www.sheepsqueezers.com

Introduction to MySQL

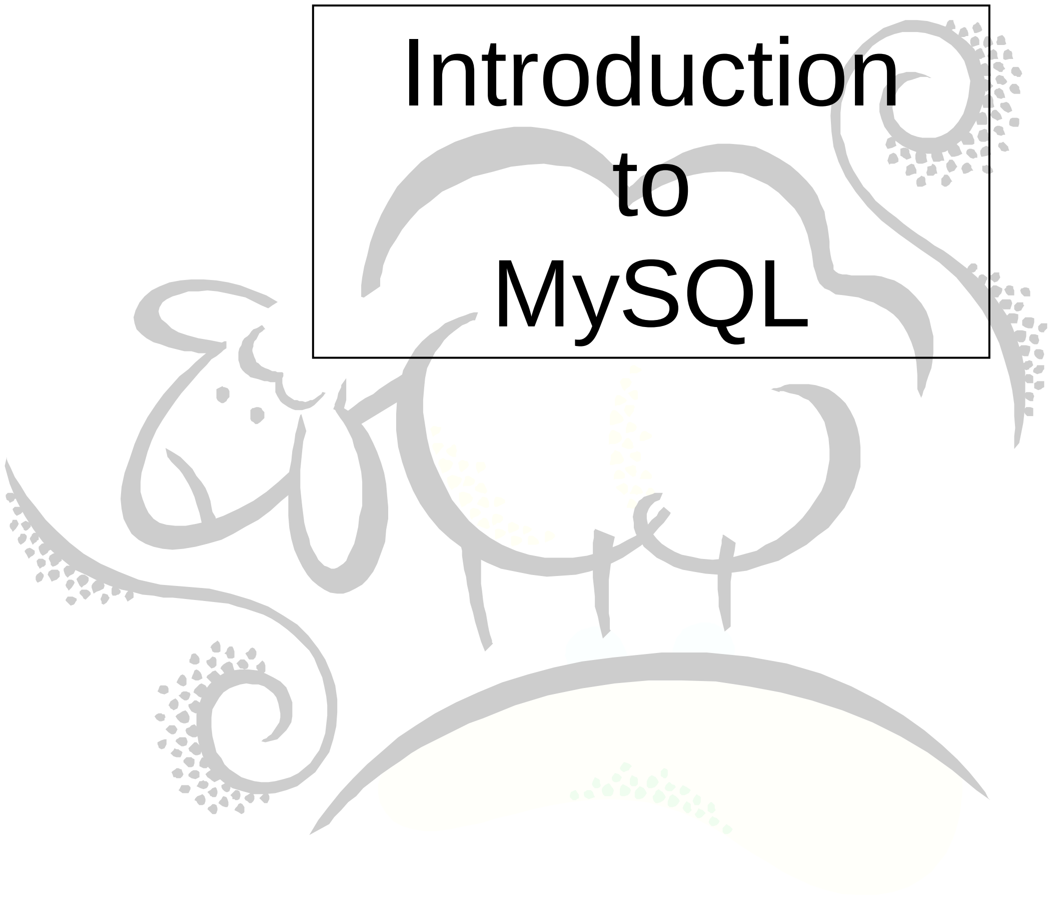
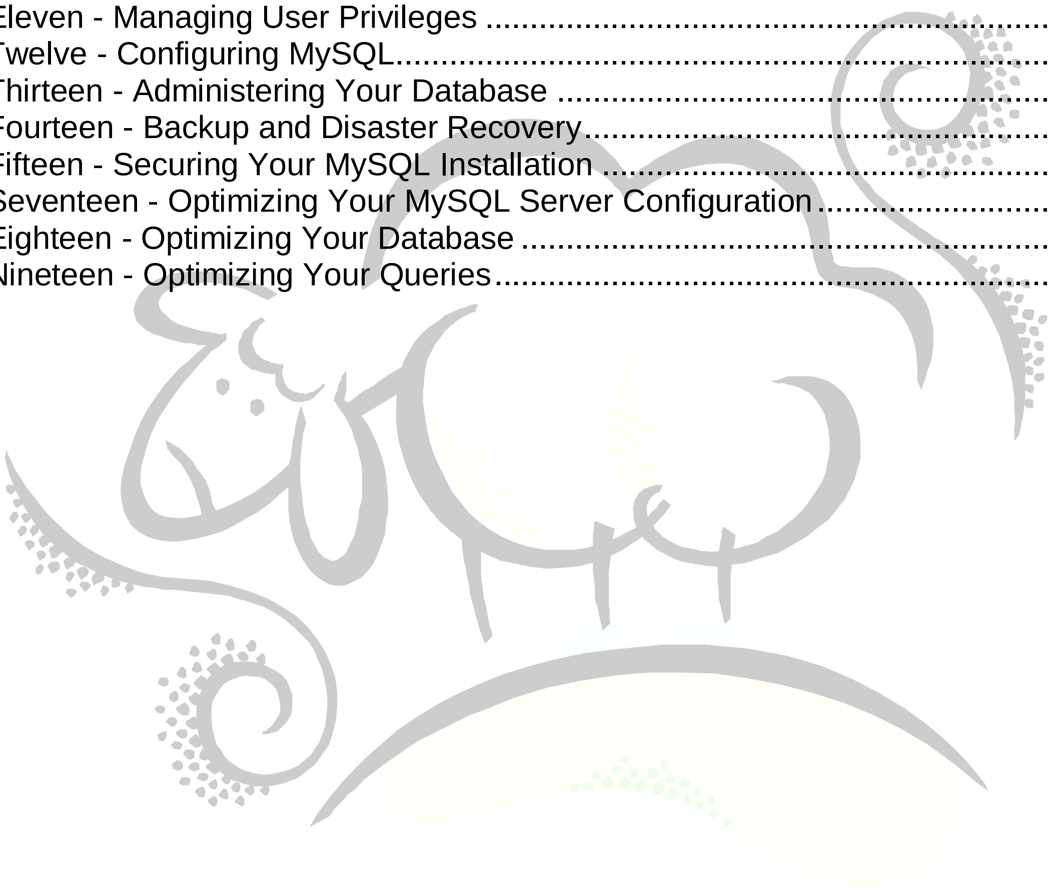
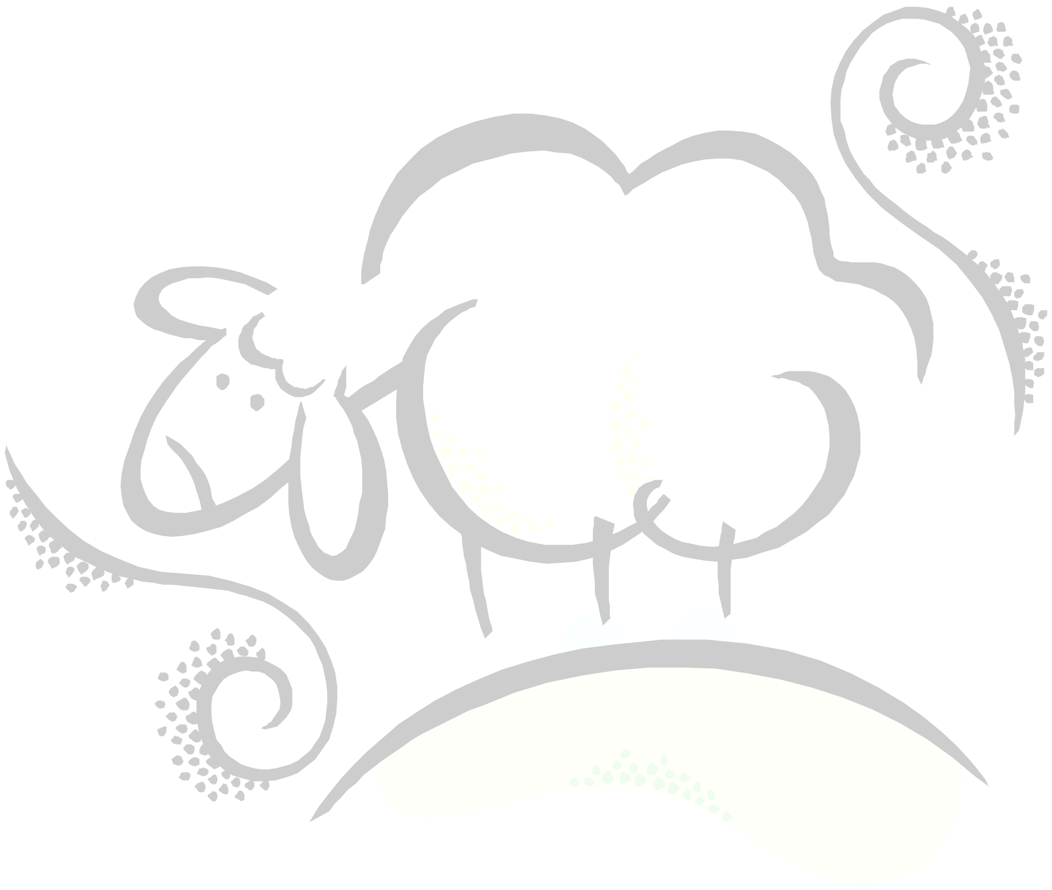


