

---

# CUBRID 2008 R4.0 QA Completion Report

---

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

---

---

# .Table of Contents

---

CUBRID 2008 R4.0 QA Completion Report	1
1. Test Overview	3
1.1 Test Objectives	4
1.2 Test Environment	4
1.2.1 TEST PROCEDURES	4
1.2.2 HARDWARE TEST ENVIRONMENT	6
1.3 Test Category	6
2. Test Results	8
2.1 Functionality Test Results	9
2.1.1 BASIC QUERY TEST	9
2.1.2 BASIC UTILITY AND OTHER SCENARIO TESTS	9
2.1.3 HA FEATURE TEST	9
2.2 Performance Test Results	10
2.2.1 CUBRID BASIC PERFORMANCE TEST	10
2.2.2 CUBRID INDEX VOLUME PERFORMANCE TEST	12
2.2.3 NBD BENCHMARK PERFORMANCE TEST	17
2.3 Stability Test Results	18
2.4 Other Test Results	20
2.5 Quality Index	21
3. Conclusions	22
Appendix	24
I. Functionality Test Scenarios	25
II. Performance Test Scenario	28
III. Stability Test Scenario	35
IV. Scenario-based Code Coverage Results	37

---

# 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 (hereinafter referred to as R4.0), which is under development for release in June 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 and that of CUBRID 2008 R3.1 (hereinafter referred to as R3.1), we tested to determine whether the performance of R4.0 was regressed or improved.

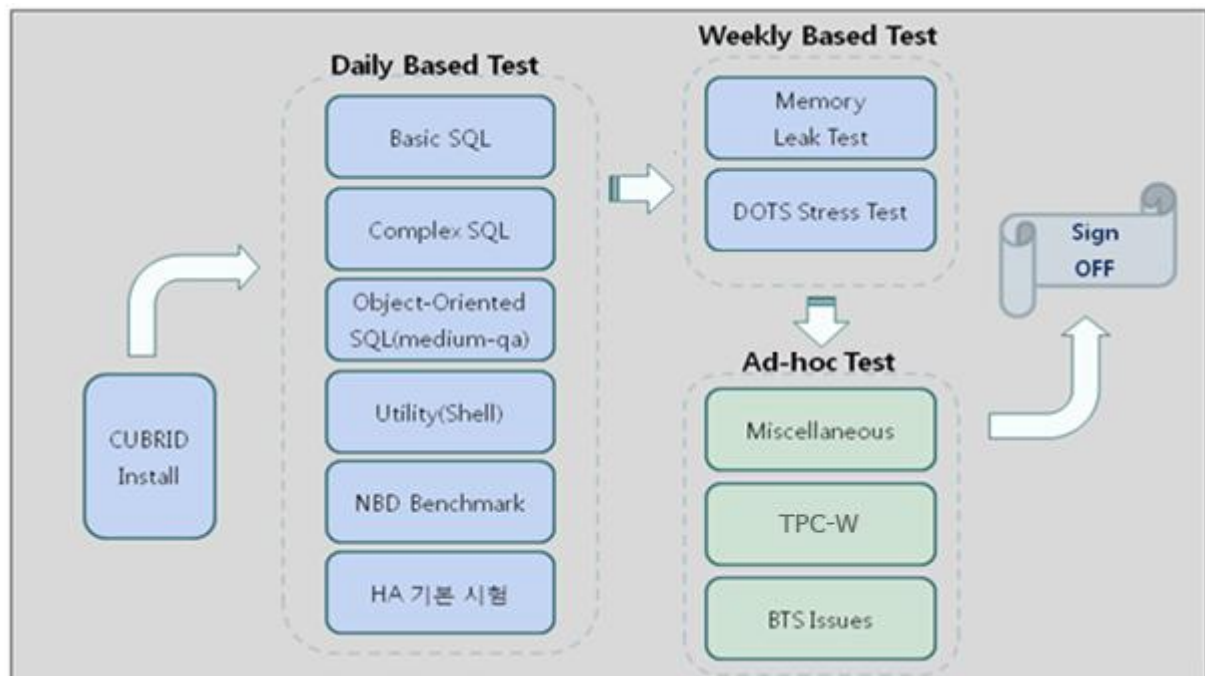
- CentOS 4.6 (32/64-bit) or compatible
- Windows 2003 (32/64-bit) or compatible
- Final test build: 8.4.0.0241 (Linux 64bit/32bit, Windows 64bit/32bit)

## 1.2 Test Environment

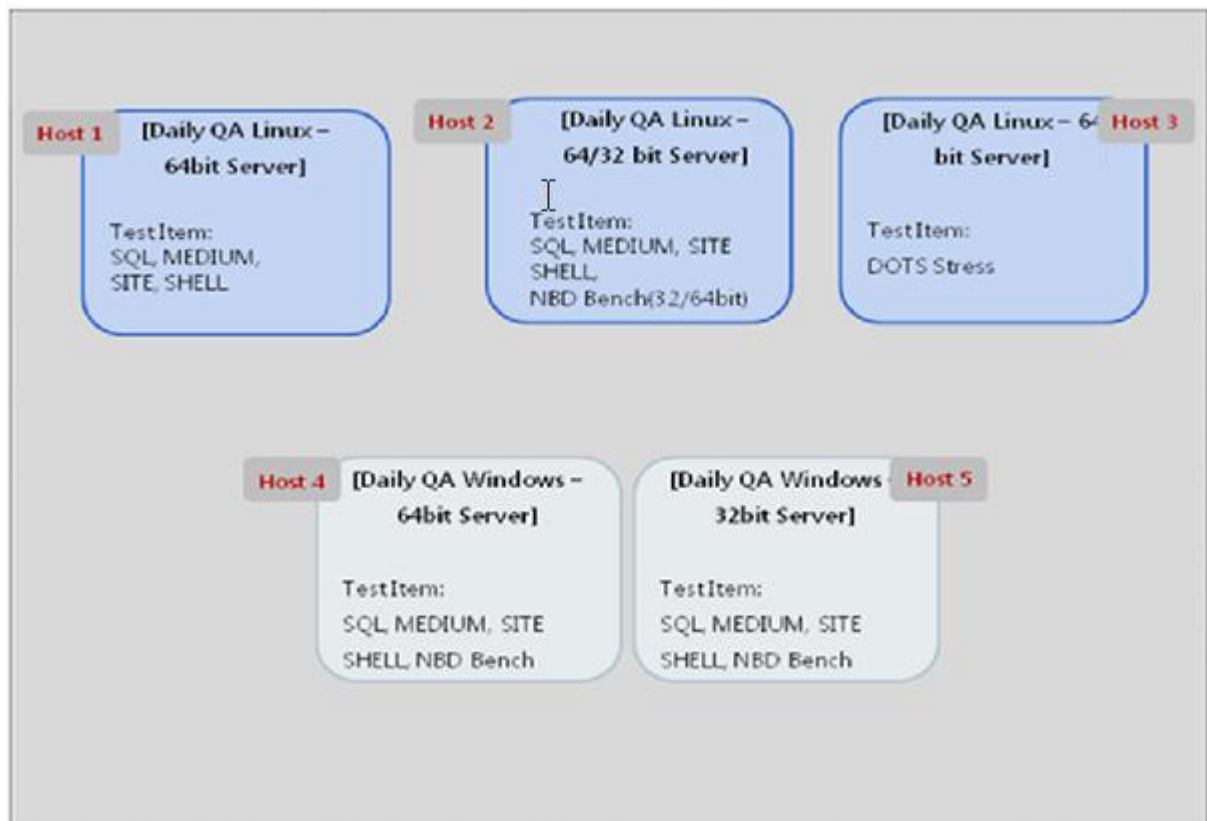
### 1.2.1 Test Procedures

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.

