MySQL 사용자를 위한 CUBRID

2009.10.10

CUBRID

- about CUBRID
- schema in CUBRID
- query in CUBRID
- PHP func. in CUBRID

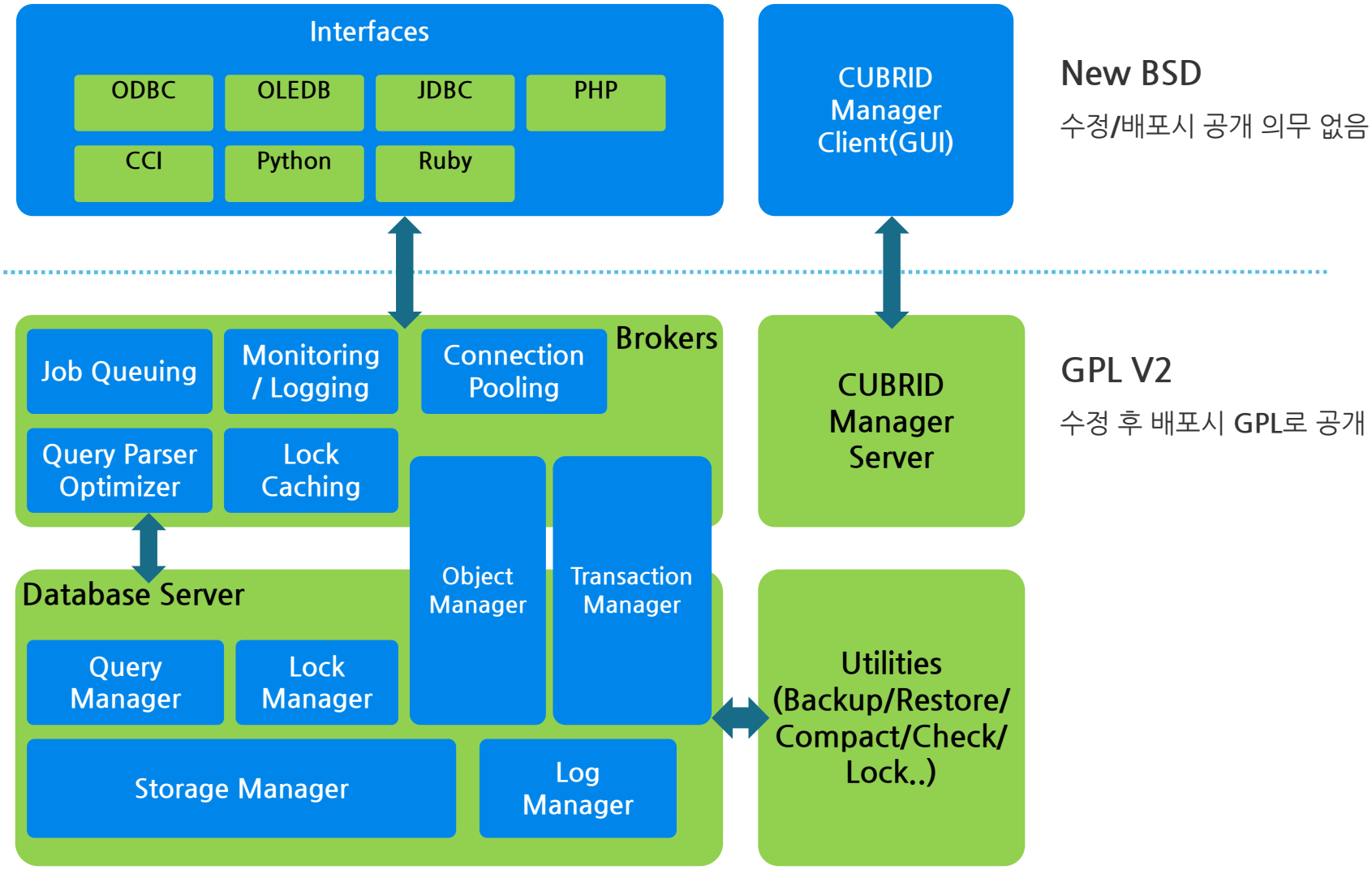
CUBRID

CUBRID 소개

- 오픈 소스 라이선스 데이터베이스
- 인터넷 서비스 최적화 지향
- 대용량 데이터베이스 및 64bit 지원
- HA 및 복제 지원
- 테이블 분할 지원
- 질의 수행 계획, 결과 캐쉬(서버 / 클라이언트)
- JAVA 기반의 stored procedure
- GUI 기반의 관리/질의 수행 도구 (CUBRID manager)