Table of Contents



Introduction

This short document is a brief introduction to the MySQL database. The text herein was gleaned from the book *MySQL Tutorial* by Luke Welling and Laura Thomson (ISBN: 0-672-32584-5) and are effectively my notes from the book. No copyright infringement is intended at all and, yes, I typed in everything you see here by hand…no copy-and-paste was used since a PDF was not available for this book at the time this document written (around 2006)!



Chapter One - Installing MySQL

```
1. Get mySQL at www.mysql.com/downloads/index.html.
2. rpm -i MySQL-server-VERSIOn.i386.rpm MySQL-client-VERSON.i386.rpm
3. Look in /etc/my.cnf for the global options file.  (Also, can have one for
    each user: ~/.my.cnf)
4. In my.cnf, set buffer pool size to 50-80% of memory:

      set-variable = innodb_buffer_pool_size=70M
      set-variable = innodb_additional_mem_pool_size=10M
      innodb_data_file_path=ibdata1:10M:autoextend
      set-variable = innodb_log_file_size=20M
      set-variable = innodb_log_buffer_size=8M
      innodb_flush_log_at_trx_commit=1

5. Login: mysql -u root
6. Start server manually: mysqld --standalone
7. Set root password:  set password for root@localhost=password('mypwd')
8. Delete anonymous accounts:

                     use mysql;
                     delete from user where User='';
                     delete from db where User='';
                     flush privileges;

9. Create an acct:

   grant create, create temporary tables, delete, execute, index, insert,
         lock tables,select,show databases, update
   on *.*
   to username identified by 'password';

10. Note that you may need to issue these next two lines to allow the use to log in:
         set password for ‘user’@’ip-address’=OLD_PASSWORD(‘password’);
         grant all on *.* to user@localhost identified by ‘password’;
```

Chapter Two - Quick Tour

1. The default location of the MySQL files is /usr/local/mysql

2. Executables in /bin:

   a. mysqladmin - used to perform many administrative functions
   b. myisamchk - used to check and repair damaged MyISAM tables
   c. mysqldump - used to backup your databases
   d. mysqlbinlog - used to read the contents of the binary log, essential for disaster recovery.
   e. mysqlshow - used to get info about databases and tables

3. User Interfaces: mysql monitor, MySQL Control Center, and phpMyAdmin

4. Get MySQL Control Center from www.mysql.com/downloads/mysqlcc.html and phpMyAdmin from: www.phpmyadmin.net

5. mysql Monitor, login:

```
mysql --i-am-a-dummy -h hostname -u username -p
```

   can replace --i-am-a-dummy with --safe-updates

6. To see all databases in mysql: show databases;

7. To switch to a particular database: use database_name;

8. To see all tables in the database (after use issued): show tables;
9. To see the table definition: describe table_name;

10. To run a file of commands from within a mysql monitor session:

```
source filename;
```

11. Like 10, but not logged in: mysql -u username -p < filename

12. Can login like this:

```
mysql -A -uUSERNAME -pPASSWORD DATABASE_NAME
```

1. Normalization:

   a. **First Normal Form (1NF):** Each attribute or column value must be atomic. That is, each attribute must contain a single value, not a set of values or another database row.

   b. **Second Normal Form (2NF):** Must be 1NF plus if all attributes that are not part of the primary key are fully functionally dependent on the primary key. That is, each non-key attribute must be functionally dependent on all parts of the key. If the primary key is made up of multiple columns, each other attribute in the table must be dependent on the combination of these columns.

   c. **Third Normal Form (3NF):** Must be 2NF plus attributes must be dependent on nothing but the key.

Chapter Four - Creating Databases, Tables and Indexes

Note: Treat all indentifies as case sensitive (Linux is case sensitive).

