10	product-0-6	0.9	----------
SEQUENCE	RANDOM	SEQUENCE	RANDOM

- SEQUENCE(ID,NAME,PRICE,DESCRIPTION,CREATEDATE)

ID	name	price	createdate	description
1	product-0-1	0	04/08/2011	----------
2	product-0-9	0.1	04/08/2011	----------
3	product-0-3	0.2	04/08/2011	----------
4	product-0-4	0.3	04/08/2011	----------
5	product-0-5	0.4	04/08/2011	----------
6	product-0-2	0.5	04/08/2011	----------
7	product-0-10	0.6	04/08/2011	----------
8	product-0-8	0.7	04/08/2011	----------
9	product-0-7	0.8	04/08/2011	----------
10	product-0-6	0.9	04/08/2011	----------
SEQUENCE	RANDOM	SEQUENCE	SEQUENCE	RANDOM

- SEQUENCE(ID,CHAR10,CHAR30,CHAR40,CHAR100,CHAR150)

ID	char10	char30	char40	char100	char150
1	ting two s	nse in which there is no oblig	ps the GPL v2.0 or later licenses, which	----------	----------
2	want to c	ey Software Distribution licen	s no obligation of opening derivative wo	----------	----------
3	al databas	oping and distributing various	are Distribution licenses in which there	----------	----------
4	sive open	o obligation of opening deriva	e, which allows distribution, modificati	----------	----------
5	anagement	ive works. The reason of adopt	and provide excellent cost savings to c	----------	----------
6	e source c	various CUBRID based applicat	e the Berkeley Software Distribution lic	----------	----------
7	dent softw	ter license, which allows dist	ensive open source relational database m	----------	----------
8	services	BRID does not want to create a	ng two separate licensing systems is that	----------	----------
9	ribute or	mized for Web applications, es	o different license policies for interfa	----------	----------
10	ution lice	ns. The CUBRID license policy	CUBRID does not want to create any limit	----------	----------
SEQUENCE	RANDOM	RANDOM	RANDOM	RANDOM	RANDOM

- RANDOM(ID)

ID
1
6
3
4
5
10
9
8
2
7
RANDOM

- RANDOM(ID,NAME)

ID	name
1	product-0-1
6	product-0-2
3	product-0-3
4	product-0-4
5	product-0-5
10	product-0-6
9	product-0-7
8	product-0-8
2	product-0-9
7	product-0-10
RANDOM	SEQUENCE

- RANDOM(ID,NAME,PRICE)

ID	name	price
1	product-0-1	0
6	product-0-2	0.5
3	product-0-3	0.2
4	product-0-4	0.3
5	product-0-5	0.4
10	product-0-6	0.9
9	product-0-7	0.8
8	product-0-8	0.7
2	product-0-9	0.1
7	product-0-10	0.6
RANDOM	SEQUENCE	RANDOM

- RANDOM(ID,NAME,PRICE,DESCRIPTION)

ID	name	price	description
1	product-0-1	0	
6	product-0-2	0.5	
3	product-0-3	0.2	
4	product-0-4	0.3	
5	product-0-5	0.4	
10	product-0-6	0.9	
9	product-0-7	0.8	
8	product-0-8	0.7	
2	product-0-9	0.1	
7	product-0-10	0.6	
RANDOM	SEQUENCE	RANDOM	SEQUENCE

- RANDOM(ID,CHAR10,CHAR30,CHAR40,CHAR100,CHAR150)

ID	char10	char30	char40	char100	char150
1	ting two s	nse in which there is no oblig	pes the GPL v2.0 or later licenses, which		
6	e source c	various CUBRID based applicat	s the Berkeley Software Distribution lic		
3	al databas	oping and distributing various	ses Distribution licenses in which there		
4	sive open	o obligation of opening deriva	e, which allows distribution, modificati		
5	anagement	ive works. The reason of adopt	and provide excellent cost savings to c		
10	ution lice	ns. The CUBRID license policy	CUBRID does not want to create any limit		
9	ribute or	mized for Web applications, es	o different license policies for interfa		
8	services	BRID does not want to create a	ng two separate licenses systems is that		
2	want to c	ey Software Distribution licen	s no obligation of opening derivative wo		
7	dent softw	ter license, which allows dist	ensive open source relational database m		
RANDOM	RANDOM	RANDOM	RANDOM	RANDOM	RANDOM

- RANDOM(ID,NAME,PRICE,DESCRIPTION,CREATEDATE)

ID	name	price	createdate	description
1	product-0-1	0	04/08/2011	
6	product-0-2	0.5	04/08/2011	
3	product-0-3	0.2	04/08/2011	
4	product-0-4	0.3	04/08/2011	
5	product-0-5	0.4	04/08/2011	
10	product-0-6	0.9	04/08/2011	
9	product-0-7	0.8	04/08/2011	
8	product-0-8	0.7	04/08/2011	
2	product-0-9	0.1	04/08/2011	
7	product-0-10	0.6	04/08/2011	
RANDOM	SEQUENCE	RANDOM	RANDOM	SEQUENCE

- NBD Benchmark

  This test was performed to verify CUBRID performance by using the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. For more information about NBD Benchmark, see separate documents.