CUBRID 소개 (계속)

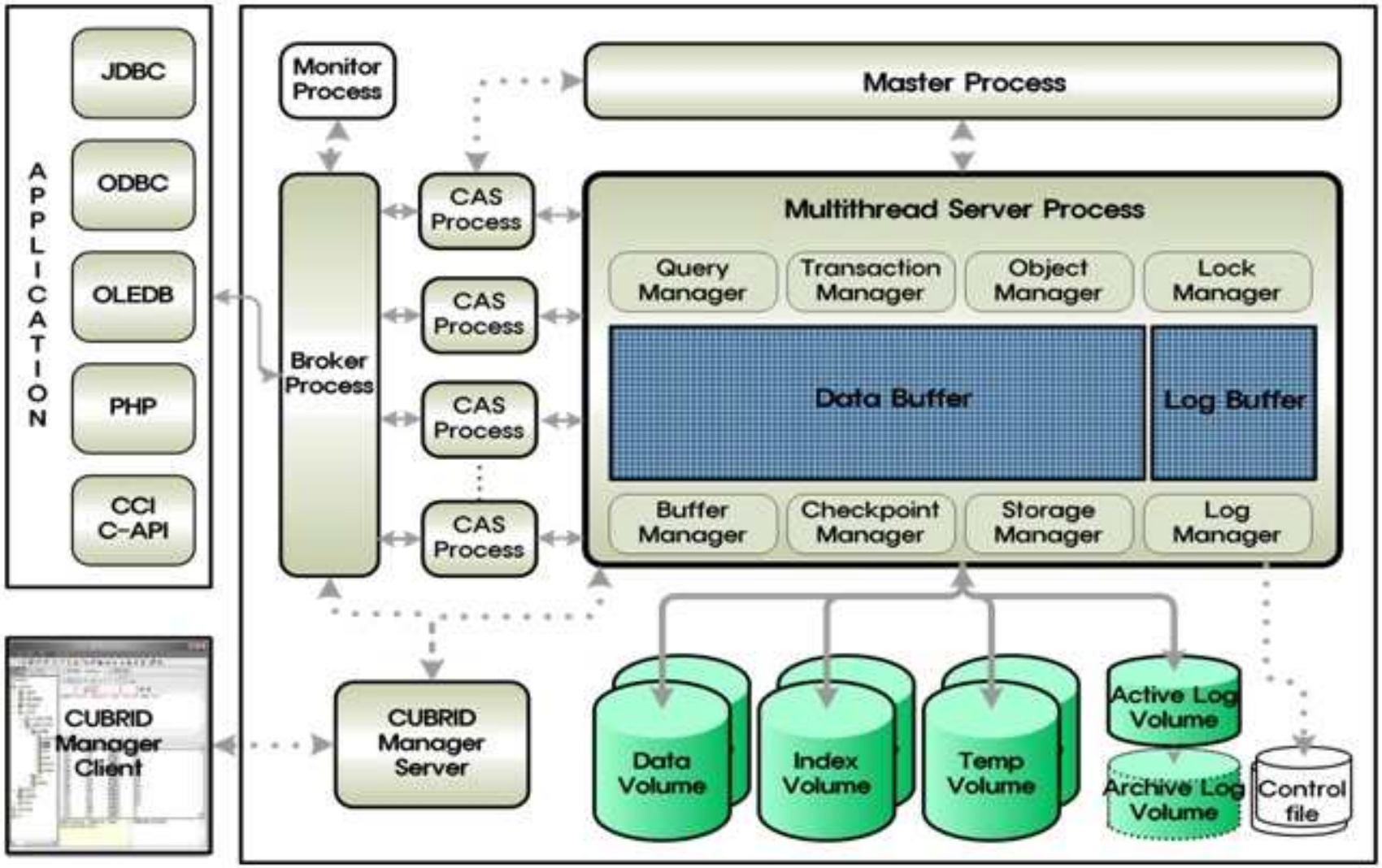
- 오픈소스 라이선스 소개



CUBRID

CUBRID 소개 (계속)