1. All identifiers can be up to 64 characters long, aliases can be up to 255 characters long.

2. To create a database:

```
              create database database_name_here;
```

3. To create tables and indexes, first must: use database_name_here;

4. To create tables:

```
  create table table_name (table_definition) [type=table_type];
```

Example:

```
drop database if exists employee;
create database employee;
use employee;
create table department (
 deptID int not null auto_increment primary key,
 name varchar(30)
) type=InnoDB;
create table employee (
 empID int not null auto_increment primary key,
 name varchar(80),
 job varchar(30),
 deptID int not null references department(deptID)
) type=InnoDB;
create table employeeSkills (
 empID int not null references employee(empID),
 skill varchar(15) not null,
 primary key (empID,skill)
) type=InnoDB;
```

5. Create table statment:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table_name [(create_definition,...)
 TYPE=type_name]
 [table_options] [select_stmt];

--or--

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table_name LIKE old_table_name;

create_definition:
 column_name data_type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT]
     [PRIMARY KEY] [reference_definition]

                              --or--

                              PRIMARY KEY (index_col_name,...)

                              --or--

                              KEY [index_name] (index_col_name,...)
```

```
                          --or--

      INDEX [index_name] (index_col_name,...)

      --or--

      UNIQUE [INDEX] [index_name] (index_col_name,...)

      --or--

      FULLTEXT [INDEX] [index_name] (index_col_name,...)

      --or--

      [CONSTRAINT    symbol]    FOREIGN    KEY    [index_name]    (index_col_name,...)
[reference_definition]

      --or--

      CHECK (expr)


     table_options:
      AUTO_INCREMENT=#
       sets the initial auto_increment value to something other than 1
      AVG_ROW_LENGTH=#
       allows you to estimate what you think will be the average row length in order
       to aid the storage engine
      CHECKSUM=1
       allows you to turn on checksum calculation for the rows in the table that may
       help you find the problem if the table becomes corrupt.  Set to 1 to turn on,
       off is default.  ONLY WORKS WITH MYISAM TABLES!
      COMMENT="string"
       comment about the table
      MAX_ROWS=#
       set that max number of rows that will be stored in the table
      MIN_ROWS=#
       set that min number of rows that will be stored in the table
      PACK_KEYS={0|1|DEFAULT}
       MySQL packs (compresses) strings in keys (CHAR,VARCHAR,TEXT).  If you set this
       value to 1, all keys will be packed; set to 0, none will be packed.
      PASSWORD="string"
       does nothing
      DELAY_KEY_WRITE = {0|1}
       Allows you to delay key updates until after the table is closed.  MyISAM table
       types only!
      ROW_FORMAT={default|dynamic|fixed|compressed}
       Allows you to specify a storage format for rows in a MyISAM table.
      RAID_TYPE={ 1 | STRIPED | RAID0} RAID_CHUNKS=# RAID_CHUNKSIZE=#
       Allows you to specify your RAID configuration for optimization purposes.
      UNION=(table_name,...)
       For MERGE table types, allows you to specify the tables to add to the merge.
      INSERT_METHOD={NO | FIRST | LAST}
       Merge tables only: don't insert additional data, insert data in first table of
        the merge, insert data in the last table of the merge.
      DATA DIRECTORY="absolute path to directory"
       Can specify where you want to the data to be stored.
      INDEXC DIRECTORY="absolute path to directory"
       Can specify where you want to the indexes to be stored.
```

a. TEMPORARY: table deleted after session closed
b. Can specify type_name as:
   i. MyISAM - very fast and supports full-text indexing
   ii. ISAM - older table type similar to MyISAM. DO NOT USE!!
   iii. InnoDB - Fully ACID-compliant storage engine that supports transactions, foreign keys and row-level locking.
   iv. BDB (Berkeley DB) - Storage engines that supports transactions and page-level locking.
   v. HEAP - table is stored completely in memory and never written to disk so they are fast but limited in size and are unrecoverable if the computer crashes.
   vi. MERGE - allows you to combine a set of MyISAM tables with the same structure so that they can be queried as if they were one table.

6. Column and Data Types
   a. Numerical Types:
      Note: Can be followed by the keywords UNSIGNED and/or ZEROFILL
      Note: NUMERIC AND DECIMAL (or DEC) are the same
      i. TINYINT - 1 byte up to 2^8 possible values
      ii. SMALLINT - 2 bytes up to 2^16 possible values
      iii. MEDIUMINT - 3 bytes up to 2^24 possible values
      iv. INT - 4 bytes up to 2^32 possible values
      v. BIGINT - 8 bytes up to 2^64 possible values
      vi. FLOAT - single-precision floating-point number. It can represent a positive number between 1.18E-38 to 3.40E38 and similar for negative numbers.
      vii. DOUBLE - double-precision floating-point number. Can also use DOUBLE and REAL and DOUBLE PRECISION. 2.23E-308 to 1.80E308, similar for negative numbers. Also, NUMERIC(w,p) and DECIMAL(w,p) but are represented exactly.
   b. String Types:
      i. CHAR(n) - fixed length string with max length of 255.
      ii. VARCHAR(n) - variable length string from 0 to 255.
      iii. TINYTEXT or TINYBLOB - hols up to 255 characters or bytes
      iv. TEXT or BLOB can hold up to 65,535 characters or bytes.
      v. MEDIUMTEXT or MEDIUMBLOB - up to 16MB
      vi. LONGTEXT or LONGBLOB - up to 4GB.
   c. ENUM('a','b',....) - allows you to list possible values. Includes NULL and error.
   d. SET - similar to ENUM.
   e. Dates and Times:
      i. DATE - stores a date. Expecting YEAR-MONTH-DAY order and are displayed as YYYY-MM-DD
      ii. TIME - stores a time and are displayed ias HH:MM:SS
      iii. DATETIME - combination of DATE and TIME: YYYY-MM-DD HH:MM:SS
      iv. TIMESTAMP - It stores the time that row was inserted or last changed.
      v. YEAR(2) or YEAR(4) - stores a year. YEAR(4) is the default and YEAR(2) stored from 1970 to 2069.

7. Creating indexes:
   Ex: create index idx_name on employee(name);
   a. Can include UNIQUE after CREATE to enforce a uniqueness constraint.
   b. Can include FULLTEXT after CREATE if we want to create a full-text index on a MyISAM table.

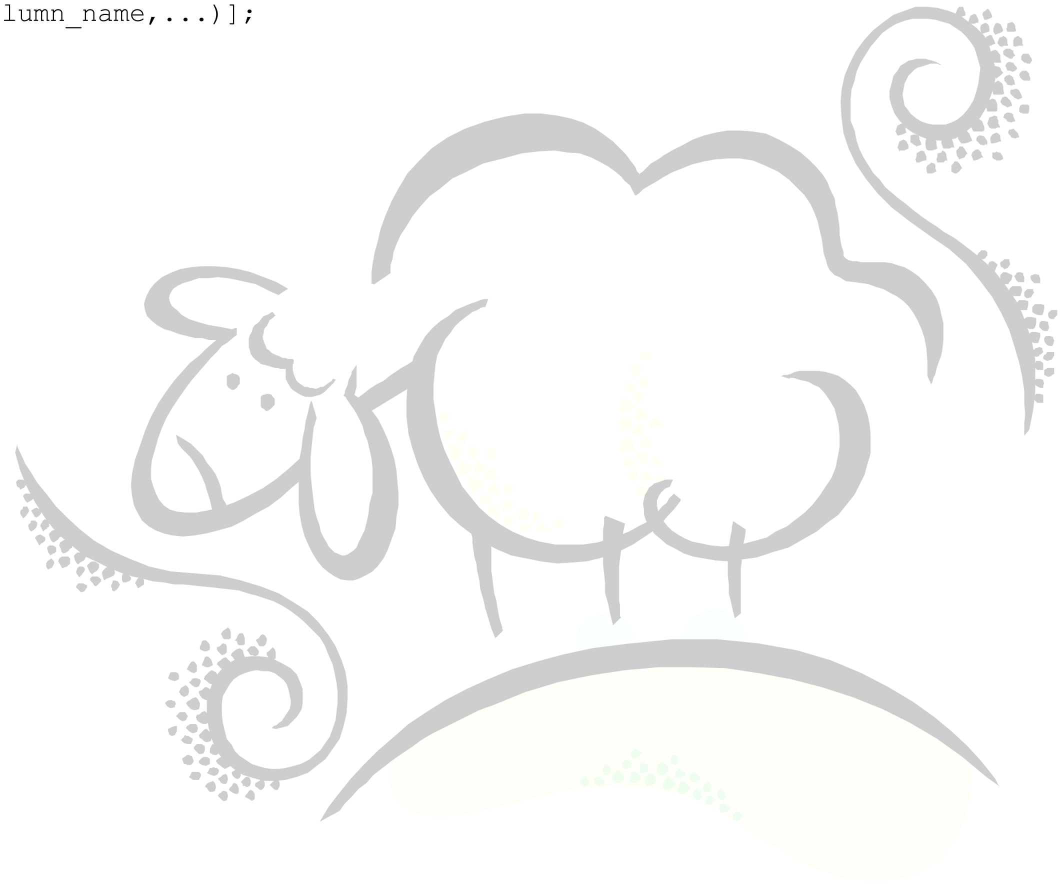
8. Dropping and deleting:
   a. To drop a table: DROP TABLE table_name;
   b. To drop a database: DROP DATABASE db_name;
   c. General form of DROP TABLE is:

```
      DROP [TEMPORARY] TABLE [IF EXISTS] table_name,....;
   d. To drop an index: DROP INDEX IDX_NAME ON TABLE_NAME;
   e. Can add an index:  ALTER TABLE table_name ADD INDEX idx_name (column_name);
   f. General form of ALTER TABLE:

      ALTER [IGNORE] TABLE table_name alter_definition, alter_definition, ...

      alter_definition:
      ADD [COLUMN] create_definition [FIRST | AFTER column_name]
      ADD [COLUMN] (create_definition, ...)
      ADD INDEX [index_name] (index_col_name,...)
      ADD PRIMARY KEY (index_col_name,...)
      ADD UNIQUE [index_name] (index_col_name,...)
      ADD FULLTEXT [index_name] (index_col_name,...)
      ADD   [CONSTRAINT   symbol]   FOREIGN   KEY   [index_name]   (index_col_name,...)
[reference_definition]
      ALTER [COLUMN] column_name (SET DEFAULT literal | DROP DEFAULT)
      CHANGE [COLUMN] old_column_name create_definition [FIRST | AFTER column_name]
      MODIFY [COLUMN] create_definition [FIRST | AFTER column_name]
      DROP [COLUMN] column_name
      DROP PRIMARY KEY
      DROP INDEX index_name
      DISABLE KEYS
      ENABLE KEYS
      RENAME [TO] new_table_name
      ORDER BY column_name
      table_options;
```

Chapter Five - Inserting, Deleting, and Updating Data

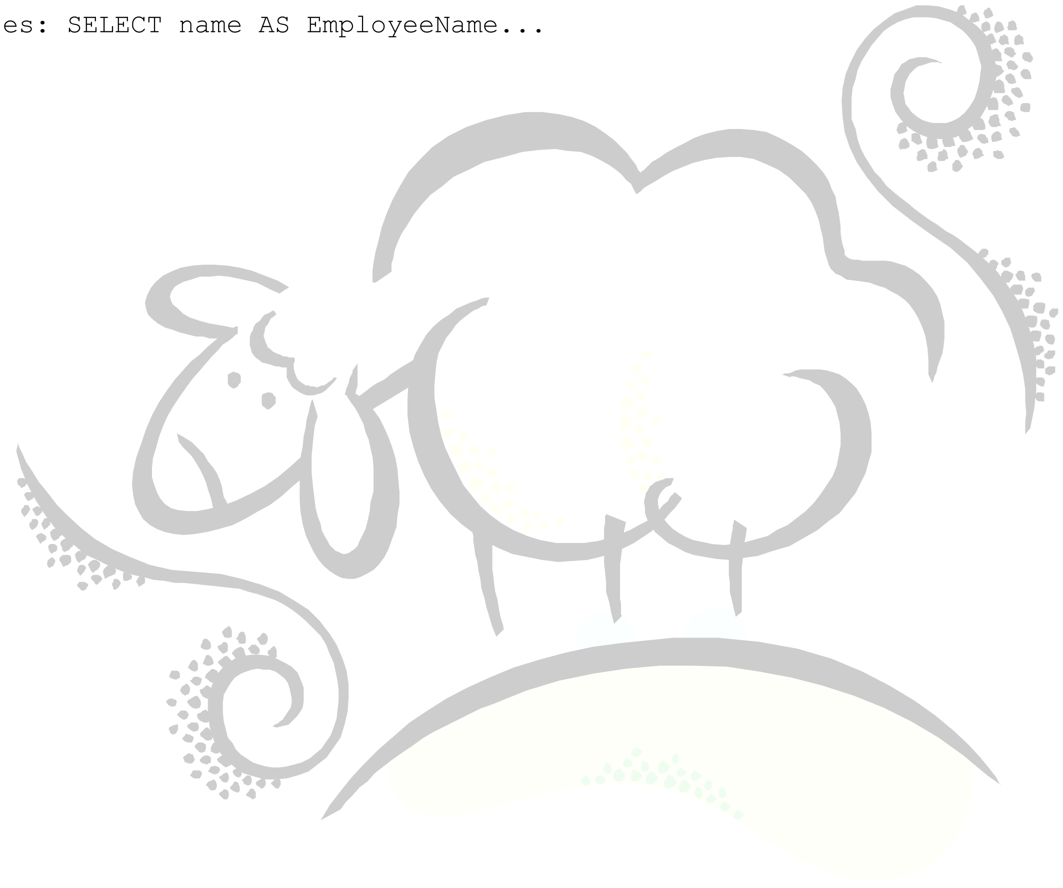
```
1. Inserting:
    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
     [INTO] table_name [(column_name,...)]
     VALUES ((expression | DEFAULT),...),...
     [ ON DUPLICATE KEY UPDATE column_name=expression,...]

