CUBRID 2008 R4.0 Beta

릴리스 노트

목차

1. 개요

릴리스 노트 정보

본 문서는 CUBRID 2008 R4.0 Beta 버전에 관한 유용한 정보를 포함한다. 릴리스 노트의 최신 버전은 CUBRID 오픈 소스 프로젝트 사이트(http://dev.naver.com/projects/cubrid)에서 확인할 수 있다.

CUBRID 2008 R4.0 Beta 버전은 CUBRID 2008 R3.1 Patch 2와 그 이전 버전까지의 모든 패치를 포함하고 있다. CUBRID 2008 R4.0 Beta 이전 버전의 패치에 대한 자세한 내용은 CUBRID 2008 R3.1의 최신 패치 버전 릴리스 노트와 CUBRID 2008 R2.2의 최신 패치 버전 릴리스 노트를 참조하도록 한다.

※ 이하 *CUBRID 2008 R4.0 Beta*는 CUBRID 2008 R4.0으로 표기한다.

릴리스 노트 개정 내역

CUBRID 2008 R4.0 버전의 릴리스 이후 릴리스 노트의 변경 사항은 아래와 같다.

작성 날짜	설명
2011년 4월	CUBRID 2008 R4.0 Beta 릴리스

참고 문서

CUBRID 2008 R4.0 제품과 함께 배포되는 문서는 아래와 같다.

문서	설명
릴리스 노트	CUBRID 릴리스 버전의 특징 및 이전 버전에서 변경된 사항과 관련된 정보를 포함한다.
매뉴얼	Quick Start Guide, CUBRID Architecture, SQL 설명서, 튜닝 안내서, 관리자 안내서, CUBRID 매니저 안내서, API References를 포함한다.

버그 리포트 및 사용자 피드백 제공 방법 안내

CUBRID 프로젝트에서는 사용자의 거침없는 버그 리포트와 솔직한 피드백을 기다리고 있으며, 아래 사이트에서 등록할 수 있다.

문서	설명
버그 리포트	CUBRID 오픈 소스 프로젝트: http://dev.naver.com/projects/cubrid/issue
사용자 피드백	CUBRID 오픈 소스 프로젝트: http://dev.naver.com/projects/cubrid/forum CUBRID 공식 사이트: http://www.cubrid.com

추가 정보 안내

CUBRID에 관한 유용한 정보는 아래 사이트에서 찾을 수 있다.

정보	사이트
CUBRID 제품 정보	http://cubrid.com/zbxe/product
CUBRID 라이선스 정보	http://cubrid.com/zbxe/bbs_oss_guide/32249
CUBRID 사용자 문서	http://cubrid.com/zbxe/developer
CUBRID 교육 서비스	http://cubrid.com/zbxe/education_overview

2. CUBRID 2008 R4.0 정보

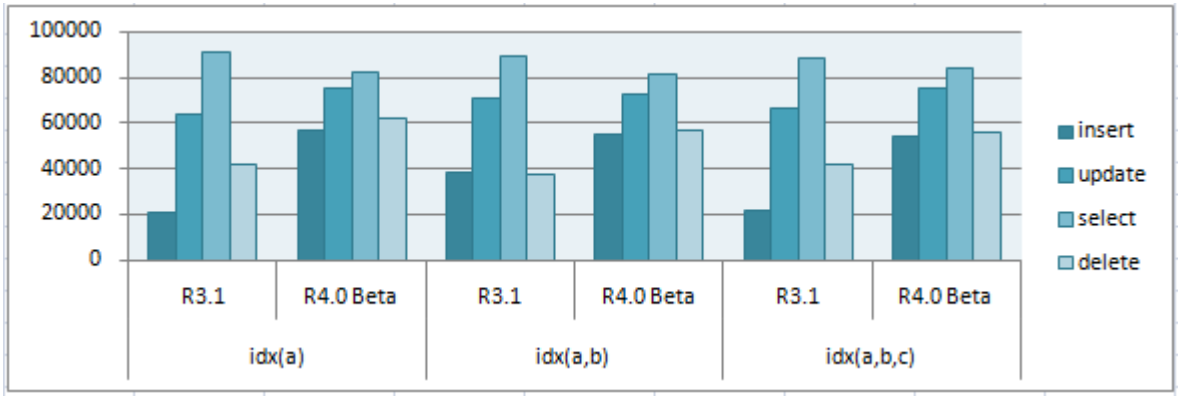
CUBRID 2008 R4.0 릴리스 특징

CUBRID 2008 R4.0 릴리스는 주로 INSERT 및 DELETE 성능을 향상시켰으며, WINDOWS 버전 성능이 2배 가까이 향상되었다. 또한, 데이터베이스 볼륨 사용량을 기존 버전 대비 절반 수준으로 대폭 감소시켰다. 응용 프로그램 개발자의 개발 생산성 향상을 위하여 다양한 SQL 함수 및 구문을 추가 및 개선하였으며, 타입 변환을 보다 용이하게 지원하고, 호스트 변수에 대한 타입 동적 바인딩을 대거 개선하였다. 질의 처리 성능을 높이기 위하여 특히 커버링 인덱스 지원, 인덱스 접근을 통한 ORDER BY 및 GROUP BY 절 처리 최적화, LIMIT 절 처리 최적화, 다중 스캔 범위 처리 최적화, 집계 함수 처리 최적화, 내림차순 인덱스 스캔 지원, LIKE 질의에 대한 인덱스 스캔 지원 등 다양한 질의 최적화 기법을 새로 추가하였다. 잠금 기법 개선을 통해 교착상태 발생 가능성을 최소화시켰다. 그리고 HA 운영 안정성 향상을 위하여 수많은 HA 안정화 이슈를 해결하였으며, HA 운영 편이성 증대를 위해 관련 유틸리티들을 추가 및 개선하였다. 또한, 15개의 PHP 함수 추가 및 CCI API 추가 지원 등 개발 인터페이스를 개선하였다. 그 외에 수많은 안정성 및 기능/성능 개선 이슈가 해결되었다.

CUBRID 2008 R4.0 버전의 주요 특징은 다음과 같다.

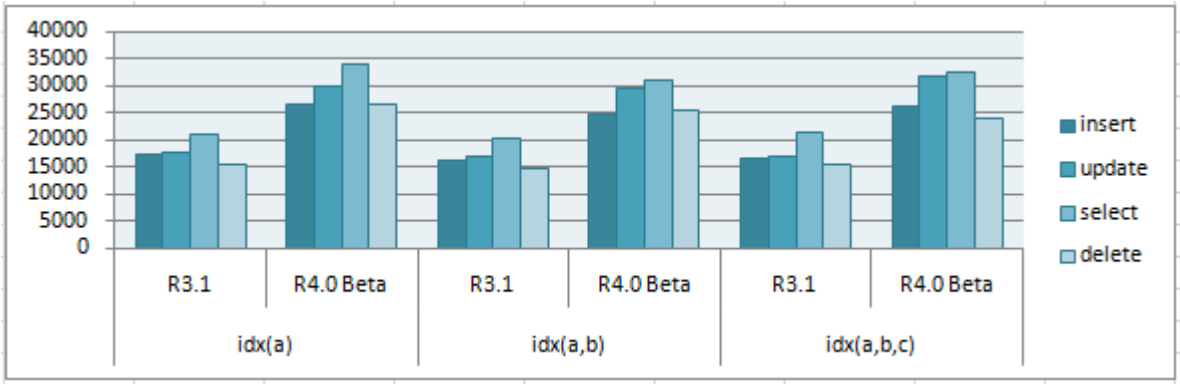
INSERT, DELETE 성능 향상

입력 성능이 기존 2008 R3.1 대비 1.5에서 2.6배 향상되었고, 삭제 성능이 1.3에서 1.5배 향상되었으며, 갱신 성능은 1.1에서 1.2배 향상되었다. 보다 자세한 사항은 QA 리포트를 참고한다.



WINDOWS 버전 성능 향상

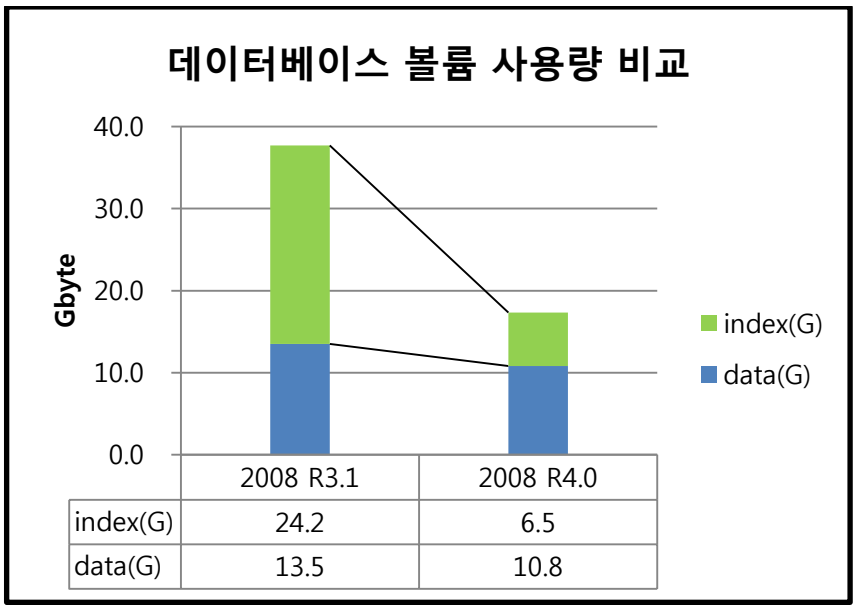
기존 2008 R3.1 버전 대비 평균 1.5에서 1.8배 향상되었다. 입력 성능은 1.8배에서 1.9배, 갱신 성능 역시 1.8배에서 1.9배, 삭제 성능은 2.2배 이상 향상되었으며, 검색 성능은 1.4배 개선되었다. 보다 자세한 사항은 QA 리포트를 참고한다.



데이터베이스 볼륨 사용량 대폭 감소

데이터베이스 구조를 개선하여 볼륨 사용량을 2008 R3.1 대비 절반 수준으로 감소시켰으며, 특히 인덱스 볼륨 사용량을 대폭 감소시켰다.

다음은 6천4백만건의 Primary Key가 있는 데이터를 INSERT하여 기본키(Primary key) 인덱스 및 데이터의 볼륨 사용량을 비교한 그래프이다.



	2008 R3.1	2008 R4.0
index(G)	24.2	6.5
data(G)	13.5	10.8

다양한 SQL 함수 및 문장 추가

날짜/시간 관련 함수, 문자열 함수, 집계 함수 등 약 20여개의 SQL 함수를 추가하였다. 또한, 데이터베이스의 다양한 정보를 쉽게 확인할 수 있는 SHOW 문, 사용자 정의 변수를 지정하는 SET 문의 추가 및 컬럼의 이름, 타입과 크기 및 제약 조건의 변경이 용이하도록 "ALTER TABLE ... CHANGE COLUMN" 문을 확장하는 등 사용자 편의를 도모하였다.

인덱스 활용 개선

ORDER BY절, GROUP BY 절, LIMIT 절이 있는 질의, SELECT 리스트의 모든 컬럼이 인덱스에 포함되는 경우 등에서 인덱스를 최적으로 활용하도록 개선하였다.

HA 기능 안정성 및 편의성 증대

HA 기능에서 발견되었던 다수의 오류를 수정하였다. 수정된 내역은 CUBRID 2008 R3.1 patch2 릴리스 노트에서 수정된 사항 중 HA 부분을 참고한다.

또한, $CUBRID/share/scripts/ha/ha_make_slavedb.sh 스크립트를 제공하여 HA 환경에서 슬레이브 데이터베이스의 재구축이 용이하도록 하였다. 또한, HA 구성이 용이하도록 별도의 HA 설정 파일(cubrid_ha.conf)을 제공하고, cubrid heartbeat 유틸리티를 대폭 개선하였다.

보다 자세한 내용은 아래의 *CUBRID 2008 R4.0에서 변경된 사항*을 참고한다.

지원 플랫폼 및 설치 권장 사양

CUBRID 2008 R4.0 버전이 지원하는 플랫폼과 설치를 위한 하드웨어/소프트웨어 요구 사항은 아래 표와 같다.

지원 플랫폼	메모리 여유 공간	디스크 여유 공간	필요 소프트웨어
- Windows 32/64 Bit XP, 2003, Vista, Windows7 - Linux 계열 32/64 Bit Linux kernel 2.4 및 glibc 2.3.4 이상	1G 이상	500MB 이상	- JRE 또는 JDK 1.6 이상 Java 저장 프로시저를 사용하는 경우 필요

2008 R4.0부터 CUBRID 매니저는 CUBRID 설치 패키지와 같이 제공되지 않으므로, 이를 사용하려면 별도로 설치해야 한다. CUBRID 설치 패키지와 CUBRID 매니저는 *다운로드 페이지* 에서 받을 수 있다.

라이선스 안내

CUBRID의 서버 엔진에는 GNU GPL v2 or later 가 적용되고 CUBRID 매니저 및 인터페이스(API)에는 BSD 라이선스가 적용된다. 보다 상세한 정보는 CUBRID 공식 사이트의 *라이선스 가이드*를 참고한다.

버전 호환성과 운용성

응용 프로그램의 호환성

- 이전 버전의 JDBC, PHP, CCI API 등을 사용하는 응용 프로그램은 CUBRID 2008 R4.0 데이터베이스에 접근할 수 있다. 다만, JDBC, PHP, CCI 인터페이스에 추가/개선된 기능을 사용하기 위해서는 CUBRID 2008 R4.0 버전의 라이브러리를 링크하거나 드라이버를 사용해야 한다.
- 새로운 예약어 추가 및 일부 질의에 대한 스펙 변경으로 인해 질의 결과가 이전 버전과 다를 수 있으므로 주의한다. 보다 상세한 사항은 업그레이드 주의 사항을 참고한다.
- GLO 클래스를 이용하여 개발된 응용은 BLOB, CLOB 타입에 맞는 응용 및 스키마로 변환하여 사용해야 한다.

CUBRID 매니저의 호환성

2008 R4.0부터 CUBRID 매니저는 CUBRID 설치 패키지와 같이 제공되지 않으므로, 이를 사용하려면 별도로 다운로드하여 설치해야 한다.

CUBRID 매니저는 CUBRID 2008 R2.1 이상 버전의 서버에 대해서 하위 호환성을 보장하며, 각 서버 버전과 일치하는 CUBRID JDBC 드라이버를 사용한다. 하지만 CUBRID 매니저에서 제공하는 모든 기능을 제대로 사용하기 위해서는 CUBRID 서버 버전보다 높은 버전의 CUBRID 매니저를 사용해야 한다. CUBRID JDBC 드라이버는 CUBRID 설치 시 $CUBRID/jdbc 디렉터리에 포함되어 있다.[1]

(2008 R2.2 이상 버전의 드라이버는 CUBRID 매니저에 기본으로 내장되어 있으며, *CUBRID 오픈 소스 프로젝트 사이트*에서 별도로 받을 수도 있다.)

데이터베이스 호환성

서버 버전을 하위 버전에서 업그레이드하는 경우, 데이터베이스의 마이그레이션 작업을 수행해야 한다. 보다 상세한 사항은 *데이터베이스 마이그레이션 절차*를 참고한다.

[1] $CUBRID는 CUBRID가 설치된 곳을 지정하는 Linux용 환경변수이며, Windows 환경에서는 %CUBRID%와 같은 형식으로 사용된다.

상호 운용성

CUBRID DB 서버와 브로커 서버를 분리하여 운영하는 경우, 서버 장비의 운영 체제가 다르더라도 상호 운용성을 보장한다. 단, DB 서버의 Bit 버전과 브로커 서버의 Bit 버전은 서로 동일해야 한다. 예를 들어, Linux용 64Bit 버전 DB 서버는 Windows용 64Bit 버전 브로커 서버와 상호 운용이 가능하지만, 32Bit 버전 브로커 서버와는 상호 운용이 불가능하다.

CUBRID 2008 R4.0의 설치 방법

제품 설치

Linux용 설치 패키지는 바이너리를 포함하는 스크립트, tar.gz 압축 파일, Linux RPM 패키지 형태로 제공되며, 설치 방법은 매뉴얼의 *CUBRID 시작> 설치와 실행> Linux에서의 설치와 실행*을 참고한다.

Windows용 설치 패키지는 설치 마법사를 이용하여 설치할 수 있다. 설치 방법은 매뉴얼의 *CUBRID 시작> 설치와 실행> Windows에서의 설치와 실행*을 참고한다. 이와 함께 *Windows 환경에서 설치 디렉터리 설정에 관한 주의사항*을 참고한다.

CUBRID 환경 변수 및 OS 환경 변수 설정

CUBRID를 사용하기 위해서는 CUBRID 환경 변수와 관련 OS 환경 변수를 설정해야 한다. (*관련 매뉴얼 참고*) 특히, CUBRID 매니저와 Java 저장 프로시저를 사용하기 위해서는 Java 환경 변수를 설정해야 한다. (*관련 매뉴얼 참고*)

CUBRID 2008 R4.0으로 업그레이드하는 방법

업그레이드 주의 사항

- **기존 환경 설정 파일 보관**

  이전 버전의 $CUBRID/conf 디렉터리의 환경 설정 파일(**cubrid.conf**, **cubrid_broker.conf**, **cm.conf**)과 $CUBRID_DATABASES 디렉터리의 데이터베이스 위치 정보 파일(**databases.txt**)을 보관한다.[2]

- **데이터베이스 마이그레이션**

  CUBRID 2008 R3.1부터 GLO를 지원하지 않으며 LOB 타입이 GLO 기능을 대체하게 되었으므로, GLO를 이용한 응용 및 스키마는 LOB 타입에 맞게 수정해야 한다. (아래의 *GLO 클래스 사용자의 마이그레이션 참고*)

  CUBRID 2008 R4.0은 2008 R3.x 이하 버전의 데이터베이스와 호환되지 않으므로, 이전 데이터베이스를 CUBRID 2008 R4.0으로 마이그레이션해야 한다. (아래의 *2008 R3.x 및 이전 버전에서 2008 R4.0으로 마이그레이션 참고*)

- **복제 또는 HA 환경 재구성**

  2008 R4.0부터는 복제 기능을 지원하지 않으므로, 이전 버전의 복제 기능을 사용하는 시스템에서는 DB 마이그레이션 및 HA 환경으로 재구성할 것을 권장한다. 또한, CUBRID 2008 R2.0, 2008 R2.1에서 제공된 Linux Heartbeat 기반의 HA 기능을 사용하는 시스템도 보다 안정적인 운영을 위해 DB 마이그레이션 및 CUBRID Heartbeat 기반의 HA 환경으로 재구성해야 한다. (아래의 *HA 환경에서 데이터베이스 마이그레이션 참고*)

---

[2] $CUBRID 또는 $CUBRID_DATABASES 형식의 환경변수는 Linux 환경에서 사용하는 형식이며 Windows 환경에서는 %CUBRID% 또는 %CUBRID_DATABASES%와 같은 형식으로 사용해야 한다.

