LINUX KOREA
LINUX IS THE INTERNET

MySQL Reference

1999

리눅스코리아(주)

목 차

I. 데이터베이스와 데이터베이스 언어

'데이터베이스는 필요에 의해서 저장된 데이터의 집합이다'라고 말한다. 즉 정보라는 자원을 사용자의 다양한 요구에 맞춰서 정리되어 합쳐져 있는 데이터의 집합을 말한다. 모아진 집합이기 때문에 데이터베이스를 사용하기 위해서는 소프트웨어가 필요하다. 이러한 프로그램을 데이터베이스 관리프로그램(DBMS)라고 부른다. 즉 저장된 데이터에 대한 사용자의 처리를 담당하는 프로그램을 말하는 것이다. 사용자는 DBMS에 자신이 원하는 작용을 하도록 명령을 내려주게 되면 그 외 상응하는 작용을 하게 되는 것이다. 이때 사용자가 DBMS에 내려주는 명령을 데이터베이스 언어라고 하고 보통은 프로그래밍 언어의 형태를 띄고 있는데 필요한 규칙에 따라 형식화된 명령이다.
DBMS의 중요한 특징은 다음 두가지로 나눌 수 있다.

- 독립성
- 무결성(일관성)

독립성이라는 말은 사용자는 DBMS가 데이타를 어디에 어떠한 형태로 저장을 하는지에 대해서 알 필요가 없다는 말이다. 무결성은 실생활의 규칙이 데이터로 구성되는 경우이므로 이러한 실생활의 규칙에 맞아야 한다는 것이고 두 개의 서로 다른 부분에 있는 데이터베이스 데이터가 서로 모순되지 않아야 한다는 것을 의미한다.

1 SQL

SQL 은 형식적이고 수학적인 모델을 근거로 하고 있다. 개념의 정의의 집합으로 구성된 이러한 이론을 관계형 모델이라고 하며 이 관계형 모델은 codd가 논문으로 발표한 것으로 이러한 관계형 모델은 데이터베이스 언어를 위한 이론적 기초를 구성하였다. 이 모델은 데이터베이스에 데이터를 기록하도록 하는 작고 간단한 많은 개념들로 구성되었고, 정보를 조작하기 위한 다양한 연산자도 가지고 있었다. 관계형 모델은 다양한 데이터베이스 언어를 개발하기 위한 예제로서 공헌하게 되었다. 이러한 데이터베이스 언어는 관계형 모델의 이론과 개념에 근거를 두고 있다. 따라서 이러한 언어를 관계형 데이터베이스 언어라 하는데, SQL은 이러한 언어중 하나이다. SQL(Structured Query Language)은 1960년대 후반에 IBM에서 개발한 것으로 현재는 데이터 베이스 언어의 표준으로 자리잡았다. SQL은 데이터베이스 정의, 데이터베이스 조작, 트랜잭션 처리라는 주요한 세가지 기능을 제공한다.
SQL의 문법은 ANSI의 표준으로 잡혀 있지만 각 데이터 베이스 벤더마다 조금씩의 차이점이 있다. 대부분의 SQL은 다음과 같은 작업을 할 수 있는 내용을 가지고 있다.

- 데이터베이스 정의
- 데이터베이스 조작
- 트랜잭션 처리

다음은 몇가지 SQL 문의 형식이다.

- CREATE DATABASE 데이터베이스이름

데이터베이스 이름은 문자와 숫자, '_' 를 포함하는32 byte 이내로 작성한다. 이 명령은 새로운 데이터베이

스 공간을 생성시킨다.

▪ DROP DATABASE 데이터베이스이름

존재하는 데이터베이스를 제거한다. 데이터베이스상에 존재하는 테이블도 모두 삭제되므로 매우 조심하여 작업하여야 한다. 한번 삭제되면 복구는 불가능하므로 데이터베이스 상의 테이블과 데이터가 모두 백업이 되었는지 확인하고 작업한다.

▪ CREATE TABLE 테이블이름 (정의형식, ...)

데이터베이스상에 테이블을 생성시키는 명령이다. 자세한 옵션은 다음과 같다.

정의형식:

컬럼이름 자료형 [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT] [PRIMARY KEY] [reference_definition]

or PRIMARY KEY (index_column_name,...)

or KEY [index_name] KEY(index_column_name,...)

or INDEX [index_name] (index_column_name,...)

or UNIQUE [index_name] (index_column_name,...)

or FOREIGN KEY index_name (index_column_name,...) [reference_definition]

or CHECK (expr)

자료형:

TINYINT[(length)] [UNSIGNED] [ZEROFILL]

or SMALLINT[(length)] [UNSIGNED] [ZEROFILL]

or MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]

or INT[(length)] [UNSIGNED] [ZEROFILL]

or INTEGER[(length)] [UNSIGNED] [ZEROFILL]

or BIGINT[(length)] [UNSIGNED] [ZEROFILL]

or REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]

or DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]

or FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]

or DECIMAL[(length,decimals)] [UNSIGNED] [ZEROFILL]

or NUMERIC[(length,decimals)] [UNSIGNED] [ZEROFILL]

or CHAR(length) [BINARY],

or VARCHAR(length) [BINARY],

or DATE

or TIME

or TIMESTAMP

or DATETIME

or TINYBLOB

or BLOB

or MEDIUMBLOB

or LONGBLOB

or TINYTEXT

or TEXT

or MEDIUMTEXT

or ENUM(value1,value2,value3,...)

or SET(value1,value2,value3,...)

인텍스컬럼이름:

컬럼이름 [(length)]

참조정의:

REFERENCES 테이블이름 [(인텍스컬럼이름, ...)]

[MATCH FULL | MATCH PARTIAL]

[ON DELETE 참조옵션]

[ON UPDATE 참조옵션]

참조옵션:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

▪ ALTER [IGNORE] TABLE 테이블이름 변경방법 [, 변경방법 ...]

존재하는 테이블을 변경 및 컬럼의 추가 작업시 사용된다.

변경방법:

ADD [COLUMN] 정의형식

or CHANGE [COLUMN] 이전컬럼이름 정의형식

or ALTER [COLUMN] 컬럼이름 { SET DEFAULT literal | DROP DEFAULT }

or ADD INDEX [인덱스이름] (인덱스컬럼이름, ...)

or ADD UNIQUE [인덱스이름] (인덱스컬럼이름, ...)

or DROP [COLUMN] 컬럼이름

or DROP PRIMARY KEY

or DROP INDEX 키이름

or RENAME AS 새로운테이블이름

▪ DROP TABLE 테이블이름 [, 테이블이름...]

존재하는 테이블을 제거할 경우 사용되는 명령이다.

▪ DELETE FROM 테이블이름 WHERE 조건

존재하는 테이블 상의 데이터를 삭제할 때 사용하는 명령이다.

▪ SELECT 문

존재하는 테이블 상의 데이터를 조회할 때 사용하는 명령이다. 기본적인 사용법은 다음과 같다.

```
SELECT [STRAIGHT_JOIN] [DISTINCT | ALL] 검색방법,...
    [INTO OUTFILE 'file_name' ...]
[ FROM 테이블참조
  [WHERE 조건 ]
  [GROUP BY 컬럼,...]
```

[HAVING 조건]

[ORDER BY 컬럼 [ASC | DESC] ,...] [LIMIT [offset,] rows]

[PROCEDURE 프로시저이름]]

▪ JOIN 문

테이블참조, 테이블참조

테이블참조 [CROSS] JOIN 테이블참조

테이블참조 LEFT [OUTER] JOIN 테이블참조 ON 조건표현식

테이블참조 LEFT [OUTER] JOIN 테이블참조 USING (컬럼)

테이블참조 NATURAL LEFT [OUTER] JOIN 테이블참조

{ 테이블참조 LEFT OUTER JOIN 테이블참조 ON 조건표현식 }

▪ INSERT 문

존재하는 테이블에 데이터를 입력할 때 사용한다.

INSERT INTO 테이블 [(컬럼이름,...)] VALUES (표현,...)

or INSERT INTO 테이블 [(컬럼이름,...)] SELECT ...

▪ REPLACE 문

이 문장은 INSERT문과 거의 유사하게 동작한다. 같은 레코드가 있을 경우 이 레코드를 삭제한 후 INSERT 작업을 한다. 같은 레코드가 없을 경우는 INSERT와 같이 동작한다.

REPLACE INTO 테이블 [(컬럼이름,...)] VALUES (표현,...)

or REPLACE INTO 테이블 [(컬럼이름,...)] SELECT ...

▪ LOAD DATA INFILE 문

Server 상에 위치하는 text 파일로 부터 데이타를 읽어서 테이블에 입력작업을 할 경우 사용한다. 매우 속도가 빠르다.

```
LOAD DATA [NFILE 'text_file_name.text' [REPLACE | [GNORE]
    [NTO TABLE 테이블이름
[FIELDS [TERMINATED BY ',' [OPTIONALLY] ENCLOSED BY '"' ESCAPED BY '\\' ]]
[LINES TERMINATED BY '\n']
[(Field1, Field2,...)]
```

테이블로 부터 데이터를 읽어서 text 파일에 데이터를 저장할 경우 다음과 같이 사용한다.

```
SELECT ...
 [NTO OUTFILE 'interval.txt' fields terminated by ','
 enclosed by '"'
 escaped by '\\' lines terminated by '\n'
FROM ...
```

▪ UPDATE 문

테이블 상에 존재하는 데이터를 변경할 경우 사용한다.

```
UPDATE 테이블 SET 컬럼=expression,... WHERE where_definition
```

▪ SHOW

MySQL 상의 각종 정보를 보여준다. 데이터베이스, 테이블, 컬럼등을 확인할 수 있다.

```
        SHOW DATABASES [LIKE wild]
or      SHOW TABLES [FROM 데이터베이스] [LIKE wild]
or      SHOW COLUMNS FROM 테이블 [FROM 데이터베이스] [LIKE wild]
or      SHOW [NDEX FROM 테이블 [FROM 데이터베이스]
or      SHOW STATUS
or      SHOW VARIABLES [LIKE wild]
```

▪ EXPLAIN

SELECT 문과 이때 요구되는 테이블에 대한 정보를 준다. 일반 SELECT 문의 처음에 EXPLAIN 을 추가하면 동작한다.

EXPLAIN SELECT select_options

▪ DESCRIBE

존재하는 테이블에서 컬럼 정보를 가지고 온다.

(DESCRIBE | DESC) 테이블 [컬럼]

▪ LOCK TABLES

테이블에 lock 을 설정하여 타인이 읽지 못하게 할 때 사용한다. 주의할 점은 한사람이 lock 을 실행하면 이 사람이 사용하는 모든 테이블이 lock되므로 사용 후 꼭 unlock를 사용하여 lock를 풀도록 한다.

LOCK TABLES 테이블이름 [AS alias] READ|WRITE [, 테이블이름 READ|WRITE]

...

UNLOCK TABLES

▪ SET OPTION

현재 사용중인 세션이 사용되는 동안 지속적으로 영향을 준다.

SET [OPTION] SQL_VALUE_OPTION=value, ...

▪ GRANT 문

MySQL에서의 특권은 MySQL 허가 테이블을 이용하여 다루어 진다.

GRANT (ALL PRIVILEGES | (SELECT, INSERT, UPDATE, DELETE,

REFERENCES (컬럼리스트), USAGE))

ON 테이블 TO 사용자, ... [WITH GRANT OPTION]

▪ CREATE INDEX 문

ALTER TABLE 을 사용하여 새로운 index 를 생성할 수 있다.

CREATE [UNIQUE] INDEX 인덱스이름 ON 테이블이름 (컬럼이름,...)

▪ DROP INDEX 문 (호환명령)

ALTER TABLE 을 사용하여 index 를 제거할 수 있다.

DROP INDEX 인덱스이름

▪ 주석

한줄의 주석은 # 로 하고 주석의 내용이 여러줄일 경우는 /* */ 를 이용하여 주석을 처리할수 있다.

▪ CREATE FUNCTION

MySQL 의 기본 함수인 ABS(), constr() 처럼 새로운 사용자 정의 함수를 생성하는 함수이다. 사용자정의 함수의 소스는 C, C++로 작성되어야 하고 동적으로 읽혀지는 것이 요구되어 진다. 예제로서 배포판에 보면 sql/udf_example.cc 가 있고 5개의 새로운 사용자정의함수가 작성되어 있다.

CREATE FUNCTION <함수이름> RETURNS [string|real|integer]

SONAME <공유라이브러리이름>

DROP FUNCTION <함수이름>

▪ FUNCTIONS

집단함수	COUNT, SUM, AVG, MAX, MIN, VARIANCE, STDDEV
날짜와 시간 함수	ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, NEW_TIME, NEXT_DAY, SYSDATE
수학함수	ABS, CEIL, FLOOR, COS, COSH, SIN, SINH, TAN, TANH, EXP, LN, LOG, MOD, POWER, SIGN, SQRT
문자함수	CHR, CONCAT, INITCAP, LOWER, UPPER, LPAD, RPAD, LTRIM, RTRIM, REPLACE, SUBSTRING, TRANSLATE, INSTR, LENGTH
변환함수	TO_CHAR, TO_NUMBER,

상용이든 쉐어웨어든 데이터베이스가 그 자신의 힘을 100% 발휘하기 위해서는 개발자의 주의 깊은 테이블 설계와 효율적인 인덱싱 정책이 필요하다. 잘못된 인덱싱 정책은 오히려 데이터 베이스의 동작 속도를 감소시킬 수도 있고 효율적인 인덱싱만으로도 60%이상의 액세스 효율성을 향상시킬 수도 있다. 이러한 데이터베이스 관리 정책은 많은 시행착오에서 오는 경험에 의한 노하우인 경우가 많으므로 개발자의 입장에서는 많은 노력과 시간을 투자해야만이 원하는 효율을 얻을 수 있을 것이다.

II MySQL

1. MySQL은 무엇인가

MySQL은 관계형 데이터베이스이다. 이것은 공개용 데이터베이스로서 일반 상용 데이터베이스와 비교하여 뒤질것이 없는 매우 뛰어난 관계형 데이터베이스이다. 물론 안정성 및 무결성의 측면에서 다른 상용 데이터베이스에 비하여 약간은 떨어지지만 매우 안정적이라고 할 수 있다. 관리만 잘 이루어진다면 다른 공개 데이터베이스에 대해서 보안이나, 각종 함수도 많아서 프로그램에 용이하다. 공개용 데이터베이스로는 mSQL, PostgreSQL, MySQL 등이 있는데 그 중 전세계적으로 볼때 MySQL이 많이 애용되고 있다고 할 수 있다. PHP와도 연결이 용이하고 각종 공개용 웹서버와 연결도 간편하다. C, C++, Java 그리고 ODBC 도 제공이 된다. ODBC는 windows3.1, win95, NT용 모두 제공이 되고 있다. Access와 자료를 연동할 수 있으며 Excel 은 물론 Delphi(델파이)와는 DBE 3.2를 사용하여 데이터연동이 가능하며 C++ Builder 와는 DBE 3.0을 통하여 사용 가능하다. 이 MySQL을 상업적으로 사용할 경우 DB 개발자들에게 조금의 개발 후원금을 제공하면 제품에 포함하여 개발된 프로그램과 함께 판매도 할 수 있으며 개인적인 용도로만 사용한다면 무료로 사용할 수 있다.

2. MySQL 설치 가능 OS

MySQL의 설치는 매우 간단하다. 공식 사이트로 가서 소스를 가지고 온 후 컴파일을 하여 설치하여도 되고, 또는 컴파일된 실행 파일을 가지고 와서 실행하여도 무방하다. 현재 MySQL은 각종 OS에서 설치되고 실행될 수 있다. 실행 가능한 OS와 패키지는 다음과 같다.

AIX 4.x with native threads
BSDI 2.x with the included MIT-pthreads package
BSDI 3.0, 3.1 and 4.x with native threads
DEC UNIX 4.x with native threads
FreeBSD 2.x with the included MIT-pthreads package
FreeBSD 3.x with native threads
HP-UX 10.20 with the included MIT-pthreads package
HP-UX 11.x with the native threads.
Linux 2.0+ with LinuxThreads 0.7.1 or glibc 2.0.7
NetBSD 1.3 Intel and NetBSD 1.3 Alpha
OpenBSD 2.x with the included MIT-pthreads package
OS/2 Warp 3, FixPack 29 and OS/2 Warp 4, FixPack 4
SGI Irix 6.x with native threads
Solaris 2.5, 2.6 and 2.7 with native threads on SPARC and x86
SunOS 4.x with the included MIT-pthreads package
SCO OpenServer with a recent port of the FSU Pthreads package
SCO UnixWare 7.0.1
Tru64 Unix
Win95, Win98 and NT

3. MySQL 을 설치할 경우 필요한 것

MySQL 을 설치할 경우 필요한 것이 있다. 그것은 다음과 같다. 만약 RPM 패키지를 설치하는 경우에는 rpm -Uvh MySQL-버전 하면 설치가 되기 때문에 여기서는 소스로 설치하는 경우에 필요한 사항들이다.

압축툴 : GNU gunzip
GNU tar
컴파일러 : ASI C++ compiler
gcc 2.8.1 이상, egcs 1.0.2 이상
gcc 2.7.x 버전은 버그가 있으므로 MySQL을 설치하기 위해서는 꼭 2.8.x을 사용.
Make 툴 : GNU make 3.75 이상

6. 설치 기본 사항

바이너리 배포본은 자신이 원하는 장소에 설치할 수 있지만 해당 장소에 다음과 같은 디렉토리가 생긴다. (기본장소는 보통 /usr/local/mysql 이다)

'bin'	- 클라이언트 프로그램, mysqld 서버
'data'	- 로그파일, 데이터베이스
'scripts'	- mysql_install_db
'share'	- 에러메세지 파일
'sql-bench'	- 벤치마크

소스배포본은 설치하고 서정하고 컴파일 해야 하는데 기본적으로 /usr/local 에 설치되는데 역시 다음과 같은 디렉토리들이 생긴다.

'bin'	- 클라이언트 프로그램과 스크립트
'include/mysql'	- 인클루드 파일
'libexec'	- mysqld 서버
'info'	- .info 형식의 문서파일
'share/mysql'	- 에러메시지 파일
'sql-bench'	- 벤치마크와 crash-me
'var'	- 데이터베이스와 로그파일

소스로 설치되는 경우에는 바이너리에 비해서 설치되는 장소가 조금은 차이가 있다.

- mysqld 서버는 '/usr/local/mysql/bin' 대신에 '/usr/local/libexec' 디렉토리에 설치된다.
- 데이터 디렉토리는 '/usr/local/mysql/data' 대신에 '/usr/local/var' 디렉토리로 들어간다.
- mysql_install_db는 '/usr/local/mysql/scripts' 대신에 '/usr/local/bin' 디렉토리에 설치된다.

7. 시스템 관련사항

▪ MIT-pthread 는 AF_UNIX 프로토콜을 지원하지 않는다. 그래서 만약 MIT-pthread를 사용하는 경우에는 TCP/IP를 통한 접속만을 할 수 있기 때문에 약간 속도가 느리다. 만약 MySQL의 바이너리를 만들고 난 후에 로컬 서버로 접속할 수 없다면 TCP/IP를 통한 접속을 해본다.
▪ 소스로 설치하거나 바이너리로 설치해도 항상 libc나 glibc 버전에 주의해야 한다. 실행시에 나오는 에러메시지중의 거의 대부분은 해당 라이브러리 버전이 맞지 않아서 그런 경우이다. 그리고 이러한 단점을 피하려면 모두 정적으로 링크해서 바이너리를 생성하면 약간은 에러상황을 피해갈 수 있을 것이다.

4. 설치

■ MySQL 의 설치방법

다음과 같은 순서로 MySQL을 설치한다.

1. 공식 홈페이지에서 MySQL을 받는다. 자신에게 필요한 패키지를 받는데 여기서는 소스를 받아서 원하는 처리를 해야 하므로 가장 최신버전의 소스를 받아야 한다. 만약 바이너리가 필요한 경우에는 자신에게 설치된 배포본에 따라서 필요한 패키지를 받으면 된다.

홈페이지 : http://www.mysql.com/download.html

2. root 아이디로 다음의 작업들을 해야 한다.

3. 다음의 명령으로 내용을 푼다.

```
tar xvzf mysql-VERSION.tar.gz
```

여기서 mysql-VERSION.tar.gz 은 방금 받은 파일의 이름이다.

4. 생성된 디렉토리로 이동한다.

```
cd mysql-VERSION
```

5. 환경을 설정한다. 만일 어떤 옵션을 추가하기를 원한다면 다음과 같아 한다.

```
./configure --help
```

이 경우 각종 옵션이 나오므로 이 내용을 보고 변경하기 바란다. 더 자세한 사항은 영문 메뉴얼을 읽어보기 바란다.
대부분의 경우는 환경설정 없이 그냥 해도 문제가 없지만 한글을 사용하려고 한다거나 또는 그외 필요한 장소로 설치를 유도하거나 하는 경우에는 도움말를 잘 읽고 그대로 해야 한다. 다음에는 설정에 관련된 다양한 예이다.

- 가장 일반적인 경우

```
./configure
```

- 한글을 사용하는 경우

```
./configure --with-charset=euc_kr
```

※ 참고사항

■ 다국어사용

한글뿐만이 아니라 다국어를 지원하기 때문에 원하는 지원코드를 입력하면 된다. 지원하는 코드는 다음과 같다.

big5' 'cp1251' 'cp1257' 'czech' 'danish' 'dec8' 'dos' 'euc_kr' 'german1' 'hebrew' 'hp8' 'hungarian' 'koi8_ru' 'koi8_ukr' 'latin1' 'latin2' 'sjis' 'swe7' 'tis620' 'ujis' 'usa7' 'win1251' 'win1251ukr'

그리고 서버와 클라이언트 사이의 문자를 변경하려면 다음을 사용한다.

SET OPTION CHARACTER SET

- 클라이언트 프로그램만으로 사용할 때 (서버 기능없이 클라이언트로만 사용한다)

./configure --without-server

- 기본 설치디렉토리를 '/usr/local/mysql'로 변경할 때

./configure --prefix=/usr/local/mysql

- 기본설치디렉토리를 '/usr/local' 로 하고,
  데이터베이스 디렉토리를 '/usr/local/mysql/data' 로 변경

./configure --prefix=/usr/local --localstatedir=/usr/local/mysql/data

- 소켓디렉토리를 /path/to/socket/dir 로 변경할 때 (기본값은 '/tmp' 또는 '/var/run')

./configure --with-unix-socket-path=/path/to/socket/dir

- 정적으로 프로그램을 컴파일 하는 경우

./configure --with-client-ldflags=-all-static --with-mysqld-ldflags=-all-static

- libg++이나 libstdc++이 없이 gcc만 사용할 때 (미리 환경변수로 설정해 놓는다.)

CC=gcc
CXX=gcc
./configure

- 'DEFAULT' 필드를 사용하지 않도록 하려면

CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS
./configure

▪ 디버깅코드 옵션을 주려면

./configure --with-debug

▪강제로 MIT_pthreads를 실행할 때

./configure --with-mit-threads

6. 만약 환경설정에 문제가 있다면

rm config.cache
make clean

또는

make distclean

7. 5항의 실행이 끝나면 다음의 명령을 실행한다.

make

컴파일시 'sql_yacc.cc' 부분에서 다음과 같은 에러가 난다면 다음과 같이 환경을 잡아주고 다시 해본다.

에러)

Internal compliter error: program cc1plus got fatal signal 11
또는 Out of virtual memory
또는 Virtual memory exhausted

원인)

'gcc' 가 매우 많은 양의 메모리를 요구하기 때문이다.

해결)

./configure --with-low-memory

8. 7항의 작업이 끝나면 이제 컴파일된 실행파일을 인스톨하여야 한다.

shell> make install

컴파일된 모든 실행 가능한 파일이 설치가 되고 환경이 설정된다.

9. 설치가 끝난후 아래의 명령을 실행하여 기초 테이블과 admin 유저를 설치한다. 이 명령은 단 한번만

하는 것으로서 두 번 실행하면 않된다. 현재 8항에서 작업한 후 디렉토리를 변경하지 않고 아래의 명령을 입력한다.

```
shell> ./scripts/mysql_install_db
```

이것은 스크립트로서 실행하면 기초테이블 6개를 설치한다. ('user', 'db', 'host', 'tables_priv', 'columns_priv', 'func')

10. 설치 디렉토리를 디폴트로 하였다면 /usr/local/bin/ 디렉토리에 실행파일이 모두 설치되었을 것이다. 디폴트로 설치하였다고 가정하고 다음의 명령을 수행하여 본다.

```
shell> bin/safe_mysqld &
```

11. 에러없이 실행되면 아래의 명령으로 설치된 프로그램을 확인해 본다.

```
shell> mysqladmin version
```

```
mysqladmin  Ver 6.3 Distrib 3.22.22, for pc-linux-gnu on i686
TCX Datakonsult AB, by Monty

Server version          3.22.22-1
Protocol version        10
Connection              localhost via TCP/IP
TCP port                3306
UNIX socket             /tmp/mysql.sock
Uptime:                 16sec

Running threads:1  Questions: 20  Reloads: 2  Open tables: 3
```

다음과 같은 명령도 내려보면 현재의 상태를 볼 수 있을 것이다.

```
shell> mysqladmin variables
```

12. MySQL 서버를 중지시키려면 다음과 같이 한다.

```
shell> mysqladmin -u root shutdown
```

13. 간단한 테스트를 한다. 아래와 같은 명령을 해본다.

```
shell> mysqlshow
```

Databases
mysql
test

shell> mysqlshow mysql

Tables
columns_priv
db
func
host
tables_priv
user

shell> mysql -e "select host,db,user from db" mysql

host	db	user
%	test	
%	test_%	

14. 벤치마크 테스트를 해본다.

벤치마크에 관심이 있는 사람은 다음과 같은 명령어로 테스트를 해보자.

shell> run-all-test

프로그램을 실행하고 나면 상당한 시간동안 테스트가 이루어지고 결과는 output 디렉토리에 저장된다. 결과를 확인해서 자신의 시스템이 얼마만큼의 성능을 가지는지 확인해보는 것도 좋을듯하다.

테스트를 위해서라면 다음의 프로그램도 사용될 수 있다.

shell> crash-me

이 프로그램은 테스트라기보다는 시스템을 최대한 사용하여 제한점까지 확인을 하는 것이다.

위의 프로그램전에 반드시 Perl 모듈들이 미리 설치되어 있어야 한다. 만약 하지 않았다면 다음의 순서에 따라서 설치를 한다.

15. Perl 모듈의 설치

우선 다음의 프로그램들이 필요하다. 역시 MySQL의 다운로드 장소에서 받아오면 된다.

Data-Dumper

DBI 1.06
Msql-Mysql-modules 1.2014

Perl 버전이 5.004 버전이후이어야 한다. Perl 버전을 확인하는 방법은 다음의 명령을 사용한다.

perl -V 또는
perl --version
rpm -qi perl (RPM 패키지를 설치한 경우)

그런후에 다음의 프로그램들을 적당한 디렉토리에 풀어서 설치를 한다. 설치순서는 'Data-Dumper', 'DBI', 'Msql-Mysql-modules' 순으로 해야 한다. 설치시에는 다음과 같이 한다.
Data-Dumper를 설치하는 경우 해당 디렉토리에서 다음과 같이 실행한다.

perl Makefile.PL
make
make test
make install

이러한 순서대로 설치하면 적당한 디렉토리(레드햇 시스템이라면 /usr/lib/perl)에 설치된다. ---- (?) 확인해보자

'DBI' 와 'Msql-Mysql-modules' 도 이와같은 방식으로 설치하자.