    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] table_name [(column_name,...)]
      SELECT ...

    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] table_name
      SET column_name=(expression|DEFAULT),....
      [ ON DUPLICATE KEY UPDATE column_name=expression,...]

2. Replacing (an INSERT if not there, replaces row if it is there):
    REPLACE [LOW_PRIORITY | DELAYED] [IGNORE]
     [INTO] table_name [(column_name,...)]
     VALUES ((expression | DEFAULT),...),...

    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] table_name [(column_name,...)]
      SELECT ...

    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] table_name
      SET column_name=(expression|DEFAULT),....

3. Deleting:
    DELETE [LOW_PRIORITY] [QUICK]
     FROM table_name
     [WHERE where_definition]
     [ORDER BY ...]
     [LIMIT rows]

    DELETE [LOW_PRIORITY] [QUICK] table_name[.*], table_name[.*],...
     FROM table_reference
     [WHERE where_definition]

    DELETE [LOW_PRIORITY] [QUICK]
     FROM table_name[.*], table_name[.*],...
     USING table_reference
     [WHERE where_definition]

4. Truncate: TRUNCATE TABLE table_name;

5. Updating:
    UPDATE [LOW_PRIORITY] [IGNORE] table_name
     SET column_name=expression,....
     [WHERE where_definition]
     [ORDER BY...]
     [LIMIT rows]

    UPDATE [LOW_PRIORITY] [IGNORE] table_name, table_name, ...
     SET column_name=expression,....
     [WHERE where_definition]

6. Bulk loading data with LOAD DATA INFILE
```

```
LOAD DATA LOCAL FILE '/my/local/file.txt'
 INTO TABLE table_name
 FIELDS TERMINATED BY ','
 LINES TERMINATED BY '\n'
 IGNORE 2 LINES
 (name, job, deptID);

LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'filename.txt'
 [REPLACE | IGNORE]
 INTO TABLE table_name
 [FIELDS
   TERMINATED BY '\t'
   [[OPTIONALLY] ENCLOSED BY '']
   [ESCAPED BY '\\'] ]
 [LINES TERMINATED BY '\n']
 [IGNORE number_of LINES]
 [(column_name,...)];
```



Chapter Six - Querying MySQL

```
1. SELECT [DISTINCT] columns
    FROM tables
    WHERE conditions
    GROUP BY group
    HAVING group_conditions
    ORDER BY sort_columns
    LIMIT limits;

  (the above is the abbreviated syntax...see next chapter)

2. Note that you can specify a database and table as:

              DATABASE_NAME.TABLE_NAME

3. Aliases: SELECT name AS EmployeeName...
```



Chapter Seven - Advanced Queries

1.
```
SELECT column,...
 FROM table_nameA,table_nameB,...
```

2. Without a WHERE clause you get a Cartesian Product (Full Join or Cross Join)

3. JOIN, CROSS JOIN, INNER JOIN, STRAIGHT JOIN, LEFT JOIN, RIGHT JOIN

4. Full Syntax for SELECT:
```
SELECT [STRAIGHT JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
  select_definition
 [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
 [FROM table_references
 [WHERE where_definition]
 [GROUP BY {unsigned integer | column_name | formula} [ASC | DESC],...]
 [HAVING where_definition]
 [ORDER BY {unsigned integer | column_name | formula} [ASC | DESC],...]
 [LIMIT [offset,] rows | rows OFFSET offset]
 [PROCEDURE procedure_name(argument_list)]
 [FOR UPDATE | LOCK IN SHARE MODE] ]
```

   a. STRAIGHT JOIN - forces optimizer to join the tables in the order listed on FROM
   b. SQL_SMALL_RESULT and SQL_BIG_RESULT tells MySQL that you expect the result set to have a few or large number of rows.
   c. SQL_BUFFER_RESULT tells MySQL that it must put the result set into a temp table. You can used this when it takes significant time to send the results to the client to avoid having the queried tables locked for that time.
   d. SQL_CACHE and SQL_NOCACHE tells MySQL whether to cache the results.
   e. SQL_CALC_FOUND_ROWS is for use with the LIMIT clause. It tells MySQL to work out how many rows woul have been returned if there had been no LIMIT clause. We can retrieve this usine SELECT FOUND_ROWS();
   f. PROCEDURE must be a proc written in C++.
   g. FOR UPDATE is for the InnoDB table type only. Sets an exclusive lock.
   h. LOCK IN SHARE MODE is the the InnoDB table type only. Sets a shared lock.

Chapter Eight - Using MySQL Built-In Functions with SELECT

1. No DUAL as in Oracle, can just "SELECT 2+2;"

2. Operators: +,-,*,/, IS NULL, IS NOT NULL, =,!= or <>,<,<=,>,>=

3. MySQL string comparisons are case insensitive. Use the BINARY keyword to make it sensitive:

   ```
   SELECT * FROM DEPT WHERE NAME = BINARY 'MARKETING';
   ```

4. n BETWEEN min AND max – range

5. n in (set) - Set membership

6. <=> - NULL safe equal. This will return true (1) if we are comparing two NULLs.

7. n IS NULL - tests whether n is null

8. ISNULL(n) - test for a null value n

9. AND or &&
   OR or ||
   NOT or !
   XOR

10. Control Flow:
    IF (e1,e2,e3)
    if e1 is true then e2 is returned, otherwise e3 is returned.

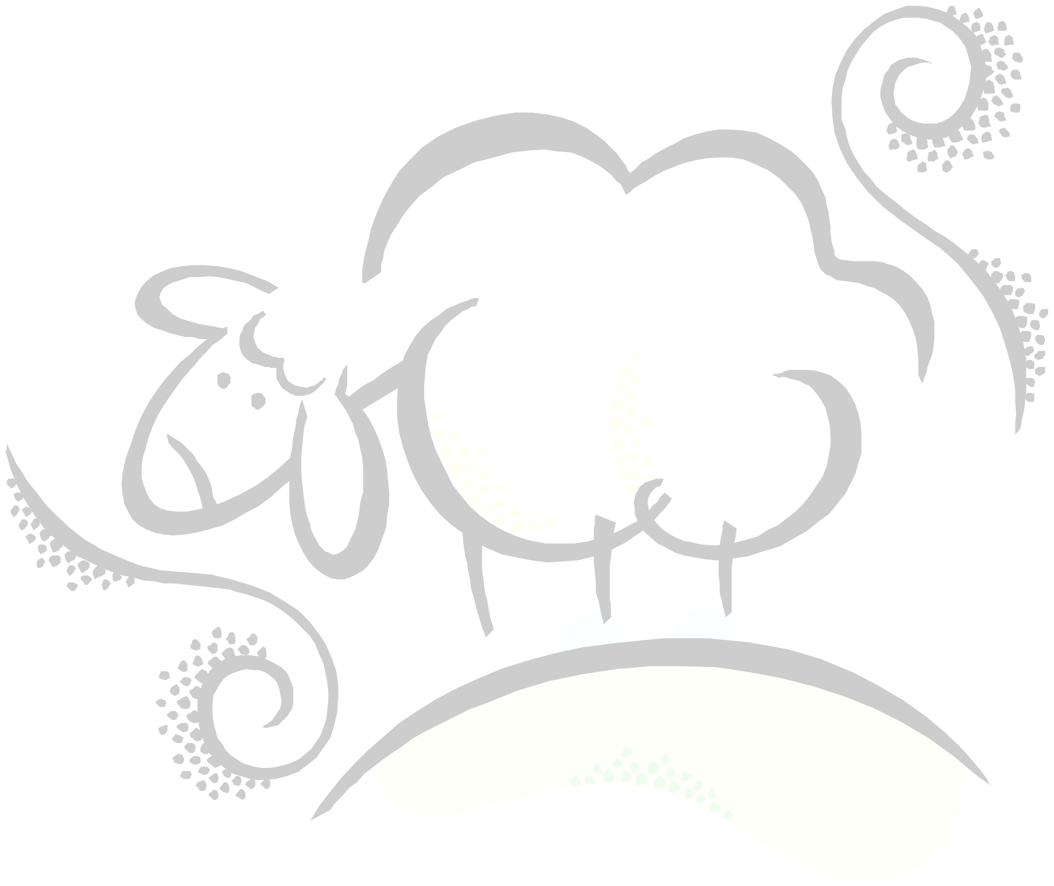
11. Case Statement:

    ```
    CASE value
     WHEN [compared-value] THEN result
     WHEN [compared-value] THEN result
     WHEN [compared-value] THEN result
     ELSE result
    END
    --or--
    CASE
     WHEN [condition] THEN result
     WHEN [condition] THEN result
     ELSE result
    END
    ```

12. String Functions:
    a. concat(s1,s2,...) - Concatenates the strings
    b. conv(n,old_base,new_base) - converts the number n from old to new base
    c. length(s) – length of string
    d. load_file(filename) - Returns the contents of the file stored at filename as a string
    e. locate(needle,haystack,position) - Returns the starting position of the needle in the haystack.
    f. lower(s),upper(s) - lower case or upper case
    g. quote(s) - quotes the string for safe insertion into the database.
    h. replace(target,find,replace) - Returns a string based on target wit all incidences of find replaced with replace.
    i. soundex(s) - Soundex of string s
    j. substring(s,position,length) - substring.
    k. trim(s),ltrim(s),rtrim(s) - removes leading and trailing whitespace from s