HA 환경 구성은 매뉴얼의 *관리자 안내서 > CUBRID HA*를 참고하여 재설정해야 한다.

- **새로 추가된 예약어 검사**

CUBRID 설치 패키지 또는 *다운로드 페이지*에서 배포되는 CUBRID 2008 R4.0용 예약어 검출 스크립트인 **check_reserved.sql**을 이용하여 예약어 사용 여부를 검사할 수 있으며, 예약어로 지정된 식별자를 사용하고 있을 경우 식별자를 수정해야 한다. (*관련 매뉴얼 참고*)

데이터베이스 마이그레이션 절차

2008 R3.x 및 이전 버전에서 업그레이드하는 사용자를 위한 절차 바로 가기

GLO 클래스 사용자를 위한 절차 바로 가기

- **2008 R3.x 및 이전 버전에서 2008 R4.0으로 마이그레이션**

2008 R3.x 및 이전 버전의 사용자는 아래의 표와 같이 2008 R4.0으로 데이터베이스 마이그레이션을 해야 한다.

또한, 기존의 GLO 클래스를 사용하고 있는 경우에는 추가 작업이 필요하다. (아래의 *GLO 클래스 사용자의 마이그레이션 참고*)

아래는 **cubrid unloaddb/loaddb** 유틸리티 및 *다운로드 페이지*에서 별도 배포되는 **check_reserved.sql** 예약어 검출 스크립트를 이용하여 마이그레이션을 수행하는 방법이다. (*관련 매뉴얼*과 본 문서의 *주의 사항> GLO 클래스 지원 중단에 따른 주의 사항*을 참고)

특히, 2008 R4.0에서 cubrid createdb 유틸리티의 기본 페이지 크기가 기존의 4K에서 16K로 변경되었으므로 데이터베이스 생성 시에 유의해야 한다.

<table>
<tr><th>단계</th><th>Linux 환경</th><th>Windows 환경</th></tr>
<tr><td>C1 단계:
CUBRID Service 종료</td><td><code>% cubrid service stop</code></td><td>CUBRID Service Tray>[Exit]를 선택한다.</td></tr>
<tr><td>C2 단계:
예약어 검출 스크립트 실행</td><td colspan="2">예약어 검출 스크립트가 위치하는 디렉터리에서 아래 명령을 실행한다. 검출 결과를 확인하여 마이그레이션 진행 또는 식별자 수정 작업을 진행한다. (허용되는 식별자는 관련 매뉴얼 참고)
<code>% csql -S -u dba -i check_reserved.sql testdb</code></td></tr>
<tr><td rowspan="2">C3 단계:
이전 버전 DB 언로드</td><td colspan="2">이전 버전의 databases.txt 및 conf 디렉터리 내 설정 파일을 별도 디렉터리에 보관한다. (C3a)
cubrid unloaddb 유틸리티를 실행하고 이때 생성된 파일을 별도 디렉터리에 보관한다. (C3b)
<code>% cubrid unloaddbdb -S testdb</code>
기존 데이터베이스를 삭제하거나, 별도 디렉터리로 이동한다. (C3c)
<code>% cubrid deletedb testdb</code></td></tr>
<tr><td></td><td>이전 버전의 CUBRID를 제거한다.</td></tr>
<tr><td>C4 단계:
2008 R4.0 설치</td><td colspan="2">본 문서의 CUBRID 2008 R4.0의 설치 방법을 참고한다.</td></tr>
<tr><td rowspan="2">C5 단계:
데이터베이스 생성 및
데이터 로딩</td><td></td><td><code>% cubrid service stop</code></td></tr>
<tr><td colspan="2">데이터베이스를 생성할 디렉토리로 이동한 후, 데이터베이스를 생성한다. (C5a)
<code>% cd $CUBRID/databases/testdb</code>
<code>% cubrid createdb testdb</code>
(C3c)에서 보관한 파일을 가지고 cubrid loaddb 유틸리티를 실행한다. (C5b)
<code>% cubrid loaddb -S -s testdb_schema -d testdb_objects testdb</code></td></tr>
<tr><td>C6 단계:
새 버전 DB 백업</td><td colspan="2"><code>% cubrid backupdb -S testdb</code></td></tr>
<tr><td>C7 단계:
CUBRID 환경 설정 및
CUBRID Service 구동</td><td>환경 설정 파일을 수정한다. 이때, (C3a)에서 보관한 이전 버전의 환경 설정 파일을 새 버전에 맞게 일부 수정한다. (파라미터 설정은 관련 매뉴얼 참고)
<code>% cubrid service start</code></td><td>CUBRID Service Tray> [Service Start]를 선택하여 서비스를 시작한다.
명령 프롬프트 창에서 데이터베이스 서버를 구동한다.
<code>% cubrid server start testdb</code></td></tr>
</table>

```
% cubrid server start testdb
```

- **GLO 클래스 사용자의 마이그레이션**

GLO 클래스를 사용하는 경우, 2008 R3.1부터는 GLO 클래스를 지원하지 않으므로 BLOB 또는 CLOB 타입을 사용하도록 응용과 스키마를 변경해야 한다. 변경 작업이 용이하지 않다면 마이그레이션을 보류할 것을 권장한다.

HA 환경에서 데이터베이스 마이그레이션 절차

2008 R2.2 이상 버전에서 업그레이드 하는 사용자를 위한 절차 바로 가기

2008 R2.0 또는 2008 R2.1에서 업그레이드 하는 사용자를 위한 절차 바로 가기

- **2008 R2.2 이상 버전에서 2008 R4.0으로 HA 마이그레이션**

아래는 응용 프로그램1, 응용 프로그램2, 브로커1, 브로커2, 마스터 데이터베이스, 슬레이브 데이터베이스를 각각 별도 서버에 구축한 환경에서 CUBRID 업그레이드가 수행되는 중에도 읽기 전용(Read Only) 상태로 서비스를 유지하기 위한 시나리오이다. 시나리오의 장비 구성 및 CUBRID 업그레이드 상태를 설명하기 위하여, 각 장비와 기호를 다음과 같이 정의하였다.

장비	장비 기호
[응용 프로그램1] 서버	A1
[응용 프로그램2] 서버	A2
[브로커1] ReadWrite 모드	B1_rw
[브로커2] ReadOnly 모드	B2_ro
[DB 노드1] (마스터 상태)	N1_m
[DB 노드2] (슬레이브 상태)	N2_s

CUBRID 업그레이드 여부	기호
업그레이드 이전	장비 기호 앞에 o
업그레이드 이후	장비 기호 앞에 n

초기 설정은 브로커1이 DB 노드1(마스터 데이터베이스 상태)에 연결되어 있고 브로커1은 읽기/쓰기(RW), 브로커2는 읽기 전용(RO)으로 설정되어 있으며, 응용 프로그램은 2개의 장비에서 각각 수행 중이고 브로커1이 정지하면 브로커2에 접속하도록 설정되어 있다고 가정한다. 위에서 언급한 노드 상태를 기호로 표현하면 다음과 같다.

oA1	oA2
oB1_rw	oB2_ro
oN1_m	oN2_s

다음은 위와 같은 상황에서 CUBRID 업그레이드를 진행하기 위한 시나리오이다.

단계	설명	단계 수행 후 서버 상태
H1 단계: [브로커1] 브로커1을 ReadOnly로 전환	업그레이드 중 읽기 서비스만 가능하도록 하기 위해 현재 읽기/쓰기 모드로 동작 중인 브로커1을 읽기 전용(RO : ReadOnly)으로 변경한 후	oA1 oA2

<table>
<tr>
<td></td>
<td>상태를 확인한다.
<pre>% cubrid broker stop
% vi $CUBRID/conf/cubrid_broker.conf
…
ACCESS MODE=RO

% cubrid broker start

% cubrid broker status
@ cubrid broker status

% broker - cub cas [16139,33000]
 ……
 KEEP_CONNECTION:AUTO, ACCESS_MODE:RO
 ……</pre></td>
<td><table><tr><td>oB1_ro</td><td>oB2_ro</td></tr><tr><td>oN1_m</td><td>oN2_s</td></tr></table></td>
</tr>
<tr>
<td>H2 단계: [DB 노드1]
DB 노드1에서 C1~C6 단계를 수행하고
cubrid_port_id, ha_port_id 변경 후
데이터베이스 시작</td>
<td>DB 노드 1에서 2008 R4.0 마이그레이션 C1 ~ C6 단계를 수행하여, CUBRID 업그레이드, 마이그레이션 및 R4.0 데이터베이스를 백업한다.

cubrid_port_id, ha_port_id를 기존과 다른 값으로 변경한다.
(cubrid_port_id는 DB 노드와 브로커 간 통신에 사용하는 포트이고,
ha_port_id는 HA로 이중화된 DB 노드 간 통신에 사용하는 포트이다.)
예) cubrid_port_id를 1523 -> 1524로 변경
<pre>% vi $CUBRID/conf/cubrid.conf
cubrid_port_id=1524</pre>
ha_node_list, ha_db_list를 cubrid_ha.conf에 설정한다. (기존 버전의 cubrid.conf와 cubrid-ha 파일 참고)
ha_mode=on으로 cubrid_ha.conf에 설정한다.
그 외 나머지 파라미터를 설정한다. (관련 매뉴얼 참고)
<pre>% vi $CUBRID/conf/cubrid_ha.conf
[common]
ha mode=on
ha port id=59902
ha_node_list=cubrid-ha@master:slave
ha_db_list=testdb
…</pre>(C3a)에서 저장한 databases.txt를 $CUBRID_DATABASES에 복사하고, HA를 구동한다.

<pre>% cubrid heartbeat start</pre>HA 구동 후 서버의 HA 모드를 확인하여, “to-be-active’ 상태인 경우 cubrid changemode 유틸리티를 이용하여 서버의 HA 모드를 “active”로 변경한다.
<pre>% cubrid changemode testdb@localhost
The server `testdb@localhost''s current HA running
mode is to-be-active.
% cubrid changemode -f -m active testdb@localhost
The server `testdb@localhost''s current HA running
mode is active.</pre></td>
<td><table><tr><td>oA1</td><td>oA2</td></tr><tr><td>oB1_ro</td><td>oB2_ro</td></tr><tr><td>nN1_m</td><td>oN2_m</td></tr></table></td>
</tr>
<tr>
<td>H3 단계: [브로커1]
브로커 노드 업그레이드 및 cubrid_port_id,
BROKER_PORT 변경</td>
<td>브로커1 노드를 업그레이드하고, 사용하는 모든 브로커 포트를 변경한다. (cubrid_port_id는 브로커와 DB 노드 간 통신에 사용하는 포트이고,
BROKER_PORT는 브로커와 응용 프로그램 간 통신에 사용하는 포트이다.)
cubrid_port_id는 H2단계에서 바꾼 것과 같은 값으로 변경한다.

예) 브로커 노드의 cubrid_port_id를 1523 -> 1524로 변경
<pre>% vi $CUBRID/conf/cubrid.conf
cubrid_port_id=1524</pre>
예) BROKER_PORT를 1111 -> 2222로 변경
<pre>% vi $CUBRID/conf/cubrid_broker.conf
[%BROKER1]</pre></td>
<td><table><tr><td>oA1</td><td>oA2</td></tr><tr><td>nB1_ro</td><td>oB2_ro</td></tr><tr><td>nN1_m</td><td>oN2_m</td></tr></table></td>
</tr>
</table>

<table>
<tr><td></td><td>BROKER_PORT=2222</td><td></td></tr>
<tr><td>H4 단계: [응용 프로그램1]
인터페이스 업그레이드 및
연결 포트 변경</td><td>응용 프로그램1의 인터페이스를 업그레이드하고, 연결 포트를 변경한다.
(JDBC 설정, CCI 설정 참고)
예) JDBC를 이용한 응용 프로그램에서 연결 포트를 1111 -> 2222로 변경
Connection connection = DriverManager.getConnection("jdbc:cubrid:primary broker:2222:testdb:::?charset=utf8&althosts=secondary broker:2222&rctime=30", "dba", "");</td><td><table><tr><td>nA1</td><td>oA2</td></tr><tr><td>nB1_ro</td><td>oB2_ro</td></tr><tr><td>nN1_m</td><td>oN2_m</td></tr></table></td></tr>
<tr><td>H5 단계: [응용 프로그램2]
인터페이스 업그레이드 및
연결 포트 변경</td><td>응용 프로그램2의 인터페이스를 업그레이드하고, 연결 포트를 변경한다.
작업 과정은 H4 단계와 같다.</td><td><table><tr><td>nA1</td><td>nA2</td></tr><tr><td>nB1_ro</td><td>oB2_ro</td></tr><tr><td>nN1_m</td><td>oN2_m</td></tr></table></td></tr>
<tr><td>H6 단계: [브로커2]
브로커 노드 업그레이드 및
BROKER_PORT 변경</td><td>브로커2 노드를 업그레이드하고, 사용하는 모든 브로커 포트, cubrid_port_id를 변경한다. 작업 과정은 H3 단계와 같다.</td><td><table><tr><td>nA1</td><td>nA2</td></tr><tr><td>nB1_ro</td><td>nB2_ro</td></tr><tr><td>nN1_m</td><td>oN2_m</td></tr></table></td></tr>
<tr><td>H7 단계: [DB 노드2]
DB 노드2에서 C1~C5를
수행하고 H2에서 생성한
백업본 복구,
cubrid_port_id, ha_port_id
변경 후
데이터베이스 시작</td><td>DB 노드 2에서 2008 R4.0 마이그레이션 C1 ~ C4 단계를 수행하여, CUBRID 업그레이드를 완료한 후, H2 단계에서 생성한 DB 노드1의 데이터베이스 백업본(예: testdb_bk*)을 DB 2노드에 복구한다.
% scp user1@master:~/DB/testd/testdb_bk0v000 .
% scp user1@master:~/DB/testdb/log/testdb bkvinf ./log/.
% cubrid restoredb testdb

cubrid_port_id, ha_port_id를 H2 단계와 같은 값으로 변경한다.

(C3a)에서 저장한 databases.txt를 $CUBRID_DATABASES에 복사하고, cubrid.conf 및 cubrid_ha.conf를 설정하고(관련 매뉴얼 참고), HA를 구동한다.
% cubrid heartbeat start</td><td><table><tr><td>nA1</td><td>nA2</td></tr><tr><td>nB1_ro</td><td>nB2_ro</td></tr><tr><td>nN1_m</td><td>nN2_s</td></tr></table></td></tr>
<tr><td>H8 단계: [브로커1]
브로커1을 ReadWrite로 전환</td><td>브로커 1을 다시 읽기/쓰기 가능한 모드로 변경하여 재 시작한 후, 상태를 확인한다.
% cubrid broker stop
% vi $CUBRID/conf/cubrid_broker.conf
…
ACCESS_MODE=RW
% cubrid broker start
% cubrid broker status

@ cubrid broker status

% broker - cub cas [16139,33000]
……
KEEP_CONNECTION:AUTO, ACCESS_MODE:RW
……</td><td><table><tr><td>nA1</td><td>nA2</td></tr><tr><td>nB1_rw</td><td>nB2_ro</td></tr><tr><td>nN1_m</td><td>nN2_s</td></tr></table></td></tr>
</table>

- **2008 R2.0 또는 2008 R2.1에서 2008 R4.0으로 HA 마이그레이션**

CUBRID 2008 R2.0 또는 2008 R2.1의 HA 기능을 사용하는 경우, 서버 버전 업그레이드, 데이터베이스 마이그레이션을 수행하고 HA 환경을 새롭게 구축한 후 해당 버전에서 사용되었던 Linux Heartbeat 패키지를 제거해야 한다.

위의 H1~H8 단계를 수행한 후, 아래의 H9 단계를 수행한다.

단계	설명
H9 단계: 기존 Linux heartbeat 제거	이하의 작업은 마스터 및 슬레이브 서버에서 root 계정으로 수행한다. [root@master ~]# chkconfig --del heartbeat // 슬레이브 서버에서 동일 작업 수행

복제 기능 사용 시 주의사항

CUBRID 2008 R4.0 버전부터 복제 기능이 제거되었으므로, 기존의 복제 기능을 사용 중인 환경에서 이중화 환경을 구축하려면 HA 기능을 사용해야 한다. 서버 버전 업그레이드 및 데이터베이스 마이그레이션을 수행한 후, HA 환경을 새롭게 구축할 수 있다. HA 환경 구축과 관련하여, 온라인 매뉴얼의 *관리자 안내서 > CUBRID HA* 를 참고한다.

3. CUBRID 2008 R4.0에서 변경된 사항

새로 추가된 기능

CUBRIDSUS-4562 사용자 정의 변수를 지정하고 제거할 수 있는 SET, DROP VARIABLE 구문 추가

사용자 정의 변수를 지정하고 제거할 수 있는 SET 구문, DROP VARIABLE 구문이 추가되었다.

```
SET @a = 0;
SELECT @a := @a+1 AS row no, i FROM t;

SET @a:=3;
PREPARE stmt FROM 'SELECT i FROM t WHERE i < ?';
EXECUTE stmt USING @a;

DROP VARIABLE @a;
```

CUBRIDSUS-5164 PHP 함수 추가 및 개선

다음과 같은 PHP 함수를 추가하였다. 보다 자세한 설명은 *관련 매뉴얼*을 참고한다.

PHP 함수 이름	설명
cubrid_client_encoding	현재의 CUBRID 연결 문자셋을 나타내는 문자열을 반환
cubrid_close	현재 실행 중인 모든 트랜잭션을 종료
cubrid_db_name	cubrid_list_dbs 함수의 결과로부터 데이터베이스 이름을 반환
cubrid_fetch_array	질의 결과로부터 한 레코드를 얻어 배열을 반환
cubrid_get_autocommit	자동 커밋 설정 상태를 확인
cubrid_lob_export	CLOB 타입의 데이터를 파일로 저장
cubrid_lob_close	cubrid_lob_get 함수가 반환하는 CLOB 타입의 저장소 파일을 닫음
cubrid_lob_get	CLOB 타입 값을 리소스 배열로 반환
cubrid_lob_send	CLOB 타입 데이터를 읽어서 웹 브라우저에 바로 전달
cubrid_lob_size	CLOB 타입 데이터의 크기를 반환
cubrid_next_result	cubrid_execute 하나로 여러 개의 질의 실행 시 다음 질의의 결과 반환
cubrid_ping	연결 상태를 검사
cubrid_query	한 개의 질의만 실행 시 사용
cubrid_set_db_parameter	CUBRID_PARAM_ISOLATION_LEVEL, CUBRID_PARAM_LOCK_TIMEOUT 설정
cubrid_set_autocommit	자동 커밋 설정

또한, cubrid_connect_with_url 함수의 URL 문자열의 끝에 "autocommit=true/false" 지정을 통하여 자동 커밋 여부를 설정할 수 있도록 확장하였다.

```
$con = cubrid_connect_with_url("cci:CUBRID:localhost:31930:TEST:dba::?autocommit=true");
```

CUBRIDSUS-685 컬럼 타입 및 제약 조건 변경 기능 추가

ALTER TABLE ... CHANGE/MODIFY COLUMN 문을 사용하여, 컬럼의 이름, 타입과 크기 및 제약 조건을 변경할 수 있다.

```
다음은 타입 변경 및 제약 조건을 변경하는 예이다.
CREATE TABLE t1 (a INTEGER);

-- 컬럼 a 의 이름을 b 로 변경한다.
ALTER TABLE t1 CHANGE a b INTEGER;

-- 컬럼 a 에 NOT NULL 제약조건을 추가한다.
ALTER TABLE t1 CHANGE a a INTEGER NOT NULL;
또는 ALTER TABLE t1 MODIFY a INTEGER NOT NULL;
```

```
--. 컬럼 col1 의 크기를 10 에서 20 으로 바꾼다.
CREATE TABLE t1 (col1 CHAR(10));
ALTER TABLE t1 MODIFY col1 CHAR(20);

-- 컬럼 col1 의 INT 타입을 BIGINT 타입으로 변환하고, DEFAULT 1 제약 조건을 제거한다.
CREATE TABLE t1 (col1 INT DEFAULT 1);
ALTER TABLE t1 MODIFY col1 BIGINT;

-- 컬럼 col1 의 INT 타입을 BIGINT 타입으로 변환하고, DEFAULT 1 제약 조건을 유지하려면 다음과 같이 한다.
ALTER TABLE t1 MODIFY col1 BIGINT DEFAULT 1;
```

CUBRIDSUS-4498 php5용 CUBRID php 라이브러리 소스 추가

php5를 지원하기 위해 php5 소스를 별도 디렉터리(contrib/php5)에 추가하였으며, php4 사용자를 위해 기존 소스는 contrib/php4에 유지하였다. php4 모듈의 개발은 중단되었으므로, php5를 사용할 것을 권장한다.