16. 자동으로 MySQL 서버 시작과 종료

시작)
shell> mysql.server start

종료)
shell> mysql.server stop

위와 같이 해주면 시작과 종료가 이루어지는데 부팅시 자동으로 시작되게 하려면 레드햇 시스템이라면 /etc/rc.d/rc.local 에 다음의 내용을 넣어준다. 이때 경로는 자신이 설치한 디렉토리를 넣어주어야 한다.

/bin/sh -c 'cd /usr/local/mysql ; /bin/safe_mysqld &'

이렇게 하면 시작시에 자동으로 실행된다.

레드햇 시스템에서 init에 mysql.server를 넣어두고 각 런레벨에서 이것을 링크로 하여 실행시키게 만들 수도 있다.

프로그램이 실행되면 다음 두 개의 프로세스가 실행된다.

/usr/local/bin/safe_mysqld

/usr/local/libexec/mysqld

5. 시작

■ 옵션파일

MySQL이 시작시에 필요한 옵션을 주어서 실행을 시킬 수 있지만 매번 그렇게 하기가 불편하기 때문에 옵션을 가지고 있는 파일이 있다. mysql.server 에 옵션을 주기 위해서는 /etc/my.cnf 파일에 필요한 사항을 넣어주면 된다. 다음은 간단한 예이다.

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/tmp/mysqld.sock
port=3306

[mysql.server]
user=mysql
basedir=/usr/local/mysql
```

옵션파일이 다음의 위치에 있을 수 있다.

```
'/etc/my.cnf'          - 시스템전체 옵션
'DATADIR/my.cnf'       - 서버지정 옵션
'~/.my.cnf'            - 개인지정 옵션
```

DATADIR 은 MySQL의 데이터 디렉토리(일반적으로는 '/usr/local/mysql/data' 또는 '/usr/local/var' 에 위치하고 있다)이다.

옵션파일의 지정법은 다음과 같다.

```
[group]                 - 옵션을 지정하길 원하는 프로그램이나 그룹의 이름
옵션                    - 옵션을 지정한다. 명령행에서 --option 과 같다.
옵션=값                 - 옵션을 지정한다. 명령행에서 --option=값 과 같다.
set-variable=variable=값  - mysqld 변수에 사용되어지는 값을 지정하는데 명령행에서
                            --set-variable variable=값 과 같다.
```

client 그룹은 mysqld 프로그램이 아니라 모든 MySQL 클라이언트 프로그램에 적용되는 것이다. 이것은 주로 서버에 접속하기 위한 패스워드를 지정할 때 사용할 수 있다.

다음은 옵션파일의 전형적인 예이다.

```
[client]
port=3306
socket=/tmp/mysql.sock
```

```
[mysqld]
port=3306
socket=/tmp/mysql.sock
set-variable=key_buffer=16M
set-variable=max_allowed_packet=1M

[mysqldump]
quick
```

이러한 옵션파일은 소스로 설치한 경우라면 설치디렉토리의 'support-files' 디렉토리의 'my-example.cnf' 라는 파일을 복사하여 사용하면 된다. 만약 바이너리 배포본을 선택한 경우라면 'share/mysql' 밑에 있다.

6. MySQL 의 컬럼 형식

MySQL은 다양한 컬럼형식을 지원하지만 기본적으로 세가지로 분류할 수 있다. 이 가지는 수치형, 날짜와시간형, 문자열(문자) 형이다.

다음은 각 형을 나열해 놓은 것이다.

수치형)

TINYINT[M]	[UNSIGNED] [ZEROFILL]	-128에서 127까지의 정수형. unsigned 라면 0에서 255이다.
SMALLINT[M]	[UNSIGNED] [ZEROFILL]	-32768에서 32767까지의 정수형. unsigned 라면 0에서 65535이다.
MEDIUMINT[M]	[UNSIGNED] [ZEROFILL]	-8388608에서 8388607까지의 정수형. unsigned 라면 0에서 16777215이다.
INT[M]	[UNSIGNED] [ZEROFILL]	-2147483648에서 2147483647까지의 정수형. unsigned 형이라면 0에서 4294967295이다.
INTEGER[M]	[UNSIGNED] [ZEROFILL]	INT와 같다.
BIGINT[M]	[UNSIGNED] [ZEROFILL]	-9223372036854775808에서 9223372036854775807까지의 정수형. unsigned 형이면 0에서 18446744073709551615까지이다. 이 값을 가지고 산술연산을 수행할때는 주의가 필요하다. 63bit 이상 넘어가기 때문에 만약 이 형의 값 두 개를 곱하게 되는 연산을 하는 경우에는 63bit를 초과하여 원하지 않는 값이 나오게 된다.
FLOAT(정밀도)	[ZEROFILL]	정밀도는 4나 8이다. FLOAT(4)는 단정도이고 FLOAT(8)은 배정도이다.
FLOAT[M,D]	[ZEROFILL]	-3,402823466E+38 ~ -1,175494351E-38 0 -1,175494351E-38 ~ 3,402823466E+38
DOUBLE[M,D]	[ZEROFILL]	-1,7976931348623157E+308 ~ -2,2250738585072014E+308 0 2,2250738585072014E-308 ~ 1,7976931348623157E+308
DOUBLE PRECISION[M,D]	[ZEROFILL]	
REAL[M,D]	[ZEROFILL]	DOUBLE와 같다.
DECIMAL(M,D)	[ZEROFILL]	unpacked 부동소수점수
NUMERIC(M,D)	[ZEROFILL]	DECIMAL과 같다.

날짜형)

DATE	날짜형. '1000-01-01'에서 '9999-12-31' 까지를 표현한다. MySQL은 DATE값은 'YYYY-MM-DD' 형식으로 보여주지만 입력시에는 다른 형태로 입력해도 된다.
DATETIME	날짜와시간조합형. '1000-01-01 00:00:00'에서 '9999-12-31 23:59:59'까지의 범위를 가진다. MySQL은 DATETIME형은 'YYYY-MM-DD HH:MM:SS'형식으로 표현한다. 그러나 역시 다른 형태로 입력을 할 수 있다.
TIMESTAMP[M]	timestamp형이다. 범위는 '1970-01-01 00:00:00'에서 2106년까지이다. MySQL은 TIMESTAMP 값은 YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, YYMMDD 형태로 표시한다. 이것은 M 값이 14(또는 없는 경우), 12, 8, 6 일경우에 맞추어서 이루어진다. 이 값은 INSERT나 UPDATE 할 때 유용한데 이경우에 자동적으로 이전 작업시간으로 설정된다. 그래서 필요하다면 NULL 값을 주어서 현재 시간으로 설정할수도 있다.
TIME	시간형. '-838:59:59'에서 '838:59:59' 까지의 범위를 가진다. TIME 형은 'HH:MM:SS' 형태로 보여진다.
YEAR	해를 나타낸다. 1901에서 2155그리고 0000 값을 가진다. YEAR형는 YYYY 형태로 보여진다.

문자열형)

CHAR(M)	[BINARY]	고정길이 문자열로서 오른쪽은 항상 공백으로 채워진다. M 값은 1에서 255까지이다. 뒤의 공백은 자료가 추출될때는 제거된다. 만약 BINARY 키워드가 주워지지 않았다면 기본적으로 CHAR 형은 대소문자를 구별하지 않고 정렬이나 비교를 한다.
VARCHAR(M)	[BINARY]	가변길이 문자열이다. M값은 1에서 255까지이다.
TINYBLOB TINYTEXT		최대길이가 255 (2^8 -1)개인 문자형이다.
BLOB TEXT		최대길이가 65535(2^16-1)개인 문자형이다.
MEDIUMBLOB MEDIUMTEXT		최대길이가 16777215(2^24 - 1)개인 문자형이다.
LONGBLOB LONGTEXT		최대길이가 4294967295(2^24 - 1)개인 문자형이다.
ENUM('값1','값2',...)		열거형이다. 문자열 객체는 오직 한 개의 값을 가지지만 값들중에서 선택된다. ENUM은 최대 65535개의 서로다른 값을 가진다.
SET('값1', '값2',...)		설정형. 문자열 객체는 0 이나 그외의 값들중에서 선택할 수 있다. 최대 64개의 멤버를 가질 수 있다.

■ 컬럼타입의 크기

수치형)

컬럼 타입	크기
TINYINT	1 바이트
SMALLINT	2 바이트
MEDIUMINT	3 바이트
INT	4 바이트
INTEGER	4 바이트
BIGINT	8 바이트
FLOAT(4)	4 바이트
FLOAT(8)	8 바이트
FLOAT	4 바이트
DOUBLE	8 바이트
DOUBLE PRECISION	8 바이트
REAL	8 바이트
DECIMAL(M, D)	M 바이트 (만약 M이 D보다 작으면 D+2)
NUMERIC(M, D)	M 바이트 (만약 M이 D보다 작으면 D+2)

날짜형)

컬럼타입	크기
DATETIME	8 바이트
DATE	3 바이트
TIMESTAMP	4 바이트
TIME	3 바이트
YEAR	1 바이트

문자열형)

컬럼타입	크기
CHAR(M)	M 바이트, 1 <= M <= 255
VARCHAR(M)	L <= M 고 1 <= M <= 255 일 때 L+1 바이트,
TINYBLOB,TINYTEXT	L < 2^8 일때 L+1 바이트
BLOB,TEXT	L < 2^16 일때 L+2 바이트
MEDIUMBLOB,MEDIUMTEXT	L < 2^24 일 때 L+3 바이트
LONGBLOB,LONGTEXT	L < 2^32 일 때 L+4 바이트
ENUM('value1', 'value2', ...)	1또는 2 바이트. 열거갯수에 따라 다르다.
SET('value1', 'value2', ...)	1, 2, 3, 4 또는 8 바이트. 역시 값에 따라 다르다.

7. 호환성

MySQL은 기존의 ANSI SQL92 규약을 따르고 있으며 확장된 부분이 있다. 다음은 확장된 부분이다.

1. 컬럼타입 : MEDIUMINT, SET, ENUM, BLOB, TEXT

2. 필드속성 : AUTO_INCREMENT, BINARY, UNSIGNED, ZEROFILL

3. 모든 문자열 비교는 기본적으로 대 소문자를 구별하지 않으며 현재의 문자셋에 의해 정렬 순서가 정해진다. 이것을 원하지 않으면 컬럼을 BINARY 속성으로 정의해야 하며 이런 경우에는 MySQL 서버 호스트가 사용하는 ACSII 순서에 따라서 문자열을 비교한다.

4. MySQL은 데이터베이스를 디렉토리로 만들고 테이블은 파일이름으로 만든다. 이것은 생각해야 할 두 가지가 있는데 먼저 파일이름의 대소문자를 구별하는 운영체제(대부분의 유닉스, 리눅스)에서는 MySQL 데이터베이스 이름과 테이블 이름은 대소문자를 구별한다. 그래서 소문자로 만드는 것이 좋다. 두 번째는 테이블의 백업, 이름바꾸기, 옮기기, 삭제, 복사를 위해 표준시스템 명령을 사용할 수 있다. 예를 들면 테이블의 이름을 바꾸려면 해당하는 테이블의 '.ISD', '.ISM'과 '.frm' 파일의 이름을 변경하면 된다.

5. SQL 문에서 db_name.tbl_name 문을 이용하여 다른 데이터베이스의 테이블에 접근할 수 있다. 일부 SQL 서버는 같은 기능을 지원하지만 이것을 사용자영역(user space)라고 부른다. MySQL은 다음과 같은 TABLESPACES를 지원하지 않는다

```
create table ralph.my_table.. IN my_tablespaece
```

6. 수치형 컬럼에서 LIKE를 사용할 있다.

7. SELECT 문에서 INTO OUTFILE과 STRAIGHT_JOIN을 사용할 수 있다. SELECT 형식을 참조한다.

8. EXPLAIN SELECT 는 테이블에서 어떻게 조인이 되었는지에 대한 정보를 보여준다.

9. CREATE TABLE 구문에서 INDEX와 KEY를 사용할 수 있다.

10. ALTER TABLE에서 CHANGE col_name, DROP col_name 또는 DROP INDEX를 사용한다.

11. ALTER TABLE에서 IGNORE 사용할 수 있다.

12. ALTER TABLE 문에서 다중 ADD, ALTER, DROP, CHANGE 사용

13. IF EXISTS 키워드를 이용하여 DROP TABLE 사용

14. 한 테이블 이상에서 DROP TABLE 사용

15. 오라클과 호환되는 LOAD DATA INFILE을 사용한다.

16. OPTIMIZE TABLE을 사용한다.

17. 문자열은 ' ' 만이 아니라 ' " ' 를 사용한다.

18. 에스케이프 '\' 문자 사용

19. SET OPTION 문 사용

20. GROUP BY 부분에서 모든 컬럼을 사용할 필요할 필요가 없다. 이러한 기능은 일반적인 질의가 아닌 특정한 질의에서 성능을 향상시킨다.
* ANSI SQL 에서는 여러 테이블을 이용하여 GROUP BY를 사용할 때 사용하고자 하는 모든 컬럼에 GROUP BY를 지정해 주어야 한다. 이렇게 되는 경우 불필요한 연산이 수행될 수 있는데 MySQL에서는 이러한 것을 없앤 것이다.

21. 다른 SQL 환경을 사용하던 사용자를 위해 많은 기능에서 알리어스를 지원한다. 예를 들어 모든 문자 기능은 ANSI SQL과 ODBC구문을 지원한다.

22. 포팅이 가능하도록 SQL CODE에서 STRAIGHT_JOIN같은 MySQL만의 키워드 사용을 지원하기 위해 이런 키워드를 /*! */ 주석안에 내장할 수 있다. 주석문안의 내용은 '!' 로 시작한다. 이런 경우 MySQL 에서는 주석문을 다른 MySQL과 같이 해석하지만 다른 SQL 서버에서는 이러한 확장기능을 사용하지 않고 건너뛸 수 있다. 예는 다음과 같다.

```
SELECT /*! STRAIGHT_JOIN */ * from table1, table2 WHERE ...
```

23. DELETE + INSERT 대신에 REPLACE 사용

8. MySQL 제한사항

1. sub-select

다음은 MySQL에서 작동하지 않는다.

```
SELECT * FROM table1 WHERE id IN (SELECT id FROM table2);
```

MySQL에서는 오직 INSERT .. SELECT .. , REPLACE .. SELECT .. 만을 지원한다. 즉 INSERT나 REPLACE 시에만 사용할 수 있다는 것이다. 그래서 독립적인 sub-select 문은 다음버전에서 지원될 예정이다. 대신에 현재는 IN() 함수를 사용할 수 있다.

2. SELECT INTO TABLE

MySQL 은 아직 SELECT ... INTO TABLE ..을 지원하지 않는다. 현재 MySQL은 오직 SELECT ... INTO OUTFILE ..., 만을 지원한다.

3. 트랜잭션

현재 MySQL에서는 트랜잭션이 지원되지 않는다. 앞으로 지원예정인데 트랜잭션이라기 보다는 최하수준(atomic) 작동을 지원할 것이다. 즉 롤백이 없는 트랜잭션과 비슷하다. 최하수준 작동을 사용하여 insert나 select 모든 명령의 그룹을 실행 할 수 있으며 어떤 쓰레드와도 충돌하지 않게 보장해 줄 것이다. 현재는 LOCK TABLES 와 UNLOCK TABLES 명령을 사용하여 다른 쓰레드와 충돌하지 않도록 한다.

※ COMMIT-ROLLBACK을 어떻게 대치할 수 있을까?
MySQL은 COMMIT-ROLLBACK 을 지원하지 않는다. 문제는 COMMIT-ROLLBACK을 효과적으로 다루기 위해서는 MySQL에서 현재 사용하는 것과 완전히 다른 테이블 설계가 필요하다는 것이다. Mysql은 또한 테이블을 자동 클린업하는 추가적인 스레드와 더 많은 디스크를 사용할 수 있는 기능이 필요하다. 이러한 기능은 현재보다 mysql을 2-4배 느리게 만든다. MySQL은 대부분의 다른 SQL 데이터베이스보다 훨씬 더 빠르다. (전형적으로 최소 2 ~3배 빠름) 이러한 이유는 MySQL에 COMMIT-ROLLBACK이 없기 때문이다.

현재 개발자들은 MySQL 서버의 성능을 높이는데 관심을 가지고 있고 또 주력할 예정이라고 한다. 대부분 COMMIT-ROLLBACK 기능이 정말로 필요한 경우는 드물다. 그리고 추가적으로 COMMIT-ROLLBACK이 없는 편이 더 좋은 성능을 낼 수 있다.

일반적으로 트랜잭션이 필요한 루트는 LOCK TABLES를 사용해 코드를 짤 수 있다. 또한 레코드를 업데이트할 때 커서를 사용할 필요가 없다.

개발자들은 트랜잭션과 커서를 TODO에 넣었지만 우선권이 높은 것은 아니다. 이러한 기능을 수행한다면 CREATE TABLE의 옵션으로 될 것이다. 이것은 옵션으로 지정한 테이블에서만 작동하며 그 테이블은 느리게 될 것이라는 것을 의미한다.

개발자들이 원하는 것은 100% 보편적인 데이터베이스보다는 정말로 빠른 데이터베이스가 필요하다. COMMIT-ROLLBACK 기능을 수행하더라도 속도에 손상이 없다면 개발자들이 지원할 것이라고 한다. 그러나 우선순위가 높은 것은 아니어서 TODO에 있는 순서대로 처리될 것으로 보이며 그다지 빠른 시일내에 이 작업 마무리 될 것 같진 않다.

현재의 문제는 실제로 ROLLBACK에 있다. ROLLBACK없이 LOCK TABLES을 이용하여 여러 종류의 COMMIT를 사용할 수 있다. ROLLBACK을 지원하기 위해 MySQL은 업데이트가 된 모든 예전 레코드를 저장하고 ROLLBACK이 일이났을 때 시작 시점으로 돌아갈 수 있도록 바꾸어야 한다. 이러한 일을 하는 것은 현재 그다지 어렵지 않다. 현재의 isamlog 라는 것이 이런경우를 위해서 사용할 수 있다. 그러나 ALTER/DROP/CREATE TABLE에서 ROLLBACK을 수행하는 것이 무척 어렵다. 그래서 ROLLBACK기능이 추가되기 전에는 이러한 것을 할 수 없다.

ROLLBACK 사용을 피하기 위해 다음의 전략을 사용할 수 있다:

- 접근하기 원하는 모든 테이블에 락을 사용. LOCK TABLES ...
- 조건 테스트(Test conditions)
- 모든 것이 제대로 된다면 업데이트를 한다.
- UNLOCK TABLES

일반적으로 가능한 ROLLBACK을 이용해 트랜잭션을 사용하는 것보다는 이러한 방법이 훨씬 더 빠르다. 그렇지만 항상 사용 가능한 것은 아니다. 이러한 방법으로 해결할 수 없는 유일한 상황은 업데이트 중 누군가가 스레드를 죽였을 때이다. 이런 경우 모든 락은 해제가 된다. 그렇지만 업데이트의 일부는 실행되지 않을 것이다.

물론 단일 오퍼레이션에서 레코드를 업데이트하는 함수를 사용할 수 있다. 다음의 테크닉을 사용하며 매우 효율적인 애플리케이션을 만들 수 있다:

- 현재 값과 관련되어 있는 필드를 수정
- 실제로 변화가 생겼을 때만 필드를 업데이트

예를 들어, 어떤 고객 정보를 업데이트 할 때 오직 바뀐 데이터만 업데이트를 한다. 그리고 변화된 데이터의 테스트는 UPDATE 문에서 WHERE 절을 사용하여 할 수 있다. 레코드가 업데이트되지 않았다면 클라이언트에 다음과 같은 메시지를 준다:

"당신이 바꾼 데이터 일부가 다른 사용자에 의해 바뀌었습니다."

그러고 나서 윈도우에서 예전의 레코드와 현재의 레코드를 비교하여 보여준다. 그러면 사용자는 어떤 고객 정보 레코드를 사용할지 결정할 수 있다.

이렇게 하면 "컬럼 Locking"과 비슷하다. 그렇지만 실제로는 더 빠르다. 왜냐하면 현재의 값과 관련되어 있는 값의 컬럼만 업데이트하기 때문이다. 전형적인 UPDATE은 다음과 비슷할 것이다:

```
UPDATE tablename SET pay_back=pay_back+'relative change';
```

```
UPDATE customer
  SET
    customer_date='current_date',
    address='new address',
    phone='new phone',
    money_he_owes_us=money_he_owes_us+'new_money'
  WHERE
    customer_id=id AND address='old address' AND phone='old phone';
```

지금 보듯이 이렇게 하면 매우 효율적이면서도 설사 다른 클라이언트가 pay_back 이나 money_he_owes_us 컬럼의 값을 바꾸었을 때라도 제대로 작동한다.

대부분의 경우, 사용자는 테이블에서 유일한 값(identifiers)을 관리하기 위해 ROLLBACK과 테이블 락을 사용하고 싶어한다. 이것은 AUTO_INCREMENT 컬럼과 SQL LAST_INSERT_ID() 펑션, 또는 mysql_insert_id() 의 C API 펑션을 사용하여 더욱 효율적으로 사용할 수 있다.

TcX에서는 언제나 이런 문제를 해결할 수 있기 때문에 low-level lock을 필요로 하지 않는데 어떤 경우에는 low-level lock이 필요하다. 그러나 이런 경우는 극소수이다. low-level lock을 원하면 테이블에서 플래그 컬럼을 사용할 수 있다. 다음과 같다:

```
UPDATE tbl_name SET row_flag=1 WHERE id=ID;
```

만약 row가 발견되고 row_flag가 원래의 row에서 이미 1이 아니라면 영향을 받은 row의 숫자로서 1일 반환한다.

4. stored procedure 와 trigger

stored procedure는 서버에서 컴파일되고 저장될 수 있는 SQL 명령이다. 이런 기능이 수행되면 클라이언트는 전체 질의를 다시 할 필요가 없고 stored procedure를 참조할 수 있다. 이런 기능이 있으면 질의는 한번만 해석되고 서버와 클라이언트간의 주고받아야 하는 데이터가 줄어들게 되어 속도가 향상된다. 그러나 아직 지원하고 있지 않다.
트리거는 특별한 이벤트가 발생했을 때 나타나는 stored procedure 이다. 예를 들어 트랜잭션 테이블에서 레코드가 삭제되고 모든 트랜잭션이 지워질 때 상응하는 테이블을 삭제할 수 있는 stored procedure를 설치할 수 있다. 이 기능은 앞으로도 지원하지 않을 것이다.

> 전반적으로 트랜잭션 처리와 트리거등은 데이터베이스의 속도를 저하시킨다. MySQL은 이러한 속도에 영향을 미칠 수 있는 부분을 제거하여 빠른 속도를 내는 것이다. 이러한 부분이 자기가 사용하는 데이터베이스에서 얼마나 중요한가 판단을 해보아야 한다. 트리거와 같은 경우는 자료의 무결성을 보장하기 위해서 필요하다.

5. 외래키(Foreign Keys)

SQL 문에서 외래키는 테이블을 조인할 때 사용하지 않지만 대부분 참조 무결성을 확인할 때 사용한다. SELECT 문에서 다중 테이블에서 자료를 가져오길 원하면 테이블을 조인해서 처리할 수 있다.