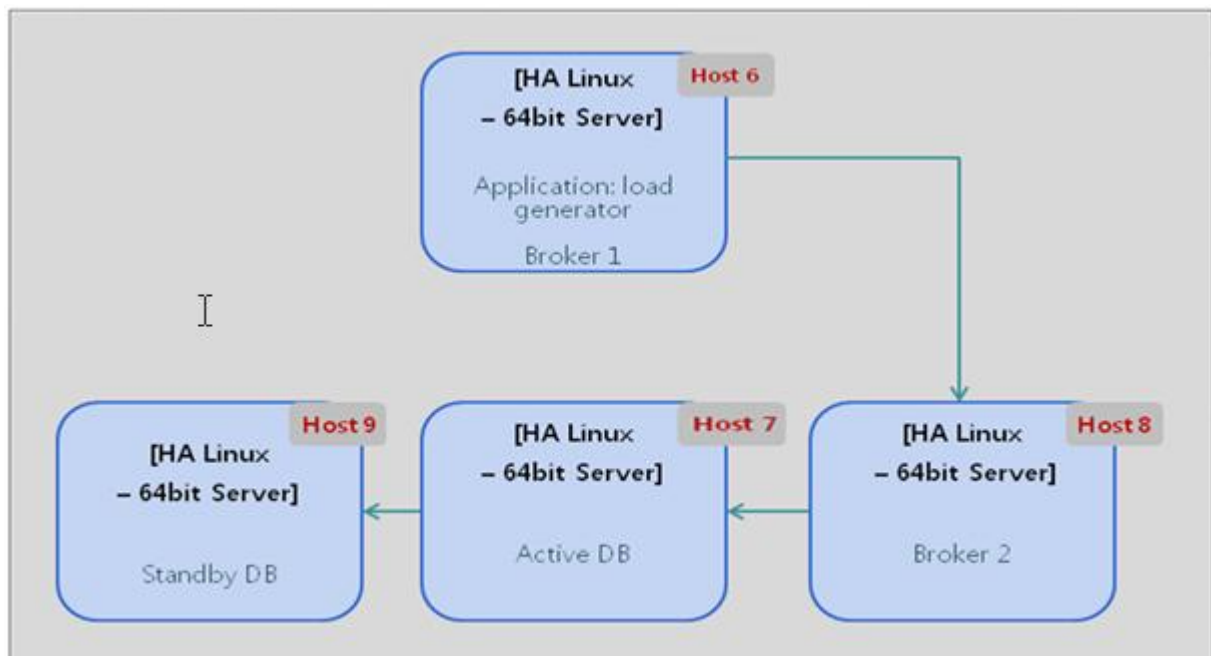
- CUBRID test procedures



- 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
<b>Host 1</b>	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (dual-core) * 1	4 GB	SATA 500G * 2 (No Raid)
<b>Host 2</b>	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
<b>Host 3</b>	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
<b>Host 4</b>	Windows 2003 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
<b>Host 5</b>	Windows 2003 (32-bit)	Xeon 2.10 GHz (quadcore) * 1	4 GB	SATA 500G * 2 (No Raid)
<b>Host 6</b>	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
<b>Host 7</b>	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
<b>Host 8</b>	Cent OS 4.6 (64-bit)	Xeon 2.10 GHz (quadcore) * 1	8 GB	SATA 500G * 2 (No Raid)
<b>Host 9</b>	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 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 10778 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
<b>SQL query test</b>	8595	8595	100%
<b>MEDIUM query test</b>	970	970	100%
<b>SITE query test</b>	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 498 shell scripts to verify DBMS conformity.

Test Category	Total Number of Scenarios	Number of Successful Tests	Success Rate
<b>Utility</b>	195	195	100%
<b>Bug regression</b>	206	206	100%
<b>Environment variable</b>	7	7	100%
<b>Other</b>	90	90	100%

### 2.1.3 HA Feature Test

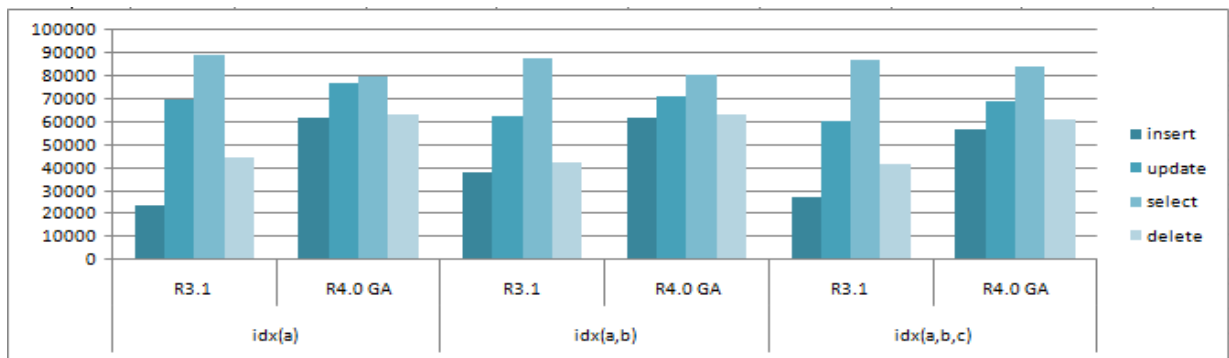
Test Category	Total Number of Scenarios	Number of Successful Tests	Success Rate
<b>Data replication test</b>	5	5	100%
<b>Node fault test</b>	16	16	100%
<b>Process fault test</b>	8	8	100%
<b>Broker fault test</b>	8	8	100%
<b>Run replication test scenarios</b>	115	115	100%

## 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 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 (64-bit)

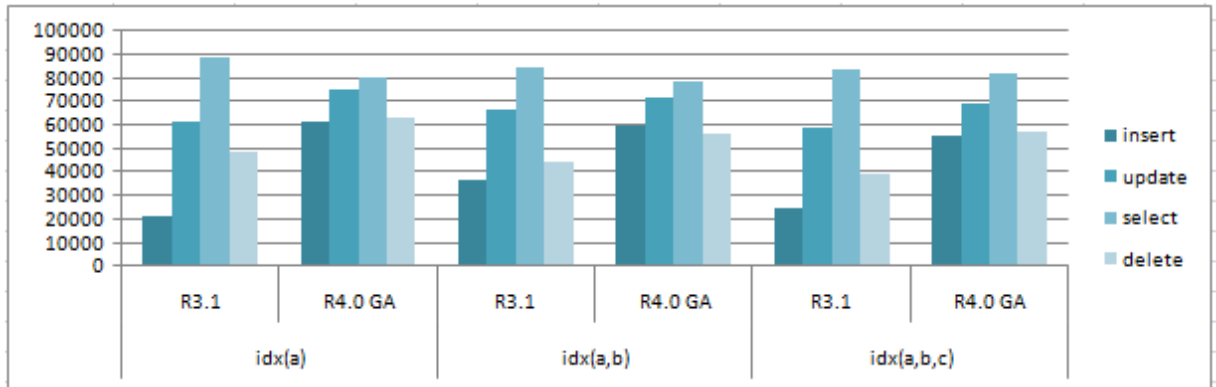


(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	R4.0	Perfor mance Ratio	R3.1	R4.0	Perfor mance Ratio	R3.1	R4.0	Perfor mance Ratio
Insert	23307	61824	265%	37841	61298	162%	27167	56288	207%
Update	69208	76940	111%	62501	70744	113%	60543	69128	114%
Select	88837	79279	89%	87429	80285	92%	87037	84120	97%
Delete	44662	63237	142%	41905	62790	150%	41476	61091	147%
Total	226014	281280	124%	229676	275117	120%	216223	270627	125%