- CUBRID 구조



CUBRID

CUBRID 스키마

- 테이블
  - 개수 무제한
  - 이름에 한글, 영문자, 숫자, _, #, % 사용 가능, 첫글자는 문자(한글, 영문)
  - 이름 최대 길이는 255자
- 컬럼
  - 테이블 당 최대 6,400 개 생성 가능
  - 이름에 한글, 영문자, 숫자, _, #, % 사용 가능, 첫글자는 문자(한글, 영문)
  - 이름 최대 길이는 255자
- index
  - 테이블 당 최대 6,400 개 생성 가능
  - 이름지정 가능
- 제약조건
  - NULL: NULL 값을 허용
  - NOT NULL : NULL 값을 허용하지 않음
  - unique : 중복된 값 허용하지 않음, default 와 같이 선언 불가
  - primary key / foreign key

✓ 예약어 사용시 “”로 감싸주어야 함. 예) “date”

✓ 데이터베이스 이름 최대 길이 : 128자

CUBRID 스키마 (계속)

- 데이터 타입
  - 숫자형
    - smallint: 2bytes, -32768 ~ 32767
      - MySQL : TINYINT, SMALLINT
    - integer: 4bytes, -2147483648 ~ 2147483647
      - 자리수 지정 않됨
      - MySQL : MEDIUMINT, INTEGER
    - bigint: 8bytes, -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
    - numeric: numeric(10), numeric(10,2), numeric(38)
    - float:-10e38 ~ 10e38, 유효자리수 초과하는 수치 정밀도 보장 불가
    - double:-10e308 ~ 10e308, float 와 마찬가지로 유효자리수 문제
  - YEAR, ENUM 은 지원되지 않음

CUBRID 스키마 (계속)

- 문자형
  - byte단위 처리
  - 입력되는 codeset 유지
    - 내부적으로 codeset 지정하지 않음
  - char:고정길이형
    - 다른 컬럼과 같이 저장
    - 길이가 일정하거나 인덱스로 사용될 때
    - select시 rtrim() 필요
  - varchar:가변길이형
    - 최대1G(string)
      - MySQL : VARCHAR, MEDIUMTEXT, TEXT, LONGTEXT 일부
    - 다른 컬럼과 다른 곳에 저장
    - 데이터길이와 저장된 위치에 대한 포인터 저장에 대한 overhead
    - update시 길이 변하는 경우 overhead

CUBRID 스키마 (계속)

- 날자/시간형
  - '' 값 허용 않함, NULL 또는 표현 가능한 값 입력 허용
  - date:날자만 저장, 기본형 mm/dd/yyyy
  - time:시간만 저장, 기본형 hh:mi:ss
  - datetime: 1/100초, 'mm/dd[/yyyy] hh:mi[:ss.ff]' 또는 '[yyyy-]mm-dd hh:mi[:ss.ff]'
  - timestamp:1970/1/1 09:00:00 이후 경과한 초, 최대 2038/1/19 03:14:07
  - to_char() 등 사용시 인덱스 사용 불가, 상수 사용시 기본형 사용 권장
    - 예) where reg_date = '12/25/2009' and reg_time = '18:32:50'
- NULL is not '' (empty string)
  - string_field = ''
  - string_field is null
  - string_field is not null
  - string_field = null (결과 예상 못함)
- 엄격한 데이터 형 비교
  - where 1 = '1' : 에러
  - select 1 + '1' : 에러

CUBRID 스키마 (계속)

- 테이블 생성
  - create table 명령 이용
  - 테이블 복제: 테이블 생성후 insert … select 수행
  - dual 대신 db_root 사용

```
create table <테이블이름>
        [ ( 컬럼정의절 [, {, 컬럼정의절 }] ) ]

컬럼정의절:
        컬럼이름  데이터타입  [default 값] [제약사항]

데이터타입:
        int, bigint, char, varchar, numeric, date, time, timestamp, datetime
제약사항:
        primary key, foreign key, not null, null, unique

예)
create table my_company (
        comp_id      int not null unique,
        comp_name varchar(100) default '',
        seq_no       int auto_increment not null  // not null 과 auto increment 순서 주의
)
```