```
SELECT * from table1, table2 where table1.id = table2.id
```

6. 뷰(View)

MySQL은 뷰를 지원하지 않지만 지원할 예정이다.

7. '--'를 사용한 주석

일부 다른 SQL 데이터베이스는 '--' 로 주석을 시작한다. 그러나 MySQL 은 '--'를 지원하지 않는다.

9. 권한설정

1. 접근 제어 단계 1 : 연결 확인(인증)

MySQL 서버에 접속하려고 할 때 서버는 사용자 확인과 비밀번호를 통해 접속을 허용하거나 거부한다. 사용자 확인이 안되면 서버는 접속을 완전히 거부한다. 사용자 확인이 되면 서버는 연결을 받아들이고 2번째 단계로 들어가며 요청을 기다린다.

사용자 확인은 두 가지 정보에 기반하고 있다:

- 접속하는 호스트
- MySQL 사용자 이름

사용자 확인은 user 테이블의 세가지 범위 필드(Host, User, Password)를 사용하여 수행된다. 서버는 user 테이블 엔트리의 호스트이름과 사용자 이름이 맞으며, 비밀번호가 정확할 때만 접속을 받아들인다.

아래와 같이 user 테이블의 범위 필드값을 지정할 수 있다:

- Host 값은 호스트 이름이나 IP 숫자 또는 로컬 호스트를 가리키는 'localhost' 가 될 것이다.
- Host 필드에서 '%' 와 '_' 의 와일드카드 문자를 사용할 수 있다.
- '%'의 Host 값은 모든 호스트 이름을 나타낸다. 공백의 호스트 값은 '%'와 같다. 특정한 호스트에 대한 이러한 값은 당신의 서버에 연결할 수 있다는 것을 참고하자.
- 와일드카드 문자는 User 필드에는 허용되지 않는다. 그렇지만 모든 유저에 해당하는 공백으로 둘 수 있다. 연결을 하려는 목록에 공백 사용자 이름이 있다면 클라이언트에서 실제로 지정한 이름 대신에 그 사용자는 익명 사용자, 이름이 없는 사용자로서 간주된다.
- Password 필드는 공백으로 될 수 있다. 이것은 아무런 비밀번호나 사용할 수 있다는 것을 의미하는 것은 아니며 사용자는 비밀번호를 지정하지 않고 연결을 해야 한다는 의미이다.

아래의 테이블은 연결 요청에 적용하는 user 테이블 목록의 Host, User 값이 어떻게 조합되는지를 보여주는 예제이다:

Host	User	의미
'thomas.loc.gov'	'fred'	thomas.loc.gov 에서 연결하는 fred
'thomas.loc.gov'	''	thomas.loc.gov 에서 연결하는 모든 사용자
'%'	'fred'	모든 호스트에서 연결하는 fred
'%'	''	모든 호스트에서 연결하는 모든 사용자
'%.loc.gov'	'fred'	loc.gov 도메인의 모든 호스트에서 연결하는 fred
'x.y.%'	'fred'	x.y.net, x.y.com, x.y.edu 등에서 접속하는 fred (이것은 아마도 유용하지 않을 것이다)
'144.155.166.177'	'fred'	144.155.166.177의 IP 주소에서 접속하는 fred
'144.155.166.%'	'fred'	144.155.166 클래스 C 서브넷의 모든 호스트에서 접속하는 fred

Host 필드에서 IP에 와일드카드를 사용할 수 있기 때문에 (예를 들어 '144.155.166.%' 는 서브넷의 모든 호스트에 적용된다) 144.155.166.somewhere 와 같은 호스트 이름을 이용하여 부당하게 이용할 가능성이 생길 수 있다. 이러한 것을 막기 위해 MySQL은 숫자와 도트(.)으로 시작하는 호스트이름은 허용하지 않는다. 1.2.foo.com 과 같은 호스트라면 이러한 호스트이름은 승인(grant) 테이블의 Host 컬럼과 매치되지 않는다. IP 숫자만이 IP 와일드 카드 값과 매치시킬 수 있다.

만약 한 개 이상의 user table 목록이 있다면 서버는 어떻게 user table을 선택할까? 이런 경우에는 user table의 정렬 순서에 따라 해결을 하며, 정열은 서버가 시작할 때 수행이 된다. user table이 다음과 같다고 가정해보자:

Host	User	...
%	root	...
%	jeffrey	...
localhost	root	...
localhost		...

서버가 테이블을 읽을 때, 먼저 특정하게 지정된 값이 있는 호스트부터 목록을 정렬한다. (Host 컬럼에서 '%'는 "모든 호스트"를 의미하여 최소한도로 지정하는 것이다) 목록에서 호스트값이 같으면 먼저 특정하게 지정된 사용자가 있는 것부터 정열한다. (공백으로 되어 있는 User 값은 "모든 사용자"를 의미하여 최소한도로 지정하는 것이다.) 이렇게 하면 정열된 user 테이블은 다음과 같다:

Host	User	...
localhost	root	...
localhost		...
%	jeffrey	...
%	root	...

정렬된 순서에 따라 매칭 알고리즘이 적용되며 먼저 매칭되는 것을 사용한다. localhost에서 jeffrey가 연결을 하려할때, Host 컬럼에서 'localhost' 목록이 먼저 매칭된다. 물론 사용자 이름이 공백인 목록은 연

결하는 호스트네임과 사용자 이름에 매칭된다. ('%'/'jeffrey' 목록 또한 매칭이 된다. 그러나 테이블에서 처음으로 매칭되는 것은 아니다.)

다른 예제가 있다. user 테이블이 다음과 같다고 가정해보자:

Host	User	...
% thomas.loc.gov	jeffrey	... ...

정렬된 테이블은 다음과 같다:

Host	User	...
thomas.loc.gov %	 jeffrey	... ...

첫번째로 thomas.loc.gov에서 jeffrey가 연결하는 것이 매칭되며, whitehouse.gov에서 jeffrey가 연결하는 것은 두번째로 매칭이 된다.

서버에 연결할 때 문제가 생기면, user 테이블을 출력하여 어떤 것이 먼저 매칭되는지 직접 정렬을 하면 된다.

2. 접근 제어 단계 2 : 요청 인증

연결되었다면 서버는 단계 2로 들어간다. 연결이 성사되었을 때 각 요구에 대해 사용자가 수행하려는 연산의 유형에 기반하여 서버는 사용자가 충분한 권한을 가지고 있는지 점검한다.
여기서 승인 테이블의 권한 필드가 작동한다. 권한은 user, db, host, table_priv, columns_priv 테이블의 정보를 사용한다. GRANT 와 REVOKE 명령을 이용하여 권한 테이블을 다룰 수 있다.

user 테이블의 승인 권한은 사용자에게 전체적인 기반을 제공하며 현재의 데이터가 어떤 것인지와는 상관이 없다. 예를 들어, user 테이블에서 사용자에게 delete 권한을 승인했다면 서버 호스트에서 어떤 데이터베이스의 레코드라도 삭제할 수 있다! 다르게 말해서 user 테이블 권한은 슈퍼유저 권한이며, 슈퍼유저(서버나 데이터베이스 관리자 등)에게만 user 테이블에 대한 권한을 승인하는 것이 좋다. 다른 사용자에게는 user 테이블에서 권한을 'N'으로 설정하고, db와 host 테이블을 사용하여 특정 데이터베이스에 기반한 권한승인을 하는게 좋다.

db와 host 테이블은 특정 데이터베이스의 권한을 승인한다. 각 테이블의 Host 와 Db 필드에서 와일드카드 문자 '%' 와 '_' 를 사용할 수 있으며 값이 공백이면 필드 범위(scope fields)에서 모든 값을 허용한다. '%' Host 값은 "모든 호스트"를 의미한다. db 테이블에서 Host 값이 공백이면 "host 테이블에서 더 자세한 정보를 문의하라"는 의미이다. User 값이 공백이면 익명 사용자로 간주된다.

서버가 시작할 때 db 와 host 테이블을 읽고 정열을 한다.(동시에 user 테이블을 읽는다.) db 테이블은 Host, Db, User 순으로 필드 범위를 정열하며 host 테이블은 Host, Db 순으로 필드 범위를 정열한다. user 테이블과 같이 특정하게 지정되어 있는 값이 먼저 정열되고 최소한도로 지정된 값이 나중에 정열

된다. 서버에서 매칭되는 목록을 찾을때, 가장 먼저 발견한 것을 사용한다.

tables_priv 와 columns_priv 테이블은 특정한 테이블과 컬럼에 관련된 권한을 승인한다. db와 host 테이블의 Host 필드와 같이 와일드카드를 Host 필드에서 사용할 수 있다. 그렇지만 Db, Table_name, Column_name 필드에서는 와일드카드나 공백값을 사용할 수 없다.

Host 테이블에서만 와일드카드를 사용할 수 있지만 tables_priv 와 columns_priv 테이블은 db 테이블과 비슷하게 정열이 되며 정열은 간단하다.

요청 인증 과정은 아래에서 설명한다. 접근-점검 소스 코드에 친숙하다면, 여기서 설명하는 것은 코드에서 사용된 알고리즘과는 약간 다르다는 것을 알 수 있다. 여기서의 설명은 코드가 실제로 작동하는 방식과 동일하다. 단지 설명을 간단하게 하는데서 차이가 있는 것이다.

관리자 요청에 대해서(shutdown, reload 등) 서버는 단지 user 테이블만 체크를 한다. 왜냐하면 user 테이블에서만 관리자 권한을 지정하기 때문이다. 목록에서 요청된 연산을 허용하면 접근이 허용되며 아닌 경우에는 접근이 거부된다. 예를 들어, mysqladmin shutdown을 실행하고자 하는데 user 테이블 목록에서는 사용자에게 shutdown 권한을 승인하지 않으면, db나 host 테이블을 체크하지 않더라도 접근이 거부된다. (이러한 테이블에는 Shutdown_priv 컬럼이 없기 때문에 이렇게 할 필요도 없다)

데이터베이스와 관련된 요청에 대해(insert, update 등) 서버는 먼저 user 테이블 목록에서 사용자의 전체(슈퍼유저) 권한을 점검한다. 목록에서 요청한 연산을 허용하면 접근이 승인된다.

user 테이블에서 전체적인 권한이 불충분하면, 서버는 db 와 host 테이블을 점검하여 데이터베이스에 관련된 권한을 결정한다:

① 서버는 db 테이블에서 매칭되는 Host, Db, User 필드를 찾는다. 연결하려는 사용자의 호스트 이름과 Mysql 사용자 이름이 Host 와 User에 매칭된다. 사용자가 접근하기 원하는 데이터베이스는 Db 필드에 매칭된다. 적합한 Host 와 User 목록이 없으면 접근은 거부된다.

② 매칭되는 db 테이블 목록이 있고 Host 필드가 공백이 아니면, 목록은 사용자의 데이터베이스 관련 권한을 정의한다.

③ 매칭되는 db 테이블 목록의 Host 필드가 공백이면, host 테이블에서 어떤 호스트가 데이터베이스에 접근할 수 있는지 판단한다는 것을 의미한다. 이런 경우, 더 자세한 정보를 위해 host 테이블에서 매칭되는 Host 와 Db 필드를 찾는다. host 테이블에 매칭되는 목록이 없으면 접근은 거부된다. 매칭되는 목록이 있으면 사용자의 데이터베이스 관련 권한은 db 와 host 테이블 목록에서 권한을 intersection 하여 결정된다. 다시 말해서, db와 host 테이블 둘 다 'Y'로 되어있을 때 권한이 설정된다. 이러한 방법으로 db 테이블에서 일반적인 권한을 승인할 수 있으며, 그러고 나서 host 테이블 목록을 사용해 host를 기반으로 하여 선택적으로 권한을 제한할 수 있다.)

db 와 host 테이블 목록을 이용해 데이터베이스와 관련된 권한 승인을 결정한 후, 서버는 이러한 정보를 user 테이블에서 승인한 전체적인 권한에 추가한다. 그 결과가 요청한 연산을 허용하면 접근이 허용된다. 다른 방법으로, 서버는 tables_priv 와 columns_priv 테이블에서 사용자의 테이블과 컬럼 권한을 점검하고 사용자의 권한에 추가한다. 그 결과에 따라 접근이 허용되거나 거부된다.

왜 서버에서 전체적인 사용자 엔트리 권한에 데이터베이스, 테이블, 컬럼에 관련된 권한을 추가하는지가 명확하지 않다. 이런 경우 사용자 권한은 초기에 요청된 연산에 대하여 불충분하다. 요청은 한가지 유형 이상의 권한이 필요하기 때문이다. 예를 들어, INSERT ... SELECT 문을 수행할 때 insert 와 select 권한 둘 다 필요하다. 사용자의 권한은 user 테이블에서 한가지 권한을 승인하고 db 테이블 엔트리에서 다른 권한을 승인할 것이다. 이런 경우, 사용자는 이러한 요청을 수행하기 위해 필요한 권한을 가지고 있다. 그렇지만 서버는 자체적으로 다른 테이블에 대해서는 두 엔트리에 의해 승인된 권한이 조합되어야 한다.

host 테이블은 "안전한" 서버 목록을 유지하는데 사용할 수 있다. TcX에서는, host 테이블에는 지역 네트웍의 모든 시스템이 포함되어 있다. 여기서는 모든 권한이 허용된다.

안전하지 않는 호스트를 가리키기 위해 host 테이블을 사용할 수 있다. 안전하다고 생각되지 않는 공개 지역에 위치한 public,your,domain 시스템이 있다고 가정해보자. 사용자는 사용자 네트웍의 모든 호스트에 접근할 수 있으며, host 테이블 엔트리가 다음과 같은 시스템만 제외한다 :

Host	Db	...
public,your,domain	%	...(all privileges set to 'N')
%,your,domain	%	...(all privileges set to 'Y')

당연히 접근 권한이 원하는 대로 되어 있는지 언제나 승인 테이블에서 목록을 점검해야 한다. (예를 들어 mysqlaccess 를 사용한다.)

3. 권한 변경시 적용 방법

mysqld 가 시작할 때, 모든 승인 테이블 내용이 메모리로 올라가고 이때부터 유효하게 된다.

GRANT, REVOKE, SET PASSWORD 를 이용해 승인 테이블에 변경을 하면 바로 서버에서 인식을 한다.

권한 테이블을 직접 변경했다면(INSERT, UPDATE 등을 사용하여), 서버에서 승인 테이블을 재가동하도록 하기 위해 FLUSH PRIVIEGES 문이나 mysqladmin flush-privileges 를 실행해야 한다. 그렇게 하지 않으면 서버를 다시 시작하기 전까지 변경된 권한이 적용되지 않는다.

서버에서 권한 테이블이 변경되었다는 것을 감지했을 때, 이미 존재하던 클라이언트 연결은 다음과 같이 영향을 받는다:

① 테이블과 컬럼 권한 변경은 클라이언트의 다음 요청부터 적용된다.
② 데이터베이스 권한 변경은 다음의 USE db_name 명령부터 적용된다.
③ 전체적인 권한과 비밀번호 변경은 클라이언트가 다음에 연결할 때부터 적용된다.

4. 초기 MySQL 권한설정

MySQL을 설치하고 나서, mysql_install_db 스크립트를 실행해서 초기 접근 권한을 설정해야 한다. mysql_install_db 스크립트는 mysqld 서버를 시작하고, 다음과 같이 승인 테이블의 권한을 초기화한다:

▪ mysql root 사용자는 슈퍼유저이며 모든 것을 할 수 있다. 로컬 호스트에서만 연결할 수 있다.
주의 : 처음에 root 비밀번호는 비어있다. 그래서 누구나 비밀번호없이 root로 연결할 수 있고 모든 권한을 승인받는다.

▪ 익명 사용자는 'test' 나 'test_' 로 시작하는 데이터베이스에 대한 모든 권한을 승인받는다. 모든 사용자가 로컬 호스트에서 연결할 수 있으며 익명 사용자로 간주된다.

▪ 다른 권한은 거부된다. 예를 들어 일반 사용자는 mysqladmin shutdown 이나 mysqladmin processlist 를 사용할 수 없다.

설치했을 때 초기 권한이 폭넓게 설정되어 있기 때문에 가장 먼저 mysql root 사용자의 비밀번호를 설정해야 한다. 다음과 같이 설정하면 된다. (PASSWORD() 함수를 이용해 비밀번호를 설정해야 한다!)

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('new_password')
           WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

또는

```
shell> mysqladmin -u root password new_password
```

두 번째의 경우는 PASSWORD() 라는 함수를 사용하지 않아도 되므로 편리하다. 그리고 SQL문이라도 grant 명령을 이용해서도 설정할 수 있다.

첫번째 방법을 사용하면 직접 user 테이블의 비밀번호를 업데이트한다. 이경우 서버가 다시 승인 테이블을 읽도록 해야 한다. (FLUSH PRIVILEGES 사용한다) 왜냐하면 다른 방법으로는 변경 사항을 알릴 수 없기 때문이다.

※ 승인 테이블을 다시 읽지 않아서 이전에 설정했던 비밀번호가 제대로 안되는 경우도 있다.

root 비밀번호가 설정되었으면 서버에 root로 접속할때마다 비밀번호를 명시해야 한다.

추가로 셋업을 하거나 테스트할 때는 비밀번호를 설정할 필요가 없기 때문에 root 비밀번호를 빈값으로 남겨두고 싶을 것이다. 그렇지만 실제 작업을 하기 전에는 반드시 비밀번호를 설정했는지 확인해야 한다.

기본권한을 어떻게 설정하는지 mysql_install_db 스크립트를 살펴보자. 다른 사용자에게 권한을 어떻게 설정할지 이것을 기본으로 사용할 수 있다.

위에서 설명한 것과 다르게 초기 권한을 설정하기 원하면, mysql_install_db 스크립트를 실행하기 전에 수정하면 된다.

완전하게 승인 테이블을 다시 만들기 위해 MySQL 데이터베이스를 포함하는 디렉토리의 '*ISM' 과 '*.ISD' 파일을 제거해야 한다. (이 디렉토리는 database 디렉토리에서 'mysql'이라는 이름이 붙어있다. mysqld --help 해서 database 디렉토리의 목록을 볼 수 있다.) 원하는대로 권한을 수정한 후 mysql_install_db 스크립트를 실행하자.

5. MySQL에 새로운 사용자 권한 추가하기

다음 두가지 방법으로 사용자를 추가할 수 있다.

- GRANT 문 사용
- MySQL 승인 테이블 직접 조작.

GRANT 문을 사용하는 것이 더 편리하고 다른 DBMS에서 사용되기 때문에 일관성이 있다.

아래의 예제는 새로운 사용자를 설정하기 위해 어떻게 MySQL 클라이언트를 사용하는지 보여 준다. 이 예제는 이전에 설명했던 것과 같이 기본값에 따라 권한을 설정하는 것으로 가정한다. 이것은 설정을 바꾸기 위해 mysqld가 실행되고 있는 같은 시스템에 있어야 한다는 것을 말한다. 초기값은 localhost에서만 접근이 가능하게 되어 있다. 또한 MySQL root 사용자로 접속해야 하고 root 사용자는 MySQL 데이터베이스에 대한 insert 권한과 reload 관리자 권한이 있어야 한다. root 사용자의 비밀번호를 바꾸었으면, 아래와 같이 MySQL 명령행 상태에서 비밀번호를 명시해야 한다.

GRANT 문을 이용해 새로운 사용자를 추가할 수 있다:

```
shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
          IDENTIFIED BY 'something' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
          IDENTIFIED BY 'something' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

위 GRANT 문에서는 세 명의 사용자를 설정한다:

monty : 어느 곳에서든 서버에 연결할 수 있는 완전한 슈퍼유저이지만 비밀번호를 사용해야 한다. monty@localhost 와 monty@"%"를 사용한 GRANT 문에 대해서 알아야 한다. localhost 목록을 추가하지 않으면, mysql_install_db에 의해 생성된 localhost의 익명 사용자 목록이 로컬 호스트에서 접속할 때 우선권을 갖는다. 왜냐하면 지정된 Host 필드 값이 있으며 정열 순서에서 먼저 오기 때문이다. 승인 테이블의 정열 순서가 특정한 Host를 지정한 것부터 시작하는 것을 기억하자.

admin : 비밀번호 없이 localhost에서 접속할 수 있으며 reload와 process 관리자 권한을 승인받은 사용

자. 이 경우 사용자가 mysqladmin processlist 뿐만 아니라 mysqladmin reload, mysqladmin refresh, mysqladmin flush-* 명령을 실행할 수 있다. 데이터베이스와 관련된 권한은 승인되지 않았다. 이것은 추가적인 GRANT 문을 사용해 나중에 승인할 수 있다.

dummy : 비밀번호없이 연결할 수 있지만 오직 localhost에서만 연결 가능한 사용자. 권한 유형(privilege type)이 USAGE 이기 때문에 전체적인 권한이 'N'로 설정되어 있다. USAGE는 아무런 권한도 설정하지 않는다. 나중에 데이터베이스와 관련된 권한을 승인할 수 있다.

또한 동일한 사용자 접근 정보를 INSERT 문을 통해 직접 추가할 수 있으며 이 경우에는 서버가 승인 테이블을 다시 읽도록 알려주어야 한다. 물론 FLUSH PRIVILEGES 사용한다.

```
shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('something'),
                'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y')
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('something'),
                'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y')
mysql> INSERT INTO user SET Host='localhost',User='admin',
                 Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
                       VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

MySQL 버전에 따라 위에서 'Y' 값이 다를 수 있다는 것을 기억하자. 3.22.11 버전 이후에서 사용할 수 있는 확장된 INSERT 문은 여기서 admin 사용자에게 사용되었다.

슈퍼유저를 설정하기 위해 권한필드를 'Y'로 한 user 테이블 목록만 만들면 된다는 것을 기억하자. db나 host 테이블 목록은 필요없다. 관리자 권한은 db나 host 테이블과는 전혀 관련이 없다. db는 접속할 수 있는 데이터베이스에 대해 상세하게 설정하고 host 테이블은 db테이블을 좀 더 정교하게 설정하기 위해 필요한 것이다. 관리자 권한은 오직 user 테이블만 관련되어있다.

마지막 INSERT 문(dummy 사용자)에서는 user 테이블의 권한 컬럼이 명확하게 설정되지 않았다. 왜냐면 이 컬럼의 기본값은 'N'로 되어 있기 때문이다.

다음의 예제에서는 custom 이라는 사용자를 추가한다. custom은 localhost, server.domain, whitehouse.gov에서 접속할 수 있다. localhost에서는 bankaccount 데이터베이스에만 접속할 수 있으며 whitehouse.gov에서는 expenses 데이터베이스에, 모든 세 호스트상에서는 customer 데이터베이스에 접속하길 원한다. 모든 세 호스트상에서 stupid라는 비밀번호를 사용하길 원한다.

GRANT 문을 이용 이러한 사용자 권한을 설정하기 위해 다음의 명령을 실행하자:

```
shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
           ON bankaccount.*
           TO custom@localhost
```

```
            IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
            ON expenses.*
            TO custom@whitehouse.gov
            IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
            ON customer.*
            TO custom@'%'
            IDENTIFIED BY 'stupid';
```

승인 테이블을 직접 수정해 사용자 권한을 설정하려면 다음의 명령을 사용하자. 마지막에 FLUSH PRIVILEGES를 사용해야 한다는 것을 기억하자.

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
       VALUES('localhost','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
       VALUES('server.domain','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
       VALUES('whitehouse.gov','custom',PASSWORD('stupid'));
mysql> INSERT INTO db
       (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       VALUES
       ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
       (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       VALUES
       ('whitehouse.gov','expenses','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
       (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       VALUES('%','customer','custom','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

처음의 세가지 INSERT 문은 custom 사용자가 비밀번호를 사용하여 다양한 호스트에서 접속할 수 있도록 user 테이블 목록을 추가한다. 그렇지만 그에게 어떠한 퍼미션도 승인하지 않는다. 모든 권한은 기본값으로 'N' 이다. 다음의 세가지 INSERT 문은 적절한 호스트에서 접속을 할 때, custom 에게 bankaccount, expenses, customer 데이터베이스에 대한 권한을 승인하는 db 테이블 목록을 추가한다. 일반적으로 승인 테이블을 직접 수정하였으면, 변경된 권한을 적용하기 위해 서버가 승인 테이블을 다시 읽도록 해 주어야 한다.

특정한 사용자가 특정한 도메인의 시스템에서 접속할 수 있도록 설정하고자 한다면, 다음과 같이

GRANT 문을 설정할 수 있다:

```
mysql> GRANT ...
           ON *.*
           TO myusername@"%.mydomainname.com"
           IDENTIFIED BY 'mypassword';
```

승인 테이블을 직접 수정하려면 다음과 같이 한다:

```
mysql> INSERT INTO user VALUES ('%.mydomainname.com', 'myusername',
           PASSWORD('mypassword'),...);
mysql> FLUSH PRIVILEGES;
```

이러한 승인 테이블을 다루기 위해서 xmysqladmin, mysql_webadmin, xmysql 프로그램을 사용할 수도 있다. http://www.mysql.com/Contrib 에서 이러한 유틸리티를 찾을 수 있다.

6. 비밀번호 설정 방법

앞의 예제는 중요한 원칙을 보여준다.
INSERT 나 UPDATE 문에서 공백이 아닌 비밀번호를 저장할 때 반드시 암호화하기 위해 PASSWORD() 함수를 사용해야 한다. user 테이블은 비밀번호를 일반 텍스트가 아니라 암호화된 형태로 저장하기 때문이다. 이러한 사실을 잊어버리면 다음과 같이 비밀번호를 설정하려고 할 것이다:

```
shell> mysql -u root mysql
mysql> INSERT INTO user (Host,User,Password)
       VALUES('%','jeffrey','bLa81m0');
mysql> FLUSH PRIVILEGES;
```

플레인텍스트 값 'bLa81m0' 은 user 테이블에 비밀번호로 저장이 된다. jeffrey라는 사용자가 이 비밀번호를 사용해 서버에 연결하려고 할 때 MySQL 클라이언트를 이 비밀번호를 암호화해서 그 결과를 서버로 보낸다. 서버는 암호화된 비밀번호('bLa81m0'이 아니다)를 user 테이블의 비밀번호(플레인텍스트 'bLa81m0' 값이다)와 비교한다. 비교는 실패하고 서버는 연결을 거부한다:

```
shell> mysql -u jeffrey -pbLa81m0 test
Access denied
```

비밀번호는 user 테이블에 입력될 때 반드시 암호화되어야 하기 때문에, INSERT 문은 다음과 같이 사용해야 한다:

```
mysql> INSERT INTO user (Host,User,Password)
       VALUES('%','jeffrey',PASSWORD('bLa81m0'));
```

또한 SET PASSWORD 문을 사용할 때도 PASSWORD() 함수를 사용해야 한다:

```
mysql> SET PASSWORD FOR jeffrey@"%" = PASSWORD('bLa81m0');
```

참고 : PASSWORD() 함수는 비밀번호 암호화를 수행한다. 그렇지만 유닉스에서 비밀번호를 암호화하는 방법과는 다르다. 유닉스 비밀번호와 MySQL 비밀번호가 동일할 때 PASSWORD()가 유닉스 비밀번호 파일에 암호화되어 저장된 값과 같다고 생각하면 안 된다.

GRANT ... IDENTIFIED BY 문이나 mysqladmin password 명령을 사용해 비밀번호를 설정하면 PASSWORD() 함수는 필요없다. 둘 다 비밀번호를 암호화해서 저장한다:

```
mysql> GRANT USAGE ON *.* TO jeffrey@"%" IDENTIFIED BY 'bLa81m0';
shell> mysqladmin -u jeffrey password bLa81m0
```

당연히 GRANT 문이나 mysqladmin password 명령을 사용하는게 편하겠지요? MySQL의 암호화 알고리즘이 유닉스와 다르듯 유닉스 계정과 MySQL의 사용자는 전혀 다르다는 것도 다시 한번 기억하고 있어야 합니다.

7. 접근 거부 에러가 나는 이유

MySQL 서버에 연결하려 할 때 접근 거부 에러가 나면, 아래에서 설명하는 것에 따라 해결 방법을 찾을 수 있다:

▪ 초기 승인 테이블 내용을 설정하기 위해 MySQL을 설치한 후 mysql_install_db 스크립트를 실행하였는가? 실행하지 않았다면 스크립트를 실행하자. 다음 명령을 이용해 초기 권한을 시험해 볼 수 있다:

```
shell> mysql -u root test
```

에러없이 서버에 접속할 수 있을 것이다. MySQL 데이터베이스 디렉토리에 'user.ISD' 파일이 있는지 확인해보아야 한다. (일반적으로 MySQL 설치 디렉토리 '/var/mysql/user.IDS' 이다)

▪ 설치를 새로하고 난 후 , 서버에 연결하고 사용자와 접근 권한을 설정해야 한다:

```
shell> mysql -u root mysql
```

초기에 MySQL root 사용자만 비밀번호가 없기 때문에 서버에 연결할 수 있다. 보안문제가 있기 때문에, 다른 MySQL 사용자를 설정하기 전에 먼저 root의 비밀번호를 설정해야 한다. root로 접속하려하는데 다음의 에러가 났다고 가정하자:

```
Access denied for user: '@unknown' to database mysql
```

이것은 user 테이블에 User 컬럼 = root 라는 목록이 없고, mysqld가 사용자 클라이언트의 호스트이름을 해석할 수 없다는 것을 의미한다. 이런 경우 --skip-grant-tables 옵션을 이용해 서버를 다시 시작해야 하고 '/etc/hosts' 를 편집하거나 '\windows\hosts' 파일에 사용자 호스트 목록을 추가해야 한다.

▪ 3.22.11 이전 버전에서 3.22.11이나 이후 버전으로 업데이트했다면, mysql_fix_privilege_tables 스크립

트를 실행했는가? 하지 않았다면 실행한다. mysql 3.22.11에서 GRANT 문 기능이 가능해지면서 승인 테이블 구조가 바뀌었다.

▪ (INSERT 나 UPDATE 문을 사용해) 승인 테이블을 직접 고쳤고 변화가 아직 반영되지 않은 것으로 보이면, FLUSH PRIVILEGES 문을 사용하거나 mysqladmin flush-privileges 명령을 사용해 서버가 승인 테이블을 다시 읽도록 해야 한다는 것을 기억하자. 그렇지 않으면 서버가 재시작하기 전까지는 변화된 것이 반영되지 않는다. root 비밀번호를 설정하고 나서 권한을 flush 하기까지는 비밀번호를 명시할 필요가 없다. 왜냐면 서버는 아직 비밀번호를 바꾸었는지 모르기 때문이다.

▪ 세션 중간에 권한이 변경된 것으로 보이면 슈퍼유저가 바꾸었을 것이다. 승인 테이블을 재시작하는 것은 새로운 클라이언트 접속에 영향을 미치지만 이미 존재하고 있던 연결은 이전의 설명과 같다.

▪ 시험하기 위해, mysqld 대몬에 --skip-grant-tables 옵션을 주어 시작하자. 그러고 나서 MySQL 승인 테이블을 변경할 수 있고 변경된것이 원하는대로 작동하는지를 체크하는 mysqlaccess 스크립트를 사용할 수 있다. 원하는대로 수정이 되었으면 mysqld 서버가 새로운 승인 테이블로 시작할 있도록 mysqladmin flush-priveleges 를 실행한다.

주의 : 승인테이블을 재로딩하는 것은 --skip-grant-tables 옵션을 무효화한다. 이를 통해 서버를 다운시키고 다시 재시작하지 않고도 승인 테이블을 시작할 수 있다.

▪ 펄, Python, ODBC 프로그램에서 접근하는데 문제가 있다면, mysql -u user_name db_name 또는 mysql -u user_name -pyour_pass db_name 으로 서버에 접속을 시도해보자. (-p 와 비밀번호사이에는 공백이 없다. 또한 --password=your_pass 형태로도 사용할 수 있다) MySQL 클라이언트로 접속이 되면 프로그램에 문제가 있는 것이며 접근 권한에는 문제가 없다.

▪ 비밀번호가 제대로 작동하지 않으면, INSERT, UPDATE, SET PASSWORD 문에서 비밀번호를 설정하면서 PASSWORD() 함수를 반드시 사용해야 한다는 것을 기억하자. PASSWORD() 함수는 GRANT ... INDENTIFIED BY 문이나 mysqladmin password 명령을 사용했다면 불필요하다.

▪ localhost 는 지역 호스트 이름과 같은 말이다. 또한 호스트를 명백하게 설정하지 않은 경우 클라이언트에서 연결하려는 호스트의 기본값이다. 그러나 MIT-pthreads를 사용하는 시스템에서는 localhost로의 연결이 제대로 작동하지 않는다. (localhost 연결은 유닉스 소켓을 통해 만들어진다. 그렇지만 MIT-pthreads에서는 유닉스 소켓을 지원하지 않는다.) 이와 같은 시스템에서 문제를 피하려면, 서버에 호스트 이름을 명확하게 말해주기 위해 --host 옵션을 사용해야 한다. 그러면 mysqld 서버에 TCP/IP 연결을 만든다. 이경우, 서버 호스트의 user 테이블 목록에 실제 호스트이름이 있어야 한다. (서버와 동일한 호스트에서 클라이언트 프로그램을 실행한다고 하더라도 마찬가지이다.)

▪ mysql -u user_name db_name 으로 데이터베이스에 접속하려 할 때 접근 거부 에러가 나면, user 테이블에 문제가 있을 것이다. mysql -u root mysql 를 실행하여 점검하고 다음의 SQL 문을 사용하자:

```
mysql> SELECT * FROM user;
```

여기서 사용자의 호스트이름과 MySQL 사용자 이름과 맞는 Host 와 User 컬럼의 목록이 포함되어 있어야 한다.

▪ Access denied 에러 메시지는 접속하려는 사용자와 호스트 이름, 그리고 비밀번호를 사용했는지 여부를 보여줄 것이다. 일반적으로 user 테이블에 에러 메시지에서 보여준 호스트 이름과 사용자 이름과 정확하게 맞는 목록을 가지고 있어야 한다.

▪ 다른 시스템에서 MySQL 서버에 접속할 때 다음의 에러 메시지가 나오면, user 테이블에 연결을 하려고 하는 호스트 이름이 없다는 것을 말한다:

Host ... is not allowed to connect to this MySQL server

서버 호스트에서 명령행 유틸리티인 mysql을 사용하여 user 테이블에 연결하고자 하는 사용자/호스트 이름을 추가하여 해결할 수 있다. MySQL 3.22 를 사용하고 있지 않고 연결하고자 하는 시스템의 IP 숫자나 호스트 이름을 모른다면, user 테이블에 Host 컬럼 값으로 '%' 목록을 입력하고 서버 시스템에서 --log 옵션을 사용해 mysqld 를 재시작하자. 클라이언트 시스템에서 연결을 시도한 후 MySQL 로그에는 어떻게 실제로 연결을 했는지에 대한 정보가 들어있다. 그러고나서 '%'를 로그에 나온 실제 호스트 이름으로 바꾼다. 그렇지 않으면 보안에 문제가 생길 수 있다.

▪ mysql -u root test 는 작동을 하는데 mysql -h your_hostname -u root test 에서 접근에 에러가 나면, user 테이블에 정확한 호스트 이름이 없을 것이다. 일반적인 문제는 user 테이블의 Host 값에는 완전하지 않은 호스트 이름 - 도메인은 빼고 호스트 이름만 넣은 경우 - 이 들어가 있는데 시스템의 네임 해석 루틴은 FQDN(Fully-Qualified Domain Name) 으로 처리하는 경우이다. 예를 들어, user 테이블에 호스트 이름이 'seongju' 으로 되어있는데, DNS는 MySQL에 호스트 이름이 'seongju.subnet.se'라고 알려줄 수 있으며 이 경우는 제대로 작동하지 않을 것이다. user 테이블에 Host 컬럼 값으로서 해당하는 IP 숫자나 호스트 이름을 추가하자. 대신, user 테이블에 Host 값으로 와일드카드 문자를 포함할 수 있다. 예를 들어, 'seongju.%'. 그러나 호스트이름 끝에 '%'를 사용하는 것은 안전하지 않으며 권하지도 않는다.

▪ mysql -u user_name test 는 작동하는데 mysql -u user_name other_db_name 은 작동하지 않는다면 db 테이블에 other_db_name 목록이 없는 경우이다.

▪ 서버 시스템에서 mysql -u user_name db_name 은 작동을 하는데, 다른 클라이언트 시스템에서 mysql -u user_name db_name 은 작동을 하지 않는다면, user 테이블이나 db 테이블에 클라이언트 시스템의 목록이 없는 것이다.

▪ 왜 Access denied 가 나는지 해결하지 못하면, user 테이블에서 와일드카드('%' 또는 '_')를 포함하고 있는 Host 값을 가진 목록을 모두 제거하자. 매우 일반적인 에러는 Host='%' 그리고 User='some user'로 입력을 하고 나서, 이렇게 하면 같은 시스템에서 연결할 때 localhost를 지정할 수 있도록 허용한다고 생각하는 것이다. 이것이 제대로 작동하지 않는 이유는 기본 권한에 Host='localhost' 와 User='' 목록이 포함되어 있기 때문이다. Host 값에 '%' 보다 더 분명한 'localhost' 목록이 있기 때문에, localhost에서 접속할 때 새로운 목록보다 먼저 선택이 된다. 정확한 절차는 두번째 항목으로 Host='localhost' 와 User='some_user' 를 입력하거나 Host='localhost' 와 User='' 를 제거하는 것이다.

▪ 다음의 에러가 나는 경우

Access to database denied

db 나 host 테이블에 문제가 있을 것이다. 선택한 db 테이블의 목록에 Host 컬럼이 비어있다면, host 테이블에 db 테이블 목록에 적용되는 호스트 이름이 있는지를 확인해야 한다. 일반적으로 db 테이블에 host 값을 비워두는 경우, host 테이블에서 접근하는 호스트를 제어할 수 있다.

▪ 다음의 에러가 나는 경우:

Access to database denied

SELECT ... INTO OUTFILE 또는 LOAD DATA INFILE 의 SQL 문을 사용하는 경우, user 테이블 목록에서 file 권한이 설정되어 있지 않았을 것이다.

▪ 다른 모든 것이 실패하였을 경우, mysqld 대몬을 디버깅 옵션으로 시작하자. (예를 들어, --debug=d, general, query) 그러면 각 명령에서 생기는 정보와 함께, 접속을 시도하는 호스트와 사용자에 대한 정보를 출력할 것이다.

▪ mysql 승인 테이블에서 다른 문제에 부딪쳤고 이 문제를 메일링리스트에 알려야겠다고 느끼면, mysql 승인 테이블을 덤프하여 제공해야 한다. mysqldump mysql 명령을 사용해 테이블을 덤프할 수 있다. 언제나 mysqlbug 스크립트를 사용하여 발생하는 문제를 보내야 한다.

▪ 다음의 에러가 나는 경우, mysqld 대몬이 실행되지 않거나 잘못된 소켓이나 포트로 연결하려고 시도하는 경우:

Can't connect to local MySQL server
Can't connect to MySQL server on some_hostname

먼저 mysqld 대몬이 실제로 작동하는지 ps를 이용해 확인한다. 소켓 파일이 있는지 확인하고 점검을 해보아야 한다. (일반적으로 '/tmp/mysql.sock' 이다) 또는 telnet host_name 3306 으로 접속을 시도해보자. 더 자세한 정보를 위해 mysqladmin version 과 mysqladmin -h host_name version 을 사용해 볼 수 있다. 물론 참고할 것이 있는지 MySQL 데이타 디렉토리의 에러 로그 파일을 점검해보자.

▪ 클라이언트 프로그램은 설정 파일이나 환경 변수에서 지정한 연결 패러미터를 사용할 수 있다는 것을 기억하자. 명령행에서 지정하지 않았는데 클라이언트가 잘못된 기본 연결 패러미터를 보내는 것으로 생각되면, 홈 디렉토리에서 환경변수나 '.my.cnf' 파일을 점검해보자. 비록 여기서 지정한 연결 패러미터와는 관계가 멀지만 시스템의 전반적인 MySQL 설정 파일을 점검해 볼 수 있다. 클라이언트에서 아무런 옵션도 주지 않았는데 Access denied 에러 메시지가 나오는 경우, 옵션 파일 중에서 예전의 비밀번호를 지정하지 않았는지 확인해 본다.

8. 크랙커에 대비하여 MySQL을 안전하게 하는 방법

MySQL 서버에 연결할 때 일반적으로 비밀번호를 사용해야 한다. 비밀번호는 연결할 때 단순한 텍스트로 전송되지 않는다.

서버/클라이언트 연결은 암호화되지 않기 때문에 주의해야 한다. 모든 정보는 연결을 볼 수 있는 누구라도 읽을 수 있는 텍스트로 전송된다. 이 문제에 대해 걱정이 되면 문제를 어렵게 하기 위해 압축 프로토콜(MySQL 3.22 이상 버전)을 사용할 수 있다. 보안을 더 확실하게 하기 위해 SSH를 설치할 수 있다. (http://www.cs.hut.fi/ssh 참고) 이것을 이용해 MySQL 서버와 클라이언트 사이에 암호화된 TCP/IP 연결을 사용할 수 있다.

MySQL 시스템의 보안을 유지하게 위해 다음의 제안을 신중하게 고려하자:

• 모든 MySQL 사용자가 비밀번호를 사용. 어떤 사용자가 비밀번호가 없으면 'mysql -u 사용자이름'을 이용해 간단하게 그 사용자로 로그인할 수 있다는 것을 기억하자. 이것은 클라이언트/서버 애플리케이션의 일반적인 작동방법이다. mysql_install_db 스크립트를 실행하기 전에 이 스크립트를 수정하여 모든 사용자의 비밀번호를 바꿀 수 있다. 또는 MySQL root 사용자의 비밀번호를 바꿀 때는 다음과 같이 하면 된다:

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('new_password')
           WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

• MySQL 데몬을 유닉스의 root 사용자로 시작하지 말자. mysqld 는 다른 사용자가 실행할 수 있다. 또한 보안을 더 엄격하게 하기 위해 mysql이라는 유닉스 사용자를 만들 수 있다. 다른 유닉스 사용자로 mysqld를 실행하면 user 테이블에서 root 사용자 이름을 바꿀 필요가 없다. mysqld를 다른 유닉스 사용자가 시작하기 위해 mysql.server 스크립트를 수정하면 된다. 일반적으로 su 명령을 사용한다.

• mysqld를 실행할 수 있는 사용자만이 데이터베이스 디렉토리에 읽기/쓰기 권한을 가지고 있는지 확인.

• 모든 사용자에게 process 권한을 주지 말자. mysqladmin processlist 을 출력하면 현재 실행하는 쿼리의 내용을 볼 수 있다. 그러므로 이러한 명령을 실행할 권한이 있는 사용자는 다른 사용자의 UPDATE user SET password=PASSWORD('no_secure') 질의를 볼 수 있다. MySQL은 process 권한을 가진 사용자를 위해 추가적인(extra) 연결을 저장한다. 그래서 MySQL root 사용자는 모든 일반적인 연결이 사용되었어도 로그인하고 점검을 할 수 있다.

• 모든 사용자에게 file 권한을 주지 말자. 이러한 권한이 있는 사용자는 mysqld 데몬의 권한이 있는 파일 시스템의 어느 곳에라도 파일을 저장할 수 있다. 좀 더 안전하게 하기 위해 SELECT ... INTO OUTFILE로 생성되는 모든 파일은 모든 사용자가 읽기만 할 수 있으며 이미 존재하는 파일을 덮어씌울 수 없다. file 권한은 LOAD DATA INFILE , SELECT .. INTO OUTFILE 문을 이용하여 서버에 파일을 저장하고 읽을 수 있는 권한을 허용한다. 이러한 권한을 가진 사용자는 MySQL 서버가 읽고 쓸 수 있는 파일을 읽고 쓸 수 있는 권한이 허용된다. 일반 사용자에게 이런 권한을 줄 필요는 없다. 필요한 부분만 권한을 주는 것이 좋다.

• DNS 를 신뢰하지 못한다면 승인 테이블에서 호스트이름 대신 IP를 사용하자. 기본적으로 mysqld 의 --secure 옵션은 호스트이름을 안전하게 한다. 어떤 경우 와일드카드 문자가 포함된 호스트이름 값을

사용할 때는 매우 조심해야 한다.

▪ mysql.server 스크립트에서 유닉스 root 사용자의 비밀번호를 넣는다면, 이 스크립트는 오직 root만이 읽을 수 있도록 해야 한다.

다음의 mysqld 옵션은 보안과 관련되어 있다:

--secure : gethostbyname() 시스템 콜에 의해 리턴된 IP 숫자가 원래의 호스트이름을 해석한 것과 같은지를 점검한다. 이것은 어떤 사람이 다른 호스트 이름을 에뮬레이터해서 접근하는 것을 어렵게 만든다. 이 옵션은 또한 호스트이름이 온전한지에 대한 점검을 추가 한다. 해석하는데 때로는 시간이 많이 걸려서 MySQL 3.21에서는 기본적으로 설정이 되어 있지 않다. MySQL 3.22에서는 호스트이름을 캐쉬하고 이 옵션이 기본적으로 설정되어 있다. 함수 gethostbyname()은 호스트 이름을 인자로 받아 그에 해당하는 IP 주소 및 기타 정보를 해당하는 구조체에 담아 그 구조체의 포인터를 리턴하는 함수입니다. 쉽게 말해서 호스트 이름을 넣으면 해당 IP 주소를 찾아주지요.

--skip-grant-tables : 이 옵션을 사용하면 서버가 권한 시스템을 전혀 사용하지 않는다. 그러면 모든 사용자가 모든 데이터베이스에 접속할 수 있다. (mysqladmin reload 를 실행하여 실행중인 서버가 승인 테이블을 사용하도록 할 수 있다.)

--skip-name-resolve : 호스트이름이 해석되지 않는다. 승인 테이블의 모든 Host 컬럼값은 반드시 IP 숫자이거나 로컬호스트이어야 한다.

--skip-networking : 네트웍을 통한 TCP/IP 연결을 허용안함. mysqld와 모든 연결은 유닉스 도메인 소켓을 통해 만들어진다. 이 옵션은 MIT-pthreads를 사용하는 시스템에서는 제대로 작동을 하지 않는다. 왜냐면 MIT-pthreads 패키지는 유닉스 소켓을 지원하지 않기 때문이다. 리눅스를 사용하는 사람들에게는 상관없다. 리눅스는 유닉스 도메인 소켓을 지원하기 때문이다.

11. 유틸리티

mysqlclient 라이브러리를 사용하여 서버와 통신을 하는 모든 MySQL 클라이언트 프로그램은 다음의 환경변수를 사용한다.

환경변수	내 용
MYSQL_UNIX_PORT	기본 소켓; 로컬호스트에서 접속할 때 사용
MYSQL_TCP_PORT	기본 TCP/ip port
MYSQL_PWD	기본 패스워드
MYSQL_DEBUG	디버깅할때 Debug-trace 옵션
TMPDIR	임시 테이블/파일이 생성되는 디렉토리
MYSQL_HISTFILE	이전명령어들을 저장한다.

위의 환경변수 중에서 MYSQL_PWD를 사용하는 것은 보안에 문제가 있을 수 있다.

모든 MySQL 프로그램은 매우 다양한 옵션이 있다. 그러나 모든 MySQL 프로그램에서 --help 옵션을 제공한다. --help 옵션을 이용해 프로그램의 다양한 옵션에 대한 모든 정보를 볼 수 있다. 예를 들어, mysql --help 를 해보면 알 수 있다.

1. mysql

간단한 SQL 쉘이며 GNU readline 호환성 있다. 상호대화식으로도 할 수 있고 일괄적(비대화식)으로도 사용할 수 있다. 대화식으로 사용하는 경우, 질의 결과는 아스키-테이블 형식으로 출력된다. 비대화식으로 사용할 경우, 결과는 탭으로 분리된 형식으로 출력된다. 이러한 출력 형식은 명령행라인 옵션을 이용해서 바꿀 수 있다. 이러한 경우에 다음과 같이 스크립트를 사용할 수 있다.

```
shell> mysql database < script.sql > output.tab
```

클라이언트에서 메모리가 부족해서 문제가 생기면 --quick 옵션을 사용한다. 그러면 질의 결과를 가져오기 위해 mysql_store_result() 대신 mysql_use_result()를 사용한다.

2. mysqlaccess

호스트와 사용자와 데이터베이스의 연결에 대한 접근 권한을 점검하는 스크립트이다.

3. mysqladmin

MySQL의 관리용 유틸리티이다. 데이터베이스의 생성, 제거, reload, 그리고 refresh 등을 한다. 또한 MySQL서버로 부터 상태정보와 진행상태, 정보등을 준다.

4. mysqld

MySQL 의 서버 데몬이다. 이것이 항상 실행되고 있어야만 MySQL 데이터베이스를 사용할 수 있다.

5. mysqldump

MySQL 데이터베이스를 읽어서 파일로 저장한다. SQL 문장 또는 탭으로 분리된 텍스트 파일로 저장한다.

6. mysqlimport

하나 또는 그이상의 텍스트 파일을 개별적인 테이블로 가져올 때 사용한다. LOAD DATA INFILE에 의해 지원되는 모든 형식을 사용할 수 있다.

7. mysqlshow

데이터베이스, 테이블, 컬럼, 그리고 인덱스에 대한 정보를 보여 준다.

8. mysqlbug

이것은 스크립트로서 MySQL 버그 보고서를 사용할 때 항상 사용되어지는 것이다.

9. mysql_install_db

디폴트 특권을 사용하여 테이블의 허가를 생성시키는 스크립트이다. MySQL을 시스템에 처음 설치하였을 경우 MySQL을 실행하기 이전에 실행시키는 것으로서 이것은 설치 후 딱 한번만 실행하는 스크립트이다.

10. isamchk

MySQL 테이블 정보 보기, 점검, 최적화 복구 유틸리티이다. 내용이 많아서 마지막 장에서 설명을 한다.

11. make_binary_release

컴파일된 MySQL의 binary 릴리즈 버전을 만든다. 다른 MySQL 사용자의 편이를 위하여 컴파일된 실행버전을 만들때 사용된다.

12. msql2mysql

shell 스크립트로서 mSQL 프로그램을 MySQL로 변환한다. 모든 것을 변환하지는 않지만 변환하기에 좋은 상태로 만들어준다.

13. replace

msql2mysql에서 사용되는 실행파일이다. 문자열을 파일, 표준입력, 또는 어떤 특정 상태로 변환하는 프

로그램이다. 문자열을 교환할 때도 사용할 수 있다. 예를 들어 다음의 명령어는 파일에서 a와 b를 교체한다.

```
shell> replace a b b a -- file1 file2 ...
```

14. safe_mysqld

안정된 형태로 MySQL 데몬을 시작시킨다. 오류시 재시작 시키고, 실행정보 log를 log파일에 기록한다.

15. mysql 사용방법

mysql은 일반 유저로 실행하여도 되는 프로그램이다.

```
$ mysql -u root mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.22.22

Type 'help' for help.

mysql>
```

위와 같이 연결된다. 여기서 몇가지 간단한 테스트용 명령을 적는다.

```
mysql> show databases
    -> \g
```

Database
mysql
test

2 rows in set (0.00 sec)

```
mysql> show tables from mysql
    -> \g
```

Tables in mysql
columns_priv
db
func
host
tables_priv
user

6 rows in set (0.01 sec)

```
mysql> show columns from db
    -> \g
```

Field	Type	Null	Key	Default	Extra
Host	char(60)		PRI		
Db	char(32)		PRI		
User	char(16)		PRI		
Select_priv	enum('N','Y')			N	
Insert_priv	enum('N','Y')			N	
Update_priv	enum('N','Y')			N	
Delete_priv	enum('N','Y')			N	
Create_priv	enum('N','Y')			N	
Drop_priv	enum('N','Y')			N	
Grant_priv	enum('N','Y')			N	
References_priv	enum('N','Y')			N	
Index_priv	enum('N','Y')			N	
Alter_priv	enum('N','Y')			N	

```
13 rows in set (0,01 sec)

mysql> show index from db
    -> \g
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
db	0	PRIMARY	1	Host	A	2	NULL
db	0	PRIMARY	2	Db	A	2	NULL
db	0	PRIMARY	3	User	A	2	NULL
db	1	User	1	User	A	NULL	NULL

```
4 rows in set (0,00 sec)

mysql> show status
    -> \g
```

Variable_name	Value
Aborted_clients	0
Aborted_connects	0
Created_tmp_tables	0
Delayed_insert_threads	0
Delayed_writes	0
Delayed_errors	0
Flush_commands	1
Handler_delete	0
Handler_read_first	0
Handler_read_key	1
Handler_read_next	0
Handler_read_rnd	10
Handler_update	0
Handler_write	0
Key_blocks_used	0
Key_read_requests	0
Key_reads	0
Key_write_requests	0
Key_writes	0
Max_used_connections	0
Not_flushed_key_blocks	0
Not_flushed_delayed_row	0
Open_tables	6
Open_files	12
Open_streams	0
Opened_tables	10
Questions	19
Running_threads	1
Slow_queries	0
Uptime	4028

30 rows in set (0,00 sec)

```
mysql> show variables
    -> \g
```

Variable_name	Value
back_log	5
connect_timeout	5
basedir	/usr/local/mysql/
datadir	/usr/local/mysql/var/
delayed_insert_limit	100
delayed_insert_timeout	300
delayed_queue_size	1000
join_buffer	131072
flush_time	0
key_buffer	8388600
language	/usr/local/mysql/share/mysql/korean/
log	OFF
log_update	OFF
long_query_time	10
low_priority_updates	OFF
max_allowed_packet	1048576
max_connections	100
max_connect_errors	10
max_delayed_insert_threads	20
max_join_size	4294967295
max_sort_length	1024
net_buffer_length	16384
port	3306
protocol-version	10
record_buffer	131072
skip_locking	ON
socket	/tmp/mysql.sock
sort_buffer	2097144
table_cache	64
thread_stack	65536
tmp_table_size	1048576
tmpdir	/tmp/
version	3.22.22
wait_timeout	28800

30 rows in set (0.00 sec)

```
mysql> select * from db
    -> \g
```

실행결과 확인한다.(내용이 많아서 포함시키기가 힘듭니다.)

```
mysql>
```

위의 명령은 기본적인 명령으로서 실제 mysql 을 사용할 때 자주 사용되는 명령이다.

16. 새로운 데이타 베이스 생성과 사용자 연결

데이타베이스 생성의 방법은 두가지가 있다. mysqladmin 프로그램을 이용하여 root 아이디로 shell 상에서 생성하는 방법과 또 하나 MySQL 프로그램 상에서 생성하는 방법이다.

먼저 mysqladmin 프로그램을 이용하는 방법이다.

```
shell> mysqladmin create seongju
```

admin 인 root 에 암호를 설정한 경우 다음과 같이 실행한다.

```
shell> mysqladmin -u root -p create seongju
```

이때 Enter password: 라는 말이 나오고 자신의 패스워드를 입력한다.

이 경우 디폴트로 설치한 경우 /usr/local/var/seongju 라는 디렉토리가 생성된다. 이 디렉토리가 데이타베이스 공간으로 이용되는 곳이다. 즉 seongju 라는 데이타 베이스 공간에 테이블을 생성할 경우 seongju 라는 디렉토리로 관련된 파일이 생성되며 데이타가 저장된다.

※ MySQL과 주고 받는 각종 내용은 /usr/local/var 상의 완전한도메인명.log 파일상에 모두 저장된다. 에러도 역시 완전한도메인명.err 에 저장된다. 즉 만약 abc.linuxkorea.co.kr에서 접속을 하였다면 /usr/local/var에 abc.linuxkorea.co.kr.err 과 abc.linuxkorea.co.kr.log 이 파일이 생성되는 것이다. 만일 동작 상태를 알고 싶다면 터미날을 열어서 tail -f 완전한도메인명.log 하여 MySQL과의 주고받는 내용을 모니터링 할 수 있다.

두번째 방법으로 MySQL을 이용하는 법이다. MySQL 실행상태에서 아래와 같이 한다. 이때 관리자인 root 아이디로 MySQL에 접속한다.

```
mysql> create database seongju
    -> \g
Query OK, 1 row affected (0.00 sec)

mysql>
```

17. 데이터베이스와 MySQL 관리 테이블과의 연결 및 사용자등록, 권한설정

생성된 데이터베이스와 데이터베이스 관리및 엑세스 권한등을 정의하고 있는 mysql 상의 db, user 테이블에 방금 생성된 seongju 테이블의 내용을 정의 하자.

먼저 이 seongju 데이터베이스는 모든 호스트 상에서 접근이 가능하도록 만들기로 한다. 그리고 사용자는 id를 lee 라고하는 DB user를 생성한다고 하자.

sql 문은 다음과 같다.

```
mysql> insert into db
    -> (host,db,user,select_priv,insert_priv,update_priv,delete_priv,create_priv
    -> drop_priv, grant_priv,references_priv,index_priv,alter_priv)
    -> values ('%','seongju','lee','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y')
    -> \g
Query OK, 1 row affected (0,00 sec)

mysql>
```

위의 sql문에서 보면 lee 이라는 사용자는 seongju 데이터베이스에 대하여 select, insert, update, delete, create, drop, grant, reference, index, alter 권한이 모두 주어졌다. 이것은 하나의 seongju 데이터베이스에 대하여 각 sql 명령에 대한 권한을 개별적으로 줄 수 있다는 것을 의미한다.
select * from db 라는 sql 문을 이용하여 정상적으로 insert 되었는지 확인해 보기 바란다.