- Linux: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 (32-bit)

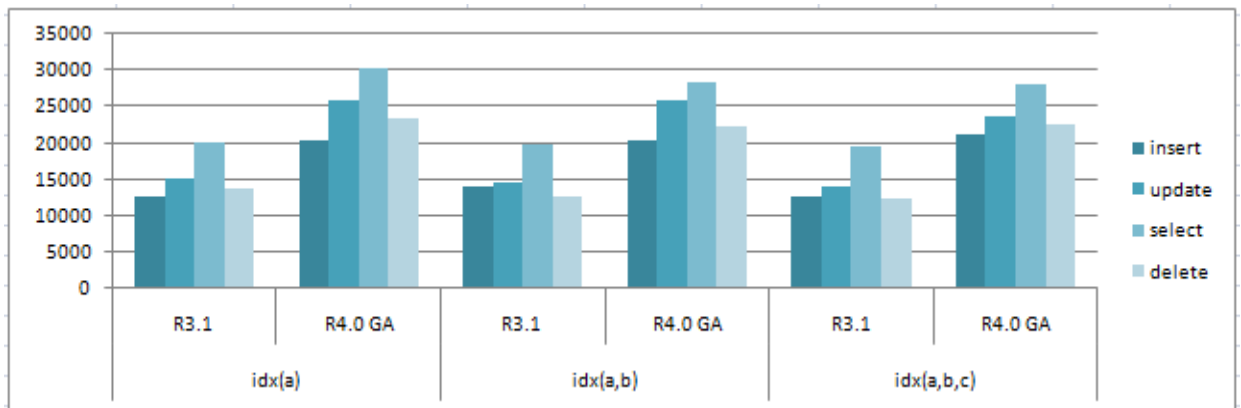
Similarly to the results for the 64-bit tests, we have found that there was significant change in performance between 32-bit CUBRID 2008 R3.1 and 32-bit CUBRID 2008 R4.0 except select operation. Select performance of 32/64-bit CUBRID 2008 R4.0 is lower about 5% than 32/64-bit CUBRID 2008 R3.1. It is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



(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	R4.0	Perfor mance Ratio	R3.1	R4.0	Perfor mance Ratio	R3.1	R4.0	Perfor mance Ratio
Insert	21462	61408	286%	36786	59842	163%	24536	55626	227%
Update	60860	74664	123%	66270	71915	109%	58413	69345	119%
Select	88571	80035	90%	83899	78620	94%	83438	81370	98%
Delete	48514	62795	129%	44048	56320	128%	39227	57325	146%
Total	219407	278902	127%	231003	266697	115%	205614	263666	128%

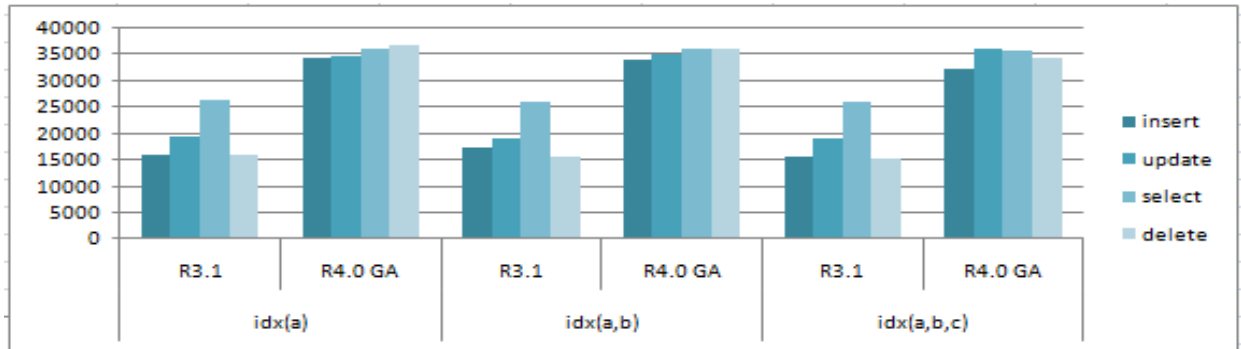
- Windows: Performance Comparison between CUBRID 2008 R3.1 (64-bit) and CUBRID 2008 R4.0 (64-bit)



(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	R4.0	Perfor mance Ratio	R3.1	R4.0	Perfor mance Ratio	R3.1	R4.0	Perfor mance Ratio
Insert	12651	20305	160%	14113	20354	144%	12618	21128	167%
Update	14969	25776	172%	14402	25702	178%	14053	23674	168%
Select	20039	30216	151%	19837	28385	143%	19511	28065	144%
Delete	13757	23279	169%	12599	22193	176%	12225	22506	184%
Total	61416	99575	162%	60951	96634	159%	58408	95373	163%

- Windows: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 (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.



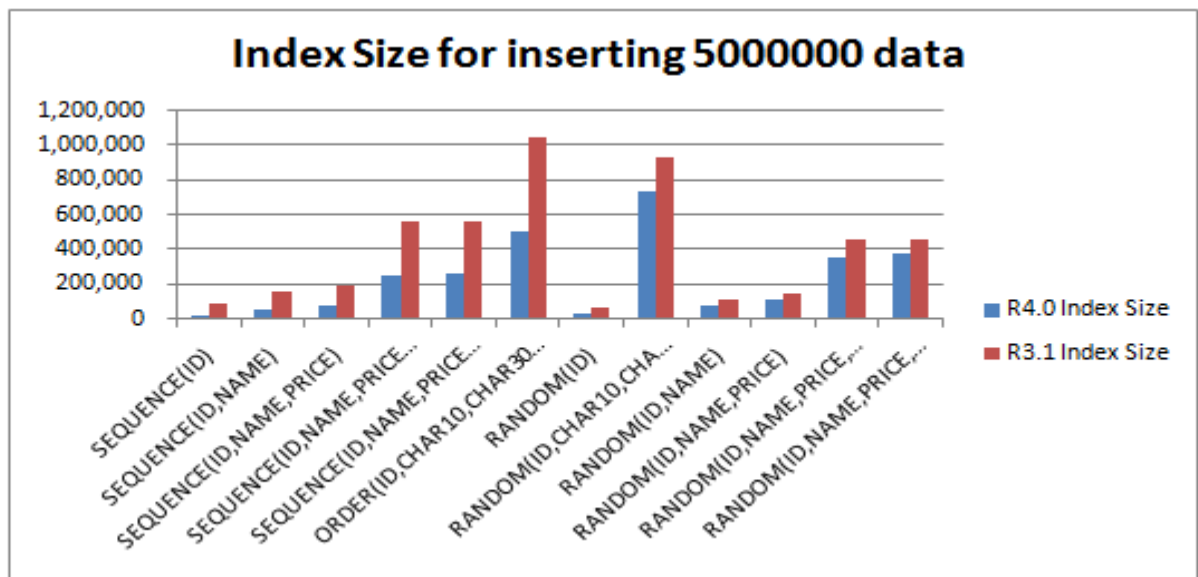
(Unit: TPS)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R3.1	R4.0	Performance Ratio	R3.1	R4.0	Performance Ratio	R3.1	R4.0	Performance Ratio
Insert	15846	34405	217%	17270	33848	196%	15537	32118	207%
Update	19411	34727	179%	19194	34838	182%	19156	35871	187%
Select	26214	35923	137%	25977	36133	139%	26031	35815	138%
Delete	15760	36640	232%	15392	36141	235%	15060	34174	227%
Total	77231	141695	183%	77833	140960	181%	75784	137977	182%