III. Stability Test Scenario

DOTS, a sub-project of an open project called "Linux Test Project," is an open test tool for testing the DBMS.

- Test Related Schema (the Number of Data in Each Table)

```
CREATE TABLE REGISTRY (
    USERID              CHAR(15) NOT NULL PRIMARY KEY,
    PASSWD              CHAR(10),
    ADDRESS             CHAR(200),
    EMAIL               CHAR(40),
    PHONE               CHAR(15)
);

CREATE TABLE ITEM (
    ITEMID              CHAR(15) NOT NULL PRIMARY KEY,
    SELLERID            CHAR(15) NOT NULL,
    DESCRIPTION         VARCHAR(250) ,
    BID_PRICE           FLOAT,
    START_TIME          DATE,
    END_TIME            DATE,
    BID_COUNT           INTEGER
);

CREATE TABLE BID (
    ITEMID              CHAR(15) NOT NULL PRIMARY KEY,
    BIDERID             CHAR(15) NOT NULL,
    BID_PRICE           FLOAT,
    BID_TIME            DATE
);
```

- Data Size and How to Create Data

  The initial number of data when starting the test is 0. Enter 1000 of data in the REGISTRY table. Next, enter 100 of data in the ITEM table as well as in the bid table. Then, update 100 times.

- Transaction types

  - ✧ INSERT transaction 1

```
INSERT INTO ITEM (ITEMID,SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT)
VALUES (?, ?, ? ,?, ?, ?, ?)
```

  - ✧ INSERT transaction 2

```
INSERT INTO BID (ITEMID,BIDERID,BID_PRICE,BID_TIME)
VALUES (?, ?, ?, ?)
```

  - ✧ SELECT transaction 1

```
SELECT SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT
FROM ITEM WHERE ITEMID = ?
```

  - ✧ SELECT transaction 2

```
SELECT BIDERID, BID_PRICE, BID_TIME FROM BID WHERE ITEMID = ?
SELECT BIDERID, BID_PRICE, BID_TIME FROM BID WHERE ITEMID = ?
```

- ✧ UPDATE transaction 1

```
SELECT SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT
FROM ITEM WHERE ITEMID =
UPDATE ITEM SET DESCRIPTION = ?,BID_PRICE = ?,START_TIME = ?,END_TIME = ? WHERE ITEMID = ?
```

- How to Generate Load
  - ✧ How to generate load

    Use two threads to generate the initial load. Each thread repeats the insert/select/update queries mentioned above. The DOTS program checks CPU usage every 5 minutes. If the Peak CPU usage does not exceed 100%, the test continues, by adding two more threads.

IV. Scenario-based Code Coverage Results

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level	Lines:	181448	241072	75.3 %
Test:	Code Coverage	Functions:	9209	10363	88.9 %
Date:	2011-04-26	Branches:	110173	187313	58.8 %
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %				

Directory	Line Coverage			Functions		Branches	
/home/xdbms/build/src/executables		71.3 %	1433 / 2009	93.4 %	57 / 61	60.0 %	452 / 753
/home/xdbms/build/src/parser		91.5 %	7461 / 8152	98.7 %	76 / 77	54.2 %	2028 / 3742
src/base		73.6 %	6318 / 8581	88.1 %	452 / 513	52.8 %	3710 / 7022
src/broker		74.8 %	8290 / 11086	91.5 %	473 / 517	56.7 %	4269 / 7531
src/cci		69.5 %	3402 / 4895	82.1 %	216 / 263	55.0 %	1502 / 2733
src/communication		70.9 %	5724 / 8074	77.1 %	296 / 384	42.5 %	1701 / 3999
src/connection		76.1 %	2718 / 3572	88.3 %	241 / 273	58.3 %	1223 / 2096
src/executables		70.6 %	12481 / 17687	83.7 %	786 / 939	53.2 %	6857 / 12894
src/heaplayers		88.3 %	68 / 77	100.0 %	11 / 11	62.5 %	20 / 32
src/jsp		83.8 %	898 / 1071	100.0 %	68 / 68	63.7 %	344 / 540
src/object		76.4 %	21559 / 28205	88.6 %	1668 / 1883	57.6 %	14866 / 25812
src/optimizer		88.9 %	8522 / 9587	98.3 %	355 / 361	76.7 %	6540 / 8530
src/parser		81.6 %	27561 / 33766	92.9 %	1165 / 1254	67.5 %	18925 / 28025
src/query		75.1 %	31918 / 42493	92.4 %	1275 / 1380	61.6 %	22490 / 36524
src/session		71.0 %	519 / 731	93.6 %	44 / 47	52.6 %	273 / 519
src/storage		71.5 %	23414 / 32768	89.4 %	1125 / 1259	56.2 %	12959 / 23055
src/thread		72.1 %	1125 / 1560	92.0 %	80 / 87	57.1 %	507 / 888
src/transaction		67.4 %	18037 / 26758	83.3 %	821 / 986	50.9 %	11507 / 22618