자 이제 lee 라는 db 사용자에 대한 등록을 할 차례이다. 다음과 같은 sql 문을 만들어 실행한다.

```
mysql> insert into user
    -> (host,user,password,select_priv,insert_priv,update_priv,delete_priv,create_priv,
    -> drop_priv,reload_priv,shutdown_priv,process_priv,file_priv,grant_priv,
    -> references_priv,index_priv,alter_priv)
    -> values ('localhost','lee',password('lee'),'N','N','N','N','N','N','N','N','N','N',
    -> 'N','N','N','N')
    -> \g
Query OK, 1 row affected (0,00 sec)

mysql>
```

위와 같이 하여도 되고 간단히 아래와 같이 하여도 무방하다.

```
mysql> insert into user
    -> (host,user,password)
    -> values ('localhost','lee',password('lee'))
    -> \g
Query OK, 1 row affected (0,00 sec)

mysql>
```

host, user, password 이외의 칼럼들은 모두 디폴트로 'N' 으로 설정되어 있으므로 위와 같이 하면 된다. 여기서 db 테이블과 user 테이블은 특권상에서 약간의 차이점이 있다. db 테이블 상에서 설정되지 않고 user 테이블 상에만 설정되어진 DB 사용자는 user 테이블 상에서 설정된 권한을 존재하는 모든 데이터베이스에 대하여 인정받는다. 이러한 이유로 일반 사용자인 경우는 user 테이블 상에는 모든 권한을 'N' 로 설정하여야 하고 각 DB user가 사용할 데이터베이스에 대한 권한을 DB 테이블에 다시 정해

주는 것이다. 즉, 데이터베이스가 여러개 존재한다고 가정하자. aaa,bbb, ccc 라는 세개의 데이터베이스가 존재할때, lee 이라는 DB 사용자는 aaa 라는 데이터베이스만 이용하게 하려고 설정하려고 한다고 하자.

이 경우는 user 테이블에는 모든 권한을 'N' 상태로 설정하고 db 테이블에는 aaa 라는 테이블에 대하여 모든 권한을 'Y' 라고 설정하면 이 lee라는 db 사용자는 aaa 라는 데이터베이스에 대하여 사용권한이 주어질 것이다. 잘 이해가 가지 않는다면 몇 번 더 자세히 읽어보고 test 해본다면 쉽게 이해할 수 있을것이다.

※ user 테이블에 사용자를 insert, update 한 경우 MySQL을 꼭 재기동시켜 주거나 권한을 다시 불러들이는 명령을 해주어야 한다.

```
shell> myaqladmin -u root -p reload
mysql> flush priviledges \g
```

이 명령은 user 테이블에서 사용자 정보를 다시 읽어서 MySQL 을 다시 실행시켜준다.

18. 텍스트 파일에서 데이터 가져오기

mysqlimport 는 명령행 인터페이스에서 LOAD DATA INFILE SQL문을 제공한다. 대부분의 옵션은 LOAD DATA INFILE 과 동일하다. 다음과 같이 사용한다

```
shell> mysqlimport [options] filename ...
```

명령행에서 지정한 텍스트 파일에 대하여, mysqlimport는 파일이름에서 확장자를 제거한다. 그리고 파일의 내용을 어떤 테이블에 넣을 것인지 결정하는데 사용한다. 예를 들어 파일이름이 'patient.txt', 'patient.text', 'patient'는 모두 patient라는 테이블 이름으로 입력된다.

mysqlimport 는 다음의 옵션을 지원한다:

-C, --compress : 서버, 클라이언트에서 압축을 지원하면 서버/클라이언트 사이에서 모든 정보를 압축한다

-#, --debug[=option_string] : 프로그램 사용 추적(디버깅용)

-d, --delete : 텍스트 파일에서 입력하기 전에 테이블을 비운다.

--fields-terminated-by=...
--fields-enclosed-by=...
--fields-optionally-enclosed-by=...
--fields-escaped-by=...
--fields-terminated-by=...
: LODA DATA INFILE 에서의 옵션과 동일한 기능을 갖는 옵션

-f, --force : 에러 생략. 텍스트 파일을 위한 테이블이 없으면, 다른 남아있는 파일을 계속 처리한다. 이 옵션이 없는 경우, 테이블이 없으면 빠져나온다

--help : 도움말 출력

-h host_name, --host=host_name : 지정한 호스트의 MySQL 서버에 데이타 입력. 기본값은 localhost

-i, --ignore : --replace 옵션의 정보 참고.

-l, --lock-tables : 텍스트 파일로 처리하기 전에 모든 테이블에 쓰기 락을 건다. 그러면 서버에서 모든 테이블이 동기화될 수 있다.

-L, --local : 클라이언트에서 입력 파일 읽음. 기본적으로, localhost에서 접속하면 텍스트 파일은 서버에 있다고 가정된다.

-pyour_pass, --password[=your_pass] : 서버에 연결할 때 사용하는 비밀번호. '=your_pass'를 지정하지 않으면 터미널에서 비밀번호를 물어봄

-P port_num, --port=port_num : 호스트에 연결할 때 사용하는 TCP/IP 숫자. (localhost가 아닌 호스트에서 접속할 때 사용. 예를 들어 유닉스 소켓을 사용하는 경우)

-r, --replace : --replace 와 --ignore 옵션은 unique key 값의 레코드가 중복되어 있을 경우에 사용된다. --replcae를 명시할 경우, 동일한 unique key 값을 가지고 있는 레코드를 대체한다. --ignore를 명시할 경우, unique key 값이 동일한 레코드는 생략된다. 두 옵션 모두 명시하지 않는다면, 중복되는 키 값이 발견되면 에러를 출력하고 텍스트 파일의 나머지 부분은 생략이 된다.

-s, --silent : 침묵 모드. 에러가 발생했을 때만 출력.

-S /path/to/socket, --socket=/path/to/socket : 로컬호스트(기본 호스트값)에서 접속할 때 사용하는 소켓 파일.

-u user_name, --user=user_name : 서버에 연결할 때 사용하는 mysql 사용자 이름. 기본값은 유닉스 로그인 이름.

-v, --verbose : Verbose 모드. 프로그램의 수행에 대한 상세한 정보 출력.

-V, --version : 버전 정보 출력.

19. 압축된 읽기 전용 테이블 만들기

pack_isam 은 10 라이센스 이상을 구입하거나 extended support 를 받을 때 사용할 수 있는 추가 유틸리티이다. 바이너리로만 배포하므로 특정한 플랫폼에서만 사용 가능하며 압축률은 40% ~ 70% 정도이다. mmap()을 사용하므로 이 함수가 없다면 문제가 된다. 압축하고 나서는 읽기 전용 테이블이 되며

BLOB 칼럼은 압축하지 못한다.

공개적으로 구할 수 있는 MySQL에서는 압축 읽기 전용 테이블을 생성하지는 못하지만 읽는 것은 가능하다.

10. DB접속방법

10.1 PHP 인터페이스

PHP 문법

■ Variable (변수)

▪ 변수는 $로 시작한다.

▪ 변수는 별도로 선언하는 부분이 없으며 (형을 미리 지정하지 않으며) 변수형은 변수가 사용된context에 따라 결정된다.

▪ 변수의 형으로는 기본적으로 integer, double, string type이 있으며 이외에 array와 object type이 있다. 변수 형은gettype(), is_long(), is_double(), is_string(), is_array(), is_object() 등의 함수로 알아낼 수 있다.

▪ 형변환은 C에서와 같은 형식으로 이루어진다. (int) (integer) (real) (double) (float) (string) (array) (object) 등의 casting operator가 있다.

▪ string conversion : 문자열이 숫자로 변환될 때에 문자열이 '.', 'e', 'E'를 포함하고 있으면 double로 그렇지 않으면 integer로 변환이 된다.

▪ variable variable : 변수의 값이 변수의 이름이 되는 것을 말한다. 예를들어 $a = "hello" 일때 $$a = "world"라고 정의하면 $hello = "world"로 정의되는 것이다.

▪ scope : user-defined function에서는 local function scope가 적용된다. 즉 function 안에서 정의된 변수는 function안에서만 의미가 있다. 주의할 점은 외부에서 정의된 변수도 user-defined function안에서는 의미가 없다는 것이다. 외부에서 정의된 변수를 사용하려면 함수 안에서 global $externalVar; 형식으로 정의를 하고 사용하거나 $GLOBALS["externalVar"] 형식으로 직접 변수를 사용하여야 한다.

▪ static variable : local function scope를 가지지만 scope를 벗어나더라도 값이 유지되는 변수를 말한다.

■ Array

▪ array는 $array[] 형식으로 사용된다.

▪ scalar array : 첨자가 숫자로 주어지는 배열이다. 예를 들면 $array[0] = 1;

▪ associative array : 첨자가 숫자가 아니라 문자열로 주어지는 배열이다. 예를 들면 $array["first"] = 1;

▪ 그냥 $array[]에 값을 지정하면 array에 값이 하나 추가된다. $array[] = 1; $array[] = 2; 는 $array[0] = 1; $array[1] = 2;와 같은 의미를 지닌다.

▪ array() 함수를 통하여 array를 만들 수도 있고, count() 함수로 element의 갯수를 얻을 수도 있다. next(), prev() 함수나 each() 함수를 통하여 element들을 참조할 수도 있다.

■ External Variables (외부변수)

▪ HTML form (GET/POST) : form으로 전달된 값은 form에서 name field로 지정한 이름의 변수로 지정이 된다. form에서 type = "image"인 경우에는 마우스로 클릭한 위치가 name_x, name_y 형태로 추가로 전달된다.

▪ HTTP Cookie : browser로 cookie를 전달하려면 SetCookie(name, value, timeout) 함수를 이용한다. client로부터 전달되는 cookie는 PHP 변수로 변환이 된다.

▪ Environment Variable : 환경변수 또한 PHP 변수처럼 사용할 수 있다. 환경변수는 getenv()라는 함수를 이용해 얻을 수 있으며 putenv() 함수로 환경변수를 지정할 수도 있다.

■ Expression

▪ expression이란 값으로 환산되는 것을 말한다.

▪ assignment는 expression이므로 $a = $b = 5; 같은 문장을 쓸 수 있다.

▪ pre and post increment / decrement : $a++, ++$a, $a--, --$a

▪ comparison operator는 boolean 값을 갖는 expression이다.

▪ operator와 assignment의 결합 : $a += 3; $b = ++$a; $c = double(--$b); $d = $c += 10;

▪ boolean : 숫자에서 0은 false, 0이 아닌 값은 true이다. 문자열에서 ""와 "0"은 false, 나머지는 true이다. array에서는 element가 하나도 없으면 false, 하나라도 있으면 true이다.

■ Statement

▪ if 문

```
if (expression) {
        do anything 1;
} elseif (expression) {
        do anything 2;
} else
```

```
        do anything 3;
```

또는

```
if (expression) :
        do anything 1;
elseif (expression) :
        do anything 2;
else :
        do anything 3;
endif;
```

- while 문

```
while (expression)
        do anything;
```

또는

```
while (expression) :
        do anything;
endwhile;
```

- do...while 문

```
do {
        do anything;
} while (expression);
```

- for 문

```
for (expr1; expr2; expr3)
        do anything;
```

- switch 문

```
switch (expression) {
        case ... :
                do anything;
                break;
        default :
                do anything;
}
```

- loop에서 break를 이용한 제어가 가능하다.

■ Function (함수)

▪ 함수는 function 이라는 keyword를 가지고 정의가 되며 별도로 return type은 지정하지 않는다.

▪ function의 기본형은 다음과 같다.

```
function funcName($arg1, $arg2, $arg3, ..., $argn) {
        do anything;
        return $retval;
}
```

▪ return value는 list와 object를 포함하여 어떤 type이든 될 수 있다. 예를 들어 array를 return하려면

```
return array(0, 1, 2);
```

▪ argument는 default로 pass by value이다. pass by reference로 하려면 argument definition에서 변수명 앞에 &를 붙여주면 된다 (function funcName(&$arg1)). function이 pass by value로 정의가 되었더라도 함수를 부를때 argument에 &를 붙여서 넘기면 pass by reference가 된다 (call : doFunc(&$var)).

▪ default parameter : C++에서 사용하는 방식으로 default parameter를 정의할 수 있다

```
function funcName($var = 1) { }
```

■ Operator

▪ operator들은 C언어에서의 operator와 비슷하며 다음과 같은 operator들이 있다.

▪ arithmetic operator : +, -, *, /, %
▪ string operator : . (concatenation)
▪ assignment operator : =
▪ bitwise operator : &, |, ~
▪ logical operator : and (&&), or (||), xor, !
▪ comparison operator : ==, !=, <, >, <=, >=

■ Class

▪ class keyword를 사용하여 class를 정의하며 instance 생성은 new operator를 이용한다.

▪ class의 상속은 extends keyword를 사용한다.

▪ constructor 정의는 class와 같은 이름을 갖는 상수를 정의함으로써 이루어진다. constructor는 default parameter를 가질 수 있다.

예)

```
class testClass {
        var $value;
        function testClass($defValue = "test") { $value = $defValue; }
        function doSet($setValue) { $value = $setvalue; }
        function doPrint() { echo $value; }
}
class testLineClass extends testClass {
        function doPrintLine() { echo("$value\n"); }
}

$test = new testClassLine;
$test->doSet("hello");
$test->doPrint();
$test->doPrintLine();
```

■ 그밖에

▪ require : #include와 똑같은 의미로 사용된다.

▪ include : include 문장을 만날 때마다 지정한 파일을 포함한다. require는 #include처럼 무조건 파일을 포함시키지만 include는 loop나 if 문 등에서 사용할 수 있으며 필요한 경우에만 파일을 포함하도록 할 수 있다.

⑧ 예제 - FILE Upload 와 MySQL BLOB 화일 입출력

인터페이스 예제

간단한 이미지의 입출력이 가능한 Gallery를 만들어 보자. BLOB 란 binary large object 이다.
BLOB 컬럼 타입을 이용하면 일반적인 Int나 char와는 달리 이미지나 바이너리 화일을 테이블에 넣을 수 있다.

BLOB 필드는 소팅이나 INDEX생성은 할 수 없다. MySQL에서 지원하는 BLOB타입은 다음 4가지다.

TINYBLOB (TINYTEXT)
- A BLOB or TEXT column with a maximum length of 255 (2^8 - 1) characters.

BLOB (TEXT)
- A BLOB or TEXT column with a maximum length of 65535 (2^16 - 1) characters.

MEDIUMBLOB (MEDIUMTEXT)
- A BLOB or TEXT column with a maximum length of 16777215 (2^24 - 1) characters.

LONGBLOB(LONGTEXT)
- A BLOB or TEXT column with a maximum length of 4294967295 (2^32 - 1) characters.

필요한 데이터베이스와 테이블은 다음과 같다.

데이터베이스 : test

테이블 정의에 대한 SQL 문 :

```
use test ;
CREATE TABLE gallery1(
        id int NOT NULL auto_increment,
        image blob NOT NULL, # 이미지의 바이너리
        title varchar(100) DEFAULT '' NOT NULL, # 이미지 제목
        width smallint(6) DEFAULT '0' NOT NULL, # 가로크기
        height smallint(6) DEFAULT '0' NOT NULL, # 세로크기
        filesize int , # 파일크기
        detail text , # 이미지 설명
        PRIMARY KEY (id)
) ;
```

너무 큰 그림은 작동이 부자연 스럽고 확장자 검사를 하여 gif, jpg, jpeg만 업로드가 가능합니다.

■ 이미지 출력 부분

```
<?php

//-------------------------------------------------------------------------------
// 이미지 출력하기
//-------------------------------------------------------------------------------
 $connect=mysql_connect( "", "mysql", "") or die( "SQL server에 연결할 수 없습니다.");
 mysql_select_db("test",$connect);

 $que1="select * from gallery where id=$id" ;
 $result=mysql_query($que1,$connect );
 $row=mysql_fetch_array($result);

 Header( "Content-type:  image/jpeg");
 echo $row[image];

?>
```

■ 이미지 저장 부분

```
<HTML>
<HEAD>
<TITLE> PHP3 GALLERY 예제 </TITLE>
</HEAD>
<BODY BGCOLOR="#006699" LINK="#99CCFF" VLINK="#99CCCC" TEXT="#FFFFFF">

<BR><P>
<CENTER><IMG SRC=./gallerytitle.gif width=270 height=48 BORDER=0 ALT="PHP3
GALLERY"></CENTER>


<?php

//-------------------------------------------------------------------------------
// MySQL DB 접속
//-------------------------------------------------------------------------------

 $connect=mysql_connect( "", "mysql", "") or die( "SQL server에 연결할 수 없습니다.");
 mysql_select_db("test",$connect);


//-------------------------------------------------------------------------------
//이미지 저장하기
//-------------------------------------------------------------------------------
if($mode=='insert')
{

 if (!empty($image_name)) //파일이 선택 되었으면
       {
            $name_s = explode(".",$image_name);
             $ext_s=strtolower($name_s[1]);     // 이미지 화일의 확장자 검사
             if($ext_s!=gif && $ext_s!=jpg && $ext_s!=jpeg )
               { echo (" <script>
                       window.alert('이미지 화일은 JPG , GIF 만을 지원합니다. ')
                  history.go(-1)
                  </script>
                "); exit;
            }
                   else
                   {
                $size = GetImageSize($image);
                $width =  $size[0];
```

```
                $height = $size[1];
                $imageblob = addslashes(fread(fopen($image, "r"), filesize($image)));
                $filesize = filesize($image) ;
        }
    }
  else //선택한 파일이 없으면
    {

        echo("<script name=javascript>
                window.alert('이미지를 선택하시고 제목을 입력하세요');
                history.go(-1)
               </script>");exit;
    }



$que1="    INSERT INTO gallery  VALUES ('', '$imageblob','$title', '$width',
                                    '$height','$filesize', '$detail' )" ;


 $result=mysql_query($que1,$connect );

    if($result)
        {
        echo(" <meta http-equiv='Refresh' content='0; URL=$PHP_SELF'>");
        exit;
        }
     else
        {
         echo("<script name=javascript>
                window.alert('입력중 오류가 발생했습니다.');
                history.go(-1)
               </script>");exit;

        }

}

//-------------------------------------------------------------------------
//이미지 저장하기 끝!
//-------------------------------------------------------------------------
```

```
//--------------------------------------------------------------------------------
// 새이미지 선택하기 폼
//--------------------------------------------------------------------------------

if($mode=='new')
{
echo("
<form action='$PHP_SELF' method='POST' enctype='multipart/form-data'>

<INPUT TYPE=hidden name=mode value=insert>
<table>
<tr>
<td>올릴 이미지:</td><td><input type='file' name='image'></td>
</tr>

<tr>
<td>제목</td><td><input type='text' name='title'></td>
</tr>
<tr>

<tr>
<td>설명</td><td><textarea name='detail'></textarea></td>
</tr>


<tr>
<td></td><td><input type='submit' name='submit' value='이미지 전송 시작'></td>
</tr>

</table>
</form>");
}
//--------------------------------------------------------------------------------
// 새이미지 선택하기 폼 끝!
//--------------------------------------------------------------------------------


//--------------------------------------------------------------------------------
// 이미지 출력하기
//--------------------------------------------------------------------------------
if(!$mode)
{
```

```
 $que1="select id,title,width,height ,filesize,detail from  gallery order by id DESC " ;

 $result=mysql_query($que1,$connect );

 $row=mysql_fetch_array($result);

 echo("<table bordr=1 width=90% align=center><tr>
 <td>이미지</td>
 <td>제목</td>
 <td>파일크기</td>
 <td>설명</td>
 </tr>");

 while($row)
  {

   echo   ("<tr><td><img   src=./view.html?id=$row[id]   width=$row[width]   height=$row[height]
></td>
                    <td>$row[title]</td>
                    <td>$row[filesize] byte</td>
                    <td>$row[detail]</td>  </tr>
   ");

   $row=mysql_fetch_array($result);

  }

 echo("</table>");
}


//-------------------------------------------------------------------------------
// 이미지 출력하기 끝!
//-------------------------------------------------------------------------------

if($mode!='new')
{ echo("<CENTER><A HREF='$PHP_SELF?mode=new'>새이미지 올리기</A></CENTER>");
}

?>

</BODY>
</HTML>
```

■ 소스에 대한 설명

몇가지 주요 사항은 다음과 같다.

▪ 화일 업로드를 위해서는 FORM 태그를 사용합니다. 한가지 주의 할점은 파일전송을 위해 반드시

```
enctype='multipart/form-data'
```

부분이 태그 내부에 정의되어 있어야 한다.

▪ 전송할 화일은 <input type='file' name='image'> 를 통해 선택한다.

이때 전송된 화일은 서버상의 /tmp 디렉토리에 임시로 저장되며 임의의 이름이 부여됩니다. 이름은 $image 에 저장되며 실제 사용자 로컬 하드에서 사용되었던 이름은 $image_name 에 저장된다.

▪ 실제로 이미지를 테이블에 저장하는 핵심적인 루틴을 다음이다.

```
$size = GetImageSize($image); // tmp 디렉토리에 올라온 이미지 크기
$width = $size[0];
$height = $size[1];
$imageblob = addslashes(fread(fopen($image, "r"), filesize($image)));
$filesize = filesize($image) ;

$que1=" INSERT INTO gallery VALUES ('', '$imageblob','$title', '$width',
'$height','$filesize', '$detail' )" ;

$result=mysql_query($que1,$connect );
```

* GetImageSize() 함수는 이미지의 가로, 세로 등등을 알려준다.(매뉴얼 참조)

```
$imageblob = addslashes(fread(fopen($image, "r"), filesize($image)));
```

위의 부분이 바로 화일을 읽어 그 크기만큼 DB저장 가능한 BLOB형태로 처리한다.

▪ 이미지를 출력해 주는 부분은 다음이다.

```
<img src=./view.html?id=$row[id] ..
```

실제 view.html 의 내부는 의외로 간단하다. DB에 접속한 후 다음과 같이 처리한다.

```
$que1="select * from gallery where id=$id" ;
$result=mysql_query($que1,$connect );
$row=mysql_fetch_array($result);
Header( "Content-type: image/jpeg");
```

```
echo $row[image];
```

위의 방법으로 BLOB 화일 입출력을 할 수 있다.

실제 UPLOAD 된 화일을 저장하는 방법에는 두 가지가 있다.

▪ 업로드 된 화일을 서버의 특정 디렉토리에 복사하고 화일위치+이름을 테이블에 저장하는 방법으로 여러곳에서 사용이 가능하다고 입출력 루틴이 별도로 필요하지 않다는 장점이 있으나 화일이 수가 많아지면 관리가 힘들고 실제 파일에 변동이 생길 경우 테이블의 내용을 업데이트 해 주는 루틴이 필요하게 됩니다.

▪ 본 강의에서 사용한 방법으로 화일을 직접 DB 에 BLOB 형태로 넣는 겁니다. 별도의 입출력 루틴이 필요하지만 파일의 수가 많아질수록 유리합니다. 상업용 페이지를 운영하는데 좋은 방법일 것 같다.

11. 웹에서의 사용자인증

1. 인증소개

MySQL을 이용한 사용자인증(User Authentication)은 다음과 같이 한다.

서버 측 스크립트에 Header()함수를 사용하면 웹서버에서는 "Authentication Required" 메시지를 클라이언트 브라우저에 보내게 되고 그러면 흔히 보는 사용자인증 대화상자가 뜨게 된다.

여기서 사용자가 아이디, 암호를 넣게 되면 각각 $PHP_AUTH_USER, $PHP_AUTH_PW 변수에 들어가게 되고 이 값을 가지고 MySQL에 SQL문을 날리는 방법으로 인증을 처리하게 된다.

일반적으로 사용되는 아파치의 기본 인증방법(.htaccess, .htpasswd 파일을 이용하는)은 단지 사용자가 특정 디렉토리나 파일에 접근권한을 가지는가의 여부만을 판단하는데 반해 MySQL을 이용하면 사용자별로 다른 권한레벨을 주어 인증 후 각각 다른 페이지로 연결 시켜 줄 수 도 있다는 장점이 있다.

주의 할 점은 사용자 인증 기능은 PHP가 아파치 모듈로 돌아갈 때에만 가능하며 반드시 PHP 스크립트 제일 처음에 나와야 합니다.

주로 사용하는 방법은 auth.inc 파일을 따로 만들어 인증이 필요한 페이지의 시작 부분에

```
<?php
include "auth.inc" ;
?>
```

와 같이 추가 시켜주는 것이다.

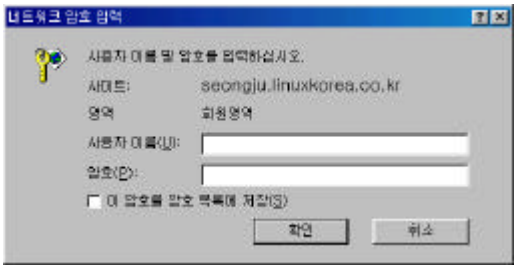
우선 test DB에 다음과 같이 member 라는 사용자 테이블을 만듭니다.

```
use test ;
create table member(
        id char(10) NOT NULL PRIMARY KEY ,
        passwd char(10),
        name char(10),
        level int
);
```

그 다음 각기 다른 권한을 가지는 사용자를 등록해 줍니다.

```
insert into member values('loveme','12345','정식사용자','3');
insert into member values('user1','user1','임시사용자','2');
insert into member values('guest','guest','나그네','1');
```

이렇게 데이터베이스에 인증에 필요한 사용자를 넣어두고 사용자 인증을 위한 페이지를 지정하고 나면 다음과 같은 인증확인 대화상자가 나타난다. 적당한 암호를 넣으면 인증이 이루어져서 다음과정으로 넘어간다.



2. auth.inc 내용

```
<?php

    cfunction authenticate() {
      Header( "WWW-authenticate: basic realm=\"BBS ADMIN 영역\" ");
      Header( "HTTP/1.0 401 Unauthorized");
      $title= "Invalid Login";
       ?>
         아이디와 암호가 필요합니다!
         <?php
       exit;
    }

    if(!isset($PHP_AUTH_USER)) {
        authenticate();
    } else {
        mysql_pconnect( '', 'mysql', '') or die( "Unable to connect to SQL server");  // MySQL 서
버 접속
        mysql_select_db( "test") or die( "Unable to select database");              // DB 선택
        $result = mysql_query( "select name , level from member
                          where id='$PHP_AUTH_USER'
              and passwd='$PHP_AUTH_PW' ");
        if(!mysql_num_rows($result))
            {
             authenticate();
            }
        else
       {
             $userinfo = mysql_fetch_array($result);
```

```
        }
         }
?>
```

3. auth.html (또는 auth.php3)

```
<?php
      include  "./auth.inc" ;
?>

<HTML>
<HEAD>
<TITLE> 사용자 인증 예제 </TITLE>
</HEAD>

<BODY bgcolor="#2B4577" text="white" link="blue" vlink="purple" alink="red">

<?php
$username =  $userinfo[0] ;    // auth.inc 에서 가져온 사용자 이름
$userlevle = $userinfo[1] ;    //          "              사용자 LEVEL

echo( " <script>
            window.alert('어서오세요 $username 님')
         location.href='level$userlevle.html'
        </script>
     ");
?>

</BODY>
</HTML>
```

12. 문제해결

1. 테이블 유지보수 및 파손 복구에 isamchk 사용하기

데이터베이스 테이블의 정보를 얻을 때, 테이블 점검, 복구 및 최적화 할때 isamchk 유틸리티를 사용할 수 있다. 다음의 섹션은 어떻게 isamchk를 사용하는지(옵션에 대한 상세한 설명 포함), 테이블유지계획을 어떻게 설정할 것인지, 어떻게 isamchk의 다양한 기능을 수행하기 위해 isamchk를 사용하는지에 대해 설명하고 있다.