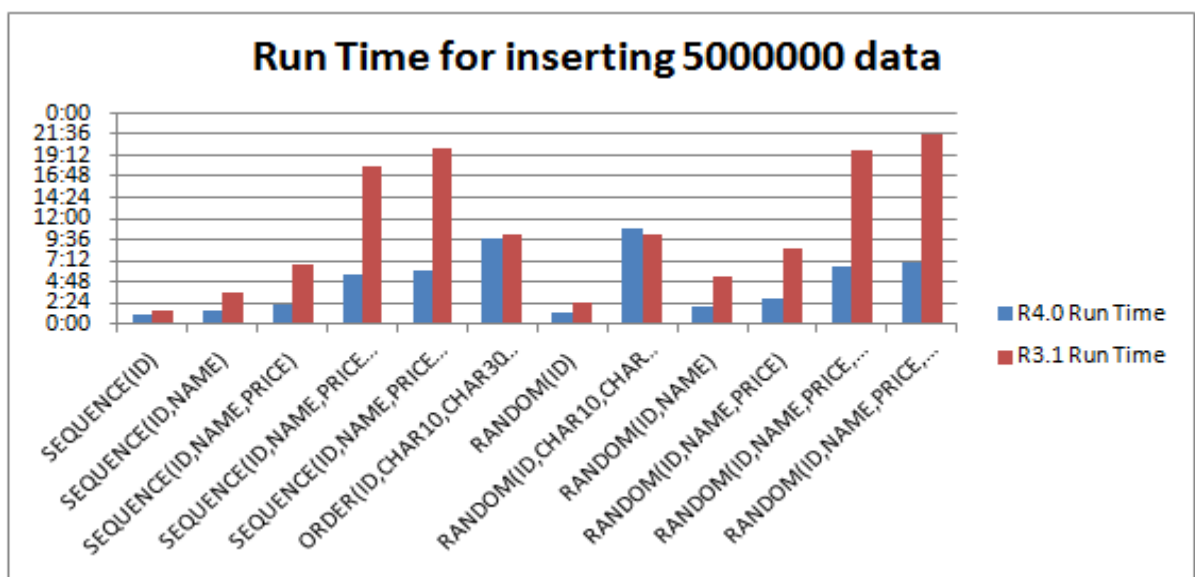
### 2.2.2 CUBRID Index Volume Performance Test

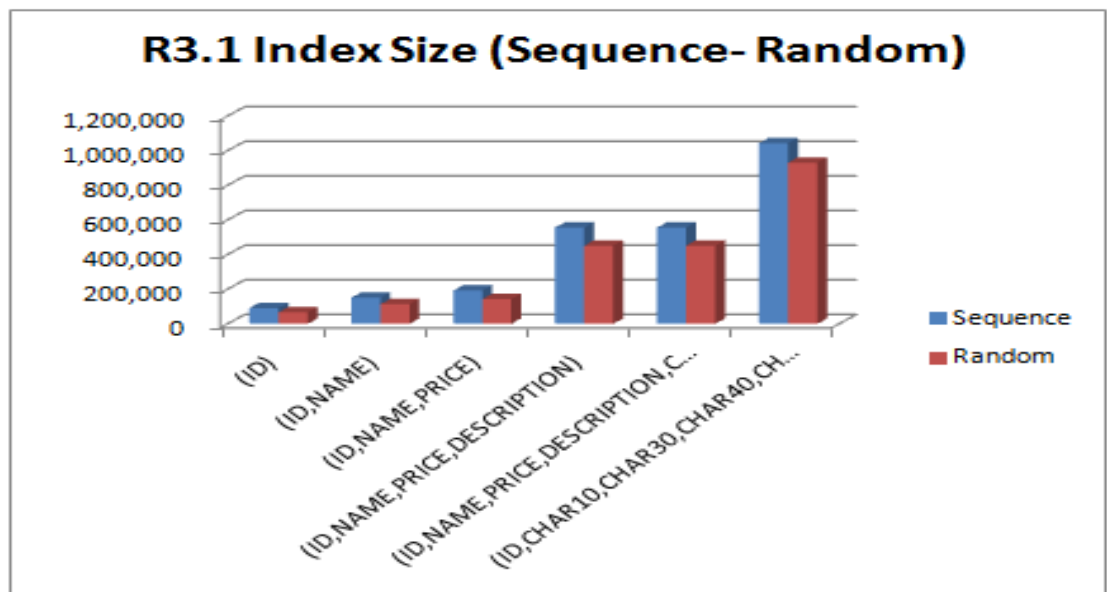
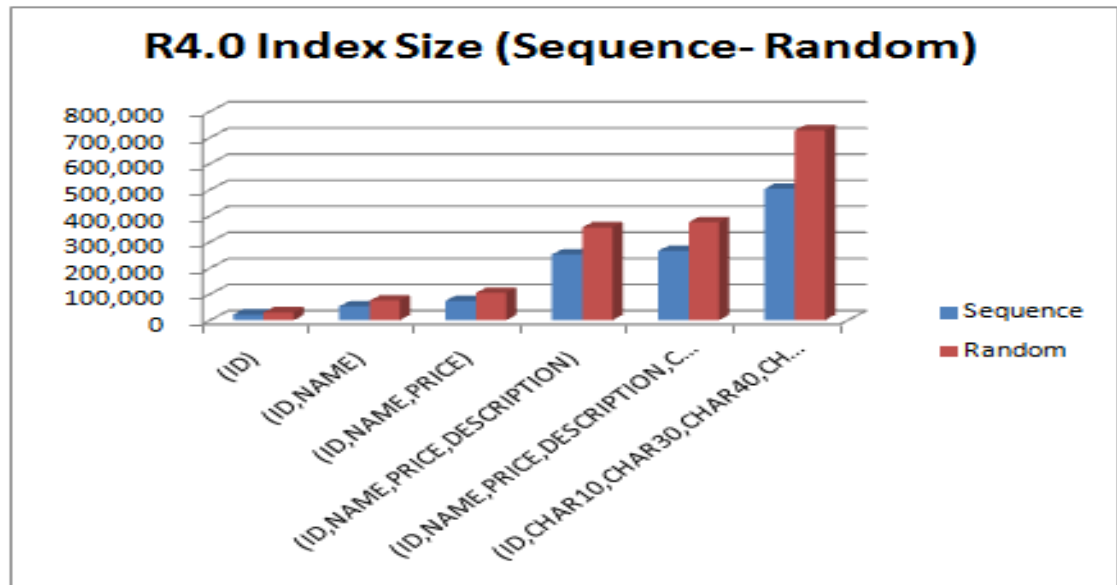
This test was performed to check index volume reduction amount in case of various index types and data generation types. Especially we have focused on inserted data sequence, random and ordered. For more information about test scenarios, see the appendix. As shown in the table below, we have found that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

- Linux: Performance Comparison between CUBRID 2008 R3.1 and CUBRID 2008 R4.0 (64-bit)  
On Linux 64-bit, the overall test results has significant change. Not only index size of CUBRID 2008 R4.0 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.