```
13. String Comparison Functions:
    a. LIKE - Performs string wildcard matching
    b. RLIKE - Performs regular expression matching
    c. STRCMP - string comparison, just like the strcmp() function in C (-1,1 or 0)
    d. MATCH - Performs full-text searching.

14. Using LIKE:
    a. SELECT * FROM DEPT WHERE NAME LIKE "%re_kjsdf"
       Note that % matches zero or more characters and the underscore matches exactly
       one character.

15. Numeric Function:
    a. abs(n) - absolute value
    b. ceiling(n) - n rounded up to nearest integer
    c. floor(n) - n rounded down to nearest integer
    d. mod(n,m) - integral remainder after n/m
    e. n div m - integral quotient
    f. power(n,m) - n**m
    g. rand(n) - returns a random number between 0 and 1.
    h. round(n[,d]) - round n to nearest integer.  If you specify d then round to
       nearest decimal point.
    i. sqrt(n) - square root of n

16. Date and Time Functions:
    a. adddate(date, INTERVAL n type)
       subdate(date, INTERVAL n type)
        type can be: SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, MINUTE:SECOND (n should
        be 'm:s'), HOUR:MINUTE('h:m'), DAY_HOUR('d h'), YEAR_MONTH('y-m'),
        HOUR_SECOND('h:m:s'), DAY_MINUTE('d h:m'), and DAY_SECOND('d h:m:s').
    b. curdate(), curtime(),now() - current date, time, date/time.
    c. date_format(date,format)
       time_format(time,format)
        consult manual for format string
    d. dayname(date) - Monday, Tuesday, ...
    e. extract(type FROM date) - returns the value of type in date. If you specify
       YEAR then it returns the year portion of the date (see (a))
    f. unix_timestamp([date]) - Returns the Unix timestamp or the number of seconds
       since 01JAN1970.

 17. Cast Functions:
     a. cast(expression AS type)
     b. convert(expression, type)
      types are BINARY,CHAR,DATE,DATETIME,SIGNED (INTEGER) and UNSIGNED (INTEGER).

 18. Other Functions:
     a. benchmark(count,expression) - evaluates expression count number of
        times...used for benchmarking.
     b. encrypt(s[,salt]) - encrypts s using a Unix crypt system call.
     c. found_rows() - see SQL_CALC_FOUND_ROWS above.
     d. last_insert_id() - returns the last automatically generated AUTO_INCREMENT
        value.
     e. md5(s) - returns the 128-bit MD5 hash of string s.  Use this to store
        usernames and passwords.
     f. password(s) - Calculates a password string for the string s.