CUBRIDSUS-4601 날짜/시간 함수 추가

다음과 같은 날짜/시간 함수를 추가로 제공한다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

SQL 함수 이름	설명
FROM_DAYS	입력 인자로부터 날짜를 반환
FROM_UNIXTIME	'YYYY-MM-DD HH:MM:SS' 형태의 날짜와 시간을 반환
TIME	시간 부분을 추출하여 시간 형태의 문자열을 반환
YEAR	1- 9999 범위의 년도를 반환
MONTH	1-12 범위의 월수를 반환
DAY	1-31 범위의 일을 반환
DAYOFMONTH	1-31 범위의 일을 반환
DAYOFWEEK	1-7 범위의 요일을 반환
DAYOFYEAR	1-366 범위의 년의 날수를 반환
HOUR	시를 반환
MAKEDATE	입력 인자로부터 날짜를 반환
MAKETIME	입력 인자로부터 시간을 반환
MINUTE	0-59 범위의 분을 반환
QUARTER	1-4 범위의 년 분기(QUARTER)를 반환
SEC_TO_TIME	시, 분, 초를 포함한 시간을 반환
SECOND	0-59 범위의 초를 반환
TIME_TO_SEC	0-86399 범위의 초를 반환
TIMEDIFF	두 개의 입력 인자 간의 시간 차이를 반환
TO_DAYS	366-652424 범위의 날 수를 반환
WEEK	0-53 범위의 주를 반환
WEEKDAY	0-6 범위의 요일을 반환
UTC_DATE	UTC 날짜를 'YYYY-MM-DD' 형태로 반환
UTC_TIME	UTC 시간을 'HH:MM:SS' 형태로 반환

CUBRIDSUS-4602 문자열 함수 추가

다음과 같은 문자열 함수를 추가로 제공한다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

SQL 함수 이름	설명
ELT	조건 결과값에 따라 나열한 문자열 중 하나를 반환
INSERT	입력한 문자열의 위치부터 정해진 길이만큼의 부분 문자열을 삽입
REPEAT	입력 문자열에 대해 반복 회수만큼의 문자열을 반환
SPACE	지정한 숫자만큼의 공백 문자열을 반환
SUBSTRING_INDEX	문자열로부터 구분자가 나타나는 횟수 이전까지의 부분 문자열을 반환

CUBRIDSUS-4604 집계 함수 추가

다음과 같은 집계 함수를 추가로 제공한다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

SQL 함수 이름	설명
GROUP_CONCAT	그룹에서 NULL이 아닌 값들을 연결하여 결과 문자열을 반환
STDDEV_POP	모 표준편차를 반환
STDDEV_SAMP	표본 표준편차를 반환
VAR_POP	모 분산을 반환
VAR_SAMP	표본 분산을 반환

CUBRIDSUS-4602 기타 함수 추가

다음과 같은 기타 함수를 추가로 제공한다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

SQL 함수 이름	설명
INDEX_CARDINALITY	테이블에서 인덱스 카디날리티 정보를 반환
MD5	128 비트 체크섬(checksum)을 반환

CUBRIDSUS-4603 SHOW 문 추가

테이블, 컬럼, 인덱스, 뷰, 권한을 확인할 수 있는 SHOW 문을 추가하였다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

SHOW 문	설명
SHOW TABLES	전체 테이블 이름 목록을 출력
SHOW COLUMNS	테이블의 컬럼 정보를 출력
SHOW INDEXES	인덱스 정보를 출력
SHOW GRANTS	사용자 계정에 부여된 권한을 출력
SHOW CREATE VIEW	지정한 뷰의 CREATE VIEW 문을 출력

CUBRIDSUS-3998 외래 키의 이름을 추가로 지정할 수 있도록 문법 확장

제약 조건 이름 외에 별도의 외래 키(FOREIGN KEY)의 이름을 추가로 지정하는 것이 가능하도록 SQL 문법을 확장하였다. 테이블 생성 혹은 변경 시에 외래 키 정의를 위해 다음과 같이 제약 조건의 이름만 쓰는 방식과 제약 조건과 외래 키의 이름을 쓰는 방식을 둘 다 허용한다.

```
CREATE TABLE y (i INT, j INT, FOREIGN KEY fk1 (j) REFERENCES x (i));
CREATE TABLE z (i INT, j INT, CONSTRAINT fk0 FOREIGN KEY fk2 (j) REFERENCES x (i));
ALTER TABLE y ADD FOREIGN KEY fk1 (j) REFERENCES x (i);
ALTER TABLE y ADD CONSTRAINT fk0 FOREIGN KEY fk1 (j) REFERENCES x (i);
```

CUBRIDSUS-685 AUTO_INCREMENT의 초기값 변경 기능 추가

ALTER TABLE 문장을 통해 AUTO_INCREMENT의 초기값을 변경할 수 있도록 하였다.

```
CREATE TABLE tbl (id INT AUTO INCREMENT, val STRING) AUTO INCREMENT = 3;
ALTER TABLE tbl AUTO_INCREMENT = 23;
```

CUBRIDSUS-4428 CCI API에 CLOB 타입에 대한 함수 추가

CLOB 타입과 관련하여 다음과 같은 CCI API 함수를 제공한다.

CCI API 함수 이름	설명
cci_clob_free()	CLOB 구조체에 대한 메모리를 해제
cci_clob_new()	CLOB 데이터가 저장될 빈 파일을 하나 생성하고 clob 구조체를 할당
cci_clob_read()	CLOB 데이터 파일을 읽음

cci_clob_size()	CLOB 데이터 파일의 크기를 반환
cci_clob_write()	CLOB 데이터 파일을 기록

CUBRIDSUS-4077 CCI API에 자동 커밋 설정/확인 함수 및 CCI API 관련 브로커 설정 파라미터 추가

CCI API에 자동 커밋 모드를 확인하고 설정하도록 cci_get_autocommit(), cci_set_autocommit() 함수를 추가하였다. 또한, connect_with_url() 함수의 URL에 자동 커밋 모드를 설정할 수 있도록 하였다.

```
URL=cci:CUBRID:localhost:31930:TEST:dba::?autocommit=true
```

이와 관련한 브로커 파라미터인 CCI_DEFAULT_AUTOCOMMIT 파라미터를 추가하여 CCI API로 개발된 응용 클라이언트의 자동 커밋 실행 여부를 브로커 설정으로 가능하도록 하였으며, cubrid_broker.conf 파일에 다음과 같이 설정할 수 있다. 이 파라미터는 CCI API 및 CCI API로 개발된 인터페이스(PHP, ODBC, OLE DB)를 사용하는 응용 클라이언트에 영향을 끼치며, JDBC로 개발된 응용 클라이언트에는 영향을 끼치지 않는다.

```
CCI_DEFAULT_AUTOCOMMIT=ON
```

CUBRIDSUS-5200 cubrid spacedb 유틸리티에 요약 출력 기능 추가

cubrid spacedb 유틸리티에 데이터 볼륨(DATA), 인덱스 볼륨(INDEX), 일반 볼륨(GENERIC), 임시 볼륨(TEMP), 일시적 임시 볼륨(TEMP TEMP) 별로 전체 공간(total_pages), 사용 공간(used_pages), 빈 공간(free_pages), 볼륨 개수(volume_count)를 합산하여 출력하도록 -s(--summarize) 옵션을 추가하였다.

```
% cubrid spacedb -s test
Summarized space description for database 'test' with pagesize 16384. (log pagesize: 16384)

Purpose           total_pages      used_pages       free_pages       volume_count
------------------------------------------------------------------------------
DATA              1000             3                997              1
INDEX             2000             3                1997             1
GENERIC           5000             236              4764             1
TEMP              3000             3                2997             1
TEMP TEMP         0                0                0                0
------------------------------------------------------------------------------
TOTAL             11000            245              10755            4
```

CUBRIDSUS-3787 cubrid_rel 수행 시 OS와 Bit 버전 정보 출력

cubrid_rel 실행 시 출력하는 버전 정보에 빌드한 OS와 Bit 버전을 추가하였다.

```
$ cubrid_rel
CUBRID 2008 R4.0 (8.4.0.0193) (64bit release build for linux_gnu) (Apr 27 2011 13:40:25)
```

CUBRIDSUS-5133 데이터베이스의 기본 페이지 크기 및 최대 클라이언트 접속 수 설정 파라미터의 기본값 변경

데이터베이스 생성 시 기본으로 지정되는 페이지 크기를 4KB에서 16KB로 변경하였다.

페이지 크기는 cubrid createdb 유틸리티의 --page-size (-s) 옵션으로 지정할 수 있으며, 옵션을 지정하지 않는 경우 기본값으로 16384바이트(16KB)로 지정된다. 이에 따라 기본으로 생성되는 데이터베이스의 크기가 100MB에서 400MB로 증가하였다.

또한, 데이터베이스 서버에 동시에 연결할 수 있는 클라이언트의 최대 수를 지정하는 파라미터인 max_client의 기본값을 50에서 100로 변경하였다.

CUBRIDSUS-4222 바이트 크기 단위로 설정할 수 있는 시스템 파라미터 추가

페이지 개수 단위로 설정하는 파라미터에 상응하는 바이트 크기 단위(bytes, Kilobytes, Megabytes, Gigabytes, Terabytes)로 설정할 수 있는 시스템 파라미터를 추가하였다. 기존에 사용되던 *_pages 계열의 파라미터들을 기존과 같은 의미로 사용할 수는 있지만, 각 파라미터에 상응하는 새로운 파라미터를 사용할 것을 권장한다.

기존 파라미터	추가된 파라미터	기본값 (단위: 바이트)
data_buffer_pages	data_buffer_size	512M
log_buffer_pages	log_buffer_size	4M
sort_buffer_pages	sort_buffer_size	2M
index_scan_key_buffer_pages	index_scan_key_buffer_size	320K
index_scan_oid_buffer_pages	index_scan_oid_buffer_size	64K

또한, 데이터베이스의 기본 페이지 크기가 16KB로 변경됨(*CUBRIDSUS-5133* 참고)에 따라 추가된 파라미터의 기본값도 이를 반영하였다. 추가된 파라미터의 기본값은 위의 표를 참고한다.

CUBRIDSUS-4613 구문/타입 관련 시스템 파라미터 추가

다음과 같은 구문/타입 관련 시스템 파라미터를 추가하였다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

파라미터 이름	설명
add_column_update_hard_default	ALTER TABLE ... ADD COLUMN로 추가되는 컬럼의 값을 고정 기본값(hard_default)으로 할 것인지 지정
alter_table_change_type_strict	타입 변경에 따른 해당 컬럼 값들의 변환 허용 여부를 지정
default_week_format	WEEK 함수의 mode 기본값을 지정
group_concat_max_len	GROUP_CONCAT 함수의 최대 길이를 지정
plus_as_concat	+ 연산자를 문자열 병합 연산자로 사용할 것인지 지정
return_null_on_function_errors	일부 SQL 함수에서 오류 발생 시 NULL을 반환할 것인지 지정
no_backslash_escapes	이스케이프 문자로 백슬래시를 사용할 것인지 지정
require_like_escape_character	LIKE 절의 이스케이프 문자를 사용할 것인지 지정

CUBRIDSUS-4562 데이터베이스 연결 세션 시스템 파라미터 추가

다음과 같은 데이터베이스 연결 세션 관련 시스템 파라미터를 추가하였다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

파라미터 이름	설명
session_state_timeout	CUBRID 세션 데이터가 유지되는 시간을 정의. *CUBRIDSUS-4565* 참고

CUBRIDSUS-4563 LIMIT 절 관련 시스템 파라미터 추가

다음과 같은 LIMIT 절 관련 시스템 파라미터를 추가하였다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

파라미터 이름	설명
use_orderby_sort_limit	"... ORDER BY ... LIMIT row_count" 구문에서 질의의 정렬 및 합병(sort and merge) 과정의 중간 결과를 row_count만큼만 유지할 것인지를 지정 (*CUBRIDSUS-1396* 참고)
multi_range_optimization_limit	다중 범위(col IN (?, ?, ...,?))의 조건을 가지며 인덱스 사용이 가능한 LIMIT 절이 있는 질의에서 사용 (*CUBRIDSUS-4563* 참고)

CUBRIDSUS-5306 보관 로그 파일을 강제로 삭제하도록 할 수 있는 시스템 파라미터 추가

다음과 같이 보관 로그를 강제로 삭제할 수 있는 시스템 파라미터가 추가되었다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

파라미터 이름	설명
force_remove_log_archives	log_max_archives 에 지정된 수 만큼의 보관 로그만 남기고 나머지를 강제로 삭제할 지 여부를 지정

CUBRIDSUS-3496 브로커에서 prepare statement의 핸들 개수를 제한할 수 있는 파라미터 추가

브로커 파라미터인 MAX_PREPARED_STMT_COUNT를 추가하여, 하나의 연결 당 사용할 수 있는 prepared statement 핸들의 개수를 제한할 수 있도록 하였다. 이 파라미터 값을 사용자가 적절히 지정함으로써, 응용 프로그램의 작성 실수로 인해 시스템이 허용하는 메모리를 초과하여 prepared statement 문을 생성하는 것을 사전에 방지할 수 있게 되었다. 이 파라미터의 기본값은 2000이며, 최소값은 1이다.

CUBRIDSUS-1396 하나의 인덱스를 사용하는 여러 조건이 포함된 LIMIT 질의에서 질의 최적화가 되도록 KEYLIMIT 기능 추가

하나의 인덱스를 사용하는 여러 조건이 포함된 LIMIT 질의는 불필요한 정렬 및 탐색을 제거할 수 있도록 KEYLIMIT 기능을 추가하였다. KEYLIMT은 LIMIT 절이 있는 질의에서 인덱스의 검색 대상을 일부 범위로 제한하는 것을 의미한다. 다음 예는 col IN (?, ?, ...) 형태로 하나의 컬럼(i1)이 존재하고, 이 컬럼은 ORDER BY 절에 나타나는 컬럼(i2)의 왼쪽에 나타나므로 KEYLIMIT에 의한 질의 최적화가 가능하다.

```
CREATE TABLE t1 (i1 INT, i2 INT, i3 INT);
CREATE INDEX ON t1(i1, i2 DESC);
CREATE INDEX ON t1(i1, i2 DESC, i3);
INSERT INTO t1 VALUES (0, 0, 00), (0, 1, 01), (0, 2, 02), (0, 3, 03), (0, 4, 04);
INSERT INTO t1 VALUES (1, 0, 10), (1, 1, 11), (1, 2, 12), (1, 3, 13), (1, 4, 14);
INSERT INTO t1 VALUES (2, 0, 20), (2, 1, 21), (2, 2, 22), (2, 3, 23), (2, 4, 24);
INSERT INTO t1 VALUES (3, 0, 30), (3, 1, 31), (3, 2, 32), (3, 3, 33), (3, 4, 34);
INSERT INTO t1 VALUES (4, 0, 40), (4, 1, 41), (4, 2, 42), (4, 3, 43), (4, 4, 44);

SELECT * FROM t1
WHERE i1 IN (1,3)
ORDER BY i2 DESC LIMIT 2;

          i1          i2          i3
=======================================
           1           4          14
           3           4          34
```

참고로, 시스템 파라미터인 use_orderby_sort_limit를 사용하여, "... ORDER BY ... LIMIT row_count" 구문에서 질의의 정렬 및 합병(sort and merge) 과정의 중간 결과를 row_count만큼만 유지할 것인지를 지정할 수 있다.

개선된 기능

CUBRIDSUS-3594 데이터베이스 볼륨 사용량이 절반으로 줄어들도록 개선

인덱스 구조 및 인덱스 노드 생성 동작 등을 개선하여 일반적인 데이터베이스 사용 환경에서 인덱스 볼륨 사용량이 기존 버전 대비 절반 정도로 줄어들도록 하고, 데이터를 저장하는 힙(heap) 파일 구조를 수정하여 데이터 볼륨 사용량이 기존보다 줄어들도록 개선하였다.

CUBRIDSUS-4565 데이터베이스 연결이 지속되는 동안 사용자 정의 변수, prepared statement 등의 정보가 유지되도록 개선

데이터베이스 연결이 지속되는 동안 사용자(응용 프로그램)의 연결 정보가 유지되도록 개선하였다. 사용자 연결 정보가 유지되는 동안 다음과 같은 정보가 유지된다.

- LAST_INSERT_ID()
- ROW_COUNT()
- SET 구문으로 지정한 사용자 정의 변수
- PREPARE name FROM 문

시스템 파라미터인 session_state_timeout에 의해 지정한 시간 동안 연결이 유지된다.

CUBRIDSUS-3996 HA 안정성 및 환경 설정 개선

HA 안정성을 강화하여, 복제 지연으로 발생하는 오류를 수정하고 복제 불일치에 대처할 수 있도록 개선되었다.

또한, 기존에는 cubrid-ha 스크립트로 HA 환경을 설정하고 실행하였으나, 환경 설정은 cubrid_ha.conf 파일을 통하도록 하고 기능은 모두 cubrid heartbeat 유틸리티로 실행할 수 있도록 하였다. 또한, 슬레이브 서버를 구축하는데 필요한 작업의 편의를 위하여 별도의 스크립트($CUBRID/share/scripts/ha/ha_make_slavedb.sh)를 제공한다.