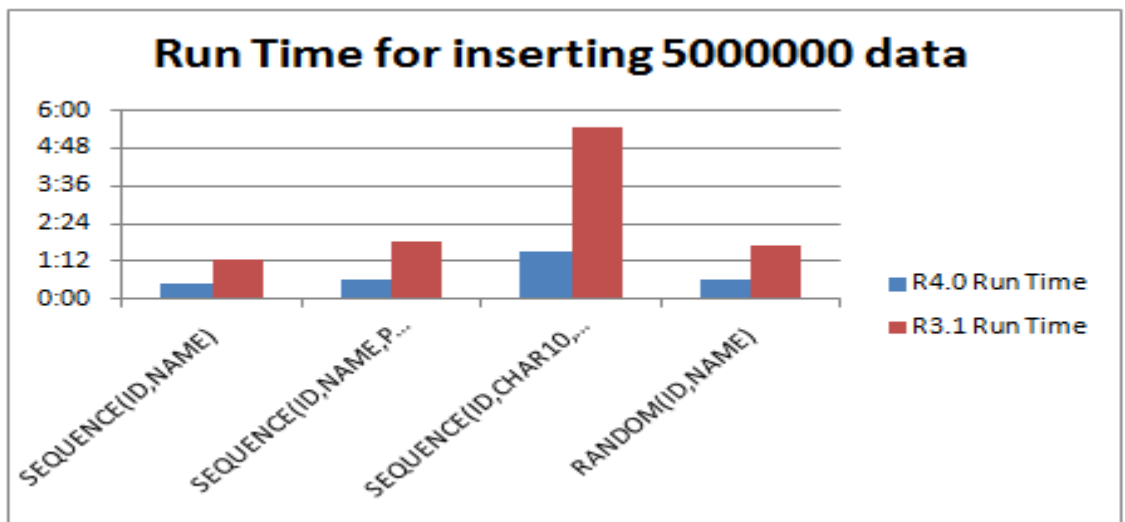
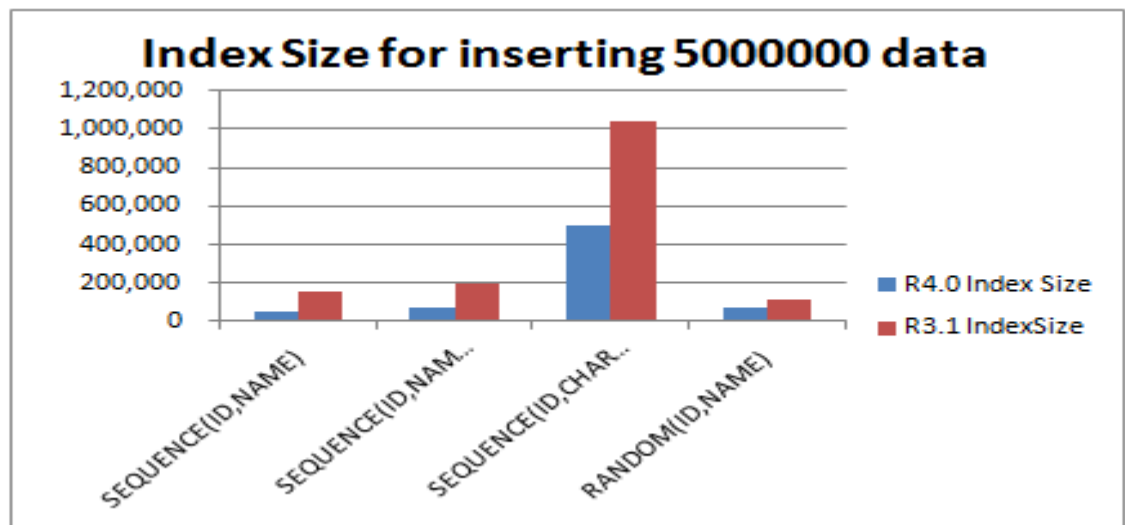


For above table, SEQUENCE(ID) means that data in column ID is inserted by sequence and ascending order. RANDOM(ID, NAME) means that data in column ID is inserted by random order and data in column NAME is inserted by sequence and ascending order. For more information about test scenarios, see the appendix.

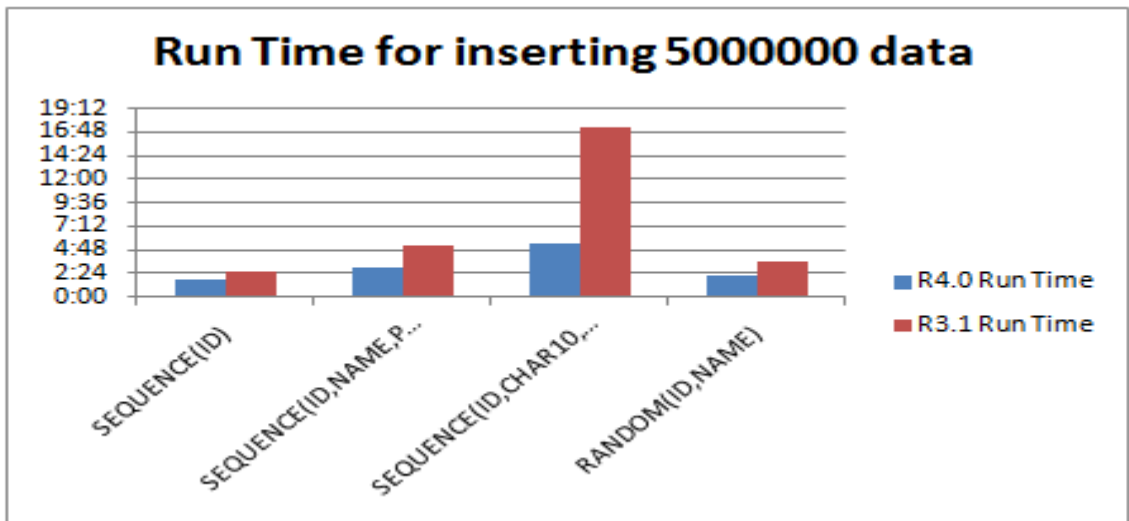
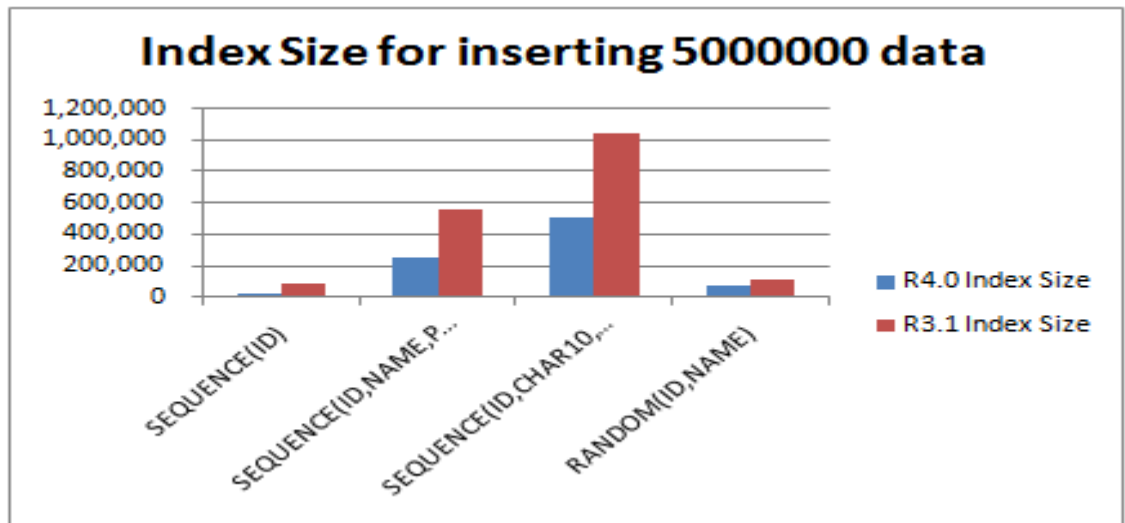




- Linux: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 (32-bit)  
On Linux 32-bit, the overall test results were the same as the Linux 64-bit results, and no significant difference was found.