19. Group By Functions:
    a. avg(column) - average
    b. count(column)
    c. min(column)
    d. max(column)
```

e. `std(column)`
f. `sum(column)`



Chapter Nine - Understaning MySQL's Table Types

1. ISAM (type=ISAM) - legacy table type replaced by MyISAM.

2. MyISAM (type=MyISAM) - This is the default. Offers very fast but not transaction-safe storage. High performance.
   a. Can be dynamic, static or compressed. Tables with fixed length rows will be created as static tables, otherwise dynamic.
   b. Compressed tables must be created with myisampack tool...but remember that compressed tables are READ ONLY!!!
   c. When using CREATE TABLE, you can specify "fulltext(column_name,...)" to create a fulltext search:

   ```
   create table article (
    articleID int not null auto_increment primary key,
    title varchar(255),
    body text,
    fulltext (title,body) );
   ```

   d. Can select from fulltext using MATCH/AGAINST:
   select title from article where match(title,body) against ('merge acquisition acquire takeover'); This will return a record if any of the four words are in title or body.
   e. Can also use a google-based boolean search:
   select title from article where match(title,body) against ('+linux +"Open Source" -desktiop Java ~Oracle' IN BOOLEAN MODE);
   Note:
    1. + - Word is compulsory
    2. - - Word must not appear
    3. < - This word is less important
    4. > - This word is more important
    5. () - group words together as a subexpression
    6. ~ - This word may appear but has a negative effect on ranking
    7. * - wildcard and may appear only at end of word
    8. " " - Matches the phrase is quotes.

3. InnoDB (type=InnoDB) - fast, transaction-safe storage engine offering transactions, row-level locking, support for foreign keys, consistent nonlocking reads in SELECTs. InnoDB has its own configuration options, its own directory and its own way of storing data. Note that MyISAM stores one table per file, InnoDB stores all its tables and indexes in a tablespace

4. BerkelyDB (BDB) Tables - transaction-safe with page-level locking.

5. Merge Tables (type=Merge) - Treats multiple MyISAM tables as a simgle table for the purposes of queries.

   ```
   create table logs (
      logid int auto_increment primary key,
      logts datetime,
      entry char(255)
   ) type=merge union=(logA,logB,logC) insert_method=last;
   ```

   Note that you can still query logA, logB and logC separately. You can close an open merge table with FLUSH TABLES.

6. Heap Tables (type=heap) - Stored in memory solely. Can use max_rows=# to limit number of rows in the heap. Cannot use auto_increment, no BLOB or TEXT support, cannot use leftmost prefix of an index to find rows, indexes will be used only to find rows with queries that use the = or <=> operators in the search clause.

Chapter Ten - Using Transactions with InnoDB Tables

```
1. START TRANSACTION;

2. ROLLBACK;

3. COMMIT;

4. 1,2 and 3 do not work on MyISAM tables.

5. set autocommit=0; turns off autocommit
   set autocommit=1; turns on autocommit.

6. Locking and Unlocking Tables:
   a. lock tables table_name,...; locks these tables.
   b. lock tables account write, account as A read, othertable low_priority write;
   c. unlock tables; unlocks the tables.

7. Transaction Isolation
   a. Serializable - No dirty reads, no non-repeatable reads, no phantom reads
   b. Repeatable Read - No dirty reads, no non-repeatable reads, possible (unlikely)
      phantom reads
   c. Read Committed - No dirty reads, possible non-repeatable reads, possible
       phantom reads
   d. Read Uncommitted - possible dirty reads, possible non-repeatable reads,
      possible phantom reads

    Example: set transaction isolation level serializable;
```

Chapter Eleven - Managing User Privileges

```
1. GRANT priv_type [(column_list)], priv_type [(column_list)], ....
    ON {table_name | * | *.* | dbname.* }
    TO username IDENTIFIED BY 'password'
    [ REQUIRE NONE |
              [{SSL | X509}]
              [CIPHER cipher [AND]]
              [ISSUER issuer [AND]]
              [SUBJECT subject] ]
    [WITH  [GRANT  OPTION  |  MAX_QUERIES_PER_HOUR  #  |  MAX_UPDATES_PER_HOUR  #  |
MAX_CONNECTIONS_PER_HOUR #]]
```

2. Privilge Levels
   a. User Levels
      i. CREATE
      ii. CREATE TEMPORARY TABLES
      iii. DELETE
      iv. EXECUTE
      v. INDEX
      vi. INSERT
      vii. LOCK TABLES
      viii. SELECT
      ix. SELECT
      x. SHOW DATABASES
      xi. UPDATE
      xii. USAGE (user can log in)
   b. Administrator-Level Privileges
      i. ALL - user has ALL privileges except WITH GRANT OPTION
      ii. ALTER
      iii. DROP
      iv. FILE
      v. PROCESS - user can see processes running
      vi. RELOAD - user can use the FLUSH statement.
      vii. REPLICATION CLIENT - user can check where the masters and slaves are.
      viii. REPLICATION SLAVE - See Ch 16.
      ix. SHUTDOWN - user can run mysqladmin shutdown.
      x. SUPER - User can connect even if MySQL has its maximum number of connections and can execute the commands
         CHANGE MASTER, KILL (thread), mysqladmin debug, PURGE MASTER LOGS, and SET GLOBAL.
      xi. WITH GRANT OPTION - use can pass on any privilges he has.