CUBRID 스키마 (계속)

- 테이블 변경
  - alter table 명령 이용

```
rename table <테이블이름> as <새 테이블이름>

alter table <테이블이름> add atrribute <필드명> <필드타입> [속성]
alter table <테이블이름> rename attribute <필드명> as <새 필드명>
alter table <테이블이름> change <필드명> default <새 값>
```

  - 필드 타입 변경
    - 새 테이블 생성을 통한 변경 : 필드 순서 유지

```
create table <새테이블> (<새 타입을 가지는 필드를 포함한 필드 정보들>)
insert into <새테이블> select ...  from <이전테이블> // 필요한 경우 cast() 등 사용
drop table <이전테이블>
rename table <새테이블> as <이전테이블>
```

    - 새 필드 생성을 통한 변경 : 필드 순서 변경됨

```
alter table <테이블> add attribute <새필드> <새로운 필드 타입>
update <테이블> set  <새필드> = <필드> // 필요한 경우 cast() 등 사용
alter table <테이블> drop attribute <필드>
alter table <테이블> rename attribute <새필드> as <필드>
```

CUBRID 스키마 (계속)

- primary key
  - 테이블 생성시, 생성후 추가 가능
  - 이름 지정 가능하며, default : pk_<테이블이름>_<필드명>[_<필드명>…]

```
create table my_table (
    id int primary key
)
create table my_table2 (
    id int,
    primary key(id)
)
create table my_table3 (
    id int,
    constraint pk1 primary key(id)
)

alter class my_table drop constraint pk_my_table_id

alter class my_table add constaint pk1 primary key(id)
```

CUBRID 스키마 (계속)

- foreign key
  - 테이블 생성시, 생성후 추가 가능
  - 이름 지정 가능하며, default : fk_<테이블이름>_<필드명>[_<필드명>…]

```
create table f_table (
    id int foreign key references my_table(id)
)
create table f_table2 (
    id int,
    foreign key(id) references my_table2(id)
)
create table f_table3 (
    id int,
    constraint fk1 foreign key(id) references my_table3(id)
)

alter class f_table drop constraint f_f_table_id

alter class f_table add constaint fk1 foreign key(id) references my_table1(id)
```

CUBRID 스키마 (계속)

- foreign key options
  - 참조하는 기본키의 수정,삭제에 대한 제한
  - on update : restrict(default), no action
  - on delete : restrict(default), cascade, no action
  - on cache object : CUBRID 에서만 제공, 개체개념 접목
    - 참조하는 테이블을 타입(사용자정의타입)으로 하는 필드추가
    - 추가되어지는 필드의 데이터는 데이터베이스 엔진이 관리
    - 간편한 검색 제공, outer join

```
create table company (
        comp_id int primary key,
        comp_name varchar(100),
)
create table client (
        comp_id int,
        foreign key (comp_id) references company(comp_id) on update restrict on delete cascade on cache object comp_oid
)

select client_name, co.comp_name from client cl, company co where cl.comp_id = co.comp_id
        → '홍길동', 'NHN'
select client_name, co.comp_name from client cl, company co where cl.comp_id = co.comp_id(+)
        → '홍길동', 'NHN'
        → '홍길순', NULL
select client_name, comp_oid.comp_name from client
        → '홍길동', 'NHN'
        → '홍길순', NULL
```

CUBRID 스키마 (계속)

- index
  - 인덱스 이름 지정 가능
  - 생성시 필드별 오름차순, 내림차순 지정 가능
  - unique index, reverse index(모든 필드 내림차순)
  - primary key, foreign key 도 index 를 통한 관리
    - 별도 생성 필요 없음

```
create index on my_table(id, name)

create unique index idx1 on my_table(id)

create index idx on my_table(id, name desc)

create revese index r_idx on my_table(name)
```

CUBRID 스키마 (계속)