CUBRIDSUS-4876 PRIMARY KEY 생성 시에 키의 정렬 순서를 지정할 수 있도록 확장

기본 키 정의 시에 키의 오름차순 또는 내림차순의 정렬 순서를 지정할 수 있도록 확장하였다.

```
CREATE TABLE pk_tbl (a INT, b INT, PRIMARY KEY (a, b DESC));
ALTER TABLE pk_tbl PRIMARY KEY (a, b DESC);
```

CUBRIDSUS-4891 IN 조건의 값이 2개 이상 존재하는 경우에도 질의 플랜 캐시를 사용할 수 있도록 개선

IN (1, 2, 3)과 같이 IN 조건의 값이 2개 이상 존재하는 질의문이 질의 플랜 캐시를 활용하여 한 번 생성된 질의 플랜이 재사용되도록 개선하였다.

CUBRIDSUS-1440 교착 상태 발생을 최소화하도록 잠금 기법 개선

교착 상태(deadlock) 발생을 최소화할 수 있도록 잠금 기법을 개선하였다. 예를 들어, 하나의 테이블에 동시에 입력하는 트랜잭션들 간에는 더 이상 교착상태가 발생되지 않는다.

CUBRIDSUS-4257 인덱스 생성 시간을 단축하도록 개선

인덱스 생성 과정을 최적화하여 전체 인덱스 생성 시간을 단축하였다. 특히, 한 트랜잭션 내에서 한 테이블에 여러 인덱스를 생성하는 경우의 성능을 대폭 개선하였다.

CUBRIDSUS-4607 질의 수행 시 정렬된 인덱스를 이용할 수 있도록 개선

ORDER BY 절을 포함한 질의를 정렬된 인덱스를 이용하여 별도의 정렬 과정을 거치지 않고 원하는 결과를 생성할 수 있는 최적화를 추가하였다. ORDER BY 절에 있는 모든 컬럼을 포함하는 인덱스를 정렬된 인덱스(ordered index)라고 하는데, 정렬된 인덱스가 되기 위한 일반적인 조건은 ORDER BY 절에 있는 컬럼들이 인덱스의 가장 앞부분에 위치하는 경우이다. 다음과 같은 질의를 수행하는 경우, tab(col1, col2)로 구성된 인덱스는 정렬된 인덱스이다.

```
SELECT * FROM tab WHERE col1 > 0 ORDER BY col1, col2
```

CUBRIDSUS-3656 ORDER BY, GROUP BY 절을 포함한 질의를 인덱스를 이용하여 별도의 정렬없이 수행하도록 개선

ORDER BY, GROUP BY 절의 컬럼으로 구성된 인덱스를 활용하여 별도의 정렬 과정 없이 질의가 수행될 수 있는 최적화를 추가하였다.

```
-- j,k 컬럼에 대한 인덱스가 존재하므로 ORDER BY j,k 에 의한 별도의 정렬 과정이 필요하지 않다.
CREATE TABLE tab (i INT, j INT, k INT);
CREATE INDEX on tab (j,k);
SELECT i,j,k FROM tab WHERE j > 0 ORDER BY j,k;

-- a,b 컬럼에 대한 인덱스가 존재하므로 GROUP BY a,b 에 의한 별도의 정렬 과정이 필요하지 않다.
CREATE INDEX ON T(a, b, c);
SELECT a, MIN(b), c, MAX(b) FROM T WHERE a > 18 GROUP BY a, b;
```

CUBRIDSUS-3655 커버링 인덱스 스캔 지원으로 질의 처리 성능 개선

질의 내에서 참조하는 모든 컬럼을 포함하는 인덱스를 활용하여 질의를 수행하는 경우 커버링 인덱스 스캔을 통해서 질의 처리 성능을 향상시켰다. SELECT 리스트, WHERE, HAVING, GROUP BY, ORDER BY 절에 있는 모든 컬럼의 데이터를 포함하는 인덱스를 커버링 인덱스(covering index)라고 한다. 이러한 커버링 인덱스는 질의 수행 시 필요한 모든 데이터를 포함하고 있기 때문에, 데이터 페이지에 대한 추가적인 접근 없이 인덱스 페이지 접근만으로도 질의 처리가 가능하다.

다음은 커버링 인덱스 스캔의 예인데, 커버링 인덱스 스캔으로 처리되는 경우에는 아래와 같이 질의 플랜의 인덱스 절에 (covers)와 같이 출력된다.

```
CREATE TABLE t (col1 int, col2 int, col3 int);
CREATE INDEX ON t (col1,col2,col3);
INSERT INTO t VALUES (1,2,3),(4,5,6),(10,8,9);
SELECT * FROM t WHERE col1 < 10;

Query plan:

iscan
   class: t node[0]
   index: i t col1 col2 col3 term[0] (covers)
   cost:  fixed 0(0.0/0.0) var 1(0.0/1.0) card 0
```

CUBRIDSUS-4563 인덱스를 이용한 다중 키 범위 조건의 정렬 수행을 최적화하도록 개선

다중 키 범위(col IN (?, ?, ..., ?))의 조건을 가지며 인덱스 사용이 가능한 질의에 대해 시스템 파라미터인 multi_range_optimization_limit의 설정에 따라 중간 값의 정렬을 진행하면서 결과를 수집하는 최적화를 수행할 수 있도록 기능을 개선하였다. LIMIT 절에 지정한 개수가 multi_range_optimization_limit 파라미터 값보다 작거나 같으면 해당 질의 최적화가 수행된다. 기존 버전에서는 LIMIT 절의 유무에 상관없이 항상 결과를 모두 수집한 후 정렬을 수행하였다.

CUBRIDSUS-2562 LIKE 절을 인덱스 스캔으로 처리하는 최적화 확장

기존에 LIKE 절을 인덱스 스캔으로 처리할 수 있는 최적화는 LIKE 'pattern%'와 같이 실제 값으로 패턴이 시작하는 경우뿐이었지만, 호스트 변수를 포함한 경우에 대해서도 인덱스 스캔할 수 있도록 개선하였다. 예를 들어, LIKE ? || '%' 와 같은 형태를 포함하여, LIKE '%' || ?, LIKE '%' || ? || '%' 형태도 인덱스 스캔을 통해서 처리된다.

CUBRIDSUS-4605 USING INDEX 절 확장

기존에는 USING INDEX 절에 명시한 인덱스만 질의 플랜 생성 시에 후보로 고려하였기 때문에, 질의 내에서 여러 테이블을 참조할 때 특정 테이블의 특정 인덱스만 지정하고 싶은 경우에도 모든 테이블에 대해서 사용할 인덱스를 지정해야만 했다. CUBRID 2008 R4.0에서는 USING INDEX 절에 명시하는 단위가 전체 테이블이 아니라 각 테이블로 변경되었다. 이로 인해서 특정 테이블은 사용할 인덱스를 지정하고, 다른 특정 테이블은 질의 최적화기에 맡길 수 있다.

예를 들어, tab1 테이블에 인덱스 idx1, idx2, tab2 테이블에 인덱스 idx3, idx4, idx5 가 정의된 경우에 tab1 테이블에 대한 인덱스만 지정하고 tab2 테이블에 대한 인덱스를 지정하지 않으면, 질의 최적화기는 tab2 테이블의 인덱스까지 고려하여 동작한다.

```
SELECT ... FROM tab1, tab2 WHERE ... USING INDEX tab1.idx1;
```

- 테이블 tab1 의 순차 스캔과 idx1 인덱스 스캔을 비교하여, 최상의 질의 계획을 선택한다.
- 테이블 tab2 의 순차 스캔과 idx3, idx4, idx5 인덱스 스캔을 비교하여, 최상의 질의 계획을 선택한다.

위의 질의를 기존 버전처럼 tab1 테이블의 idx1 인덱스만 사용하여 질의 플랜을 생성하고 싶은 경우에는, 아래와 같이 tab2 테이블에서 인덱스를 선택하지 않도록 지정해주어야 한다.

```
SELECT ... FROM tab1, tab2 WHERE ... USING INDEX tab1.idx1, tab2.NONE;
```

CUBRIDSUS-4620 역순 인덱스 생성 없이 내림차순 인덱스 스캔이 가능하도록 개선

역순 인덱스의 생성 없이도 USE_DESC_IDX 힌트를 사용하여 내림차순 인덱스 스캔이 가능하도록 개선하였다. 다음과 같이 내림차순 정렬이 있는 질의를 수행하기 위해서는 보통 역순 인덱스(reverse index)를 생성한다.

```
SELECT * FROM tab [WHERE ...] ORDER BY a DESC
```

하지만 같은 컬럼 조합에 대해 오름차순 인덱스와 내림차순 인덱스를 생성하면, 교착 상태(deadlock)의 발생 가능성이 높아진다. 이러한 경우를 줄이기 위해 별도의 내림차순 인덱스를 생성하지 않아도 내림차순 인덱스 스캔을 사용할 수 있도록 하였다. 아래와 같이 USE_DESC_IDX 힌트를 사용하여 내림차순 스캔을 사용하도록 명시할 수 있으며, 해당 힌트가 지정되지 않더라도 질의 최적화기가 내림차순 인덱스 스캔이 유리하다고 생각하는 경우에도 자동으로 해당 플랜을 생성한다.

```
SELECT /*+ USE_DESC_IDX */ * FROM di WHERE i > 0 LIMIT 3;
```

CUBRIDSUS-1484 호스트 변수를 포함한 수식을 가지는 질의의 인덱스 스캔 적용 확장

WHERE 절에 호스트 변수를 포함한 수식을 가지는 질의에 대해 적용 가능한 인덱스를 활용하지 못하고 순차 스캔을 하는 문제를 수정하여 인덱스 스캔이 가능하도록 하였다.

CUBRIDSUS-2925 불리언 값과 숫자 간 상호 타입 호환이 가능하도록 개선

불리언(boolean) 값과 숫자 간 상호 타입 호환이 가능하도록 개선하였다. 이로 인해 0이 아닌 값은 TRUE로, 0은 FALSE로 인식한다. 함수의 인자로 불리언 값 대신 숫자 입력을 하거나, 불리언 값과 숫자 간 비교 연산을 하는 것이 가능하다.

```
SELECT IF(STRCMP('test','test1'),'no','yes');
SELECT 1 FROM db_root WHERE (1 = 1) = 1;
```

CUBRIDSUS-4831 CREATE VIEW 문장에 LIMIT 절을 명시할 수 있도록 확장

CREATE VIEW 문장에 LIMIT 절을 명시할 수 있도록 확장하였다.

```
CREATE VIEW v AS SELECT * FROM t ORDER BY id ASC LIMIT 2;
```

CUBRIDSUS-4608 REPLACE 문과 INSERT ... ON DUPLICATE KEY UPDATE 문의 트리거 지원

REPLACE 문과 INSERT ... ON DUPLICATE KEY UPDATE 문 실행 시 트리거를 지원하도록 개선하였다. 다음은 해당 구문을 실행할 때 트리거 이벤트가 발생하는 예이다.

```
CREATE TABLE with_trigger (id INT UNIQUE);
INSERT INTO with trigger VALUES (11);

CREATE TABLE trigger actions (val INT);

CREATE TRIGGER trig_1 BEFORE INSERT ON with_trigger EXECUTE INSERT INTO trigger_actions VALUES (1);
CREATE TRIGGER trig 2 BEFORE UPDATE ON with trigger EXECUTE INSERT INTO trigger actions VALUES (2);
CREATE TRIGGER trig 3 BEFORE DELETE ON with trigger EXECUTE INSERT INTO trigger actions VALUES (3);

INSERT INTO with_trigger VALUES (11) ON DUPLICATE KEY UPDATE id=22;

SELECT * FROM trigger_actions;

         val
==============
           2

REPLACE INTO with_trigger VALUES (22);

SELECT * FROM trigger actions;

         val
==============
           2
           3
           1
```

CUBRIDSUS-4625 DROP TABLE 문장에 IF EXISTS 구문 추가

DROP TABLE 문장에 IF EXISTS 구문을 추가 하였다. 해당 테이블이 존재하지 않는 경우에 에러가 발생되지 않고 정상적으로 실행된다.

```
DROP TABLE IF EXISTS history, t;
```

CUBRIDSUS-5162 UPDATE 문에 ORDER BY 절을 허용하도록 확장

UPDATE 문에 ORDER BY 절을 허용하도록 확장하였다.

```
UPDATE f set b=1 WHERE b=1 ORDER BY a, b;
```

CUBRIDSUS-4601 날짜/시간 타입에서 허용하는 문자열 형식을 확장

날짜/시간 타입에서 허용하는 문자열 형식을 확장하였다. 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

```
SELECT CAST('420' AS DATE);
   cast('420' as date)
======================
  04/20/2011

SELECT CAST('91015' AS TIME);
   cast('91015' as time)
========================
  09:10:15 AM


SELECT CAST('110420091035.359' AS DATETIME);
   cast('110420091035.359' as datetime)
=======================================
  09:10:35.359 AM 04/20/2011

SELECT CAST('110420091035.359' AS TIMESTAMP);
   cast('110420091035.359' as timestamp)
=======================================
```

CUBRIDSUS-3540 JDBC의 ResultSet.findColumn 메소드의 성능 개선

JDBC의 ResultSet 클래스의 findColumn 메소드의 성능을 개선하였다.

CUBRIDSUS-4313 DROP된 테이블의 공간이나 삭제된 레코드의 공간을 최대한 재사용하도록 개선

DROP된 테이블의 공간이나 삭제된 레코드의 공간을 최대한 재사용할 수 있도록 하여 디스크 공간 사용을 최소화하도록 개선하였다.

CUBRIDSUS-5189 REUSE_OID 테이블 옵션으로 생성한 테이블 재생성 시 장시간 소요되는 문제 개선

REUSE_OID 테이블 옵션으로 생성한 테이블을 재생성하는 경우에 대량의 디스크 I/O로 인하여 장시간이 소요되는 경우가 있을 수 있었으나 이를 개선하였다.

CUBRIDSUS-2965 cubrid statdump 유틸리티가 동시성을 보장하도록 개선

동시에 여러 사용자가 cubrid statdump 유틸리티를 통해 데이터베이스 서버 실행 통계 정보를 모니터링하는 경우에 잘못된 정보를 출력할 수 있는 오류를 수정하였다. 단, 각 통계 수치의 누적 값이 8바이트 크기 자료형의 표현 범위를 초과하는 경우에 해당 수치가 초기화된 후 누적됨을 주의해야 한다.

또한, CSQL 인터프리터에서 ;.history all (서버 전체 실행 통계 조회) 명령이 제거되었다. 이러한 명령어 대신에 cubrid statdump 유틸리티를 사용해야 한다.

CUBRIDSUS-4762 INST_NUM, ORDERBY_NUM, GROUPBY_NUM을 포함하는 조건식 확장

INST_NUM, ORDERBY_NUM, GROUPBY_NUM을 포함하는 좀 더 복잡한 조건식을 사용할 수 있도록 확장하였다.

CUBRIDSUS-5125 IN 조건에 대해서 인덱스 스캔 범위 확장 최적화

멀티 컬럼 인덱스를 구성하는 컬럼 중에 질의 문에서 equi 조건이 아닌 첫 번째 조건 컬럼 이후에 위치하는 컬럼에 대한 조건은 인덱스 스캔 시에 스캔 범위가 아니라 키 필터(key filter)로만 활용될 수 있다. 예를 들어, (a, b) 컬럼에 인덱스가 구성되어 있을 때, a 컬럼에 IN 조건이 주어지면 b 컬럼에 대한 조건은 키 필터로만 활용될 수 있어 인덱스 스캔의 범위를 줄여주는 역할을 하지는 못한다. 하지만 2008 R4.0부터는 IN 조건에 대해서는 스캔 범위를 확장하는 최적화를 제공한다. 예를 들어, WHERE a IN (:a1, :a2) AND b BETWEEN :b1 AND :b2와 같은 질의에 대해서 (a = :a1 AND b BETWEEN :b1 AND :b2), (a = :a2 AND b BETWEEN :b1 and :b2)와 같은 두 개의 스캔 범위를 가지는 인덱스 스캔을 수행하도록 한다. 기존 최적화 방법보다 인덱스 스캔 범위를 더욱 한정할 수 있기 때문에, 특히 해당 인덱스가 큰 경우에 성능 향상 효과가 더욱 커진다.

CUBRIDSUS-5076 cubrid checkdb 유틸리티에 특정 테이블 지정 옵션 추가

데이터베이스의 일관성 확인 또는 복구 대상을 지정된 테이블로 한정할 수 있도록 확장되었다. "-i <테이블 목록 파일명> 또는 테이블 이름들"을 직접 지정할 수 있다.

```
-- testdb 데이터베이스에 대하여 tbl1, tbl2 테이블을 검사한다.
cubrid checkdb testdb tbl1 tbl2
```

```
-- testdb 데이터베이스에 대하여 tbl_list.txt 파일에 포함된 테이블을 검사한다.
cubrid checkdb -r -i tbl_list.txt testdb
```

CUBRIDSUS-3844 HA 환경에서 마스터 브로커 셧다운 시 JDBC로 구현된 응용 프로그램의 데이터베이스 연결이 느려지는 문제 개선

HA 환경에서 마스터 브로커가 셧다운(shutdown)된 경우에 JDBC로 구현된 응용 프로그램이 마스터 데이터베이스로 먼저 연결을 시도한 후에 슬레이브 데이터베이스에 연결하였으나, 처음 연결하는 스레드만 마스터 데이터베이스로 한 번 연결을 시도하고 이 후 연결하는 스레드는 슬레이브 데이터베이스로 바로 연결하도록 개선하였다.

CUBRIDSUS-1826 COMMITTED READ 이상의 격리 수준에서 트랜잭션 처리 문제 수정

COMMITTED READ 이상의 격리 수준에서 각 레코드에 설정해야 할 잠금과 호환되지 않는 테이블 잠금만을 획득한 상태에서 커밋되지 않은 데이터를 읽을 수 있는 문제를 수정하였다.

CUBRIDSUS-3697 DEFAULT VALUES 절을 포함한 INSERT 문장을 서버에서 직접 실행할 수 있도록 수정

DEFAULT VALUES 절을 포함한 INSERT 문장이 서버에서 직접 수행되는 경우에도 정상적으로 입력될 수 있도록 수정하였다. insert_execution_mode 파라미터의 값을 6으로 설정하면 위와 같은 INSERT 문장을 서버에서 실행하게 되며, 시스템 파라미터와 관련된 사항은 관련 매뉴얼을 참고한다.

```
CREATE TABLE ids (id int);
ALTER TABLE ids CHANGE column id int DEFAULT 22;
INSERT INTO ids DEFAULT VALUES;
SELECT * FROM ids;
```

CUBRIDSUS-4067 Windows에서도 브로커의 파라미터를 동적으로 변경할 수 있도록 개선

Linux에서만 broker_changer 유틸리티를 사용하여 브로커의 파라미터를 동적으로 변경할 수 있었으나, Windows에서도 동적으로 변경이 가능하도록 개선하였다. 단 Windows Vista 이상 버전에서는 broker_changer와 cubrid broker를 실행하는 명령 프롬프트 창을 관리자 권한으로 실행해야 한다. (*주의 사항> Windows Vista 이상 버전에서 CUBRID 유틸리티를 사용한 서비스 제어 시 권장 사항* 참고)

```
-- LONG_TRANSACTION_TIME 파라미터를 동적으로 변경하는 예
% broker_changer broker1 LONG_TRANSACTION_TIME 30.00
```

CUBRIDSUS-4137 Windows에서의 스레드 동기화 방식 개선

Windows에서 시스템 리소스 사용을 최소화하도록 스레드 동기화 방식을 개선하였다.