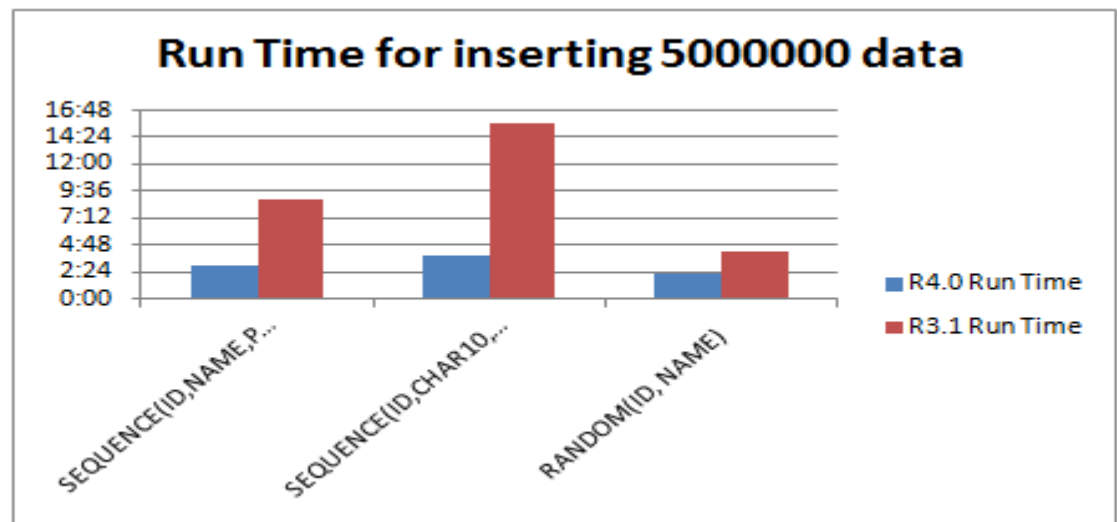
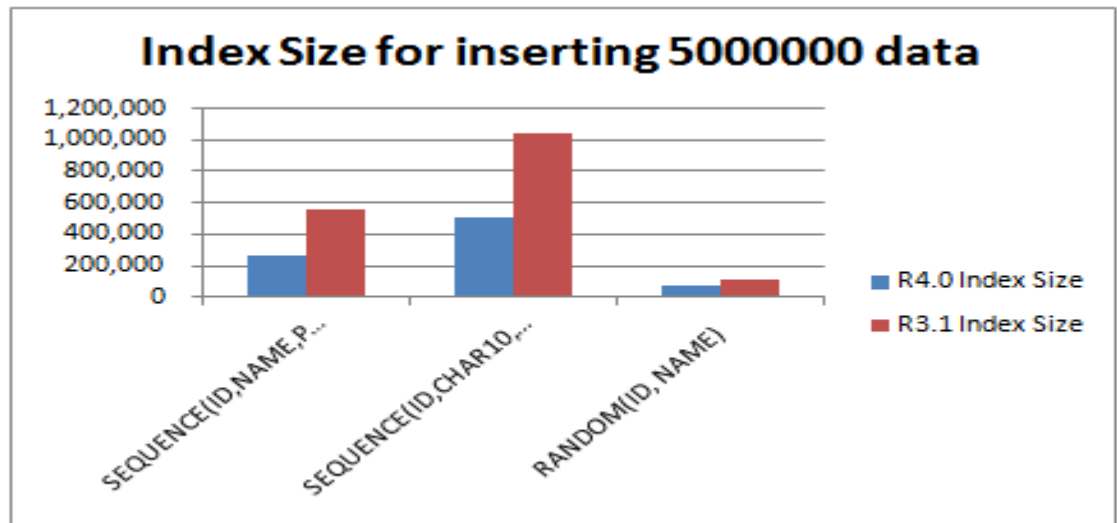


- Windows: Performance Comparison between CUBRID 2008 R3.1 (64-bit) and CUBRID 2008 R4.0 (64-bit)  
On Windows 64-bit, the overall test results were the same as the Linux 64-bit results, and no significant difference was found.



- Windows: Performance Comparison between CUBRID 2008 R3.1 (32-bit) and CUBRID 2008 R4.0 (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.