1.1. isamchk 명령어 사용법

isamchk 는 다음과 같이 사용한다:

```
shell> isamchk [options] tbl_name
```

옵션은 isamchk로 무엇을 할 것인지 지정한다. 아래에서 설명한다. (isamchk --help 명령으로 옵션의 목록을 볼 수 있다) 옵션이 없을 때, isamchk는 단지 테이블을 점검한다. 더 많은 정보를 얻으려 하거나 특정한 작업이 필요하면 아래에서 설명하는데로 옵션을 지정한다.

tbl_name은 점검하기 원하는 데이터베이스 테이블이다. 데이터베이스 디렉토리가 아닌 다른 곳에서 isamchk를 실행하면, 파일의 경로를 지정해야 한다. 왜냐하면 isamchk는 데이터베이스의 위치에 대해서 알지 못하기 때문이다. 실제로, isamchk는 작업하려는 파일이 데이터베이스 디렉토리에 있는지 아닌지 신경을 쓰지 않는다; 데이터베이스 테이블에 해당하는 파일을 다른 곳으로 복사하고 그곳에서 복구 작업을 할 수 있다.

원한다명 isamchk의 명령행에서 여러개의 테이블을 사용할 수 있다. 또한 이름을 인덱스 파일 이름('.ISM' 가 붙음)으로 지정할 수 있으며 이런 경우 '*.ISM' 패턴을 사용하여 디렉토리의 모든 테이블을 지정할 수 있다. 예를 들어 데이터베이스 디렉토리에 있다면 다음과 같이 디렉토리의 모든 테이블을 점검할 수 있다:

```
shell> isamchk *.ISM
```

데이터베이스 디렉토리에 있지 않으면, 디렉토리의 경로를 지정하여 모든 테이블을 점검할 수 있다.

```
shell> isamchk /path/to/database_dir/*.ISM
```

또한 MySQL data 디렉토리의 경로를 사용한 와일드 카드를 지정하여 모든 데이터베이스의 모든 테이블을 점검할 수 있다:

```
shell> isamchk /path/to/datadir/*/*.ISM
```

isamchk는 다음의 옵션을 지원한다:

-a, --analyze
Analyze the distribution of keys. This will make some joins in MySQL faster. 키의 분포를 분석. mysql에서 특정한 조인을 빠르게 만든다.

-#, --debug=debug_options
디버그 로그 출력. debug_options 문자는 흔히 'd:t:o,filename' 이다.

-d, --description
테이블의 정보 출력

-e, --extend-check
테이블을 매우 상세하게 점검. 아주 특정한 경우에만 필요하다. 일반적으로 isamchk는 이 옵션이 없어도 모든 에러를 찾을 수 있다.

-f, --force
이전의 오래된 임시 파일을 덮어씀. 테이블을 점검할때 -f를 사용하면(-r 없이 isamchk를 실행) isamchk는 점검하는 동안 에러가 발생하는 테이블에서 자동으로 -r 옵션을 시작한다.

--help
도움말 출력.

-i, --information
점검을 한 테이블의 통계 정보 출력.

-k #, --keys-used=#
-r 과 함께 사용. NISAM 테이블 핸들러에 첫 # 인덱스만 업데이트하라는 것을 알려준다. 높은 수의 인덱스는 해제된다. 이것은 insert를 빠르게 할때 사용한다. 해제된 인덱스는 isamchk -r 을 사용하여 재활성화된다.

-l, --no-symlinks
복구할때 심볼릭 링크를 따르지 않는다. 일반적으로 isamchk는 심볼릭 링크가 가리키는 테이블을 복구한다.

-q, --quick
빠르게 복구하기 위해 -r 과 함께 사용. 일반적으로 원래의 데이타 파일은 건드리지 않는다. 두번째 -q를 지정하여 원래의 데이타 파일을 사용하도록 할 수 있다.

-r, --recover
복구 모드. 유일하지 않는 unique 키만 제외하고 거의 모든 것을 복구한다.

-o, --safe-recover
복구 모드. 구식 복구 방법을 사용; -r을 사용하여 복구하는 것보다 느리다. 그렇지만 -r이 다룰 수 없는 몇가지 경우에 사용할 수 있다.

-O var=option, --set-variable var=option
변수값 설정. 설정가능한 변수는 아래에서 설명.

-s, --silent
침묵 모드. 에러가 발생할 때만 출력을 한다. 두개의 -s(-ss)를 사용하면 isamchk에서 매우 조용하게 작업을 할 수 있다.

-S, --sort-index
인덱스 블럭 정열. 애플리케이션에서 "read-next" 속도를 향상시켜준다.

-R index_num, --sort-records=index_num
인덱스에 따라 레코드를 정렬. 이 작업을 하면 데이타를 집중시킬 수 있고 이 인덱스를 사용한 SELECT 와 ORDER BY 작업의 속도를 증가시킬 수 있다. (처음에는 정렬하는 시간이 매우 느리다!) 테이블의 인덱스 번호를 찾기 위해 SHOW INDEX를 사용한다. SHOW INDEX는 isamchk에서 사용하는 것과 같은 순서로 테이블의 인덱스를 보여준다. 인덱스는 1번부터 시작하여 번호가 매겨진다.

-u, --unpack
pack_isam 으로 압축된 테이블의 압축 해제.

-v, --verbose
Verbose 모드. 정보를 출력. -d 와 -e 와 함께 사용할 수 있다. 여러개의 -v를 사용(-vv, -vvv)하여 더 자세하게 볼 수 있다.

-V, --version
isamchk 버전 출력.

-w, --wait
테이블에 락이 걸려 있으면 기다림.

--set-variable (-O) 옵션의 설정 가능한 변수는 다음과 같다:

keybuffer	default value: 520192
readbuffer	default value: 262136
writebuffer	default value: 262136
sortbuffer	default value: 2097144
sort_key_blocks	default value: 16
decode_bits	default value: 9

1.2. isamchk 메모리 사용법

메모리 할당은 isamchk를 실행할 때 중요하다. isamchk는 -O 옵션에서 지정한 것 이상으로 메모리를 사용하지 않는다. 매우 큰 파일에서 isamchk를 사용하려 하면, 얼마마 많은 메모리를 사용할 것인지 먼저 결정해야 한다. 기본값은 문제를 고치는데 3M를 사용한다. 더 많은 값을 사용해 더 빠르게 isamchk를 사용할 수 있다. 예를 들어 32M 이상 램을 가지고 있다면 다음과 같은 옵션을 사용할 수 있다. (사

용자가 지정한 옵션에 추가하여)

```
shell> isamchk -O sortbuffer=16M -O keybuffer=16M \
           -O readbuffer=1M -O writebuffer=1M ...
```

-O sortbuffer=16M 를 사용하면 대부분의 경우에는 충분하다.

isamchk 는 TMPDIR의 임시 파일을 사용한다는 것에 주의하자. 만약 TMPDIR이 메모리 파일 시스템을 가리킨다면 쉽게 메모리 에러에서 벗어날 수 있다.

1.3. 테이블 유지보수 설정

문제가 생길때를 기다리는 것보다 정기적으로 테이블을 점검하는 게 좋다. 유지보수 계획를 위하여 isamchk -s 를 사용해 테이블을 점검할 수 있다. -s 옵션을 사용하면 isamchk가 침묵 모드로 작동을 하며 에러가 발생했을 때만 메시지를 출력한다.

서버를 시작할때 테이블을 점검하는 것도 좋은 생각이다. 예를 들어 업데이트 도중에 시스템이 리부팅을 했을 때마다 일반적으로 영향을 받은 모든 테이블(이것을 "expected crashed table"이라고 한다)을 점검해야 한다. 만약 오래된 '.pid' (프로세스 ID) 파일이 재부팅후에 남아 있다면 최근 24시간 동안 변경이 된 모든 테이블을 점검하기 위해 safe_mysqld에 isamchk를 실행하는 테스트를 추가해야 한다. ('.pid' 파일은 mysqld가 시작할때 만들어지며 일반적으로 mysqld가 종료될때 제거된다. 시스템이 시작할때 '.pid' 파일이 있다는 것은 mysqld가 비정상적으로 종료되었다는 것을 나타낸다.)

더 좋은 테스트는 최근에 변경된 시간이 '.pid' 파일보다 최근인 테이블을 점검하는 것이다.

또한 일반적인 시스템 운영중에 정기적으로 테이블을 점검할 수 있다. TcX에서는 'crontab' 파일에 다음의 라인을 사용하여 일주일에 한번씩 우리의 중요한 테이블을 점검하도록 cron 작업을 돌린다:

```
35 0 * * 0 /path/to/isamchk -s /path/to/datadir/*/*.ISM
```

이렇게 하면 손상된 테이블에 대한 정보를 출력하여 필요할때 테이블을 점검하고 복구할 수 있다.

몇년동안 우리는 예상하지 못하게 테이블이 손상(하드웨어 문제가 아닌 다른 이유로 문제가 생긴 테이블)된 경우가 없어서 우리에겐 일주일만으로도 충분하다.

우리만큼 MySQL에 대해 신뢰를 할 때까지 최근 24시간동안 업데이트된 모든 테이블에 대해 매일 밤마다 isamchk -s 를 실행할 것을 추천한다.

1.4. 테이블 정보 얻기

테이블에 대한 정보나 통계를 얻기 위해 아래의 명령을 사용하자. 뒤에서 자세하게 정보에 대해 설명할 것이다.

isamchk -d tbl_name
테이블에 대한 정보를 얻기 위해 "describe(설명) 모드"로 isamchk를 실행. MySQL 서버를 --skip-locking 옵션을 사용해 시작하면, isamchk는 서버가 실행되는 동안 업데이트된 테이블에서 에러가 난 것을 보고한다. 그러나 isamchk는 describe 모드에서 테이블을 변경하지 못하기 때문에 데이터를 읽을 위험이 없다.

isamchk -d -v tbl_name
isamchk가 수행하는 것에 대해 더 자세한 정보를 보기위해 -v 옵션을 추가하여 verbose 모드로 수행할 수 있다.

isamchk -eis tbl_name
테이블에서 가장 중요한 정보만 보여준다. 전체 테이블을 다 읽어야 하기 때문에 속도가 느리다.

isamchk -eiv tbl_name
-eiv 와 비슷하지만 현재 무엇이 진행되고 있는지 보여준다.

▪ isamchk -d 출력 예제

```
ISAM file:      company.ISM
Data records:           1403698  Deleted blocks:         0
Recordlength:               226
Record format: Fixed length

table description:
Key Start Len Index   Type
1   2     8   unique  double
2   15    10  multip. text packed stripped
3   219   8   multip. double
4   63    10  multip. text packed stripped
5   167   2   multip. unsigned short
6   177   4   multip. unsigned long
7   155   4   multip. text
8   138   4   multip. unsigned long
9   177   4   multip. unsigned long
    193   1           text
```

▪ isamchk -d -v 출력 예제

```
ISAM file:      company.ISM
Isam-version:  2
Creation time: 1996-08-28 11:44:22
Recover time:  1997-01-12 18:35:29
```

```
Data records:            1403698  Deleted blocks:              0
Datafile: Parts:         1403698  Deleted data:                0
Datafilepointer (bytes):       3  Keyfile pointer (bytes):     3
Max datafile length: 3791650815  Max keyfile length: 4294967294
Recordlength:                226
Record format: Fixed length

table description:
Key Start Len Index    Type                        Root Blocksize Rec/key
1   2     8   unique   double                    15845376      1024        1
2   15    10  multip.  text packed stripped      25062400      1024        2
3   219   8   multip.  double                    40907776      1024       73
4   63    10  multip.  text packed stripped      48097280      1024        5
5   167   2   multip.  unsigned short            55200768      1024     4840
6   177   4   multip.  unsigned long             65145856      1024     1346
7   155   4   multip.  text                      75090944      1024     4995
8   138   4   multip.  unsigned long             85036032      1024       87
9   177   4   multip.  unsigned long             96481280      1024      178
    193   1            text
```

▪ isamchk -eis 출력 예제

```
Checking ISAM file: company.ISM
Key:  1:  Keyblocks used:  97%  Packed:   0%  Max levels:  4
Key:  2:  Keyblocks used:  98%  Packed:  50%  Max levels:  4
Key:  3:  Keyblocks used:  97%  Packed:   0%  Max levels:  4
Key:  4:  Keyblocks used:  99%  Packed:  60%  Max levels:  3
Key:  5:  Keyblocks used:  99%  Packed:   0%  Max levels:  3
Key:  6:  Keyblocks used:  99%  Packed:   0%  Max levels:  3
Key:  7:  Keyblocks used:  99%  Packed:   0%  Max levels:  3
Key:  8:  Keyblocks used:  99%  Packed:   0%  Max levels:  3
Key:  9:  Keyblocks used:  98%  Packed:   0%  Max levels:  4
Total:    Keyblocks used:  98%  Packed:  17%

Records:            1403698     M.recordlength:      226   Packed:              0%
Recordspace used:      100%     Empty space:           0%  Blocks/Record:    1.00
Recordblocks:       1403698     Deleteblocks:          0
Recorddata:       317235748     Deleted data:          0
Lost space:               0     Linkdata:              0

User time 1626.51, System time 232.36
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 627, Swaps 0
```

```
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 639, Involuntary context switches 28966
```

▪ isamchk -eiv 출력 예제

```
Checking ISAM file: company.ISM
Data records: 1403698    Deleted blocks:        0
- check file-size
- check delete-chain
index  1:
index  2:
index  3:
index  4:
index  5:
index  6:
index  7:
index  8:
index  9:
No recordlinks
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 2
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
- check data record references index: 3
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
- check data record references index: 5
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 6
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 7
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 8
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 9
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:   9%  Packed:   17%

- check records and index references
[LOTS OF ROW NUMBERS DELETED]
```

```
Records:            1403698     M.recordlength:      226   Packed:              0%
Recordspace used:       100%    Empty space:           0%  Blocks/Record:    1,00
Recordblocks:       1403698     Deleteblocks:          0
Recorddata:       317235748     Deleted data:          0
Lost space:               0     Linkdata:              0

User time 1639,63, System time 251,61
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 10580, Swaps 0
Blocks in 4 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 10604, Involuntary context switches 122798
```

다음은 앞의 예제에서 사용한 테이블의 데이터와 인덱스 파일 크기이다:

```
-rw-rw-r--   1 monty    tcx      317235748 Jan 12 17:30 company.ISD
-rw-rw-r--   1 davida   tcx       96482304 Jan 12 18:35 company.ISM
```

isamchk가 출력하는 정보 타입에 대한 설명은 아래와 같다. "keyfile"은 인덱스 파일이다. "Record"와 "row"는 같은 말이다.

ISAM file
: ISAM (index) 파일 이름.

Isam-version
: ISAM 포맷 버전. 현재는 항상 2.

Creation time
: 데이타 파일 생성 시간.

Recover time
: 인덱스/데이타 파일이 최근에 복구된 시간.

Data records
: 테이블에 있는 레코드 수.

Deleted blocks
: 지워진 블락이 차지하고 있는 공간. 이러한 공간을 줄이기 위해 테이블의 최적화를 할 수 있다.

Datafile: Parts
: 동적인 레코드 포맷을 위해, 얼마나 많은 데이터 블락이 있는지를 알림.
단편화된 레코드가 없는 최적화된 테이블에서는 Data records 와 같다.

Deleted data

회수하지 않은 지원진 데이타의 바이트 수.
이렇나 공간을 없애기 위해 테이블을 최적화 할 수 있다.

Datafile pointer
데이타 파일 포인터의 크기로 바이트수. 일반적으로 2,3,4,5 바이트이다. 대부분의 테이블은 2바이트로 관리를 한다. 그렇지만 아직까지 mysql에서는 제어를 할 수 없다. 고정 테이블에서 이는 레코드 주소(address)이다. 동적 테이블에서 이는 바이트 주소이다.

Keyfile pointer
인덱스 파일 포인터의 크기로 바이트. 일반적으로 1,2,3 바이트이다. 대부분의 테이블은 2바이트로 관리를 한다. 그렇지만 이 크기는 mysql에서 자동으로 계산이 된다. 항상 블락 주소이다.

Max datafile length
테이블의 데이터 파일(.ISD 파일)의 최대 크기. 바이트.

Max keyfile length
테이블의 key file(.ISM 파일)의 최대 크기. 바이트.

Recordlength
각 레코드가 차지하고 있는 공간. 바이트.

Record format
테이블의 레코드를 저장하는데 사용된 포맷. 위에서 보여준 예제는 고정 길이를 사용하고 있다. 다른 값은 Compressed 와 Packed 이다.

table description
테이블의 모든 키의 목록. 각 키에 대한 자세한 설명은 다음과 같다:

Key
키의 숫자.

Start
레코드에서 인덱스가 시작하는 위치.

Len
인덱스 부분의 길이. packed numbers, 컬럼의 총길이가 되어야 한다.
문자열에서는 인덱스된 컬럼의 총 길이보다 작아야 한다. 왜냐하면 문자열 컬럼 앞에 인덱....

Index
이 인덱스에 같은 값이 여러개 존재할 수 있는지 없는지를 나타냄.

Type
인덱스 부부의 데이터 타입. packed, stripped, empty 옵션을 가진 NISAM 데이타 타입이다.

Root

루트 인덱스 블락의 주소.

Blocksize

각 인덱스 블락의 크기.기본값은 1024이다. 그렇지만 이 값은 컴파일 할때 변경할 수 있다.

Rec/key

최적화기(optimizer)에서 사용하는 통계적인 값. 이 키에 대해서 얼마나 많은 레코드가 있는지를 알려준다. unique 키는 항상 1의 값을 가진다. 이 값은 isamchk -a로 테이블이 로딩된 후에(또는 매우 많이 변경되었을때) 업데이트된다. 전혀 업데이트되지 않으면 기본값으로 30이 주어진다.
위의 첫번째 예제에서, 9번째 키는 두부분을 가진 멀티-파트 키이다.

Keyblocks used

키블락이 사용하고 있는 비율.(%) 예제의 테이블은 isamchk로 재조직화(reorganize)되었기 때문에 값이 매우 높다.(이론적인 최대값에 매우 근접)

Packed

MySQL은 일상적인 접미사를 사용하여 키를 줄인다. 이것은 CHAR/VARCHAR/DECIMAL 키에 사용될 수 있다. 이름과 같은 long 문자열에서 공간 사용을 상당히 줄인다. 위의 네 번째 예제에서 4번째 키는 10 문자 long 이고 60%의 공간이 줄었다.

Max levels

이 키를 위해서 B 트리가 얼마의 깊이를 가지나. 규모가 큰 테이블이 긴 키를 가진 경우에 높은 값이 나온다.

Records

테이블의 레코드수.

M.recordlength

평균 레코드 길이. 고정 길리 레코드의 테이블에서 이 값은 레코드 길이와 같다.

Packed

MySQL은 문자끝의 공백을 제거한다. Packed value는 이렇게 해서 절약된 공간의 비율을 말한다.

Recordspace used

데이타 파일이 사용하는 공간의 비율.

Empty space

데이타 파일이 사용하지 않는 공간의 비율.

Blocks/Record

레코드당 평균 블락수.(즉, 단편화된 레코드가 몇개의 링크로 구성되어 있는지) 고정-포맷 테이블에서는 항상 1이다. 이 값은 가능한한 1에 가깝게 유지해야한다. 이 값이 너무 커지면 isamchk로 테이블을 최적화(reorganize)해야 한다.

Recordblocks
사용하는 블락(링크)수. 고정 포맷에서 이값은 레코드수와 같다.

Deleteblocks
삭제된 블락(링크)수.

Recorddata
데이타 파일이 사용하는 바이트수.

Deleted data
데이터 파일에서 삭제된(사용하지 않는) 바이트수.

Lost space
레코드가 매우 짧은 길이로 업데이트되면 약간의 공간을 잃게 된다. 이 값은 이러한 공간의 bytes 합계이다.

Linkdata
동적 테이블 포맷을 사용할 때,레코드 조각은 포이터로 링크된다.Linkdata는 이런 포인터에서 사용하는 저장 공간의 합이다.

pack_isam으로 테이블을 압축했으면, isamchk -d 는 각 테이블 컬럼에 대한 추가적인 정보를 출력한다.

1.5. 파손 복구에 isamchk 사용하기.

MySQL에서 데이타를 저장하는데 사용하는 파일 포맷은 광범위하게 테스트되었다. 그렇지만 데이터베이스 테이블의 손상될 수 있는 외부적인 상황이 항상 있다:

- 쓰기 도중에 mysqld 프로세스가 죽었을때.
- 예상치 못하게 컴퓨터가 셧다운되었을때(예를 들어, 컴퓨터의 전원이 나가는 경우)
- 하드웨어 에러

MySQL 데이터베이스에서 data의 손상을 체크하고 이에 대처하는 방법에 대해서 설명한다.

손상 복구 작업을 할 때 데이터베이스의 각 테이블 tbl_name은 데이터베이스 디렉토리의 세 파일에 조응한다는 것에 대해서 이해하고 있는 것이 중요하다:

파 일	용 도
'tbl_name.frm'	테이블 정의(형식) 파일
'tbl_name.ISD'	데이타 파일
'tbl_name.ISM'	인덱스 파일

세가지 파일 타입은 다양한 방법으로 손상을 당한다. 그렇지만 대부분의 문제는 데이터 파일과 인덱스 파일에서 생긴다.

isamchk는 '.ISD' (데이타) 파일의 복사복을 만들어 작업을 한다. 이전의 '.ISD' 파일을 제거하고 새로운 파일을 이전의 파일 이름으로 바꾸면서 복구 작업을 마친다. --quick 옵션을 사용하면 isamchk는 임시 '.ISD' 파일을 만들지 않는다. 대신 '.ISD' 파일이 정확하다고 가정하여 '.ISD' 파일은 손대지 않고 새로운 인덱스 파일만 생성한다. isamchk는 자동으로 'ISD' 파일이 손상되었는지 확인하고 손상되었을 경우 복구 작업을 중지하므로 --quick 옵션을 사용하는 것은 안전하다. 두개의 -quick 옵션을 사용할 수 있다. 이런 경우 특정한 에러(중복된 키 등)에서 취소를 하지는 않지만 '.ISD' 파일을 수정하여 문제를 해결하려고 한다.일반적으로 두개의 --quick 옵션을 사용하는 것은 복구작업을 수행하기 위한 디스크 공간이 거의 없을 경우에만 유용하다. 이런 경우 isamchk를 수행하기전에 최소한 백업을 해 놓아야 한다.

1.5.1. 에러가 났을때 테이블 점검 방법

테이블을 점검하기 위해 다음의 명령을 사용한다:

isamchk tbl_name

모든 에러의 99.99%를 발견할 수 있다. 이 경우 발견하지 못하는 것은 데이터 파일과 관련된 손상이다.(일반적으로 거의 생기지 않는다) 테이블을 점검하고자 한다면 일반적으로는 아무런 옵션을 주지 않거나 -s 나 --silent 옵션을 주어 isamchk를 수행하는 것이다.

isamchk -e tbl_name

이 옵션은 모든 데이터를 완전하게 점검한다.(-e 는 "extended check" 를 의미한다) 모든 키가 정확한 레코드를 가리키고 있는지 점검한다. 많은 키를 가진 큰 테이블에서는 시간이 많이 걸린다. isamchk는 일반적으로 첫번째 에러를 발견하면 실행을 멈춘다.
더 많은 정보를 얻고자 한다면, --verbose (-v) 옵션을 추가할 수 있다.
이 옵션을 추가하면 isamchk는 최대 20개의 에러가 있을 때까지 계속 실행을 한다.
일반적으로는 간단한 isamchk (테이블 이름 외에 아무런 인수도 없는)만으로 충분하다.

isamchk -e -i tbl_name

위의 명령과 같다. 그렇지만 -i 옵션을 붙이면 isamchk가 정보의 통계를 출력한다.

1.5.2. 테이블 복구방법

손상된 테이블의 징후는 일반적으로 질의가 갑자기 중지되고 다음과 같은 에러를 낸다:

- 'tbl_name.frm' is locked against change
- Can't find file 'tbl_name.ISM' (Errcode: ###)
- Got error ### from table handler (Error 135 is an exception in this case)
- Unexpected end of file
- Record file is crashed

이런 경우, 테이블을 고쳐야 한다. isamchk는 일반적으로 잘못된 것을 감지하고 대부분을 고친다.

복구 과정은 아래에서 설명하는대로 4단계가 있다. 시작하기 전에 먼저 데이터베이스 디렉토리로 이동하고(cd 명령 이용) 테이블 파일의 퍼미션을 확인해야 한다. mysqld를 실행할 수 잇는 유닉스 사용자가 읽을 수 있는지 확인해야 한다. (또한 작업을 하려는 사용자. 왜냐하면 점검하려는 파일에 접근해야 하기 때문이다) 파일을 수정해야 한다면 파일에 쓰기 권한이 있어야 한다.

■ 1단계 : 테이블 점검

isamchk *.ISM 또는 (충분한 시간이 있다면 isamchk -e *.ISM). 불필요한 정보를 보지 않으려면 -s (silent) 옵션을 사용한다.

isamchk에서 에러가 있다고 알리는 테이블만 고쳐야 한다. 이런 테이블의 경우는 2단계로 넘어간다.

점검하면서 에러를 만났을 때(out of memory 에러 등) 또는 isamchk가 기능을 멈추었을 때 3단계로 넘어간다.

■ 2단계 : 쉽고 안전한 복구

먼저 isamchk -r -r tbl_name을 시도한다. (-r -q는 "빠른 복구 모드"를 의미) 이경우 데이타 파일은 손대지 않고 인덱스 파일 복구를 시도한다. 데이터 파일이 제대로 되어 있고 삭제 링크가 데이타 파일 내의 정확한 위치를 가리키고 있다면, 원활하게 작동을 하고 테이블을 고칠 것이다.

다음 테이블을 고치자. 그게 아니라면 다음 과정을 사용한다:

① 진행하기 전에 데이타 파일의 백업본 만들기
② isamchk -r tbl_name 사용.(-r 는 "복구 모드" 의미) 그러면 데이타 파일에서 정확하지 않은 레코드와 삭제된 레코드를 제거하고 인덱스 파일을 재구성한다.
③ 앞의 과정이 실패하면, isamchk --safe_recover tbl_name을 사용. Safe recovery 모드는 구식 복구 방법을 사용하며 일반적인 복구 모드로 할 수 없는 몇가지 경우에 사용할 수 있다. (그렇지만 더 느리다)

점검하면서 에러를 만났을 때(out of memory 에러 등) 또는 isamchk가 기능을 멈추었을 때 3단계로 넘어간다.

■ 3단계 : 어려운 복구

인덱스 파일의 첫 16k 블락이 파괴되거나 정확하지 않은 정보를 가지고 있을 때, 또는 인덱스 파일이 없는 경우에만 이번 단계까지 온다. 이경우 새로운 인덱스 파일을 만들어야 한다. 다음과 같이 하자:

① 데이타 파일을 안전한 장소로 이동.
② 새로운(빈) 데이타와 인덱스 파일을 만들기 위해 table description 파일을 사용:

```
shell> mysql db_name
mysql> DELETE FROM tbl_name;
mysql> quit
```

③ 이전의 데이타 파일을 새롭게 만든 데이터 파일로 복사. (이전의 데이타 파일을 새로운 파일로 옮기지는 말자; 잘못되었을 경우 복사본을 유지하길 원할 것이다)

2단계로 가자. isamchk -r -q는 이제 제대로 작동을 할 것이다. (무한 루프가 되면 안된다)

▪ 4단계 : 매우 어려운 복구

description 파일 또한 손상을 입었을 경우에만 이번 단계까지 온다. description 파일은 테이블을 만든 이후에 변경이 되지 않기 때문에, 이러한 경우는 결코 생겨서는 안된다.