CUBRIDSUS-3607 Windows용 CUBRID 설치 시 Visual C++ 2008 재배포 패키지 자동 설치

Windows 용 CUBRID를 설치 시에 Microsoft에서 제공하는 Visual C++ 2008 또는 2008 SP1 재배포 패키지를 별도로 다운로드 받아 설치해야 했으나, 재배포 패키지가 설치되어 있지 않은 경우에 자동으로 설치하도록 개선하였다.

CUBRIDSUS-5074 Windows에서 CUBRID 설치 제거 개선

Windows에서 CUBRID 설치 제거 작업 진행 시에 CUBRID Service Tray가 구동 중인 경우 사용자 확인을 거쳐 이를 종료하고 계속 진행할 수 있도록 하였다.

수정된 사항

CUBRIDSUS-4220 '0000-00-00 00:00:00'에 대한 TIMESTAMP 타입과 DATETIME 타입의 예외 값 추가

TIMESTAMP 타입의 0000-00-00 00:00:00 값이 최소값인 GMT 1970-01-01 00:00:00으로 DATETIME 타입의 0000-00-00 00:00:00.000 값이 최소값인 GMT 0001-01-01 00:00:00.000으로 취급되도록 하였다.

CUBRIDSUS-4613 입력 데이터의 타입 변환을 명시하지 않아도 자동 변환이 가능하도록 수정

CAST 함수로 타입을 명시하지 않아도 지정된 규칙에 맞으면 자동으로 타입 변환이 가능하도록 수정하였다.

자동으로 타입 변환이 되는 예는 다음과 같으며, 보다 자세한 사항은 *관련 매뉴얼*을 참고한다.

```
SELECT date'2002-01-01' + '10';

  date '2002-01-01'+'10'
======================
  01/11/2002

SELECT date'2002-01-01'-'2001-01-01';

  date '2002-01-01'-'2001-01-01'
================================
                31536000000

SELECT 4 + '5.2';

             4+'5.4'
============================
 9.400000000000000e+00
```

CUBRIDSUS-3834 SELECT *, col1 FROM tbl 질의가 동작하도록 수정

"SELECT *, col1 FROM tbl" 질의와 같이 SELECT 리스트에서 전체 컬럼을 명시하는 * 부호에 뒤이어 추가적인 컬럼 이름이 명시되는 질의가 가능하도록 확장하였다.

CUBRIDSUS-3086 VIEW 정의 질의문에 FROM 절 이하가 생략된 경우 해당 VIEW 질의 시에 발생하는 오류 수정

VIEW 정의 질의문에 FROM 절 이하가 생략된 경우 해당 VIEW에 대해 질의를 수행하면 오류가 발생하였으나, 이를 수정하였다.

```
CREATE VIEW v2 AS SELECT 'a' AS x;
SELECT * FROM v2;
```

CUBRIDSUS-1048 SELECT count(*) 구문에 ORDER BY 절 사용이 가능하도록 수정

"SELECT count(*) ..." 구문에 ORDER BY 절을 명시할 경우 오류가 발생하였으나 이를 수정하였다.

CUBRIDSUS-2811 SELECT 리스트에 조건식 사용시에 발생하는 오류 수정

"SELECT (a=0) FROM tbl" 질의와 같이 SELECT 리스트에 조건식을 사용하는 경우에 아래와 같은 오류가 발생될 수 있었으나 이를 수정하였다.

```
ERROR: System error (generate var) in ../../src/parser/xasl_generation.c
```

CUBRIDSUS-3807 특정 인라인 뷰(inline-view)가 NOT IN 조건식에 포함된 질의 수행 시 결과가 틀린 오류 수정

NOT IN 조건식에 다음과 같이 "SELECT * " 형태인 인라인 뷰 테이블의 컬럼이 한 개인 경우에 잘못된 결과를 출력하는 오류를 수정하였다.

```
CREATE TABLE b (col INTEGER);
INSERT INTO a VALUES(1),(2),(3),(4);
INSERT INTO b VALUES(2),(3),(4),(5);
SELECT col FROM b WHERE col NOT IN (SELECT * FROM a);

-- 다음과 같이 잘못된 결과를 출력하였다.
col
=============
5             -- 이 레코드 하나만 출력해야 정상이다.
4
3
2
```

CUBRIDSUS-3857 집합형 데이터 타입이 IN 조건식의 값으로 지정된 경우 발생하는 오류 수정

집합형 데이터 타입이 IN 조건식의 값으로 지정된 경우에 오류가 발생하였으나 이를 수정하였다.

```
CREATE TABLE x (col SET OF INT);
CREATE TABLE y (col SET OF INT);
INSERT INTO x VALUES({0,1,2});
…
SELECT col FROM x WHERE col IN (SELECT col FROM y);
```

CUBRIDSUS-4126 질의 수행 시 적용할 인덱스 선정 순서 수정

여러 개의 인덱스가 존재하는 테이블에 대해 선택할 인덱스의 우선 순위를 수정하였다. 질의 최적화기는 질의 처리에 사용할 수 있는 인덱스가 여러 개 존재하면 각각의 비용을 비교하여 가장 비용이 작은 인덱스를 선택하게 된다. 만약 같은 비용을 갖는 인덱스가 여러 개 있을 경우 다음 순서대로 선택한다.

- 수정 이전: UNIQUE -> INDEX -> REVERSE UNIQUE -> REVERSE INDEX -> Primary Key
- 수정 이후: Primary Key -> UNIQUE -> REVERSE UNIQUE -> INDEX -> REVERSE INDEX

CUBRIDSUS-5067 GROUP BY 절과 ORDER BY 절 처리를 위해 불필요한 정렬을 수행하지 않도록 최적화

GROUP BY 절과 ORDER BY 절 처리를 위해서 불필요한 정렬 과정을 수행하지 않도록 최적화하였다.

```
SELECT b,d,e,f, COUNT(*) FROM x WHERE a=10 AND c=10 GROUP BY b,d,e,f;
UPDATE t SET c='1' WHERE c='1' ORDER BY b,d;
```

CUBRIDSUS-3949 PREFIX 인덱스 생성 시 지정하는 PREFIX 길이가 부적합한 경우 에러를 리턴하도록 수정

PREFIX 인덱스 생성 시 PREFIX 길이에 호스트 변수, 0, 또는 컬럼 크기보다 큰 값을 지정하는 경우에 에러를 리턴하도록 수정하였다.

```
CREATE TABLE tbl(col VARCHAR(10));
-- 다음과 같은 인덱스 생성은 허용하지 않는다.
CREATE INDEX i_tbl_col on tbl(col(?));
CREATE INDEX i_tbl_col on tbl(col(0));
CREATE INDEX i_tbl_col on tbl(col(50));
```

CUBRIDSUS-4712 PREFIX 인덱스를 사용하는 LIKE 질의에서 특정 조건의 질의 결과가 잘못되는 오류 수정

PREFIX 인덱스를 사용하는 질의에서 LIKE 절의 문자열이 PREFIX INDEX 크기를 초과한 경우에 질의 결과가 잘못되는 오류를 수정하였다. 예를 들어 LIKE 'abcdefgh%' 조건에 대해 %를 제외한 조건 문자열 'abcdefgh' 길이가 PREFIX INDEX의 길이 7을 초과하므로 해당 인덱스를 통해서 원하는 문자열을 찾지 못하는 오류가 존재하였다.

```
CREATE TABLE t1 (id INT, v VARCHAR(20));
CREATE INDEX i_t1_v ON t1(v(7));
INSERT INTO t1 VALUES (1, 'abcdefghijk');
-- prefix index의 크기인 7을 초과하는 조건값 사용 시 비정상 결과 - 0건 출력
SELECT id,v FROM t1 WHERE v LIKE 'abcdefgh%';
```

CUBRIDSUS-4895 질의 수행 시 PREFIX 인덱스가 있으면 항상 이를 선택했으나 실행 비용이 더 작은 인덱스를 선택하도록 수정

질의 최적화 과정에서 더 유리한 다른 인덱스가 있더라도 항상 PREFIX 인덱스를 선택하였으나, 가장 비용이 작은 인덱스를 선택하도록 수정하였다.

CUBRIDSUS-4912 컬럼 개수가 8개를 초과하는 다중 컬럼 인덱스의 인덱스 스캔 질의 결과가 잘못되는 오류 수정

다중 컬럼 인덱스를 구성하는 컬럼의 개수가 8개를 초과하는 경우에 인덱스 스캔 질의 결과가 잘못되는 오류를 수정하였다.

CUBRIDSUS-4192 NUMERIC 타입의 데이터에 대해 인덱스 검색 시 결과가 잘못될 수 있는 오류 수정

WHERE 조건에 지정한 NUMERIC 타입의 정밀도-스케일과 인덱스에 저장된 NUMERIC 타입의 정밀도-스케일이 서로 다른 경우에 인덱스를 이용한 검색 과정에서 값의 비교가 잘못되는 오류를 수정하였다.

```
CREATE TABLE tbl ( col1 NUMERIC(6,5), col2 INTEGER);
CREATE INDEX tbl_idx6 ON tbl(col1 asc, col2 asc);
-- NUMERIC 타입은 정밀도 6, 스케일 5이나 비교값인 1.005는 정밀도 4, 스케일 3으로 인식하여 정상 비교가 되지 않았다.
SELECT * FROM tbl WHERE col1 < 1.005 USING INDEX tbl idx6;

-- NUMERIC 타입과 비교값인 1.00500 모두 정밀도 6, 스케일 5로 인식하여 정상 비교가 되었다.
SELECT * FROM tbl WHERE col1 < 1.00500 USING INDEX tbl_idx6;
```

CUBRIDSUS-4997 역순(reverse) 인덱스 생성 이후 일반 인덱스 생성에 영향을 끼치는 오류 수정

역순(reverse) 인덱스를 생성한 이후 같은 컬럼 조합에 대해서 오름차순인 일반 인덱스를 생성해도 역순 인덱스로 잘못 인식되는 오류를 수정하였다.

CUBRIDSUS-4351 인덱스 스캔 조건에 컬럼 타입과 다른 타입이 지정되면 질의 결과가 잘못되는 오류 수정

인덱스 스캔 조건에 명시된 피연산자의 타입이 컬럼 타입과 다른 경우에 타입 변환으로 인해 결과가 잘못 생성되는 오류를 수정하였다.

```
SELECT int_col FROM t1 WHERE int_col < 2.3
-- 2.3을 컬럼과 같은 타입(INT)의 값으로 변환(2)한 후 값을 비교하는 오류가 존재하였다.
```

CUBRIDSUS-5233 문자 타입의 단일 컬럼 내림차순 인덱스를 가지는 테이블에 데이터 입력 과정에서 발생하는 오류 수정

문자 타입의 단일 컬럼 내림차순 인덱스를 가지는 테이블에 데이터 입력하는 과정 중에 다음과 같은 오류가 발생하였으나, 이를 수정하였다.

```
CREATE TABLE t1 (code CHAR(900));
CREATE INDEX ON t1(code DESC);
INSERT INTO t1 VALUES('00000');
INSERT INTO t1 VALUES('00001');
…
ERROR: No error message available.
```

CUBRIDSUS-4033 BETWEEN 조건식의 값을 타입 변환하는 경우 잘못된 결과를 출력하는 오류 수정

인덱스가 설정된 컬럼에 대해 BETWEEN 조건식의 값을 타입 변환하는 경우에 컬럼 타입과 호환되는 타입임에도 잘못된 결과를 출력하는 오류를 수정하였다.

```
CREATE TABLE a (col BIGINT);
CREATE INDEX ON a(col);
INSERT INTO a VALUES(-100);
...
INSERT INTO a VALUES(100);   -- 여러 건의 데이터 입력
-- BETWEEN 범위의 레코드를 출력해야 하나, 한 건도 출력하지 않는 오류가 존재하였다.
SELECT col FROM a WHERE col BETWEEN CAST(-100 AS INT) AND CAST(100 AS INT);

There are no results.
```

CUBRIDSUS-3696 CASCADE 외래 키 액션이 정의된 테이블 간에 참조 순환 관계로 인해 발생하는 오류 수정

테이블들의 외래 키가 상호 참조하는 순환 관계에 있고 CASCADE 외래 키 액션이 정의되어 있는 경우에 발생하는 오류를 수정하였다. 예를 들어, DELETE CASCADE 외래 키 액션이 정의된 경우에 DELETE 문을 수행하면 오류가 발생하였으나, 이를 감지하여 정상 수행되도록 하였다.

CUBRIDSUS-3739 GROUP BY 절에 SELECT 리스트의 위치를 명시하여 그룹핑 컬럼을 지정할 수 있도록 수정

GROUP BY 절에 SELECT 리스트의 위치를 명시하여 그룹핑 컬럼을 지정할 수 있도록 수정하였다. 아래의 두 질의는 같은 의미를 지닌다.

```
SELECT col1, AVG(col2) FROM test_tbl GROUP BY 1;  -- 새로 지원하는 형식
SELECT col1, AVG(col2) FROM test_tbl GROUP BY col1; -- 기존에 허용되던 형식
```

CUBRIDSUS-4897 SELECT 리스트에 포함되지 않은 컬럼을 기준으로 정렬하는 부질의를 포함한 질의가 에러를 발생시키는 오류 수정

다음의 예와 같이 SELECT 리스트에 포함되지 않은 컬럼을 기준으로 정렬하는 부질의(subquery)를 포함한 질의를 수행하는 경우에 정상적으로 수행하지 못하였으나 이를 수정하였다.

```
CREATE TABLE d (a int, b int);
INSERT INTO d VALUES (1, 1), (2, 2), (3, 3);
SELECT * FROM d WHERE a >= (SELECT a FROM d ORDER BY b DESC for ORDERBY_NUM() = 1);
-- 다음과 같은 오류가 발생하였다.
ERROR: Query result contains more than a single tuple.
```

CUBRIDSUS-3207 ALTER VIEW 문장이 정상 수행되지 못하는 오류 수정

이미 정의된 뷰를 변경하는 ALTER VIEW 문장이 정상적으로 수행되지 못하는 오류를 수정하였다.

```
ALTER VIEW v1 AS SELECT a FROM t1;

ERROR: The number of attributes, 3, does not match the number of columns,
1, in the query specification.
```

CUBRIDSUS-4567 ANY, SOME 수량어를 포함하는 그룹 조건식의 피연산자 리스트가 비어 있는 경우에 발생하는 오류 수정

ANY, SOME 수량어를 포함하는 그룹 조건식에서 피연산자 리스트가 비어 있는 경우에 오류가 발생하였으나 이를 수정하였다.

```
SELECT * FROM tb WHERE a=ANY{};
SELECT * FROM tb WHERE a=SOME{};

ERROR: System error (index plan generation - invalid key value)
```

CUBRIDSUS-3477 DATEDIFF 함수에서 두 개의 입력인자의 타입이 서로 다른 경우 발생하는 오류 수정

DATEDIFF 함수에서 두 개의 입력인자 중 하나는 날짜 형태의 문자열이고 다른 하나는 DATE 타입인 경우에 오류가 발생하였으나 이를 수정하였다

CUBRIDSUS-3426 NUMERIC 타입 컬럼이 허용하는 것보다 더 큰 값이 잘못 업데이트될 수 있는 오류 수정

NUMERIC 타입의 컬럼에 대해 업데이트하는 과정에 업데이트할 컬럼과 값의 정밀도/스케일이 서로 다를 때에 컬럼 타입이 허용하는 것보다 더 큰 값이 잘못 업데이트될 수 있는 오류가 있었으나 이를 수정하였다.

CUBRIDSUS-3884 제약조건 정의 시에 출력되는 에러 메시지 수정

제약조건 정의 시에 출력되는 에러 메시지를 수정하였다. 예를 들어, CLOB 타입에 PRIMARY KEY 제약 조건을 지정하려 하는 경우에 UNIQUE 제약 조건에 대한 에러 메시지가 출력되었다.

```
CREATE TABLE foo (c CLOB PRIMARY KEY);
ERROR: The attribute domain "clob" is not suitable for use with the UNIQUE integrity constraint.
```

CUBRIDSUS-3939 SYS_CONNECT_BY_PATH(), CONNECT_BY_ROOT() 함수에서 발생하는 오류 수정

SYS_CONNECT_BY_PATH(), CONNECT_BY_ROOT() 함수의 인자로 ROWNUM과 같은 지정된 경우에 오류가 발생하였으나 수정하였다.

CUBRIDSUS-4356 부질의를 포함한 ORDER SIBLINGS BY 절을 가지는 계층 질의문에서 발생하는 오류 수정

부질의를 포함한 ORDER SIBLINGS BY 절을 가지는 계층 질의문이 정상적으로 수행되지 못하는 오류를 수정하였다.

```
SELECT * FROM test tbl
WHERE c_id IN (
SELECT ID FROM
(SELECT rownum, c id AS ID,
c p id AS p id,
sortd,
use_yn
FROM test_tbl
) t
```

```
WHERE use yn = 'Y'
START WITH ID = 'CODE000001'
CONNECT BY NOCYCLE PRIOR ID = p_id
ORDER SIBLINGS BY sortd)
```

CUBRIDSUS-4875 정렬 순서가 다른 기본 키 인덱스와 고유 인덱스가 잘못 공유되는 오류 수정

컬럼 구성은 같으나 정렬 순서가 다른 기본 키(PRIMARY KEY) 인덱스와 고유(UNIQUE) 인덱스가 잘못 공유되는 오류를 수정하였다.

```
-- 기본키 인덱스와 고유 인덱스의 컬럼 구성은 같으나 마지막 컬럼인 c의 순서가 다르므로 두 인덱스를 공유할 수 없다..
CREATE TABLE i (a INT, b INT, c INT, d INT);
CREATE UNIQUE INDEX on i (a, b, c DESC);
ALTER TABLE i ADD PRIMARY KEY (a, b, c);
```

CUBRIDSUS-4889 호스트 변수를 포함한 질의문을 PREPARE할 때 또는 값을 바인딩하여 실행할 때 발생하는 타입 변환 관련 오류 수정

호스트 변수를 포함한 질의문을 PREPARE할 때 또는 PREPARE한 문장에 값을 바인딩하여 실행하는 경우에 발생하는 각종 타입 변환 관련 오류를 대거 수정하였다.

CUBRIDSUS-313: 컬럼 타입이 INTEGER인 호스트 변수를 문자열 타입으로 바인딩하는 경우, 잘못된 결과를 출력하였으나 INTEGER 타입으로 변환하여 정상 결과를 출력하도록 수정하였다.

CUBRIDSUS-1305, 1173: 컬럼 타입이 NUMERIC인 호스트 변수를 문자열 타입으로 바인딩하는 경우, 오류가 발생하였으나 NUMERIC 타입으로 자동 변환이 가능하도록 수정하였다.