- 스키마 조작 제한
  - 스키마 관련 명령어 (DDL) 수행 제한
  - 서비스 운영중 스키마 조작 제한을 통한 서비스 안정화
  - CUBRID 설정화일(cubrid.conf) 상의 설정값 조정
    - block_ddl_statement=yes
    - CUBRID broker 재구동을 통한 설정 반영

CUBRID 스키마 (계속)

- 시스템 카다로그
  - 스키마 정보를 보관한 시스템 테이블
  - 질의를 통한 스키마 정보 조회

  - 테이블 정보

    db_class

    주요 필드 : class_name, owner_name
  - 컬럼 정보

    db_attribute

    주요 필드 : class_name, attr_name, attr_type
  - 기타

    db_vclass

    db_index

    db_index_key

    db_trig

    db_partition

    db_stored_procedure

    db_auth

CUBRID 질의

- SQL-2 (SQL92) 표준
  - 예약어 사용시 “” 로 감싸주어야 함.

- 입력
  - insert into <테이블이름> [(<필드목록>)] values(<값목록>)
  - insert into <테이블이름> select … from …
  - insert into <테이블이름> set : 지원않함

- 수정
  - update <테이블이름> set <필드명> = <값> [, …] [where …]
  - update <테이블이름> set (<필드명>,<필드명> [, …]) = (select <필드명>,<필드명> [, …] from …) [where …]

- 삭제
  - delete from <테이블이름> [where …]

CUBRID 질의 (계속)

- 조건없는 수정 / 삭제 제한
  - where 절 없는 update, delete 수행 제한
  - 서비스 운영중 관리자 혹은 응용 실수로 인한 전체 데이터 수정, 삭제 제한을 통한 서비스 안정화
  - CUBRID 설정화일(cubrid.conf) 상의 설정값 조정
    - block_nowhere_statement=yes
    - CUBRID broker 재구동을 통한 설정 반영

CUBRID 질의 (계속)

- 검색
  - sub-query 지원
  - where (a,b) in (select …)
  - =, <>, >=, <=
  - 표준에 맞추어 질의

```
select 1 ➔ select 1 from db_root
where 1 ➔ where 1 = 1
```

- outer join
  - left outer, right outer join

```
select … from class1 a left outer join class2 b on a.name  = b.name
select … from class1 a, class2 b where a.name = b.name(+)
```

- group by
  - 질의 결과에 대하여 그룹화

```
select a, b, count(*) from my_table group by a, b
```

CUBRID 질의 (계속)

- 인덱스 힌트
  - 질의 수행시 사용할 인덱스 지정
  - 지정된 인덱스만 사용, 지정하지 않은 인덱스는 없는 것으로 간주
    - 지정한 인덱스와 full scan 중 비용이 낮은 것 선택
  - 조인시 필요한 인덱스 모두 지정, 서브질의는 상관없슴
  - (+) 이용하여 인덱스 사용 cost를 0 으로 강제

```
select * from my_table where a = 1 using index NONE

select * from my_table where a = 1 using index idx1

select * from my_table where a = 1 using index idx1(+)

select * from my_table m, my_table2 t where m.a = 1 and m.b = t.b using index m.idx1, t.idx1
```

CUBRID 질의 (계속)

- 조인 힌트
  - 테이블간 조인시 조인 방법, 순서 지정
  - select 절 다음에 /*+ … */ 사이에 지정
  - 조인 방법
    - USE_IDX : 조인시 인덱스 사용
    - USE_NL : 조인시 nested loop 조인 사용
    - USE_MERGE : 조인시 sort merge 조인 사용
  - 조인 순서 지정
    - ORDERED : from 절에 명시된 순서대로 조인, order by 회피

```
select /*+ USE_IDX */ from my_table m, my_table2 t where …

select /*+ USE_IDX(t) */ from my_table m, my_table2 t, my_table3 e where …

select /*+ ORDERED */ from my_table, my_table2 where …
```

CUBRID 질의 (계속)

- serial
  - 일련번호 생성
  - 트랜잭션의 영향 받지 않음

```
create serial seq_no

select seq_no.current_value from db_root
insert into bbs(...) values(seq_no.nextval, ...)

alter serial seq_no start_with 100
```