① 백업본에서 description 파일을 복구해 3단계로 넘어간다. 또한 인덱스 파일을 복구할 수 있고 2단계로 넘어간다. 뒤의 경우 isamchk -r로 시작을 해야 한다.
② 백업본이 없지만 정확히 어떻게 테이블을 만들었는지 알고 있다면, 다른 데이터베이스에 테이블의 복사본을 만든다. 새로운 데이타 파일을 제거하고 다른 데이터베이스의 description과 인덱스 파일을 손상된 데이터베이스로 옮긴다. 이렇게 하면 새로운 description 과 인덱스 파일을 얻을 수 있지만 데이터 파일만 따로 남아있다. 2단계로 가서 인덱스 파일을 재구성한다.

1.5.3. 테이블 최적화

레코드를 삭제하거나 업그레이드 하면서 생긴 단편화된 레코드를 모으고 불필요하고 낭비된 공간을 제거하기 위해 복구 모드로 isamchk를 실행한다:

```
shell> isamchk -r tbl_name
```

SQL OPTIMIZE TABLE 문을 이용하여 같은 방법으로 테이블을 최적화할 수 있다. OPTIMIZE TABLE 은 쉽지만 isamchk가 더 빠르다.

또한 isamchk는 테이블의 성능을 향상시킬 수 있는 몇가지 옵션을 사용할 수 있다:

-S, --sort-index
: high-low 순서로 인덱스 트리 블락을 정열. 검색을 최적화하고 키에 의한 테이블 검색을 빠르게 한다.

-R index_num, --sort-records=index_num
: 인덱스에 따라 레코드를 정열. 데이타를 지역화하고 이 인덱스를 사용하는 SELECT 와

ORDER BY 오퍼레이션의 속도를 향상시킨다. (처음에는 정열을 하는 시간이 매우 느리다!) 테이블의 인덱스 번호를 확인하려면 SHOW INDEX를 사용하면 되며, SHOW INDEX는 isamchk가 인덱스를 검색하는 순서대로 테이블의 인덱스를 보여준다. 인덱스는 1번부터 번호가 매겨진다.

-a, --analyze

테이블에서 키의 분포를 분석. 나중에 테이블에서 레코드를 가져올 때 조인의 성능을 향상시킨다.

2. 일반적인 문제 해결 방법

2.1. 데이터베이스 복사

데이터베이스를 복사하는 가장 일반적인 방법은 업데이트 로그를 이용하는 것이다. 이 경우 마스터로 작동하는 데이터베이스 하나(데이터가 변경된 곳)와 슬레이브로 작동하는 다른 하나의 데이터베이스가 필요하다. 슬레이브를 업데이트하려면 단지 mysql < update_log을 하면 된다. 슬레이드 데이터베이스에 맞는 호스트, 유저, 패스워드 옵션을 지정한다. 그리고 입력값으로 마스터 데이터베이스의 업데이트 로그를 사용한다.

테이블에서 삭제한 것이 없다면, 마지막으로 복사를 한 후 (마지막으로 복사한 시간을 비교) 테이블에서 입력되거나 변경된 열을 찾아내기 위해 TIMESTAMP 컬럼을 사용할 수 있고 미러링되는 데이터베이스에 변경된 자료만 복사를 한다.

삭제에 대한 업데이트 로그와 양쪽의 timestamps를 같이 사용하여 두가지 방법으로 업데이트하는 시스템을 만들 수 있다. 그러나 이런 경우 두 개의 끝에서 변경된 동일한 데이터에서 충돌을 관리할 수 있어야 한다. 아마도 어떤 것이 업데이트되었는지 결정하기 위해 예전 버전을 유지하고 싶을 것이다.

이 경우 복사(복제)는 SQL문으로 이루어지기 때문에, 데이터베이스를 업데이트하는 문장에서 다음의 함수를 사용해서는 안된다; 여기에서는 원본 데이터베이스와 동일한 값을 반환하지 않을 수 있다:

DATABASE()
GET_LOCK() and RELEASE_LOCK()
RAND()
USER(), SYSTEM_USER() or SESSION_USER()
VERSION()

timestamp는 필요한 경우에 미러되는 곳으로 보내기지 때문에 모든 time 함수는 안전하게 사용할 수 있다. LAST_INSERT_ID() 또한 안전하게 사용할 수 있다.

2.2 데이터베이스 백업

MySQL 테이블은 파일로 저장되기 때문에 백업하기가 쉽다. 일관된 백업 작업을 위해 관련된 테이블에 LOCK TABLES를 실행하자. 단지 읽기 락만이 필요하다. 데이터베이스 디렉토리의 파일 복사본을

만드는 동안에도 다른 스레드에서는 테이블에 질의를 계속 할 수 있다. SQL 레벨의 백업을 하고자 한다면 SELECT INTO OUTFILE을 사용할 수 있다.

데이터베이스를 백업하는 다른 방법은 mysqldump 프로그램을 사용하는 것이다:

데이터베이스에 대한 폴 백업 실행:

```
shell> mysqldump --tab=/path/to/some/dir --lock-tables --opt
```

서버에서 업데이트를 하지 않는한 간단하게 모든 테이블 파일(`*.frm', `*.ISD' , `*.ISM' 파일)을 복사할 수 있다. mysqld가 실행되고 있으면 멈추어야 한다. 그러고나서 --log-update 옵션으로 다시 시작하자. ''hostname.n'의 형식을 가진 로그 파일이 생성될 것이다. n은 mysqladmin refresh, mysqladmin flush-logs, the FLUSH LOGS 문, 또는 서버를 재시작할때마다 증가하는 숫자이다. 이렇게 생긴 로그 파일을 이용해 mysqldump를 수행하고 나서 데이터베이스에 변화된 내용을 복사(복제)하는데 필요한 정보를 얻을 수 있다.

복원하고자 한다면, 먼저 isamchk -r을 사용해 테이블을 복구하자. 모든 경우 99.9%가 제대로 수행된다. isamchk가 실패하면 다음의 과정대로 따른다.

① 기존의 mysqldump 백업을 복원한다.
② 업데이트 로그에서 업데이트를 다시 수행하기 위해 다음의 명령을 실행한다:

```
shell> ls -1 -t -r hostname.[0-9]* | xargs cat | mysql
```

ls는 정확한 순서대로 로그 파일을 가져오는데 사용된다.

또한 SELECT * INTO OUTFILE 'file_name' FROM tbl_name을 이용해 선택적인 백업을 할 수 있으며 레코드가 중복되는 것을 방지하기 위해 LOAD DATA INFILE 'file_name' REPLACE ... 을 이용해 복원할 수 있다. 이경우에는 테이블에서 PRIMARY 키나 UNIQUE 키가 필요하다. REPLACE 키워드는 새로운 레코드에서 unique 키 값이 같은 이전의 레코드가 중복되는 경우 이전의 레코드를 새로운 레코드로 교체한다.

13. 최적화

1. MySQL의 최대 성능 향상 방법

1.1. 버퍼 크기 조정

mysqld 서버가 사용하는 기본 버퍼 크기는 다음의 명령으로 알 수 있다.

```
shell> mysqld --help
```

이 명령은 모든 mysqld 옵션의 목록과 설정 변수를 보여준다. 출력되는 내용은 기본값을 포함하고 있으며 다음과 비슷하다.

```
Possible variables for option --set-variable (-O) are:
back_log                 current value: 5
connect_timeout          current value: 5
join_buffer              current value: 131072
key_buffer               current value: 1048540
long_query_time          current value: 10
max_allowed_packet       current value: 1048576
max_connections          current value: 90
max_connect_errors       current value: 10
max_join_size            current value: 4294967295
max_sort_length          current value: 1024
net_buffer_length        current value: 16384
record_buffer            current value: 131072
sort_buffer              current value: 2097116
table_cache              current value: 64
tmp_table_size           current value: 1048576
thread_stack             current value: 131072
wait_timeout             current value: 28800
```

mysqld 서버가 현재 가동중이면 다음의 명령을 통해 실제 변수값을 볼 수 있다.

```
shell> mysqladmin variables
```

각 옵션은 밑에서 설명한다. 버퍼 크기, 길이, 스택 크기는 바이트이다. 'K'(킬로바이트) 나'M'(메가바이트)를 앞에 붙여 값을 지정할 수 있다. 예를 들면 16M는 16 메가바이트를 가리킨다. 대소문자는 구별하지 않는다. 16M 와 16m은 같다.

-back_log
MySQL이 가질 수 있는 최대 연결 요청의 수. 이것은 메인 MySQL스레드가 매우 짧은 시간동안 매우 많은 연결 요청을 받을 때 기능을 한다. 이때 메인 스레드가 연결을 체크하고 새로운 스레드를 시작하

는데는 약간의 시간이 걸린다.(그러나 아주 짧은 시간임) back_log 값은 MySQL이 순간적으로 새로운 요청에 답하는 것을 멈추기전에 이 짧은 시간동안 얼마나 많은 요청을 쌓아두고 있는지를 지정한다. 매우 짧은 시간동안 매우 많은 연결이 예상될때만 이 값을 증가시켜야 한다.

다른 말로 이 값은 tcp/ip 연결을 받는 listen queue의 크기이다. 각 운영체제마다 이러한 큐의 크기에 한계가 있다. Unix system call listen(2) 매뉴얼페이지에 자세한 정보가 있다. back_log값의 한계는 운영체제 문서를 확인해봐라. back_log를 최대값보다 더 높여도 효과가 없다.

-connect_timeout
Bad handshake에 반응하기 전에 연결 패킷을 mysql 서버에서 기다리는 시간.(초)

-join_buffer
(인덱스를 사용하지 않는 조인의) full-join에서 사용하는 버퍼의 크기. 버퍼는 두 테이블 사이에서 각 full-join마다 한번 할당이 된다. 인덱싱을 추가하지 못할 때 조인 버퍼를 증가시키면 full join의 속도를 향상시킬 수 있다. (일반적으로 빠르게 조인을 하는 가장 좋은 방법은인덱스를 추가하는 것이다)

-key_buffer
인덱스 블락은 버퍼링되고 모든 스레드에서 공유한다. 키 버퍼는 인덱스 블락에서 사용하는 버퍼의 크기이다. 인덱스가 많은 테이블에서 delete나 insert 작업을 많이 하면 키 버퍼값을 증가시키는 것이 좋다. 더 빠른 속도를 내려면 LOCK TABLES를 사용하자.

-max_allowed_packet
한 패킷의 최대 크기. 메시지 버퍼는 net_buffer_length 바이트로 초기화되지만 필요하면 최대 허용 패킷 바이트를 증가시킬 수 있다.기본값은 큰 패킷을 잡기에는 작다. 거대 BLOB 컬럼을 사용한다면 값을 증가시켜야 한다. 사용자가 원하는 최대 blob만큼 크게 해야 한다.

-max_connections
동시 클라이언트 숫자. mysqld가 필요로하는 파일 지시자(descriptor)의 숫자만큼 값을 늘려야 한다. 밑에서 파일 디스크립터 제한에 대한 내용을 참고하자.

-max_connect_errors
호스트에서 최대 연결 에러이상의 interrupted 연결이 있으면 더 많은 연결을 위해 호스트는 block화된다. FLUSH HOSTS 명령으로 호스트의 block을 해제할 수 있다.

-max_join_size
최대 조인 크기이상으로 레크도를 읽는 조인을 하면 에러가 난다. 만약 사용자가 where 문을 사용하지 않고 시간이 많이 걸리면서 몇백만개의 레코드를 읽는 조인을 수행하려 하면 이 값을 설정한다.

-max_sort_length
BLOB나 TEXT 값으로 정열할때 사용하는 바이트의 숫자. (각 값중 오직 첫번째 max_sort_length 바이트만 사용된다. 나머지는 무시된다)

-net_buffer_length
질의에서 통신 버퍼가 초기화되는 크기. 일반적으로 바뀌지 않지만 매우 적은 메모리를 가지고 있을 때

예상되는 질의에 맞게 세팅할 수 있다. (이것은 클라이언트에 가는 예상된 sql 문의 길이이다. 질의문이 이 크기를 넘으면 버퍼는 자동으로 max_allowed_packet 바이트까지 증가한다)

-record_buffer
순차적인 검색을 하는 각 스레드에서 각 검색 테이블에 할당하는 버퍼 크기. 순차적인 검색을 많이 하면 이 값을 증가시켜야 한다.

-sort_buffer
정렬이 필요한 각 스레드에서 할당하는 버퍼 크기. order by 나 group by 오퍼레이션을 빠르게 하려면 이 값을 증가시킨다.

-table_cache
모든 스레드에서 열 수 있는 테이블의 숫자. mysqld가 필요로 하는 파일 디스크립터의 숫자만큼 이 값을 증가시켜라. MySQL은 각 유일한 오픈 테이블에서 두개의 파일 디스크립터가 필요하다. 파일 디스크립터 제한을 참고한다.

-tmp_table_size
임시 테이블이 이 값을 넘으면 MySQL은 "The Table tbl_name is full"이라는 에러 메시지를 낸다. 매우 많은 group by 질의를 사용하면 이 값을 증가시켜야 한다.

-thread_stack
각 스레드의 스택 사이즈. creash-me test(**역자주 : 데이터베이스의 벤치마킹을 하는 테스트입니다. 말그대로 데이터베이스를 죽여주지요) 에서 잡히는 많은 제한은 이 값에 달려있다. 기본값은 일반적으로 충분히 크다.

-wait_timeout
연결을 끊기전에 연결 활동(activity)을 서버에서 기다리는 시간(초).

table_cache 와 max_connections는 서버가 열 수 있는 최대 파일 갯수에 영향을 미친다. 이 값을 증가시키면 운영시스템에서 오픈 파일 디스크립터의 per-process 숫자의 한계까지 올릴 수 있다. 그러나 많은 시스템에서 이 한계를 증가시킬수 있다. 이렇게 하려면 각 시스템에서 이 한계를 변화시키는 방법이 매우 다양하므로 운영체제 문서를 참고해야 한다.

table_cache 는 max_connections 와 관계가 있다. 예를 들면 200개의 연결이 있으면 최소 200 * n 의 테이블 캐쉬를 가져야 한다. 여기서 n은 조인에서 테이블의 최대 숫자이다.

MySQL은 매우 유용한 알고리즘을 사용하기 때문에 일반적으로는 매우 적은 메모리로 사용할 수 있으며 메모리가 많을 수록 성능이 더 많이 향상된다.

많은 메모리와 많은 테이블을 가졌고 중간정도 숫자의클라이언트에서 최대의 성능을 원한다면 다음과 같이 사용한다.

```
shell> safe_mysqld -O key_buffer=16M -O table_cache=128 \
           -O sort_buffer=4M -O record_buffer=1M &
```

메모리가 적고 연결이 많으면 다음과 같이 사용한다.

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=100k \
           -O record_buffer=100k &
```

또는:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=16k \
           -O table_cache=32 -O record_buffer=8k -O net_buffer=1K &
```

매우 많은 연결이 있을 때 mysqld가 각 연결마다 최소한의 메모리를 사용하도록 설정하지 않았다면 "swapping problems" 문제가 생길 것이다.

mysqld에서 옵션을 바꾸었으면 그것은 서버의 해당하는 인스턴스에만 영향을 미친다는 것을 기억하자.

옵션을 바꾸었을때의 효과를 보기 위해 다음과 같이 해보자.

```
shell> mysqld -O key_buffer=32m --help
```

마지막에 --help 옵션이 들어간 것을 기억하자. 그렇지 않으면 커맨드 라인에서 사용한 옵션의 효력은 출력에는 반영되지 않을 것이다.

1.2. 메모리 사용 방법(메모리 최적화)

아래에서 설명하는 목록은 mysqld 서버가 메모리를 사용하는 방법에 대해서 나타내고 있다. 메모리 사용과 관련된 서버의 변수 이름이 주어진다.

▪ 키 버퍼(변수 key_buffer)는 모든 스레드에서 공유한다. 서버에서 사용하는 다른 버퍼는 필요한대로 할당이 된다.

▪ 각 연결은 각 스레드마다의 특정한 공간을 사용한다. 스택(64k, 변수 thread_stack) , 연결 버퍼(변수 net_buffer_length), result 버퍼 (변수 net_buffer_length) 등. 연결 버퍼와 result 버퍼는 필요할때 max_allowed_packet 까지 동적으로 증가된다. 질의가 수행될 때 현재의 질의문의 복사문이 또한 할당이 된다.

▪ 모든 스레드는 같은 기본 메모리를 공유한다.

▪ 메모리 맵은 아직 지원이 안된다. 왜냐하면 4GB의 32비트 메모리 공간은 대부분의 대형 테이블에서 충분히 크기가 않기 때문이다. 우리가 64비트 주소 공간을 가진 시스템을 가지게 될 때 우리는 메모리 맵핑을 위한 일반적인 지원을 추가할 것이다.

▪ 테이블에서 순차적인 검색을 하는 각 요청은 read 버퍼에 할당이 된다. (변수 record_buffer)

▪ 모든 조인은 한번에 수행이 되며 대부분의 조인은 임시 테이블을 생성하지 않고 수행이 된다. 대부분의 테이블은 메모리 기반(HEAP) 테이블이다. 거대 길이의 레코드를 가졌거나 BLOB 컬럼을 포함한 임시 테이블은 디스크에 저장이 된다. 현재의 문제는 메모리 기반 테이블이 tmp_table_size를 초과했을 때 "The table tbl_name is full"이라는 에러가 생기는 것이다. 가까운 시일안에 필요할때 자동적으로 메모리 기반(HEAP) 테이블을 디스크 기반(NISAM) 테이블로 바꾸도록 고칠 것이다.
이 문제를 해결하기 위해서 mysqld의 tmp_table_size 옵션을 설정하여 임시 테이블 크기를 늘이거나 클라이언트 프로그램에서 SQL_BIG_TABLES라는 sql 옵션을 설정하여야 한다.
MySQL 3.20에서 임시 테이블의 최대 크기는 record_buffer*16이다. 3.20 버전을 사용하고 있다면 record_buffer의 값을 증가시켜야 한다. 또한 mysqld를 시작할 때 --big-tables 옵션을 사용하여 항상 임시 테이블을 디스크에 저장할 수 있지만 질의 속도에 영향을 미친다.

▪ 정열을 하는 대부분의 요청은 정렬 버퍼와 하나나 두개의 임시 파일을 할당한다.

▪ 대부분의 파징(parsing)과 계산은 지역 메모리에서 이루어진다. 작은 아이템에는 메모리 overhead가 필요없고 일반적인 느린 메모리 할당(slow memory allocation)과 freeing(메모리 해제)는 무시된다. 메모리는 오직 예상치 못한 거대 문자열에서 할당이 된다.(mallloc() 과 free() 사용)

▪ 각 인덱스 파일은 한번에 열리며 각 병행수행되는 스레드에서 데이터 파일은 한번에 열린다. 각 병행수행 스레드마다 테이블 구조, 각 컬럼의 컬럼 구조, 3 * n 의 버퍼 크기가 할당된다. (n은 최대 레코드 길이이며 BLOB 컬럼은 해당하지 않는다) BLOB는 BLOB 데이터의 길이에 5에서 8 바이트를 더한 값을 사용한다.

▪ BLOB 컬럼을 가진 각 테이블에서 버퍼는 거대 BLOB 값을 읽을 수 있도록 동적으로 커진다. 테이블을 검색하면 버퍼는 최대 BLOB의 값만큼 버퍼가 할당이 된다.

▪ 모든 사용중인 테이블의 테이블 핸들러는 캐쉬에 저장되며 FIFO로 관리가 된다. 일반적으로 캐쉬는 64 엔트리를 갖는다. 동시에 두개의 실행 스레드에서 테이블을 사용하면 캐쉬는 테이블의 두 엔트리를 포함한다.

▪ mysqladmin flush-tables 명령은 사용하지 않는 모든 테이블을 닫고 현재 실행되는 스레드가 끝날 때 모든 사용중인 테이블을 닫는다고 표시한다. 이것은 효과적으로 사용중인 메모리를 해제한다.

ps 와 다른 시스템 상황 프로그램은 mysqld가 많은 메모리를 사용하고 있다고 보고할 것이다. 이것은 다른 메모리 주소의 스레드-스택때문에 생긴다. 예를 들면 솔라리스의 ps 는 스택사이의 사용하지 않는 메모리를 사용하는 메모리로 간주한다. 이것은 swap -s를 이용 사용가능한 스왑을 체크하여 확인할수 있다. 우리는 mysqld를 상용 메모리 유출 측정 프로그램으로 테스팅해서 mysqld에는 메모리 유출이 없다.

1.3. 속도 향상에 영향을 미치는 컴파일/링크 방법 (컴파일시 최적화하기)

다음 테스트의 대부분은 리눅스와 MySQL 벤치마크를 가지고 수행되었지만 다른 운영 시스템에도 암시해주는 것이 있다.

static으로 링크를 할때 가장 빠른 실행 속도를 얻을 수 있다. 데이터베이스에 연결하기 위해 TCP/IP보다는 유닉스 소켓을 사용하면 더 좋은 성능을 낼 수 있다.

리눅스에서 pgcc와 -O6을 사용하면 가장 빠르다. 'sql_yacc.cc'를 이 옵션으로 컴파일하려면 gcc/pgcc는 모든 성능을 내기 위해 많은 메모리가 필요하기 때문에 180M의 메모리가 필요하다. 또한 mysql을 설정할때 libstdc++ 라이브러리를 포함하지 않기 위해 CXX=gcc라고 설정해야 한다.

• pgcc를 사용하고 모두다 -O6 옵션으로 컴파일하면 mysqld 서버는 gcc로 컴파일한 것보다 11% 빨라진다.

• 동적으로 링크하면 (-static을 사용하지 않고) 13% 느려진다.

• 유닉스 소켓을 사용하는 것보다 tcp/ip로 연결하는 것이 7.5% 느려진다.

TcX에서 제공한 mysql 리눅스 배포판은 pgcc로 컴파일되었고 정적으로 링크되었다.

1.4. 인덱스의 사용

모든 인덱스(PRIMARY, UNIQUE, INDEX())는 B-trees 에 저장된다. 문자열에서 앞뒤의 공간은 자동적으로 압축된다.

인덱스의 사용 :
• WHERE 문에서 해당하는 레코드 빨리 찾기
• 조인을 수행할때 다른 테이블에서 레코드 가져오기
• 특정 키에서 MAX() 나 MIN() 값 찾기
• 소팅이나 그룹화할때 인덱스 키를 사용하면 테이블을 정열하거나 그룹화한다. 키에 DESC가 붙으면 역순으로 인덱스를 읽는다.
• 어떤 경우에는 데이터 파일에 묻지 않고 값을 가져온다. 어떤 테이블에서 사용하는 모든 컬럼이 숫자이고 특정 키로 형성되어있으면 빠른 속도로 인덱스 트리에서 값을 가져올 수 있다.

다음 예제를 보자.

```
mysql> SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;
```

다중 컬럼 인덱스가 col1 과 col2에 있으면 해당하는 레코드를 직접 가져올 수 있다. 분리된 단일 컬럼 인덱스가 col1 과 col2 에 있으면 최적화기는 어떤 인덱스가 더 적은 레코드를 가졌는지 확인하고 레코드를 가져오기 위해 그 인덱스를 사용하도록 결정한다.

테이블이 다중 컬럼 인덱스를 가졌다면 최적화기가 레코드를 찾는데 어떤 인덱스키를 사용할 수 있다. 예를 들면 세가지 컬럼 인덱스(col1, col2, col3)를 가졌다면 (col1), (col1,col2) (col1,col2,col3) 인덱스를 사용하여 검색을 할 수 있다.

MySQL은 부분인덱스를 사용할 수 없다.

```
mysql> SELECT * FROM tbl_name WHERE col1=val1;
mysql> SELECT * FROM tbl_name WHERE col2=val2;
mysql> SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;
```

인덱스가 (col1,col2,col3)로 있다면 위의 질의중 오직 첫번째 질의만 인덱스를 사용한다. 두번째 및 세번째 질의은 인덱스된 컬럼이 포함되어 있지만 (col2) 와 (col2,col3)는 (col1, col2, col3) 인덱스에 해당하지 않는다.

mysql은 또한 LIKE의 인수가 와일드카드 문자로 시작하지 않는 상수 문자열일이라면 LIKE 비교문에서 인덱스를 사용한다. 예를 들어 다음의 SELECT 문은 인덱스를 사용한다.

```
mysql> select * from tbl_name where key_col LIKE "Patrick%";
mysql> select * from tbl_name where key_col LIKE "Pat%_ck%";
```

첫번째 문장에서는 "Patrick" <= key_col < "Patricl" 을 가진 레코드만 고려된다. 두번째 문장에서는 "Pat" <= key_col < "Pau" 을 가진 레코드만 고려된다.

다음의 SELECT 문은 인덱스를 사용하지 않는다:

```
mysql> select * from tbl_name where key_col LIKE "%Patrick%";
mysql> select * from tbl_name where key_col LIKE other_col;
```

첫번째 문장에서 LIKE 값은 와일드카드 문자로 시작하고 있다. 두번째 문장에서는 LIKE 값이 상수가 아니다.

1.5. WHERE 문에서 최적화하기

일반적으로 느린 SELECT ... WHERE 문을 빠르게 하려면 가장 먼저 확인해야 할 것이 인덱스 추가 문제이다. 다른 테이블사이에서 모든 레퍼런스(references 참조)는 일반적으로 인덱스에 의해 수행된다. SELECT 문에서 어떤 인덱스를 사용하는지 결정하기 위해 EXPLAIN 명령을 사용할 수 있다.

MySQL에서 수행하는 최적화는 다음과 같다.