CUBRIDSUS-333: orderby_num()의 조건절에 사용할 호스트 변수를 INTEGER 타입이 아닌 문자열 타입으로 바인딩하는 경우, 잘못된 결과를 출력하였으나 INTEGER 타입으로 변환하여 정상 결과를 출력하도록 수정하였다.

CUBRIDSUS-443, 352, 1480: 질의문에서 날짜 타입을 반환하는 함수와 호스트 변수를 피연산자로 사용하는 경우, 호스트 변수를 날짜/시간 타입으로 변환할 수 없다는 오류가 발생하였으나 자동 타입 변환이 가능하도록 수정하였다.

```
PREPARE s FROM 'select count(*) from t where k between sysdate - ? and sysdate';
EXECUTE s USING 1;
ERROR: Semantic: Cannot coerce host var to type date.
```

CUBRIDSUS-325, 301, 2791: 하나의 컬럼에 대한 표현식에 두 개 이상의 호스트 변수가 피연산자로 사용되는 질의의 수행이 가능하도록 수정하였다.

```
PREPARE st FROM 'INSERT INTO xoo VALUES(?*?)'; EXECUTE st USING 10,20;
```

CUBRIDSUS-825, 2166: 일부 함수의 입력 인자가 호스트 변수인 경우 자동으로 타입 변환이 되지 않는 오류를 수정하였다.

CUBRIDSUS-4704: COALESCE, STR_TO_DATE, EXTRACT와 같은 일부 SQL 함수에서는 호스트 변수를 인자로 사용할 수 없었으나, 이를 지원하도록 수정하였다.

CUBRIDSUS-4208: 컬럼 타입이 TIMESTAMP인 호스트 변수를 VARCHAR 타입으로 바인딩하는 경우, 잘못된 결과를 출력하였으나 TIMESTAMP 타입으로 변환하여 정상 결과를 출력하도록 수정하였다.

CUBRIDSUS-4900 호스트 변수 조건이 있는 컬럼과 같은 컬럼의 조건이 OR 연산자로 연결되는 경우에 질의 결과가 잘못되는 오류 수정

호스트 변수 조건이 있는 컬럼과 같은 컬럼의 조건이 OR 연산자로 연결되는 경우에 동적 질의의 질의 실행 계획이 잘못되어 질의 결과가 잘못되는 오류를 수정하였다.

```
PREPARE s FROM 'SELECT i FROM a3 WHERE (i < 3 AND i > ?) OR (i < 0)';
EXECUTE s USING 1;
```

CUBRIDSUS-3801 테이블 생성 시 중첩된 집합형 타입의 컬럼을 지정하면 오류없이 수행되는 문제 수정

테이블 생성 시 중첩된 집합형 타입의 컬럼을 정의할 때 에러를 발생시키도록 수정하였다. 다음과 같이 중첩된 집합형 타입의 컬럼은 허용하지 않는다.

```
CREATE TABLE tbl (id SET(SET(INT)));
```

CUBRIDSUS-4652 파티션 키 컬럼에 대해 호스트 변수를 바인딩하는 질의의 결과가 잘못되는 오류 수정

파티션 키 컬럼에 대해 호스트 변수를 바인딩하는 질의의 결과가 잘못되는 오류를 수정하였다.

CUBRIDSUS-4671 SELECT 리스트에 없는 컬럼을 GROUP BY 절에 포함하는 경우에 DISTINCT 결과가 잘못 출력될 수 있는 오류 수정

SELECT 리스트에 없는 컬럼을 GROUP BY 절에 포함하는 경우에 DISTINCT 결과가 잘못 출력될 수 있는 오류를 수정하였다.

```
-- 테이블 생성 및 데이터 삽입
CREATE TABLE t (a int, b int);
CREATE INDEX idx ab ON T(a, b);
INSERT INTO t VALUES (0,0);
INSERT INTO t VALUES (0,10);
INSERT INTO t VALUES (1,0);
INSERT INTO t VALUES (1,10);

-- a, b 에 대하여 DISTINCT 를 수행하여 4 건의 정상 결과를 출력하였다.
SELECT DISTINCT a, b FROM T WHERE a >= 0 GROUP BY a, b;

          a           b
============================
          0           0
          0          10
          1           0
          1          10

-- a 에 대하여 DISTINCT 를 수행하였으나 GROUP BY 에 SELECT 리스트에 없는 b 가 있어 4 건의 비정상 결과를 출력하였다.
SELECT DISTINCT a FROM t WHERE a >= 0 GROUP BY a, b;

          a
=============
          0
          0
          1
          1
```

CUBRIDSUS-4883 CLOB 타입을 NCHAR 또는 VARNCHAR로 변환 시 잘못된 값을 반환하는 오류 수정

CLOB 타입을 NCHAR 또는 VARNCHAR로 변환하는 경우 NULL을 반환하였으나, 정상적으로 값을 반환하도록 수정하였다.

CUBRIDSUS-4643 같은 컬럼에 대해 <=> 연산자와 함께 IS NULL 또는 IS NOT NULL을 AND 조건으로 하여 질의 수행 시 결과가 잘못되는 오류 수정

같은 컬럼에 대해 <=> 연산자와 함께 "IS NULL" 또는 "IS NOT NULL"을 AND 조건으로 하여 질의를 수행하는 경우, 결과가 잘못되는 오류를 수정하였다. 기존 버전에서 오류가 발생한 질의는 다음과 같다.

```
CREATE TABLE t (a INT);
```

```
INSERT INTO t VALUES (NULL);

-- 질의 결과가 없어야 하나, NULL 1 건을 출력
SELECT * FROM t WHERE a<=>NULL AND a IS NOT NULL;

NULL

-- NULL 을 출력해야 하나 질의 결과가 없음
SELECT * FROM t WHERE a<=>NULL AND a IS NULL;
```

CUBRIDSUS-2936 같은 데이터베이스 세션 내에서 ROW_COUNT() 값을 유지하도록 수정

기존에는 INSERT 문 등과 함께 ROW_COUNT()를 얻어오는 질의를 한번에 함께 실행시켜야만 정상적인 ROW_COUNT() 값을 얻어올 수 있었지만, 데이터베이스 연결 세션이 유지되는 기간 동안에 ROW_COUNT() 값을 얻을 수 있도록 수정하였다.

CUBRIDSUS-3715 TO_CHAR()의 입력 인자로 문자 타입을 허용하도록 수정

기존 버전에서는 TO_CHAR()의 입력 인자로 날짜/시간 타입과 숫자 타입만이 가능하였으나 문자 계열 타입(CHAR, NCHAR, VARCHAR, NVARCHAR)의 입력이 가능하도록 수정하였다. TO_CHAR()의 입력으로 문자 계열 타입이 오면 아무런 처리를 하지 않고 입력 인자 값을 그대로 출력한다.

CUBRIDSUS-3098 CEIL 함수의 오류 수정

CEIL 함수의 인자로 문자열을 입력하는 경우 오류 발생하지 않고 자동 타입 변환되도록 수정하였다. 또한, 입력 값이 매우 큰 경우 결과가 잘못되는 오류를 수정하였다.

CUBRIDSUS-4551, 4550 AVG 함수, VARIANCE 함수, STDDEV 함수 변경

기존에는 AVG 함수, VARIANCE 함수, STDDEV 함수의 리턴 타입이 함수의 인자 타입으로 정의되었지만, DOUBLE 타입으로 변경되었다. 예를 들어, 인자의 타입이 INTEGER인 경우 기존에는 함수 결과 타입이 INTEGER로 결정되어 소수점 이하의 값이 잘리었으나 DOUBLE 값을 리턴하도록 변경하였다.

또한, VARIANCE 함수, STDDEV 함수는 각각 기존의 표본 분산, 표본 표준편차 대신 모 분산, 모 표준편차를 리턴한다.

CUBRIDSUS-4705 순차 스캔을 하는 UPDATE 질의 수행 시 잠금 방식의 오류 수정

순차 스캔(sequential scan)하는 UPDATE 질의에 대해 테이블에 의도 배타 잠금(IX_LOCK)을 획득한 상태에서 업데이트 대상인 레코드를 읽어들이는 경우에만 해당 레코드에 배타 잠금(X_LOCK)을 획득함으로 인해서 트랜잭션의 일관성이 보장되지 못하는 오류를 수정하였다.

CUBRIDSUS-5048 DELETE 작업 철회로 인해 질의 실행 계획이 잘못될 수 있는 오류 수정

대량의 DELETE 작업 철회로 인해 질의 실행 계획이 기대와 다를 수 있는 오류를 수정하였다.

CUBRIDSUS-5058 CONNECT BY 질의에 루프가 존재하는 경우 임시 볼륨 사용량이 급격히 증가할 수 있는 오류 수정

CONNECT BY 질의에 루프(loop)가 존재하는 경우 임시 볼륨 사용량이 급격히 증가할 수 있는 오류를 수정하였다.

CUBRIDSUS-4873 ORDER BY 절에서 FLOAT, MONETARY 타입 컬럼에 의한 정렬 시 결과를 잘못 출력하는 오류 수정

ORDER BY 절에서 FLOAT, MONETARY 타입 컬럼 기준으로 정렬하는 경우 결과를 잘못 출력하는 오류를 수정하였다.

CUBRIDSUS-4824 CREATE TABLE AS SELECT 문에 LIMIT 절이 있는 경우 응응 프로그램이 비정상 종료하는 오류 수정

CREATE TABLE AS SELECT 문에 LIMIT 절이 포함된 경우 응용 프로그램이 비정상 종료하는 오류를 수정하였다.

CUBRIDSUS-4141 CSQL 인터프리터의 기본 질의 수행 방식을 변경

CSQL 인터프리터의 기본 모드를 single-line 모드로 변경하였다. 따라서, 질의를 실행시키기 위한 CSQL 세션 명령어(;run 또는 ;xrun)을 별도로 입력하지 않더라도 입력된 질의를 수행하도록 변경되었다. 기존 버전처럼 사용하려면 --no-single-line 옵션을 지정하면 된다.

CUBRIDSUS-4776 다중으로 동시에 같은 테이블 생성 시도 시 데이터베이스 서버 프로세스 멈춤(hang) 오류 수정

다중으로 동시에 같은 테이블을 생성하려고 하는 경우 데이터베이스 서버가 멈추는(hang) 오류를 수정하였다.

CUBRIDSUS-4579 BEFORE/AFTER UPDATE 트리거가 있는 테이블에 여러 건의 "INSERT ... ON DUPLICATE KEY UPDATE" 질의 수행 시 비정상 동작 오류 수정

BEFORE/AFTER UPDATE 트리거가 있는 테이블에 "INSERT ... ON DUPLICATE KEY UPDATE" 질의를 수행하여 같은 테이블에 여러 건의 레코드를 입력하는 경우, 동작을 멈추거나 비정상 종료하는 오류를 수정하였다.

CUBRIDSUS-4409 sort_buffer_pages 파라미터의 크기 단위가 데이터베이스 생성 시 정의한 페이지의 크기가 되도록 수정

정렬 수행 시 사용하는 버퍼의 페이지 개수를 지정하는 시스템 파라미터인 sort_buffer_pages의 크기 단위가 기존에는 4KB로 고정되어 있었으나, 데이터베이스 생성 시 정의한 페이지의 크기가 되도록 수정하였다. 2008 R4.0부터 sort_buffer_pages 파라미터 대신 sort_buffer_size 파라미터 사용을 권고하며, 설정과 관련한 부분은 *관련 매뉴얼*을 참고한다.

CUBRIDSUS-4394 하나의 컬럼에 대한 조건이 OR로 연결되어 있을 때 정렬 결과가 잘못되는 오류 수정

하나의 컬럼에 대한 두 개 이상의 조건 범위가 OR로 연결된 질의에 대해 ORDER BY 절 처리 생략 최적화가 잘못 적용되어 결과 순서가 잘못되는 오류를 수정하였다.

```
SELECT i FROM u WHERE (i < 3) OR (i > 7) ORDER BY i DESC;

i
=============
2
1
9
8
```

CUBRIDSUS-4536 트리거에 의한 INSERT/UPDATE/DELETE 작업의 무한 반복으로 인해 응용 프로그램이 비정상 종료할 수 있는 오류 수정

트리거의 정의에 따라 CASCADING 되는 INSERT/UPDATE/DELETE 작업의 무한 반복으로 인해 응용 프로그램이 비정상 종료할 수 있는 오류가 존재하였으나, 트리거에 의한 반복 횟수를 제한하여 오류가 발생하지 않도록 수정하였다. 기존에는 트리거의 재귀 호출 횟수의 최대치 제한이 없었지만, 최대 32회까지 지정할 수 있도록 변경되었다.

CUBRIDSUS-4491 문자형 타입의 경우 문자열 중간에 있는 NULL 문자를 문자열의 끝으로 인식하는 오류 수정

문자열 중간에 있는 NULL 문자를 문자열의 끝으로 인식하여 질의 결과가 잘못되는 오류가 존재하였으나, 실제 데이터 길이만큼 인식하도록 수정하였다.

```
CREATE TABLE t(i INT, c CHAR(10), s VARCHAR(10));
INSERT INTO t SELECT 1, 'AB'+chr(0)+'CDE', 'AB'+chr(0)+'CDE' ;
-- 질의 결과가 출력되지 않음
SELECT * FROM t WHERE i > 0 AND c LIKE '%E'
```

CUBRIDSUS-4461 JDBC에서 DatabaseMetaData의 getColumns 메소드가 BLOB/CLOB 타입을 지원하도록 수정

JDBC에서 BLOB/CLOB 타입 컬럼에 대해 DatabaseMetaData 클래스의 getColumns 메소드가 NULL을 반환하는 오류를 수정하였다.

CUBRIDSUS-4216 동일 컬럼에 다른 별칭을 부여한 SELECT 문에서 ORDER BY 절에 별칭이 아닌 컬럼 명을 사용한 경우 발생하는 오류 수정

동일한 컬럼에 대해 별칭(alias)을 다르게 부여하고 ORDER BY 절에는 해당 컬럼 명을 사용하여 질의를 수행하는 경우에 발생하는 오류를 수정하였다.

```
-- 오류가 발생하는 질의.
SELECT a a1, a a2 FROM foo ORDER BY a;
```

CUBRIDSUS-4529 ORDER BY 구문에 수식을 사용하는 경우 질의 결과가 잘못 출력되거나 오류 메시지가 발생하는 문제 수정

ORDER BY 구문에 "1+2"와 같은 수식을 사용하는 경우에 발생하는 질의 결과가 잘못 출력되거나 오류가 발생하는 문제를 수정하였다.

```
-- 다음의 경우 결과가 잘못 출력되었다.
SELECT * FROM t1 ORDER BY 1+2 DESC;
-- 다음의 경우 에러가 발생하였다.
SELECT a FROM t1 ORDER BY 1+2 DESC;
```

CUBRIDSUS-4278 ORDER BY 절의 컬럼에 NOT NULL 제약 조건이 정의된 경우에 ORDER BY 절 생략 최적화 가능하도록 수정

ORDER BY 절의 컬럼에 NOT NULL 제약 조건이 정의된 경우에 ORDER BY 절 처리 생략 최적화 가능하도록 수정하였다.

```
CREATE TABLE foo (a INT, b INT, c INT, d INT);
CREATE INDEX idx ON foo (a, b, c DESC, d DESC);

-- 아래 질의는 idx 인덱스에 의해 이미 ORDER BY 에서 지정한 정렬과 같으므로
인덱스 검색 후 추가적인 ORDER BY 정렬을 수행하지 않아도 된다.
SELECT c, d
```

```
FROM foo f
WHERE f.a = 1 AND f.b = 1
USING INDEX idx
ORDER by c DESC, d DESC LIMIT 1;

-- 기존 버전에서는 아래와 같이 ORDER BY 최적화 대상이 되는 컬럼에 NOT NULL 제약조건이 있으면 ORDER BY 최적화가
지원되지 않았다.
CREATE TABLE foo (a INT, b INT, c INT NOT NULL, d INT NOT NULL);
```

CUBRIDSUS-4235 COUNT와 같은 집계 함수를 얻어오는 질의문이 ORDER BY 절을 포함한 경우 질의 수행이 가능하도록 수정

COUNT와 같은 집계 함수를 얻어오는 질의문이 ORDER BY 절을 포함한 경우에 정상 수행되지 못하고 에러를 리턴하였으나 이를 수정하였다.

```
SELECT COUNT(*) FROM athlete ORDER BY code;

ERROR: athlete.code is not single valued. Attributes exposed in aggregate queries must also appear in
the group by clause.
```

CUBRIDSUS-4228 COUNT 함수와 LIMIT 절이 포함된 질의 결과 변경

COUNT 함수가 LIMIT 절과 함께 사용될 때 출력되는 결과가 변경되었다. 예를 들어, SELECT COUNT(*) FROM t LIMIT 1; 과 같은 질의가 기존에는 SELECT COUNT(*) FROM t WHERE ROWNUM = 1;과 같이 처리되어 결과로 항상 1이 출력되었지만, 이제는 SELECT a FROM (SELECT COUNT(*) FROM t) t(a) WHERE ROWNUM = 1;과 같이 처리되어 전체 테이블 수가 출력된다.

CUBRIDSUS-4909 컬럼 타입만 다르게 재생성된 테이블에 대해 prepared 문이 재실행되는 도중 발생하는 오류 수정

브로커 설정이 STATEMENT_POOLING=ON인 환경에서 테이블 이름은 같으나 컬럼 타입이 다르게 재생성된 테이블에 대해 prepared 문이 재실행되는 경우에 호스트 변수의 타입을 기존 타입으로 바인딩하면서 실행에 실패하는 오류를 수정하였다.

CUBRIDSUS-5071 LOB 타입 저장소의 경로가 존재하지 않으면 데이터베이스 시작에 실패하는 경우 수정

databases.txt 파일에 지정된 해당 데이터베이스의 LOB 타입 저장소 경로가 존재하지 않으면 데이터베이스 시작에 실패하였으나, notification 메시지를 로그 메시지에 남기고 데이터베이스를 시작하도록 수정하였다. LOB 타입을 실제로 사용하기 위해서 해당 LOB 타입 저장소 경로가 반드시 존재해야 한다.

CUBRIDSUS-3893 비정상적 TIME 리터럴을 정상으로 받아들이는 오류 수정

time'01:02:03:04' 리터럴과 같이 시, 분, 초 이외에 추가 문자열이 입력되어도 이를 정상 입력으로 받아들이는 오류가 존재하였으나, 이를 수정하였다.