- auto increment
  - 레코드 입력시 필드값 자동 증가
  - serial 을 이용한 관리
    - 최종 증가된 현재값 확인 가능, 세션에 영향받지 않음

```
create table my_table ( t_no int auto_increment )
create table my_table2 ( t_no int auto_increment(11, 2) )

select my_table_ai_t_no from db_root
```

CUBRID 질의 (계속)

- rownum
  - 검색 결과 레코드에 대한 일련 번호 부여
  - order by, group by 실행 전에 rownum 부여

```
select rownum, id from bbs where rownum between 11 and 20

select id from bbs where rownum between 11 and 20 group by id
select id from bbs where inst_num() between 11 and 20 group by id

select rownum, id from bbs where rownum between 11 and 20 order by id
select orderby_num(), id from bbs order by id for orderby_num() between 11 and 20
```

CUBRID 질의 (계속)

- rownum 을 이용한 페이징 방법
  - seq_no 에 대하여 최근 번호를 우선으로 해서 11번째 부터 10개 가져옴

```
select seq_no from bbs order by seq_no desc limit 11, 10

select seq_no from bbs order by seq_no desc for orderby_num() between 11 and 20
```

  - 인덱스와 rownum 을 이용하여 질의 조정
    - 인덱스를 이용하여 결과를 가져오면 인덱스 순서대로 결과를 가져옴
    - 조건이 있어야 인덱스 사용, 모든 값이 나오도록 조건 추가
    - using index (+)를 이용하여 인덱스 사용하도록 지정

```
create index r_idx on bbs(seq_no desc)

select seq_no from bbs where seq_no > () and rownum between 1 and 10
   using index r_idx(+)
```

CUBRID 질의 (계속)

- click counter
  - 주어진 필드의 값을 1만큼 증가
  - select, update 질의를 select 질의만으로 수행
  - incr() : 현재값을 넘겨주고, 그값을 1만큼 증가시킴

```
select seq_no, incr(read_cnt) from bbs
```

- count(*) 성능 개선
  - 인덱스를 이용한 count

```
create table my_table (
        id int primary key
)

select count(*) from my_table
select count(*) from my_table where id > 0
```

CUBRID 질의 (계속)

- 파티션
  - range, list, hash 지원
  - 개별 파티션 조회 가능

```
create table cal (
    yy int
)  partition by range(yy) (
    partition r_1900 values less than(2000),
    partition r_2000 values less than(3000),
    partition r_etc values less than maxvalue
)

create table zip (
    city char(20)
) partition by list(city) (
    partition l_city0 values in ('서울','부산'),
    partition l_city1 values in (NULL, '경기도')
)

create table log (
    user_id  char(20)
) partition by hash(user_id) partitions 8
```

CUBRID 질의 (계속)

- 함수 등 비교

MySQL	CUBRID	비고
NOW	systimestamp	
ifnull	nvl	
locate(str, main_str)	instr(main_str, str) position(str on main_str)	1..n
left, right	substr	left, right 가 지원되지 않으므로 substr 활용
concat	+ 또는 || 이용	NULL 과 연산시 결과가 NULL 이 되므로 주의
limit	rownum	
ascii	미지원	
reverse	미지원	

- 추가적인 내용은 http://www.cubrid.com/zbxe/32320 참고

PHP 사용하기

- PHP 설치 (간단 소개)
  - http://www.cubrid.com/zbxe/32451 참고
  - phpize 이용 : 현재 사용하는 PHP 의 것 사용
  - 공유라이브러리 패스(LD_LIBRARY_PATH)에 CUBRID PHP 모듈용 CUBRID cci library(libcascci.so) 에 대한 path 추가
  - configure 및 CUBRID PHP 모듈 빌드
  - cubrid.so, cubrid_err.msg 를 /usr/local/php/lib/php/extensionts 에 복사
  - php.ini 에 CUBRID 모듈 정보 등록

```
extension_dir = "/usr/local/php/lib/php/extensions"
extension=cubrid.so
cubrid.err_path="/usr/local/php/lib/php/extensions"
```

PHP 사용하기 (계속)