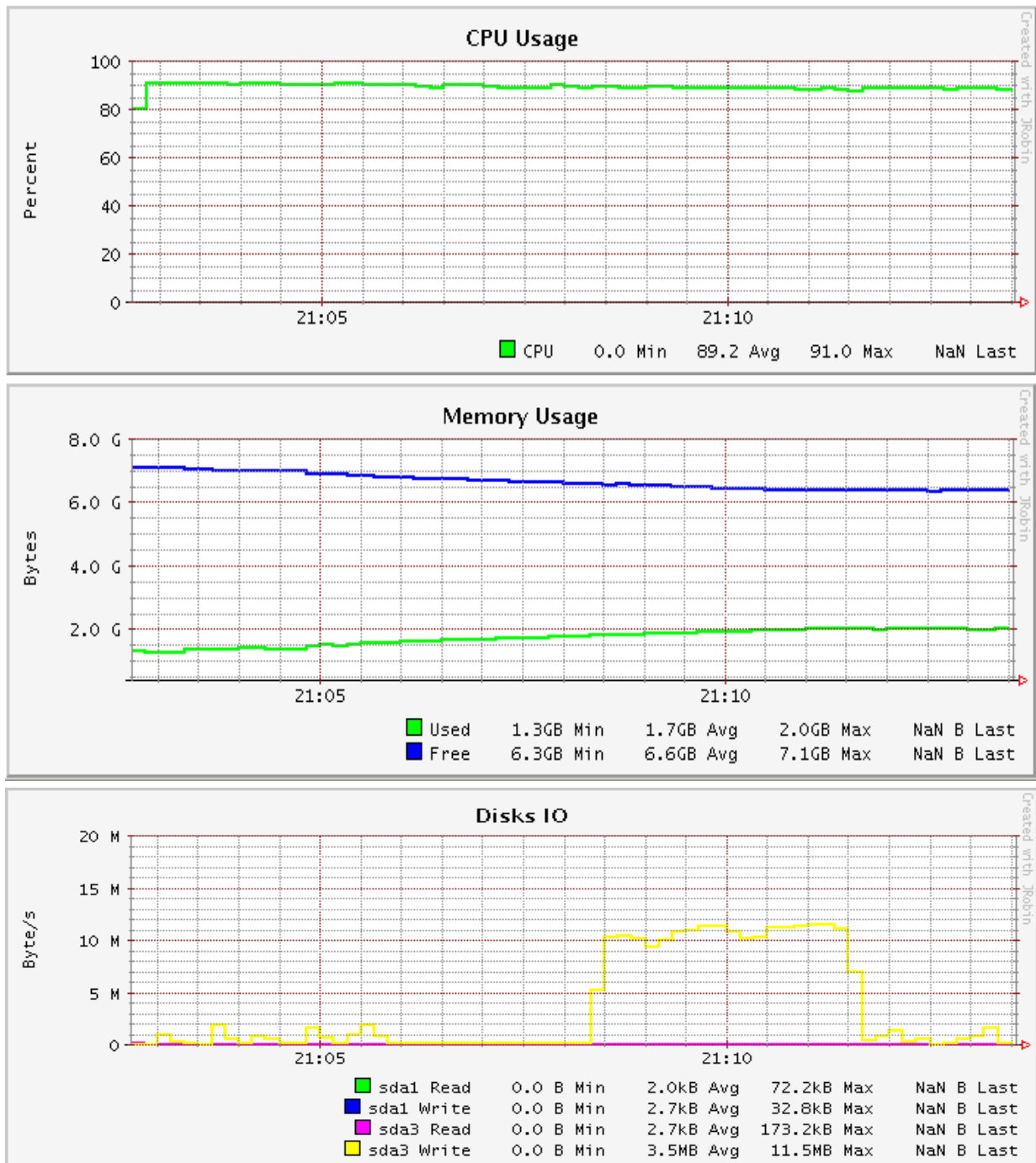


### 2.2.3 NBD Benchmark Performance Test

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 a little difference in NBD Benchmark test loads between R3.1 and R4.0. CUBRID 2008 R4.0 is slightly lower than CUBRID 2008 R3.1. It is needed to investigate it more in the future.

Platform	Version	BIT	Page View
<b>Linux 64</b>	CUBRID 2008 R3.1 (Default parameter configured)	64-bit	892
<b>Linux 32</b>	CUBRID 2008 R3.1 (Default parameter configured)	32-bit	856
<b>Linux 64</b>	CUBRID 2008 R4.0 (Default parameter configured)	64-bit	841
<b>Linux 32</b>	CUBRID 2008 R4.0 (Default parameter configured)	32-bit	817

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



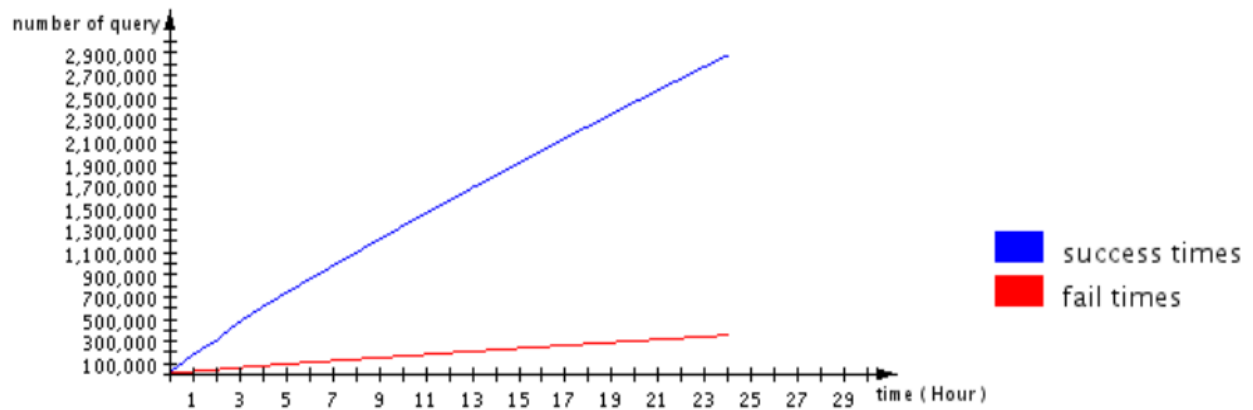
## 2.3 Stability Test Results

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

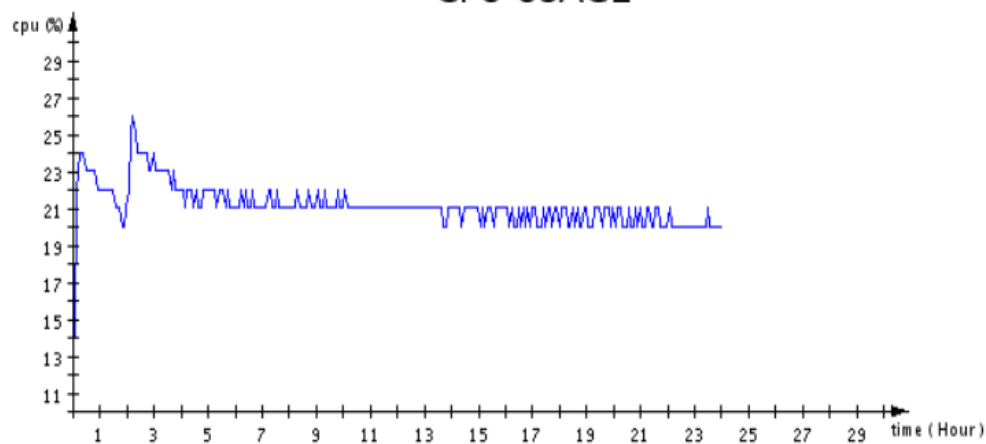
## number of SUCCESS/FAIL query

(y Axis is the total time dots test executed)



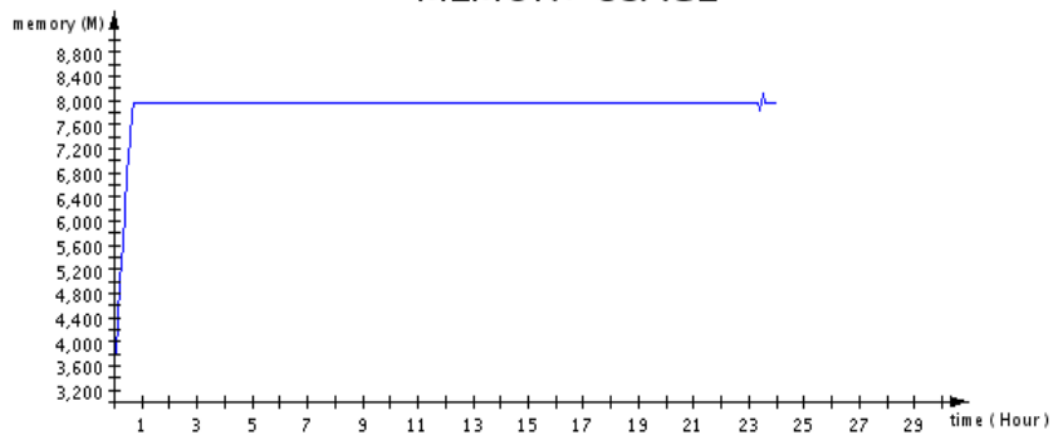
- CPU usage

## CPU USAGE



- Memory usage

## MEMORY USAGE



## 2.4 Other Test Results

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

## 2.5 Quality Index

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

Quality Name	Index	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. 887,165 LOC Total number of source code lines in the changed files 887,165 LOC
QA Scenario Code Coverage		76%	75.5%	Number of tested statements 179,387 Total number of statements 237,485
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) 426 Number of faults detected by static analysis (Level 4) 0 Total number of source code lines 786,602 LOC
Cyclomatic Code Complexity		3.3%	2.94%	Number of modules whose complexity is over 30 556 Total number of modules in a project 18,906
		12%	16.14%	Number of modules whose complexity is over 10 3,052 Total number of modules in a project 18,906

### 3. Conclusions

As described in Chapters 1 and 2, CUBRID 2008 R4.0 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 executed, and the related defects have been logged into BTS for (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 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 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 a little difference in NBD Benchmark test loads between R3.1 and R4.0. CUBRID 2008 R4.0 is slightly lower than CUBRID R3.1. It is needed to investigate it more in the future also.