CUBRIDSUS-3895 날짜 타입에 대해 4자리 수 미만인 연도 출력 변경

날짜 타입에 대해 4자리 수 미만인 연도를 출력할 때 앞에 0을 패딩하도록 수정하였다.

```
SELECT DATE'01/01/0001';

-- 기존 출력
01/01/1
-- 수정된 출력
```

```
01/01/0001
```

CUBRIDSUS-3934 IF() 함수가 포함된 질의가 기존에 수행한 다른 질의의 플랜이 잘못 재사용되는 오류 수정

WHERE 절에 IF() 함수가 포함된 경우에 서로 다른 질의임에도 불구하고 기존에 수행한 질의 플랜이 재사용되어 결국 질의 결과가 잘못되는 오류가 존재하였으나 이를 수정하였다.

```
CREATE TABLE u (i int, j int);
INSERT INTO u VALUES (0,0), (1,0), (0,1), (1,1);

SELECT * FROM u WHERE IF((i <> 0),1,99) = 1;

i j
==========================
1 0
1 1

SELECT * FROM u WHERE IF((i <> 0) AND (j <> 0),1,99) = 1;

i j
==========================
1 0 /* 질의 결과로 이 행이 포함되어서는 안 된다. */
1 1
```

CUBRIDSUS-4168 일부가 생략된 불리언 수식 처리 방식 변경

기존 버전에서는 불리언 수식의 일부가 생략된 경우에는 해당 수식을 항상 거짓으로 간주하여 질의 결과가 없었지만, 이러한 불리언 수식을 다르게 해석하도록 변경하였다.

```
-- 아래의 질의에서 WHERE c2 를 WHERE c2 <> 0 으로 질의 조건을 재작성한다.
SELECT c1 FROM t1 WHERE c2;
SELECT c1 FROM t1 WHERE c2 <> 0;
```

CUBRIDSUS-4167 WHERE 절 조건으로 타입의 범위를 벗어나는 값과 비교 시 잘못된 결과를 출력하는 오류 수정

WHERE 절 조건으로 타입의 범위를 벗어나는 값과 비교하는 경우에 잘못된 결과를 출력하는 오류를 수정하였다.

```
-- col 이 INTEGER 타입이고 INTEGER 의 최소값은 -2147483648 이다. 아래의 조건은 항상 참으로 판단하였다.
WHERE col > -2147483649
```

CUBRIDSUS-4572 LIKE 조건식의 ESCAPE 문자로 NULL 문자, 알파벳 또는 숫자를 허용하도록 수정

LIKE 조건식의 ESCAPE 문자로기존에는 '%', '_'만을 허용하였으나, NULL 문자, 알파벳 또는 숫자를 허용하도록 수정하였다.

CUBRIDSUS-4175 DATE 타입에 + 연산 시 허용하는 최대값을 초과할 수 있는 오류 수정

날짜 타입에 + 연산을 수행하는 경우에 허용하는 최대값인 '9999-12-31'을 초과할 수 있는 오류를 수정하였다.

```
SELECT 1 + DATE'9999-12-31';

01/01/10000
```

CUBRIDSUS-5092 부질의에 LIMIT 절이 포함된 경우 LIMIT 절이 적용되지 않는 오류 수정

부질의에 LIMIT 절이 포함된 경우에 LIMIT 절이 적용되지 않는 오류를 수정하였다.

```
SELECT COUNT(*) FROM (SELECT * FROM db_class LIMIT 1,2) tmp

  count(*)
=============
         39
```

CUBRIDSUS-5066 HA 환경에서 cub_server가 동작을 멈추는 경우 cub_master도 동작을 멈추는 오류 수정

HA 환경에서 데이터베이스 서버 프로세스(cub_server)가 동작을 멈추는 경우, 마스터 프로세스(cub_master)도 동작을 멈추는 문제를 수정하였다. cub_master는 cub_server의 이상 여부를 감지하기 위해 주기적으로 응답을 요청하는데, cub_server의 응답을 일정 시간 이상 수신하지 못하면 cub_server가 비정상인 것으로 판단하여 cub_server 프로세스를 재시작하도록 하였다.

CUBRIDSUS-5305 HA 환경 구축을 위해 데이터베이스 생성 후 슬레이브 서버에서 HA를 처음 시작하는 경우 실패하는 오류 수정

HA 환경 구축을 위해 데이터베이스 생성 후 슬레이브 서버에서 HA를 처음으로 구동할 때 applylogdb 프로세스의 실행이 실패하는 오류를 수정하였다.

CUBRIDSUS-4945 비슷한 이름을 가지는 테이블 생성과 접근 성능 개선

비슷한 이름을 가지는 테이블을 생성하고 접근하는 성능을 개선하였다. 특히 파티션 테이블과 같이 비슷한 이름을 가지는 테이블들을 다수 생성할 때의 성능이 대폭 개선되었다.

CUBRIDSUS-4931 지정된 보관 로그 파일 개수를 초과한 보관 로그 파일이 삭제되지 않는 문제 수정

보관 로그(archive log) 파일의 개수가 log_max_archives 시스템 파라미터 값을 초과했음에도 불구하고 불필요한 보관 로그 파일이 삭제되지 않는 문제를 수정하였다. 작업 중인 트랜잭션 로그 혹은 체크포인트에 의해 데이터 볼륨에 반영하지 못한 데이터 페이지에 대한 로그가 보관 로그 파일에 존재하는 경우 해당 보관 로그 파일은 유지되며, 최신 보관 로그 파일을 제외한 log_max_archives 값을 초과하는 나머지는 삭제된다. 참고로, force_remove_log_archives의 설정도 이러한 동작에 영향을 미친다는 점에 주의한다.

CUBRIDSUS-4833 응용 프로그램에서 데이터 입력 중 인터럽트에 의한 작업 중단 시 잘못된 메시지를 출력하는 오류 수정

응용 프로그램에서 데이터 입력으로 데이터베이스 볼륨이 자동 증가하는 도중 인터럽트에 의해 작업을 중단하는 경우에 다음과 같은 잘못된 메시지를 출력하는 오류를 수정하였다.

```
ERROR: Out of disk space in database
```

CUBRIDSUS-4100 SELECT 리스트에 ORDER BY 절의 컬럼과 같은 컬럼이 2개 이상 존재하는 경우 응용 프로그램이 비정상 종료하는 오류 수정

SELECT 리스트에 ORDER BY 절의 컬럼과 같은 컬럼이 2개 이상 존재하는 경우에 응용 프로그램이 비정상 종료하는 오류를 수정하였다.

```
-- 응용 프로그램이 비정상 종료하는 질의
SELECT a a1, a a2 FROM tbl ORDER BY a;
```

CUBRIDSUS-4096 NUMERIC 타입 비교가 잘못되는 오류 수정

NUMERIC 타입을 비교할 때 두 숫자를 공통으로 표현할 수 있는 정밀도가 NUMERIC 타입의 최대 정밀도를 넘을 경우에 정확히 비교되지 못하는 오류를 수정하였다.

```
CREATE TABLE t (n NUMERIC(38,0));
INSERT INTO t VALUES(1);
SELECT * FROM t WHERE n = 1.0;
0 rows selected.
```

CUBRIDSUS-4083 파티션 컬럼에 대해 IN 연산자에 비교값으로 집합형 값을 CAST 연산자로 타입 변환한 경우의 질의 오류 수정

파티션 테이블의 파티션 컬럼에 대해 IN 연산자에 비교값으로 집합형 값을 CAST 연산자로 타입 변환한 경우에 질의가 정상적으로 수행되지 못하는 오류를 수정하였다.

```
CREATE TABLE t(i INTEGER) PARTITION BY LIST( i ) (PARTITION p0 VALUES IN (1,2,3));
INSERT INTO t VALUES(1);
…
SELECT * FROM t WHERE i IN CAST({'1','2','3'} AS SET OF INTEGER);

ERROR: ' in ' operator is not defined on types integer and set.
```

CUBRIDSUS-5137 잠금 에스컬레이션 시도로 인해 교착 상태가 발생될 수 있는 문제 수정

한 트랜잭션이 시스템 파라미터인 lock_escalation 값을 초과하여 잠금을 획득한 상태에서 잠금 에스컬레이션을 시도하다가 교착 상태가 발생될 수 있었으나 해당 테이블에 대해 테이블 잠금을 얻지 못해서 잠금 에스컬레이션에 실패하여도 계속 트랜잭션을 진행하도록 수정하였다.

CUBRIDSUS-4066 CCI, PHP, ODBC, OLE DB 인터페이스에서 DATETIME 타입 값에 날짜, 시간의 자릿수만큼 출력하도록 수정

CCI, PHP, ODBC, OLE DB 인터페이스로 개발된 응용 프로그램에서 DATETIME 타입 값에 날짜, 시간의 자릿수만큼 출력하도록 수정하였다. '0001-01-01 00:00:00'와 같은 값을 입력하면 '1-1-1 0:0:0'으로 출력되었으나, 수정 이후 '0001-01-01 00:00:00'으로 정상 출력된다.

CUBRIDSUS-4094 cci_connect API에 주어진 연결 정보를 브로커 로그에 출력하는 부분의 오류 수정

cci_connect API에 주어진 연결 정보를 브로커 로그에 출력하는 부분의 오류를 수정하였다.

CUBRIDSUS-1047 JDBC에서 커서가 결과 집합의 마지막에 도달하더라도 결과 집합을 닫지 않고 유지하도록 수정

JDBC에서 ResultSet 클래스의 next() 메소드 호출로 레코드 값을 얻어올 경우에 커서가 결과 집합(resultset)의 마지막에 도달하여 더 이상 가져올 값이 없더라도 결과 집합을 닫지 않도록 수정하였다.

CUBRIDSUS-3799 VIEW 정의문에 포함된 ORDERBY_NUM() 함수가 정상 동작하도록 수정

VIEW 정의문에 ORDERBY_NUM() 조건을 포함하여 정렬된 레코드의 개수를 제한할 수 있도록 수정하였다.

CUBRIDSUS-3742 SHARED 제약조건인 UPDATE 대상 컬럼이 SET 절의 첫 번째에 지정되어야만 업데이트가 되는 오류 수정

SHARED 제약조건을 가진 컬럼이 SET 절의 첫 번째에 위치해야만 업데이트가 되는 오류를 수정하였다.

CUBRIDSUS-5097 컬럼 크기보다 큰 문자열을 INSERT/UPDATE 할 때 문자열이 절삭되어 입력되도록 수정

컬럼 크기보다 큰 CHAR, VARCHAR, NCHAR, VARNCHAR 타입의 문자열을 INSERT/UPDATE를 수행할 때 기존에 에러를 발생시키는 것과 달리 컬럼 크기보다 큰 문자열 부분을 잘라내고 입력되도록 수정하였다.

CUBRIDSUS-3642 멀티 컬럼 인덱스 스캔 결과자 잘못되는 오류 수정

내부적으로 질의가 재작성된 조건을 병합하는 과정의 오류로 인하여 멀티 컬럼 인덱스 스캔 결과가 잘못되는 문제를 수정하였다. 다음은 잘못된 결과를 출력하는 예이다.

```
CREATE TABLE t1 (p1 int, p2 int, p3 int);
CREATE INDEX t1 i ON t1(p1, p2, p3);
INSERT INTO t1 VALUES(1, 1, 1);
INSERT INTO t1 VALUES(1, 2, 2);
INSERT INTO t1 VALUES(1, 3, 3);
INSERT INTO t1 VALUES(1, 4, 4);

-- 아래의 조건은 (p1=? AND p2<?) OR (p1=? AND p2=? AND p3<?)로 재작성된다.
PREPARE st FROM 'SELECT * FROM t1 WHERE p1=? AND (p2<? OR (p2=? AND p3<?))';
EXECUTE st USING 1, 2, 1, 3;

-- 수행 결과가 1 건이 나와야 하나, 조건이 겹치는 부분을 중복 출력하는 오류가 존재하였다.
         p1          p2          p3
=======================================
          1           1           1
          1           1           1
```

CUBRIDSUS-5171 setTransactionIsolation, getTransactionIsolation JDBC API를 JDBC 스펙에 맞게 수정

setTransactionIsolation, getTransactionIsolation 메소드를 JDBC 스펙에 맞도록 수정하였다.

CUBRIDSUS-5093 다중으로 여러 개의 스레드가 동시 실행되는 환경에서 레코드 크기가 데이터베이스 볼륨 페이지 크기보다 큰 경우에 발생할 수 있는 오류 수정

다중으로 여러 개의 스레드가 INSERT/UPDATE/DELETE를 동시 실행하는 환경에서 레코드 크기가 데이터베이스 볼륨의 페이지 크기보다 큰 경우에 다음 에러 메시지가 간혹 발생하면서 질의 수행에 실패할 수 있는 문제를 수정하였다.

```
-18 - All page buffers are fixed
```

CUBRIDSUS-4872 UNCOMMITTED READ 격리 수준에서 UPDATE 처리 오류 수정

UNCOMMITTED READ 트랜잭션 격리 수준에서 UPDATE 질의 시 트랜잭션 처리가 잘못되는 문제를 수정하였다. 다음은 트랜잭션 처리에 오류가 발생하였던 예이다.

```
;autocommit off

CREATE TABLE foo (a INT PRIMARY KEY, b INT);
INSERT INTO foo VALUES(1, 1);
INSERT INTO foo VALUES(3, 1);
INSERT INTO foo VALUES(5, 1);
```

```
COMMIT;

T1:
UPDATE foo SET b = b + 2 WHERE a < 10 AND a <> 3;
2 row affected.

                      T2:
                      INSERT INTO foo VALUES(2, 1);
                      1 row affected. -- 여기에서 T1 이 COMMIT 할 때까지 대기해야 한다.
                      COMMIT;

T1:
UPDATE foo SET b = b + 2 WHERE a < 10 AND a <> 3;
3 row affected.
COMMIT;

<result>
a b
===========
1 5
2 3                  -- b 의 값은 1 이 되어야 한다.
3 1
5 5
```

CUBRIDSUS-4823 SA MODE로 질의 수행 중 인터럽트 시에 비정상 종료하는 오류 수정

SA MODE로 질의 수행 중 ctrl-C 등의 인터럽트 시에 비정상 종료하는 오류를 수정하였다.

CUBRIDSUS-4286 CSQL 인터프리터 실행 시 "dba"가 아닌 "DBA"로 로그인하면 "SET SYSTEM PARAMETERS" 구문을 사용할 수 없는 오류 수정

CSQL 인터프리터 실행 시 소문자 "dba"가 아닌 대문자 "DBA"로 로그인하는 경우, "SET SYSTEM PARAMETERS" 구문을 사용할 권한이 없게 되는 오류를 수정하였다.

CUBRIDSUS-5155 CSQL 인터프리터의 single line 모드에서 SQL문의 주석 처리 오류 수정

CSQL 인터프리터를 실행하는 경우, SQL문의 주석(comment) 중간에 세미콜론(;)이 포함되어 있으면 앞의 문장을 실행할 SQL 문으로 인식하여 수행하는 오류를 수정하였다.

CUBRIDSUS-5274 브로커 로그 파일에 연결 URL 정보를 출력하도록 수정

CUBRID 설치 디렉터리의 log/broker/sql_log 디렉터리에 생성되는 브로커 로그 파일에 연결 URL 정보를 출력하도록 수정하였다.

```
04/20 13:02:52.716 (0) connect db testdb user usr url cci:cubrid:192.168.0.10:30102:testdb:usr::
session id 8648
```

CUBRIDSUS-3560 cubrid unloaddb의 출력 파일 또는 VIEW의 질의 스펙 출력 시 식별자를 [ ]로 감싸도록 수정

cubrid unloaddb 유틸리티의 출력 파일 또는 VIEW의 질의 스펙에 포함되는 식별자를 [ ]로 감싸도록 수정하였다. VIEW의 질의 스펙은 CSQL의 ;sc <table> 명령어 또는 _db_query_spec 테이블에서 확인할 수 있다.

CUBRIDSUS-5120 cubrid addvoldb 작업과 데이터베이스 볼륨 자동 증가 작업이 동시에 수행되는 경우 서버가 멈추는 오류 수정

cubrid addvoldb 유틸리티를 수행하는 스레드와 데이터베이스 볼륨을 자동 추가하는 스레드가 동시에 수행되는 경우에 데이터베이스 서버 프로세스가 동작을 멈추는(hang) 현상이 발생할 수 있었으나 이를 수정하였다.

CUBRIDSUS-4474 cubrid loaddb 수행 시 특정 상황에서 오류가 발생하면서 데이터 가져오기에 실패하는 문제 수정

cubrid loaddb 유틸리티에서 객체 파일의 "%id"에 존재하지 않는 테이블이 지정된 경우, "Unknown class" 오류로 인해 데이터 가져오기(load)에 실패하는 문제를 수정하였다.

CUBRIDSUS-5187 데이터베이스 볼륨 파일 개수와 접속된 클라이언트 개수가 1024개를 초과하는 경우 발생하는 접속 제한 오류 수정

데이터베이스 볼륨 파일 개수와 접속된 클라이언트 개수가 1024개를 초과하는 경우에 데이터베이스 접속이 제한되는 오류를 수정하였다.

CUBRIDSUS-4425 Ubuntu에서 CUBRID 패키지 설치 시 demodb 데이터베이스가 생성되지 않는 오류 수정

Ubuntu 리눅스에 CUBRID 패키지를 설치할 때 demodb 생성 셸 스크립트에서 오류가 발생하였으나, Ubuntu 리눅스의 기본 셸인 dash에서도 동작이 가능하도록 수정하였다.

CUBRIDSUS-5228 CUBRID Manager 설치 패키지 별도 제공

CUBRID Manager 설치 패키지를 별도로 제공한다. CUBRID Manager를 사용하려면 CUBRID 패키지 설치 후 CUBRID Manager를 별도로 설치해야 한다. CUBRID Manager 설치 파일은 *다운로드 페이지* 에서 다운로드하여 설치하도록 한다.

이와 함께 Windows 버전의 Tray 메뉴에 포함되어있던 CUBRID Manager 바로가기를 제거하였다.

CUBRIDSUS-5075 Windows 버전의 브로커에서 통신이 오래 걸리는 경우에 발생하는 통신 오류 수정

응용 프로그램이 Windows 버전의 브로커에서 통신이 오래 걸리는 경우에 응용 프로그램과 브로커 사이에 발생할 수 있는 통신 오류를 수정하였다.

CUBRIDSUS-5096 Windows에서 브로커 로그, 서버 로그의 출력 경로가 잘못된 오류를 수정

Windows에서 브로커 로그, 서버 로그의 출력 경로가 잘못되어 에러 로그를 제대로 기록하지 못하는 오류를 수정하였다.

CUBRIDSUS-4368 Windows에서 cubrid addvoldb 또는 backupdb 유틸리티에 상대 경로가 주어지면 발생하는 오류 수정

Windows에서 cubrid addvoldb 또는 backupdb 유틸리티에 상대 경로로 파일 경로가 주어지면 해당 경로를 찾지 못하는 오류를 수정하였다.

CUBRIDSUS-4665 Windows Vista, Windows 7 이상 버전에서 CUBRID 설치 후 재부팅 시 CUBRID Service Tray가 자동 시작하지 않는 오류 수정

Windows Vista, Windows 7 이상 버전에서 CUBRID를 설치 후 재부팅하면 시작 프로그램에 등록되어 있는 CUBRID Service Tray가 자동으로 시작되도록 수정하였다.