- 불필요한 삽입어 제거

```
   ((a AND b) AND c OR (((a AND b) AND (c AND d))))
-> (a AND b ANDc) OR (a AND b AND c AND d)
```

- 상수 폴딩(folding)

```
   (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```

▪ 상수 조건 제거(상수 폴딩때문에 필요)

(B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6

▪ 인덱스에서 사용되는 상수 표현은 한번에 계산된다.

▪ WHERE 절이 없는 단일 테이블의 COUNT(*)는 테이블 정보에서 직접 값을 가져온다. 단일 테이블에서 사용된 NOT NULL 표현도 이와 같이 수행된다.

▪ 유효하지 않은 상수 표현은 미리 제거된다. mysql은 불가능하고 해당하는 레코드가 없는 SELECT 문을 빠르게 감지한다.

▪ GROUP BY 나 그룹 펑션(COUNT(), MIN() ...)을 사용하지 않으면 HAVING은 WHERE 에 합쳐진다. (HAVING 절에서는 인덱스를 사용하지 못함. 그러므로 가능한 HAVING절을 사용하지 않는게 속도면에서 좋다)

▪ 각 서브 조인에서 빠르게 WHERE 문을 계산하고 가능한한 레코드를 제외하도록 간소하게 WHERE 문이 만들어진다.

▪ MySQL은 일반적으로 최소한의 레코드를 찾기 위해 인덱스를 사용한다. =, >, >=, <, <=, BETWEEN 그리고 'something%' 처럼 앞이 와일드카드로 시작하지 않는 LIKE 문등을 사용하여 비교를 할 때 인덱스를 사용한다. (LIKE를 사용할 때 와일드카드로 시작하는 LIKE 문을 사용하면 인덱스를 사용하지 않는다. 일정한 단어로만 시작하는 컬럼에서 자료를 찾을 때 유용할 것이다.)

▪ 인덱스의 사용 관련

다음의 WHERE 문은 인덱스를 사용한다.:

```
... WHERE index_part1=1 AND index_part2=2
... WHERE index=1 OR A=10 AND index=2      /* index = 1 OR index = 2 */
... WHERE index_part1='hello' AND index_part_3=5
          /* optimized like "index_part1='hello'" */
```

다음의 WHERE 문은 인덱스를 사용하지 않는다.:

```
... WHERE index_part2=1 AND index_part3=2  /* index_part_1 is not used */
... WHERE index=1 OR A=10                   /* No index */
... WHERE index_part1=1 OR index_part2=10  /* No index spans all rows */
```

▪ 질의에서 다른 테이블보다 모든 상수 테이블을 먼저 읽는다. 상수 테이블은 다음과 같다.

① 빈 테이블이나 1개의 레코드만 있는 테이블

② WHERE 문에서 UNIQUE 인덱스나 PRIMARY KEY 를 사용하고 모든 인덱스는 상수 표현으로 된 테이블

다음의 테이블은 상수 테이블로 사용된다.

```
mysql> SELECT * FROM t WHERE primary_key=1;
mysql> SELECT * FROM t1,t2
           WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

▪ 모든 가능성을 시도하여 테이블을 조인하는데 가장 좋은 조인 조합을 찾는다. (ORDER BY나 GROUP BY의 모든 컬럼이 동일한 테이블에서 나오면 조인을 할때 이 테이블이 먼저 선택된다)

▪ ORDER BY 문과 다른 GROUP BY 문이 있을 때, 또는 ORDER BY 나 GROUP BY가 조인 큐의 첫번째 테이블이 아닌 다른 테이블의 컬럼을 포함하고 있으면 임사 테이블을 만든다.

▪ 각 테이블 인덱스를 찾고 레코드의 30%미만을 사용하는 (best) 인덱스가 사용된다. 그런 인덱스가 없으면 빠른 테이블 검색이 사용된다.

▪ 어떤 경우에는 MySQL은 데이터 파일을 조회하지 않고 인덱스에서 레코드를 읽을 수 있다. 인덱스에서 사용한 모든 컬럼이 숫자라면 질의를 처리하는데 단지 인덱스 트리만을 사용한다.

▪ 각 레코드가 출력되기 전에 HAVING 절에 맞지 않는 레코드는 건너뛴다.

다음은 매우 빠른 질의의 예이다:

```
mysql> SELECT COUNT(*) FROM tbl_name;
mysql> SELECT MIN(key_part1),MAX(key_part1) FROM tbl_name;
mysql> SELECT MAX(key_part2) FROM tbl_name
           WHERE key_part_1=constant;
mysql> SELECT ... FROM tbl_name
           ORDER BY key_part1,key_part2,... LIMIT 10;
mysql> SELECT ... FROM tbl_name
           ORDER BY key_part1 DESC,key_part2 DESC,... LIMIT 10;
```

다음의 쿼리는 인덱스 트리만을 사용하여 값을 구한다. (인덱스 컬럼은 숫자라고 가정)

```
mysql> SELECT key_part1,key_part2 FROM tbl_name WHERE key_part1=val;
mysql> SELECT COUNT(*) FROM tbl_name
           WHERE key_part1=val1 and key_part2=val2;
mysql> SELECT key_part2 FROM tbl_name GROUP BY key_part1;
```

다음의 질의는 개별적인 정열을 하지 않고 정열된 순서대로 열을 가져오는 데 인덱스를 사용한다:

```
mysql> SELECT ... FROM tbl_name ORDER BY key_part1,key_part2,...
```

mysql> SELECT ... FROM tbl_name ORDER BY key_part1 DESC,key_part2 DESC,...

1.6. 테이블 열고 닫는 방법

open 테이블의 캐쉬는 table_cache의 최대값까지 커질 수 있다. (기본값 64 ; 이 값은 mysqld에서 -0 table_cache=# 으로 바꿀 수 있다) 캐쉬가 꽉 찼을때, 그리고 다른 스레드가 테이블을 열려고 할 때, 또는 mysqladmin refresh 나 mysqladmin flush-tables를 사용할때를 제외하고는 테이블은 결코 닫히지 않는다.

테이블 캐쉬가 꽉 차면 서버는 캐쉬 엔트리를 사용하도록 조절하기 위해 다음의 절차를 사용한다.

① 가장 먼저 사용했던 순서대로 현재 사용하지 않는 테이블을 닫는다.
② 캐쉬가 꽉 찼고 어떤 테이블도 닫히지 않지만 새로운 테이블을 열어야 한다면 캐쉬가 필요한 만큼 임시적으로 확장된다.
③ 캐쉬가 임시적으로 확장된 상태이고 테이블을 사용할 수 없는 상황으로 가면 테이블을 닫고 캐쉬를 해제한다.

테이블은 각 동시병행적인 접근때마다 열린다. 동일한 테이블에 접근하는 두개의 스레드가 있거나 같은 질의에서 테이블에 두번 접근하면(with AS) 테이블을 두번 열여야 한다는 의미이다. 테이블의 첫번째 개방은 두개의 파일 디스크립터를 가진다. ; 추가적인 테이블의 개방은 하나의 파일 디스크립터를 가질 뿐이다. 처음에 개방에 사용하는 추가적은 파일 디스크립터는 인덱스 파일에 사용된다. ; 이 디스크립터는 모든 스레드에서 공유된다.

1.6.1. 데이터베이스에서 많은 수의 테이블을 만들때의 단점

디렉토리에 많은 파일이 있다면 open, close 그리고 create 오퍼레이션은 느려질 것이다. 서로 다른 많은 테이블에서 SELECT 문을 수행하면 테이블 캐쉬가 꽉 찰 때 약간의 overhead가 있을 것이다. 왜냐면 개방된 테이블이 있다면 다른 테이블은 닫혀야 하기 때문이다. 테이블 캐쉬를 크게 해서 이러한 오우버헤드를 줄일 수 있다.

1.7. 많은 테이블을 여는 이유

mysqladmin status 를 실행할 때 다음과 같이 나올 것이다:

Uptime: 426 Running threads: 1 Questions: 11082 Reloads: 1 Open tables: 12

단지 6테이블을 사용했는데 이러한 결과에 이상하게 생각할 것이다.

MySQL은 멀티스레드를 사용한다. 그래서 동시에 같은 테이블에서 많은 질의를 할 수 있다. 같은 파일에 대하여 다른 상황을 가지는 두개의 스레드에 대한 문제를 줄이기 위해 테이블은 각 동시병행적인 스레드마다 독립적으로 개방된다. 이것은 테이타 파일에서 약간의 메모리와 하나의 추가적인 파일 디스크립터를 사용한다. 모든 스레드에서 인덱스 파일은 공유된다.

1.8. 데이터베이스와 테이블에서 심볼릭 링크 사용

데이터베이스 디렉토리에서 테이블과 데이터베이스를 다른 위치로 옮기고 새로운 위치로 심볼릭 링크를 사용할 수 있다. 이렇게 하는 것을 원할 경우가 있다. 예를 들면 데이터베이스를 더 여유공간이 많은 파일시스템으로 옮기는 경우 등.

MySQL에서 테이블이 심볼링 링크되었다는 것을 감지하면 심볼링 링크가 가리키는 테이블을 대신 사용할 수 있다. realpath() call 을 지원하는 모든 시스템에서 작동한다. (최소한 리눅스와 솔라리스는 realpath()를 지원한다) realpath()를 지원하지 않는 시스템에서 동시에 실제 경로와 심볼릭 링크된 경로에 접근하면 안된다. 이런 경우에는 업데이트 된후에 테이블이 모순될 수 있다.

MySQL은 기본값으로 데이터베이스 링크를 지원하지 않는다. 데이터베이스간에 심볼릭 링크를 사용하지 않는 작동을 잘 할 것이다. mysql 데이터 디렉토리에 db1 데이터베이스가 있고 db1을 가리키는 db2 심볼릭 링크를 만들었다고 해보자:

```
shell> cd /path/to/datadir
shell> ln -s db1 db2
```

이제 db1에 tbl_a라는 테이블이 있다면 db2에도 tbl_a가 나타날 것이다. 한 스레드가 db1.tbl_a를 업데이트하고 다른 스레드가 db2.tbl_a를 업데이트하면 문제가 생길 것이다.

정말로 이 기능이 필요하면 , 'mysys/mf_format.c'에서 다음의 코드를 수정해야 한다.:

```
if (!lstat(to,&stat_buff))  /* Check if it's a symbolic link */
    if (S_ISLNK(stat_buff.st_mode) && realpath(to,buff))
```

위 코드를 다음과 같이 수정한다 :

```
if (realpath(to,buff))
```

1.9. 테이블에 락 거는 방법

MySQL의 모든 락은 deadlock-free 이다. 언제나 질의를 시작할때 한번에 모든 필요한 락을 요청하고 언제나 같은 순서대로 테이블에 락을 걸어 관리한다.

WRITE 락을 사용하는 방법은 다음과 같다:

- 테이블에 락이 없으면 그 테이블에 write 락을 건다.
- 이런 경우가 아니라면 write 락 큐에 락을 요청한다.

READ 락을 사용하는 방법은 다음과 같다:

▪ 테이블에 write 락이 없으면 그 테이블에 read 락을 건다.
▪ 이런 경우가 아니라면 read 락 큐에 락을 요청한다.

락이 해제되었을 때 락은 write 락 큐의 스레드에서 사용할 수 있으며 그러고 나서 read 락 큐의 스레드에서 사용한다.

테이블에서 업데이트를 많이 하면 SELECT 문은 더 이상 업데이트가 없을 때까지 기다린다는 것을 의미한다.

이러한 문제를 해결하기 위해 테이블에서 INSERT 와 SELECT 오퍼레이션을 많이 사용하는 경우에 다음과 같이 하면 된다. 임시 테이블에 레코드를 입력하고 한번에 임시 테이블에서 실제 테이블로 레코드를 업데이트한다.

다음의 예를 보자:

```
mysql> LOCK TABLES real_table WRITE, insert_table WRITE;
mysql> insert into real_table select * from insert_table;
mysql> delete from insert_table;
mysql> UNLOCK TABLES;
```

만약 어떤 경우에 SELECT문에 우선권을 주고 싶다면 INSERT 옵션에서 LOW_PRIORITY 또는 HIGH_PRIORITY 옵션을 사용할 수 있다. (LOW_PRIORITY를 지정하면 클라이언트에서 테이블을 읽지 않을 때까지 INSERT 문 수행이 미루어진다.)

단일 큐를 사용하기 위해 'mysys/thr_lock.c' 의 락킹 코드를 바꿀 수 있다. 이런 경우 write 락과 read 락은 같은 우선권을 가지며 어떤 애플리케이션에서는 유용할 수 있다.

1.10. 테이블을 빠르고 작게 배열하는 방법 (테이블 최적화)

다음은 테이블에서 최대의 성능을 내는 방법과 저장 공간을 절약할 수 있는 테크닉이다

▪ 가능한한 NOT NULL로 컬럼을 선언한다. 속도가 빨라지며 각 컬럼마다 1 비트를 절약할 수 있다.
▪ default 값을 가질 때 유리하다. 입력되는 값이 기본값과 다를 때만 확실하게 값이 입력된다. INSERT 문에서 첫번째 TIMESTAMP 컬럼이나 AUTO-INCREAMENT 컬럼의 값을 입력할 필요가 없다.
▪ 가능한한 테이블을 작게 만드려면 더 작은 integer 타입을 사용하자. 예를 들면 MEDIUMINT 가 보통 INT 보다 좋다.
▪ 가변 길이 컬럼이 없다면(VARCHAR, TEXT or BLOB columns), 고정 길이 레코드 포맷이 사용된다. 이 경우 속도는 더 빠르지만 불행히도(흑흑~) 낭비되는 공간이 더 많다.
▪ MySQL이 질의를 효과적으로 최적화하기 위해 많은 양의 데이터를 입력한후 isamchk --analyze를 실행하자. 이렇게 하면 동일한 값을 가진 줄의 평균 숫자를 가리키는 각 인덱스의 값을 업데이트한다. (물론 unique 인덱스에서는 항상 1이다)
▪ 인덱스와 인덱스에 따른 데이타를 정열하려면 isamchk --sort-index --sort-records=1 을 사용하자. 인덱스에 따라 정렬된 모든 레코드를 읽기 위해 unique 인덱스를 가졌다면 이렇게 하는 것이 속도를 빠

르게 하는 가장 좋은 방법이다.
▪ INSERT 문에서 가능한 다중 값 목록을 사용하자. 개별적인 SELECT 문보다 훨씬 빠르다. 데이타를 테이블에 입력할 때 LOAD DATA INFILE을 사용하자. 많은 INSERT 문을 사용하는 것보다 보통 20배 빠르다.

많은 인덱스를 가진 테이블에 데이타를 입력할때 다음의 과정을 사용하면 속도를 향상시킬 수 있다.

① MySQL이나 Perl 에서 CREATE TABLE로 테이블을 만든다.
② mysqladmin flush-tables 실행. (열린 테이블을 모두 닫음)
③ isamchk --keys-used=0 /path/to/db/tbl_name 사용. 테이블에서 모든 인덱스 사용을 제거한다.
④ LOAD DATA INFILE 를 이용 테이블에 데이타를 입력.
⑤ pack_isam을 가지고 있고 테이블을 압축하기 원하면 pack_isam을 실행.
⑥ isamchk -r -q /path/to/db/tbl_name 를 이용 인덱스를 다시 생성.
⑦ mysqladmin flush-tables 실행.

▪ LODA DATA INFILE 과 INSERT 문에서 더 빠른 속도를 내려면 키 버퍼를 증가시킨다. mysqld나 safe_mysqld에서 -O key_buffer=# 옵션을 사용하면 된다. 예를 들어 16M는 풍부한 램을 가졌다면 훌륭한 값이다.
▪ 다른 프로그램을 사용하여 데이타를 텍스트 파일로 덤프할때 SELECT ... INTO OUTFILE 을 사용하자.
▪ 연속으로 다량의 insert와 update를 할 때 LOCK TABLE을 사용하여 테이블에 락을 걸면 속도를 향상시킬 수 있다. LOAD DATA INFILE 그리고 SELECT ...INTO OUTFILE 는 원자적이기 때문에 LOCK TABLE을 사용하면 안된다.

테이블이 얼마나 단편화되었는지 점검하려면 '.ISM' 파일에서 isamchk -evi 를 실행한다.

1.11. INSERT 문에서 속도에 영향을 미치는 부분 (insert 최적화)

insert 하는 시간은 다음와 같이 구성된다:

Connect: (3)
Sending query to server: (2)
Parsing query: (2)
Inserting record: (1 x size of record)
Inserting indexes: (1 x indexes)
Close: (1)

(숫자)는 비례적인 시간이다. 이것은 테이블을 개방할때 초기의 overhead를 고려하고 있지는 않다. (매 동시병행적으로 수행되는 질의마다 발생)

테이블의 크기는 N log N(B-trees)에 따라 인덱스의 입력이 느려진다.

테이블에 락을 걸거나 insert 문에서 다중 값 목록을 사용하여 입력 속도를 빠르게 할 수 있다. 다중 값 목록을 사용하면 단일 insert 보다 5배 정도 속도가 빨라진다.

```
mysql> LOCK TABLES a WRITE;
mysql> INSERT INTO a VALUES (1,23),(2,34),(4,33);
mysql> INSERT INTO a VALUES (8,26),(6,29);
mysql> UNLOCK TABLES;
```

주요한 속도 차이는 모든 INSERT 문이 완료되고 난 후에 한번에 인덱스 버퍼가 쓰여기지 때문에 생긴다. 보통 서로 다른 여러 INSERT 문이 있으면 많은 인덱스 버퍼 플러쉬가 있을 것이다. 모든 줄을 단일 문으로 입력하면 락은 필요없다.

락킹은 또한 다중 연결 테스트의 총 시간을 줄일 수는 있다. 그러나 어떤 스레드에서는 총 대기시간은 증가할 수 있다. (왜냐면 락을 기다리기 때문이다)

예를 들어보자:

```
thread 1 does 1000 inserts
thread 2, 3, and 4 does 1 insert
thread 5 does 1000 inserts
```

락을 사용하지 않으면 2, ,3 4는 1과 5 전에 끝마칠 것이다. 락을 사용하면 2,3,4는 아마도 1이나 5 전에 끝나지 않을 것이다. 그러나 총 시간은 40% 빨라진다.

INSERT, UPDATE, DELETE 오퍼레이션은 mysql에서 매우 빠르다. 그렇기 때문에 줄에서 5개 이상의 insert나 update를 할 때 락을 추가하면 더 좋은 성능을 얻을 수 있다. 줄에 매우 많은 자료를 입력한다면 다른 스레드에서 테이블에 접근하도록 하기 위해 때때로(각 1000줄마다) UNLOCK TABLES를 사용하는 LOCK TABLES 실행하면 된다. 이렇게 하면 좋은 성능을 낼 수 있다. (열심히 입력을 하고 중간에 락을 풀었다가 다시 락을 거는 것 반복함)

물론 LOAD DATA INFILE 이 더 빠르다.

1.12. DELETE 문에서 속도에 영향을 미치는 부분 (DELETE 문 최적화)

레코드를 삭제하는 시간은 정확히 인덱스 숫자에 비례한다. 레코드를 빠르게 지우기 위해 인덱스 캐쉬의 크기를 증가시킬 수 있다. 기본 인덱스 캐쉬는 1M 이다; 빠르게 삭제하기 위해 증가되어야 한다.(충분한 메모리를 가지고 있다면 16M로 하자)

1.13. MySQL에서 최대 속도를 얻는 방법

벤치마킹을 시작한다. MySQL 벤치마크 스위트에서 어떤 프로그램을 사용할 수 있다. (일반적으로 'sql-bench' 디렉토리에 있음) 그리고 자신의 상황에 맞게 수정하자. 이렇게 해보면 문제를 해결할 수 있는 다른 해결책을 찾을 수 있으며 가장 빠른 해결책을 테스트할 수 있다.

- mysqld를 적절한 옵션으로 시작하자. 메모리가 많을수록 속도가 빠르다.

▪ SELECT 문의 속도를 빠르게 하기 위해 인덱스를 만들자.
▪ 가능한 효율적으로 컬럼 타입을 최적화하자. 예를 들면 가능한 NOT NULL로 컬럼을 정의하자.
▪ --skip-locking 옵션은 SQL 요청에서 파일 락킹을 없앤다. 속도가 빨라지지만 다음의 과정을 따라야 한다:
① isamchk로 테이블을 체크하거나 수리하기 전에 mysqladmin flush-tables 로 모든 테이블을 플러시 해야 한다. (isamchk -d tbl_name은 언제나 허용된다. 왜냐하면 이건 단순히 테이블의 정보를 보여주기 때문이다)
② 동시에 뜬 두개의 mysql 서버가 동일한 테이블을 업데이트하려 한다면 동일한 데이터 파일에 두개의 mysql 서버를 띄우면 안된다.

--skip-locking 옵션은 MIT-pthreads로 컴파일할때 기본값이다. 왜냐면 모든 플랫폼의 MIT-pthreads에서 flock()가 완전하게 지원이 되지 않기 때문이다.

▪ 업데이트에 문제가 있다면 업데이트를 미루고 나중에 하자. 많은 업데이트를 하는 것이 한번에 하나를 업데이트하는 것보다 더 빠르다.
▪ FreeBSD 시스템에서 MIT-pthreads에 문제가 있으면 FreeBSD 3.0 이후 버전으로 업데이트 하는것이 좋다. 이렇게 하면 유닉스 소켓을 사용하는 것이 가능하며(FreBSD에서 유닉스 소켓이 MIT-pthreads에서 TCP/IP 연결을 사용하는 것보다 빠르다) 그리고 스레드 패키지가 조정되어야 한다.
▪ 테이블이나 컬럼 단계를 체크하는 GRANT는 성능을 떨어뜨린다.

1.14. 로우 포맷과 다른 점은 무엇인가? 언제 VARCHAR/CHAR을 사용해야 하는가?

MySQL은 실제의 SQL VARCHAR 타입이 없다. 그대신 MySQL은 레코드를 저장하고 이것을 VARCHAR로 에뮬레이트하는데 세가지 방법이 있다.

테이블에 VARCHAR, BLOB, TEXT 컬럼이 없으면 고정 row size를 사용한다. 그외에는 동적 row size를 사용한다. CHAR 과 VARCHAR 컬럼은 애플리케이션의 관점에서 동일하게 취급된다; 둘다 trailing space는 컬럼을 가져올때 제거된다.

isamchk -d 를 이용 테이블에서 사용하는 포맷을 체크할 수 있다.
(-d 는 "테이블 묘사"를 의미)

MySQL은 세가지 다른 테이블 포맷을 가지고 있다; 고정길이, 다이나믹, 압축이 그것이다.

▪ 고정 길이 테이블
▪ 기본 포맷. 테이블에 VARCHAR, BLOB, TEXT 컬럼이 없을 때 사용.
▪ 모든 CHAR, NUMERIC, DECIMAL 컬럼은 컬럼 길이에 공백이 제거된다.
▪ 매우 빠름
▪ 캐쉬하기 쉽다
▪ 손상 후 복구가 쉽다. 왜냐면 고정된 위치에 레코드가 위치하기 때문이다.
▪ 많은 양의 레코드가 지워졌거나 운영 시스템에서 자유 공간을 늘리길 원치 않는다면 (isamchk를 이용) 재조직화할 필요없다.
▪ 보통 다이나믹 테이블보다 많은 디스크 공간을 필요로 한다.

■ 다이나믹 테이블

▪ 테이블이 VARCHAR, BLOB, TEXT 컬럼을 포함하고 있을 때 사용.

▪ 모든 문자열 컬럼은 다이나믹하다. (4보다 작은 길이를 가진 문자열 제외)

▪ 컬럼이 문자열 컬럼에서 비었거나 ('') 숫자형 컬럼에서 0(NULL 값을 가진 컬럼과 동일한 것이 아니다) 을 나타내는 비트맵이 모든 레코드 앞에 선행된다. 문자열 컬럼에서 trailing space를 제거한 후 zero의 길이를 가지거나 숫자형 컬럼이 zero의 값을 가지면 비트 맵으로 표시되고 디스크에 저장되지 않는다. 비지 않은 문자는 문자내용에 길이 바이트만큼 추가되어 저장된다.

▪ 보통 고정 길이 테이블보다 디스크 공간 절약.

▪ 줄의 길이를 확장하는 정보를 가지고 줄을 업데이트하면 줄은 단편화될 것이다. 이런 경우 더 좋은 성능을 위해 때때로 isamchk -r 을 실행해야 한다. 통계적으로 isamchk -ei tbl_name을 사용하자.

▪ 손상후 복구가 어렵다. 왜냐면 레코드가 많은 조각드로 단편화되고 링크(단편)가 없어지기 때문이다.

▪ 다이나믹 사이즈 테이블의 예상되는 열 길이 :

```
3
+ (number of columns + 7) / 8
+ (number of char columns)
+ packed size of numeric columns
+ length of strings
+ (number of NULL columns + 7) / 8
```

각 링크마다 6 바이트가 더 있다. 다이나믹 레코드는 업데이트로 레코드가 늘어날때마다 링크된다. 각 새로운 링크는 최소 20바이트일 것이며, 그래서 다음의 확장은 아마도 동일한 링크로 될 것이다. 그게 아니라면 다른 링크가 있을 것이다. isamchk -ed 로 얼마나 많은 링크가 있는지 체크할 수 있다. 모든 링크는 isamchk -r 로 제거할 수 있다.

■ 압축 테이블

▪ 읽기 전용 테이블은 pack_isam 유틸리티로 만들 수 있다. 확장 MySQL 이메일 지원을 구입한 모든 고객은 내부적인 용도로 pack_isam을 사용할 권리가 주어진다.

▪ 압축해제 코드는 모든 mysql 배포판에 있으므로 pack_isam이 없는 고객도 pack_isam으로 압축된 테이블을 읽을 수 있다. (테이블이 같은 플랫폼에서 압축되어 있는한)

▪ 매우 적은 디스크 용량을 사용.

▪ 각 레코드는 개별적으로 압축이 된다.(매우 적은 액세스 overhead) 레코드의 헤더는 테이블의 가장 큰 레코드에 따라 (1-3 바이트) 고정된다. 각 컬럼은 다르게 압축이 된다. 압축 타입은 다음과 같다:

- 일반적으로 각 컬럼마다 다른 Huffman 테이블이다.
- Suffic 공간 압축
- Prefix 공간 압축
- 0 값을 가진 숫자는 1비트로 저장.
- integer 컬럼의 값이 작은 범위를 가졌다면, 컬럼은 최대한 작은 타입으로 저장

된다. 예를 들면 BIGINT 컬럼은 모든 값이 0부터 255라면 TINIINT 컬럼(1바이트)로 저장된다.

- 컬럼이 몇가지 가능한 값으로만 구성되어 있다면, 컬럼 타입은 ENUM으로 변환된다.
- 컬럼은 위 압축 방법을 조합하여 사용한다.

- 고정 길이나 다이나믹 길이의 테이블을 다룰 수 있다. 그러나 BLOB나 TEXT 컬럼은 다룰 수 없다.
- isamchk로 압축을 해제할 수 있다.

MySQL은 다른 인덱스 타입을 지원한다. 그러나 일반적인 타입은 NISAM이다. 이것은 B-tree 인덱스이며 모든 키의 값을 합하여 (키 길이+4)*0.67로 인덱스 파일의 크기를 대강 계산할 수 있다. (이것은 모든 키가 정렬된 순서로 입력된 가장 나쁜 경우이다)

문자열 인덱스는 공간이 압축된다. 첫번째 인덱스 부분이 문자열이라면, prefix가 압축된다. 문자열 컬럼이 다량의 trailing space를 가졌거나 언제나 완전한 길이를 사용하지 않는 VARCHAR 컬럼일 때 space 압축은 인덱스 파일을 더 작게 만든다. prefix 압축은 많은 문자열에 동일한 prefix가 있을 때 유용하다.

2. MySQL 벤치마크 테스트

MySQL 벤치마크 스위트(그리고 crash-me)에 대한 기술적인 설명은 다음 장소에서 볼 수 있고 현재는 배포판에 'bench' 디렉토리에서 코드와 결과를 볼 수 있다.

http://www.mysql.com/crash-me-choose.htmy

이것은 사용자에게 주어진 SQL 수행이 제대로 수행되는지 아닌지를 알려주는 벤치마크이다.

crash-me 는 실제로 질의를 수행하여 데이터베이스에서 지원하는 기능과 능력, 제한사항 등을 측정하는 프로그램이다. 예를 들어 다음의 사항을 측정한다:

- 지원하는 컬럼 타입
- 지원하는 인덱스 숫자
- 지원하는 펑션
- 질의의 최대 크기
- VARCHAR 컬럼의 최대 크기