---

# 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: 8595		
Case Name	Path	Description
object	sql/_01_object	Performs functionality tests of objects supported by CUBRID, and has the largest number of scenarios (3332 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.
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 issue 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: 1213		
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: 498		
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: 64		
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 2008 R4.0 with CUBRID 2008 R3.1, to check whether the former's index size smaller to later.
- ♦ Compare the inserting speed of CUBRID 2008 R4.0 with CUBRID 2008 R3.1, to check whether the former's inserting speed faster to later.
- ♦ After a larget amount of data inserting operation at CUBRID 2008 R4.0 and 2008 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 2008 R4.0 will smaller than CUBRID 2008 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 2008 R4.0 is faster than CUBRID 2008 R3.1.
- ♦ For CUBRID 2008 R4.0, the index used space of order index larger than random index, For CUBRID 2008 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	.....	ting two s	case in which there is no oblig	pro the GPL v2.0 or later license, which	with the same terms as the original license. CUBRID will be provided	.....
2	9	product-0-9	0.1	04/08/2011	.....	want to c	ry Software Distribution li	e no obligation of opening derivative so	in the same manner as the original license. CUBRID will be provided	.....
3	3	product-0-3	0.2	04/08/2011	.....	al databas	oping and distributing various	are Distribution license in which the	representative of CUBRID has accepted the license in the same manner as the original	.....
4	4	product-0-4	0.3	04/08/2011	.....	sive open	a obligation of opening deriva	is, which allows distribution, modifica	representative of CUBRID has accepted the license in the same manner as the original	.....
5	5	product-0-5	0.4	04/08/2011	.....	anagement	ive works. The reason of adopt	and provide excellent cost savings to c	is or later license which allows distribution, modification and improvement of the	.....
6	2	product-0-2	0.5	04/08/2011	.....	e source c	various CUBRID based applica	is the Berkeley Software Distribution li	based applications. The CUBRID license is different from the other vendors. The license	.....
7	10	product-0-10	0.6	04/08/2011	.....	dent softw	ter license, which allows dist	tribute open source relational database in	is or later license. The license is different from the other vendors. The license	.....
8	8	product-0-8	0.7	04/08/2011	.....	services	BRID does not want to create a	ing two separate license systems is that	license policy is different from the other vendors. The license is different from the	.....
9	7	product-0-7	0.8	04/08/2011	.....	tribute or	instead of Web applications, co	a different license policies for interfa	The CUBRID license policy is different from the other vendors. The license is different from the	.....
10	6	product-0-6	0.9	04/08/2011	.....	ution lice	na. The CUBRID license policy	CUBRID does not want to create any li	is different from the other vendors. The license is different from the other vendors. The	.....
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	pts the GPL v2.0 or later license, 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 license in which there		
4	sive open	o obligation of opening deriva	c, 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	c the Berkeley Software Distribution lic		
7	dent softw	ter license, which allows dist	omative open source relational database m		
8	services	BRID does not want to create a	ng two separate license systems is that		
9	ribute or	sized 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	gts the GPL v2.0 or later license, which		
6	e source c	various CUBRID based applicat	e the Berkeley Software Distribution lic		
3	al databas	oping and distributing various	are Distribution license in which there		
4	sive open	o obligation of opening deriva	e, which allows distribution, modifiest		
5	anagement	ive works. The reason of adopt	and provide excellent cost savings to c		
10	ution lice	es. The CUBRID license policy	CUBRID does not want to create any limit		
9	tribute or	mized for Web applications, es	o different license policies for interfa		
8	services	BRID does not want to create a	ng two separate license 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	omative 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*

Current view: top level

Test: Code Coverage

Date: 2011-06-25

Legend: Rating: low: < 75 % medium: >= 75 % high: >= 90 %

	Hit	Total	Coverage
Lines:	179387	237485	75.5 %
Functions:	9215	10414	88.5 %
Branches:	108025	182599	59.2 %

Directory	Line Coverage	Functions	Branches
<a href="#">/home/xbms/build/src/executables</a>	<div><div></div></div> 71.9 % 1445 / 2009	93.4 % 57 / 61	61.1 % 460 / 753
<a href="#">/home/xbms/build/src/parser</a>	<div><div></div></div> 90.9 % 7416 / 8154	98.7 % 76 / 77	54.1 % 2028 / 3746
<a href="#">src/base</a>	<div><div></div></div> 73.8 % 6001 / 8128	87.6 % 453 / 517	52.6 % 3395 / 6460
<a href="#">src/broker</a>	<div><div></div></div> 72.8 % 8507 / 11682	85.4 % 479 / 561	55.7 % 4365 / 7838
<a href="#">src/cci</a>	<div><div></div></div> 68.9 % 3307 / 4801	81.6 % 209 / 256	54.4 % 1464 / 2692
<a href="#">src/communication</a>	<div><div></div></div> 70.2 % 5664 / 8073	77.1 % 296 / 384	41.2 % 1650 / 4001
<a href="#">src/connection</a>	<div><div></div></div> 74.9 % 2676 / 3572	87.9 % 240 / 273	57.1 % 1187 / 2079
<a href="#">src/executables</a>	<div><div></div></div> 71.9 % 12816 / 17833	85.1 % 803 / 944	54.1 % 7028 / 12979
<a href="#">src/heaplayers</a>	<div><div></div></div> 88.3 % 68 / 77	100.0 % 11 / 11	56.2 % 18 / 32
<a href="#">src/jsp</a>	<div><div></div></div> 83.5 % 894 / 1071	100.0 % 68 / 68	62.6 % 338 / 540
<a href="#">src/object</a>	<div><div></div></div> 76.1 % 21470 / 28210	88.3 % 1662 / 1883	57.6 % 14867 / 25827
<a href="#">src/optimizer</a>	<div><div></div></div> 89.3 % 8577 / 9605	98.3 % 355 / 361	77.0 % 6579 / 8548
<a href="#">src/parser</a>	<div><div></div></div> 82.1 % 27744 / 33796	93.1 % 1167 / 1254	67.9 % 19057 / 28084
<a href="#">src/query</a>	<div><div></div></div> 75.0 % 31878 / 42502	92.2 % 1272 / 1380	61.3 % 22395 / 36507
<a href="#">src/session</a>	<div><div></div></div> 71.0 % 519 / 731	93.6 % 44 / 47	52.4 % 272 / 519
<a href="#">src/storage</a>	<div><div></div></div> 73.2 % 22662 / 30974	89.4 % 1127 / 1261	58.5 % 12074 / 20632
<a href="#">src/thread</a>	<div><div></div></div> 70.7 % 1141 / 1614	91.0 % 81 / 89	54.8 % 495 / 903
<a href="#">src/transaction</a>	<div><div></div></div> 67.3 % 16602 / 24653	82.6 % 815 / 987	50.6 % 10353 / 20459