- CUBRID 함수
  - PHP 모듈 버전 정보
    - cubrid_version → mysql_get_client_info
  - 데이터베이스 연결
    - 주어진 데이터베이스에 연결
    - cubrid_connect → mysql_connect, mysql_select_db

```
$conn = cubrid_connect("db_server", 33000, "demodb", "db_user", "db_passwd");
if ($conn === false) echo("Connect error");
```

  - 데이터베이스 연결 종료
    - 종료되지 않은 트랜잭션은 강제 종료
    - cubrid_disconnect → mysql_close

```
cubrid_disconnect($conn);
```

  - 풀을 이용한 연결
    - 준비중 (mysql_pconnect)

PHP 사용하기 (계속)

- CUBRID 함수
  - 스키마 버전 정보
    - cubrid_schema → mysql_list_fields, mysql_list_tables

```
$sch_info = cubrid_schema($conn,  CUBRID_SCH_CLASS);
$sch_info = cubrid_schema($conn,  CUBRID_SCH_CLASS, "db_class");

$sch_info = cubrid_schema($conn, CUBRID_SCH_ATTRIBUTE, "db_class");
[ATTR_NAME] => class_name [DOMAIN] => 2 [SCALE] => 0 [PRECISION] => 255 [INDEXED] => 0
[NON_NULL] => 0 [SHARED] => 0 [UNIQUE] => 0 [DEFAULT] => [ATTR_ORDER] => 1 [CLASS_NAME]
=> db_class [SOURCE_CLASS] => db_class [IS_KEY] => 0
```

PHP 사용하기 (계속)

- CUBRID 함수
  - 에러 코드
    - cubrid_error_code → mysql_error_no
  - 에러 메시지
    - cubrid_error_msg → mysql_error

PHP 사용하기 (계속)

- CUBRID 함수
  - 질의 컴파일
    - cubrid_prepare
  - 컴파일된 질의에 값 할당
    - cubrid_bind
  - 질의문 수행
    - cubrid_execute → mysql_query

```
$req = cubrid_prepare($conn,  "select * from db_name where name = ?");
$req0 = cubrid_bind($req, "my_table", STRING);

$res = cubrid_execute($conn, $req);

$res = cubrid_execute($conn, "select * from db_name");
```

PHP 사용하기 (계속)

- CUBRID 함수
  - 질의 결과 레코드 개수
    - cubrid_num_rows → mysql_num_rows
  - 질의 결과 컬럼 개수
    - cubrid_num_cols → mysql_num_fields
  - 질의 수행에 영향을 받은 레코드 수 (예, update 된 레코드 수)
    - cubrid_affected_rows → mysql_affected_rows
  - 질의 결과 컬럼 이름
    - cubrid_column_names → mysql_field_name
  - 질의 결과 컬럼 타입
    - cubrid_column_types → mysql_field_type

```
$req = cubrid_execute($conn, $sql);

$rows = cubrid_num_rows($req);
$cols = cubrid_num_cols($req);

$col_names = cubrid_column_names($req);
$col_types = cubrid_column_types($req);
```

PHP 사용하기 (계속)

- CUBRID 함수
  - 질의 결과 레코드 가져오기
    - cubrid_fetch → mysql_fetch_array, mysql_fetch_field, mysql_fetch_row
    - cubrid_fetch_all : 별도 구현, http://www.cubrid.com/zbxe/46530 참고
  - 질의 결과셋 cursor 이동
    - cubrid_move_cursor → mysql_data_seek
  - 질의 결과셋 닫기
    - cubrid_close_request → my_sql_close

```
$req = cubrid_prepare($conn, "select class_name, owner_name from db_class");
$res = cubrid_execute($req);
// 또는
$req = cubrid_execute($conn, "select class_name, owner_name from db_class");

if ($req) {
        while ($rows = cubrid_fetch($req)) {
                $class_name = $row[0];
                $owner_name = $row[owner_name];
        }
        cubrid_close_request($req);
}
```

PHP 사용하기 (계속)

- CUBRID 함수
  - 트랜잭션 반영
    - cubrid_commit
  - 트랜잭션 철회
    - cubrid_rollback

```
$req = cubrid_execute($conn, "delete from my_table where id = 1");

cubrid_commit($conn);
```

감 사 합 니 다

CUBRID