CUBRIDSUS-5168 Windows에서 데이터베이스 서버가 구동되는 데 30초 초과 시 실패 메시지가 출력되는 오류 수정

Windows에서 cubrid server start <database> 명령으로 데이터베이스 서버를 구동시킬 때 30초를 초과하면 정상 구동되었음에도 실패 메시지가 출력되는 오류를 수정하였다.

CUBRIDSUS-4352 "INSERT ... ON DUPLICATE KEY UPDATE" 문 사용 시 VALUES 절에 부질의가 포함되면 응용 프로그램이 비정상 종료하는 오류 수정

"INSERT ... ON DUPLICATE KEY UPDATE" 문 사용 시 VALUES 절에 부질의가 포함되어 있으면 응용 클라이언트가 비정상 종료하는 오류를 수정하였다.

CUBRIDSUS-4549 NOT NULL과 DEFAULT NULL 제약 조건이 동시에 정의될 수 있는 오류 수정

NOT NULL과 DEFAULT NULL 제약 조건이 동시에 정의될 수 있는 오류를 수정하였다.

```
CREATE TABLE t (i int NOT NULL default NULL, j int);
-- 아래의 문장을 실행하면 오류를 반환해야 한다.
ALTER TABLE t ADD COLUMN k int NOT NULL DEFAULT NULL;
```

CUBRIDSUS-4630 PHP Extension 컴파일을 위해 configure 수행 중 발생하는 오류 수정

최근 버전의 autoconf를 지원하도록 config.m4 파일에서 매크로 명 AC_HELP_STRING을 AS_HELP_STRING으로 수정하였다.

(기여자: *네이버 개발자 센터 > CUBRID 프로젝트* 의 nori)

CUBRIDSUS-4642 cubrid --version 출력 정보 추가

cubrid --version 으로 버전 정보 출력 시 빌드된 날짜와 빌드 OS 정보를 포함하도록 수정하였다.

```
% cubrid --version
cubrid (CUBRID utilities)
CUBRID 2008 R4.0 (8.4.0.0193) (64bit release build for linux_gnu) (Apr 27 2011 13:40:25)
```

CUBRIDSUS-4656 JDBC의 getDriverVersion() 메소드가 잘못된 버전 번호를 반환하는 오류 수정

JDBC의 getDriverVersion() API가 잘못된 버전 문자열을 반환하는 오류를 수정하였다.

CUBRIDSUS-4709 디스크 캐시 정보가 잘못되어 데이터베이스 볼륨을 더 이상 사용하지 못하는 문제 수정

데이터베이스 서버가 관리하는 디스크 캐시 정보를 여러 스레드가 동시에 업데이트할 때 잘못 업데이트될 수 있는 문제를 수정하였다. 기존에는 이 정보가 잘못되어 사용 가능한 데이터베이스 볼륨에서 더 이상 페이지를 할당하지 못하여 해당 볼륨이 더 이상 활용하지 못하는 경우가 발생할 수 있었다.

CUBRIDSUS-4723 숫자 문자열의 앞, 뒤에 오는 공백을 무시하고 숫자형 타입으로 변환이 가능하도록 개선

앞, 뒤에 공백을 포함하는 숫자 문자열에서 공백을 제거하고 숫자형 타입으로 변환이 가능하도록 하였다.

```
-- '1.2     '가 1.2 라는 DOUBLE 형으로 변환된다.
SELECT * FROM a WHERE i > '1.2       ';

-- '12345 '가 BIGINT 형으로 변환된다.
SELECT CAST('12345 ' AS BIGINT);
```

CUBRIDSUS-4789 문장 집합 연산자로 구성된 질의문의 질의 결과가 잘못될 수 있는 문제 수정

UNION, INTERSECTION, DIFFERENCE 문 수행 시에 첫번째 SELECT 문장에 명시된 컬럼의 길이만큼만 출력하는 오류를 수정하였다.

```
CREATE TABLE t1(c1 char(1) not null,c2 char(2));
INSERT into t1 VALUES('1','t1');
SELECT c1,c2 FROM t1 UNION SELECT '33' c1,c2 FROM t1;

c1                  c2
============================================
  '1'                 't1'
  '3'                 't1'   -- c1 의 값이 '33'을 출력해야 한다.
```

CUBRIDSUS-4811 AUTO INCREMENT 컬럼 값 생성 시에 서버가 비정상 종료될 수 있는 문제 수정

AUTO INCREMENT 컬럼의 값을 얻어오는 과정 중에 에러가 발생할 때에 서버가 비정상 종료되는 문제를 수정하였다.

CUBRIDSUS-4923 시스템 파라미터 설정 시 값이 잘못된 경우 오류 메시지를 출력하도록 수정

SET SYSTEM PARAMETER 문장에 유효하지 않은 파라미터 값이 주어졌을 때 기존에는 해당 파라미터의 기본값으로 지정되었으나, 오류를 발생시키도록 수정하였다.

CUBRIDSUS-5170 CSQL 인터프리터의 single line 모드에서 결과에 해당하는 질의문의 라인 넘버가 잘못 출력되는 오류 수정

CSQL 인터프리터의 single line 모드로 질의문 입력 파일이 주어지는 경우에 결과에 해당하는 질의문 위치가 잘못 출력되는 오류를 수정하였다.

CUBRIDSUS-4095 인덱스 페이지 생성 시 여유 공간 비율을 변경

인덱스 페이지 생성 시 향후 업데이트를 대비하여 확보하는 여유 공간 비율은 index_unfill_factor 시스템 파라미터로 조정이 가능한데, 이 파라미터의 기본값이 0.2 (페이지 크기의 20%)에서 0.05 (페이지 크기의 5%)로 변경되었다.

CUBRIDSUS-5230 HA 환경에서 cub_master 프로세스가 비정상 종료할 수 있는 오류 수정

cub_master 프로세스의 여러 스레드가 동시에 에러 메시지를 출력하는 과정 중에 비정상 종료할 수 있는 오류를 수정하였다.

4. 주의 사항

새로 추가된 주의 사항

2008 R4.0은 그 이전 버전과 데이터베이스 볼륨이 호환되지 않음

2008 R4.0은 2008 R3.x 및 그 이전 버전과 데이터베이스 볼륨이 호환되지 않으므로, 데이터베이스를 업그레이드하려면 unloaddb/loaddb를 통해 데이터베이스를 마이그레이션해야 한다. *데이터베이스 마이그레이션 절차*를 참고한다.

CUBRIDSUS-5133 데이터베이스의 기본 페이지 크기 기본값 변경

데이터베이스 생성 시 기본으로 지정되는 페이지 크기를 4KB에서 16KB로 변경하였다. 페이지 크기는 cubrid createdb 유틸리티의 --page-size (-s) 옵션으로 지정할 수 있으며, 옵션을 지정하지 않는 경우 기본값으로 16384바이트(16KB)로 지정된다. 이에 따라 기본으로 생성되는 데이터베이스의 크기가 100MB에서 400MB로 증가하였다.

기본 제공되는 cubrid.conf 파일에 포함된 파라미터들의 기본값도 변경되었다.

CUBRIDSUS-5228 CUBRID Manager 별도 패키지 제공

CUBRID Manager를 사용하려면 별도 패키지를 다운로드하여 설치해야 한다.

CUBRIDSUS-4524 복제 기능 제거

CUBRID 2008 R4.0 버전부터 복제 기능이 제거되었으므로, 이중화 환경을 구축하려면 HA 기능을 사용해야 한다. 서버 버전 업그레이드 및 데이터베이스 마이그레이션을 수행한 후, HA 환경을 새롭게 구축할 수 있다. HA 환경 구축과 관련하여, 온라인 매뉴얼의 *관리자 안내서 > CUBRID HA* 를 참고한다.

CUBRIDSUS-5097 컬럼 크기보다 큰 문자열을 INSERT/UPDATE 할 때 문자열이 절삭되어 입력됨

컬럼 크기보다 큰 CHAR, VARCHAR, NCHAR, VARNCHAR 타입의 문자열을 INSERT/UPDATE를 수행할 때 기존에 에러를 발생시키는 것과 달리 컬럼 크기보다 큰 문자열 부분을 잘라내고 입력된다. 자세한 내용은 온라인 매뉴얼의 *CUBRID SQL 설명서 > 문자열 데이터 타입*을 참고한다.

CUBRIDSUS-5341 CUBRID 32bit 버전에서 data_buffer_size에 2G를 초과하는 값을 설정하면 데이터베이스 구동에 실패함

CUBRID 32bit 버전에서 data_buffer_size가 2G를 초과하는 값으로 설정되는 경우 데이터베이스 구동에 실패함에도 불구하고, 구동에 성공(success)하였다는 메시지가 출력된다. 32bit 버전에서는 OS의 한계로 인해 설정값이 2G를 초과할 수 없음에 주의한다.

CUBRIDSUS-4059 VARCHAR 타입의 컬럼에서 값을 가져올 때 커버링 인덱스가 적용되는 경우 뒤에 따르는 공백 문자열이 무시됨

VARCHAR 타입의 컬럼에서 값을 가져올 때 커버링 인덱스가 적용되는 경우, 뒤에 따라오는 공백 문자열은 절삭된다. 질의 수행 시 커버링 인덱스가 적용되면 질의 결과 값을 인덱스에서 가져오는데, 인덱스에는 뒤이어 나타나는 공백 문자열을 제거한 채로 값을 저장하기 때문이다. 이러한 현상을 원하지 않을 경우에는 NO_COVERING_IDX 힌트를 지정하면 된다. 커버링 인덱스에 대한 자세한 설명은 개선된 기능의 *CUBRIDSUS-3655 질의 수행 시 커버링 인덱스를 이용할 수 있도록 개선* 을 참고한다.

```
CREATE TABLE tab(c VARCHAR(32));
INSERT INTO tab VALUES('abcd'),('abcd '),('abcd ');
CREATE INDEX ON tab(c);

-- 아래 질의는 커버링 인덱스가 적용되어 3 개의 데이터가 모두 같은 조건으로 인식된다.
SELECT * FROM tab where c='abcd ' USING INDEX i tab c(+);
c
======================
'abcd'
'abcd'
'abcd'
```

기존 주의 사항

CUBRIDSUS-3757 HA 관련 주의 사항

CUBRID HA에서 트리거 및 자바 저장 프로시저를 사용할 경우 마스터 노드에서 이미 수행된 트리거 또는 자바 저장 프로시저를 슬레이브 노드에서 중복 수행하여 CUBRID HA 그룹 내의 노드 간 데이터 불일치가 발생할 수 있으므로, CUBRID HA에서는 트리거 및 자바 저장 프로시저를 사용하지 않도록 한다.

CUBRID HA는 복제 로그를 기반으로 CUBRID HA 그룹 내의 노드 간 데이터를 동기화하므로 복제 로그를 생성하지 않는 메소드를 사용하거나 CUBRID 매니저를 통해 NOT NULL 옵션 설정 작업 수행 시 CUBRID HA 그룹 내 노드 간 데이터 불일치가 발생할 수 있으므로, CUBRID HA는 메소드를 사용할 수 없고, CUBRID 매니저를 통해 작업할 수 없다.

CUBRIDSUS-5071 데이터베이스 백업/복구 시 LOB 타입 저장소는 복구되지 않음

CUBRID에서 LOB 타입의 데이터는 데이터베이스 볼륨이 아닌 별도의 저장소에 존재하므로, 데이터베이스의 백업/복구 과정에 포함되지 않는다. 즉, 데이터베이스의 백업 시 LOB 타입 저장소는 같이 백업되지 않으므로, 복구 시 LOB 타입 저장소는 복구되지 않는다. LOB 타입 저장소는 별도로 관리해야 한다.

CUBRIDSUS-3826 GLO 클래스 지원 중단에 따른 주의 사항

CUBRID 2008 R3.0 이하 버전은 GLO(Generalized Large Object) 클래스를 사용하여 Large Object를 처리하였으나, CUBRID 2008 R3.1 이상 버전 GLO 클래스를 제거하고 BLOB, CLOB 타입(이하 LOB)을 지원한다. (*관련 매뉴얼 참고*)

기존의 GLO 클래스 사용자는 다음과 같이 작업할 것을 권장한다.

- GLO 데이터를 파일로 저장한 후 어플리케이션 및 DB 스키마에서 GLO를 사용하지 않도록 수정한다.
- 데이터베이스 마이그레이션을 한다. (본 문서의 *데이터베이스 마이그레이션 절차 참고*)
- 변경한 어플리케이션에 맞게 파일을 LOB 데이터로 로딩하는 작업을 수행하도록 한다.
- 수정한 어플리케이션이 정상 동작하는지 확인한다.

참고로, cubrid loaddb 유틸리티는 GLO 클래스를 상속받거나 GLO 클래스 타입을 가진 테이블을 로딩하려는 경우, Error occurred during schema loading 에러 메시지와 함께 데이터 로딩을 중지한다.

GLO 클래스의 지원 중단에 따라 각 인터페이스 별로 삭제한 함수는 다음과 같다.

인터페이스	삭제한 함수
CCI	cci_glo_append_data
	cci_glo_compress_data
	cci_glo_data_size
	cci_glo_delete_data
	cci_glo_destroy_data
	cci_glo_insert_data

	cci_glo_load
	cci_glo_new
	cci_glo_read_data
	cci_glo_save
	cci_glo_truncate_data
	cci_glo_write_data
JDBC	CUBRIDConnection.getNewGLO
	CUBRIDOID.loadGLO
	CUBRIDOID.saveGLO
PHP	cubrid_new_glo
	cubrid_save_to_glo
	cubrid_load_from_glo
	cubrid_send_glo

CUBRIDSUS-4172 BLOB, CLOB 타입 사용 시 제약 사항

BLOB, CLOB 타입(이하 LOB)에 대하여 다음과 같은 제약 사항이 있으므로 사용에 주의한다.

- LOB 타입 컬럼 간 비교 연산(=, <>, IN, NOT IN 등)을 할 수 없으며, 이를 위해서는 문자열 또는 비트열로 타입을 변환한 후 사용해야 한다. 단, IS NULL, IS NOT NULL은 지원한다.
- PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL 제약 조건 또는 인덱스를 정의할 수 없다.
- 테이블 생성 및 수정 시, SHARED 속성을 정의할 수 없으며 DEFAULT 속성은 NULL 값에 대해서만 정의할 수 있다.
- 데이터베이스에는 파일의 위치(LOB Locator)가 저장되고 데이터는 파일로 저장되는 구조이므로, 장애가 발생하여 특정 시점으로 복구할 때 LOB Locator와 LOB 데이터의 매핑이 유효하지 않아 에러가 발생할 수 있다.
- ALTER TABLE DROP 문을 사용하여 컬럼을 삭제하거나, DROP TABLE 문을 사용하여 테이블을 삭제하는 경우 LOB Locator만 삭제되고 LOB 컬럼이 참조하는 외부 파일 시스템의 LOB 파일은 삭제되지 않고 남아있다.
- CUBRID가 제공하는 API나 CUBRID 매니저, csql을 사용하지 않고 사용자 임의로 LOB 타입의 데이터 파일을 직접 수정하면 내용이 일치됨을 보장할 수 없다.

(자세한 설명은 *관련 매뉴얼 참고*)

CUBRIDSUS-4186 Windows Vista 이상 버전에서 CUBRID 유틸리티를 사용한 서비스 제어 시 권장 사항

Windows Vista 이상 버전에서 cubrid 유틸리티를 사용하여 서비스를 제어하려면 명령 프롬프트 창을 관리자 권한으로 구동한 후 사용하는 것을 권장한다.

명령 프롬프트 창을 관리자 권한으로 구동하지 않고 cubrid 유틸리티를 사용하는 경우 UAC(User Account Control) 대화 상자를 통하여 관리자 권한으로 수행될 수 있으나 수행 결과 메시지를 확인할 수 없다.

Windows Vista 이상 버전에서 명령 프롬프트 창을 관리자 권한으로 구동하는 방법은 다음과 같다.

- [시작> 모든 프로그램> 보조 프로그램> 명령 프롬프트]에서 마우스 오른쪽 버튼을 클릭한다.

[관리자 권한으로 실행(A)]을 선택하면 권한 상승을 확인하는 대화 상자가 활성화되고, "예"를 클릭하여 관리자 권한으로 구동한다.

CUBRIDSUS-3217 JDBC에서 연결 정보를 URL 스트링으로 입력하는 경우 물음표를 반드시 명시

JDBC에서 URL 스트링으로 연결 정보를 입력하는 경우, 이전 버전에서는 물음표(?)를 입력하지 않더라도 속성(PROPERTY) 정보가 적용되었으나, CUBRID 2008 R3.0부터는 문법에 따라 반드시 물음표를 명시해야 하고 이를 생략할 경우 에러를 출력한다. 또한, 연결 정보 중 USERNAME과 PASSWORD가 없더라도 반드시 콜론(:)을 명시해야 한다.

```
URL=jdbc:CUBRID:127.0.0.1:31000:db1:::althosts=127.0.0.2:31000,127.0.0.3:31000 -에러 처리
URL=jdbc:CUBRID:127.0.0.1:31000:db1:::?althosts=127.0.0.2:31000,127.0.0.3:31000 -정상 처리
```

CUBRIDSUS-3564 마스터 프로세스와 서버 프로세스 간 프로토콜 변경 및 두 개 버전을 동시에 운영하는 경우 포트 설정 필요

마스터 프로세스(cub_master)와 서버 프로세스(cub_server) 간 통신 프로토콜 변경으로 인해 CUBRID 2008 R3.0 이상 버전의 마스터 프로세스는 하위 버전의 서버 프로세스와 통신할 수 없고, 하위 버전의 마스터 프로세스도 2008 R3.0 이상 버전의 서버 프로세스와 통신할 수 없다. 따라서, 이미 하위 버전이 설치되어 있는 환경에서 새 버전을 추가 설치하여, 두 개 버전의 CUBRID를 동시에 운영하는 경우, 각각 서로 다른 포트를 사용하도록 cubrid.conf의 cubrid_port_id 파라미터를 수정해야 한다.

CUBRIDSUS-2828 데이터베이스 이름에 @를 포함할 수 없음

데이터베이스 이름에 @이 포함되는 경우 호스트 이름이 명시된 것으로 해석될 수 있으므로, 이를 방지하기 위하여 cubrid createdb, cubrid renamedb, cubrid copydb 유틸리티 실행 시 데이터베이스 이름에 @를 포함할 수 없도록 수정하였다.

CUBRIDSUS-3267 Windows 환경에서 디렉터리 경로 설정 시 주의 사항

Windows 환경에서 CUBRID 설치 디렉터리 경로에 공백을 포함하는 경우 정상 설치가 되지 않으므로 주의한다. 또한, DB 언로드/로드/백업 등의 작업 대상 디렉터리 경로에도 공백을 포함할 수 없다.

CUBRIDSUS-3553 CUBRID 소스 빌드 후 실행 시, 매니저 서버 프로세스 관련 오류 발생

사용자가 직접 빌드하여 설치하는 경우, CUBRID와 CUBRID 매니저를 각각 빌드하여 설치해야 한다. 만약, CUBRID 소스만 checkout하여 빌드 후 cubrid service 또는 cubrid manager를 실행하면, `cubrid manager server is not installed`라는 오류가 발생한다.