```
3. REVOKE priv_type [(column_list)], priv_type [(column_list)], ...
    ON {table_name | * | *.* | db_name.*}
    FROM username [,usermame...]
```

4. Privilege Tables:
   a. user table - info about a user's global privilege set
   b. db table - stores a user's privileges for a particular database
   c. host table - MySQL consults with this table when it finds a blank host entry in the db table.
   d. tables_priv table - Expresses user privileges that relate to individual tables.
   e. columns_priv table - Expresses user privileges relating to individual columns.

Chapter Twelve - Configuring MySQL

1. /etc/my.cnf is the MySQL configuration file.

2. To obtain a complete list of mysqld options type "mysqld --help" at the Linux command prompt.

3. Setting InnoDB Configuration Options:
   a. innodb_data_file_path = ibdata1:10M:autoextend   creates a single file called ibdata1 with an initial size of 10MB and automatically makes it bigger 8MB at a time.

      filename:filesize[;filename:filesize;...][:autoextend[:max:size]]

   b. innodb_buffer_pool_size=#M
      Sets the size of the buffer used to cache InnoDB table data and indexes. More means less I/O.
   c. innodb_additional_mem_pool_size=#M
      Sets aside memory to store internal MySQL data structures.
   d. innodb_log_file=#M
      Sets the size of each log file.
   e. innodb_log_files_in_group=#
      Number of log files in each group (2 is recommended)
   f. innodb_log_buffer_size=#M
      Sets the size of the buffer in which logs are stored before they are written to disk.
   g. innodb_flush_log_at_trx_commit=#
      1=Everytime a transaction is committed, the log will be flushed to disk
      0=Log flushed almost every second
      2=Log will be written to with each commit, but flushed only one per second.

4. Internationalization
   a. Can set the default character set with --default-character-set option (default is latin1)
   b. Can set the default collation set with --default-collation option. (default is latin1-swedish_ci)

Chapter Thirteen - Administering Your Database

1. You can start the server with /etc/init.d/mysqld start OR safe_mysqld

2. You can stop the server with /etc/init.d/mysqld stop OR mysqladmin -u root -p shutdown

3. mysqlshow [-u USERNAME -p DATABASE] is the same as show databases;

4. To get info about a table: mysqlshow -u username --status employee

5. show columns from table_name; equivalent to describe table_name;

6. Server status:
   a. SHOW STATUS;
   b. mysqladmin -u username -p extended-status

7. To see the server variables:
   a. SHOW VARIABLES;
   b. mysqladmin -u username -p variables

8. Viewing processes:
   a. SHOW PROCESSLIST;
   b. mysqladmin -u username -p showprocesslist

9. Viewing grants:
   a. SHOW GRANTS FOR username@host;
   b. SHOW PRIVILEGES; - shows the privs on the system

10. Viewing Table Types:
    a. SHOW TABLE TYPES;
    b. SHOW CREATE TABLE tablename; --shows create statement for this table!!!

11. Killing a Process (see SHOW PROCESSES):
    a. kill process_id

12. Clearing Caches:
    a. FLUSH PRIVILEGES;
    b. FLUSH QUERY CACHE; --defragments the query cache and improves performance.
    c. RESET QUERY CACHE; --doesn't defrag just clear it out.

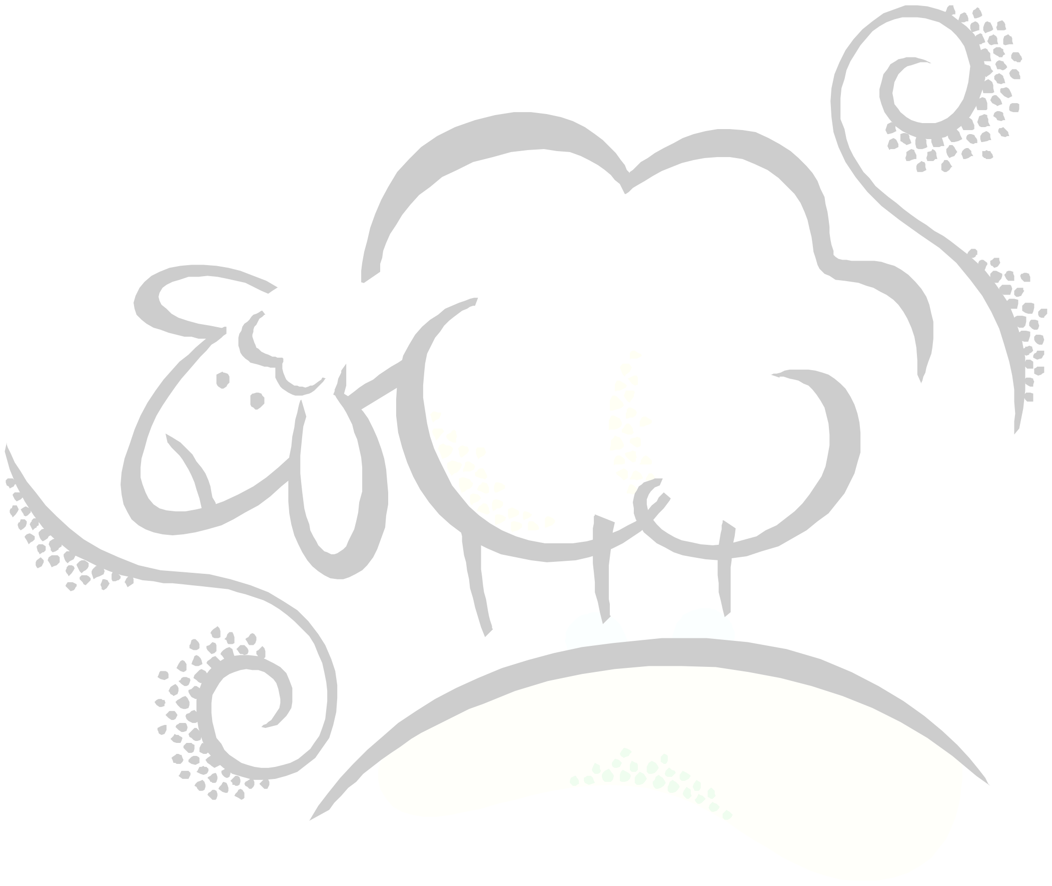
13. Understanding Log Files
    a. Error Log: Tracks all the errors that have occurred. Usually called hostname.err or mysql.err (see the option log-error=filename in my.cnf.
    b. Query Log: Logs all the queries run on the system. See log=filename option.
    c. Binary Log: Logs all the queries that change data. See log-bin=filename.
    d. Slow Query Log: Logs all queries that took longer to execute than the value stroed in the variable long_query_time. See log-slow-queries=filename.

    NOTE: All of these are text except for the binary log. You can view the binary log by running "mysqlbinlog logfile" at the Linux command prompt.

14. Log files will continue to grow so you should rotate them with the script mysql-log-rotate. Try mysqladmin flush-logs first.

15. mysqladmin Options:
    a. mysqladmin -ushecht -pscott create database
    b. mysqladmin -ushecht -pscott drop database
    c. mysqladmin -ushecht -pscott ping
    d. mysqladmin -ushecht -pscott version

```
e. mysqladmin -ushecht -pscott extended-version
f. mysqladmin -ushecht -pscott processlist
g. mysqladmin -ushecht -pscott kill id1,id2,...  Kills processes from (f).
h. mysqladmin -ushecht -pscott variables
```



Chapter Fourteen - Backup and Disaster Recovery

1. There are four ways to make a backup in MySQL:
   - a. mysqldump script to create a dump file which contains all of the SQL DDL and INSERT-VALUES to recreate the database.
   - b. mysqlhotcopy script to create a datafile. This copies the datafiles for a particular database directly.
   - c. Directly backup the files yourself. If you choose to do this, you need to shutdown or flush and lock all tables before copying to make sure they are internally consistent. (a) and (b) will do this and are safer!
   - d. Use the BACKUP TABLE and RESTORE TABLE commands to backup or restore a specified table or set of tables.

2. Using mysqldump
   - a. Backup: mysqldump --opt -u username -p password db_name > backup.sql
     - i. Options include:
       - a. --quick - no buffering of data, just a dump to the file
       - b. --add-drop-table - adds DROP TABLE
       - c. --add-locks - LOCK TABLES and UNLOCK TABLES will be added
       - d. --extended-insert - Multiline insert used rather than individual INSERTs
       - e. --lock-tables - Lock all the tables BEFORE the dump begins.
       - f. --no-data - dump structure and not the data
       - g. --databases - allow you to list more than one database for dumping
       - h. --all-databases - dump all the databases in storage
       - i. --allow-keywords - Fully-qualifies every column name with its table name.
       - j. --opt - equivalent to a,b,c,d,e above.
   - b. Restore:
     - i. Create the database first
     - ii. mysql -u username -p password db_name < backup.sql

3. Using mysqlhotcopy
   - a. Copies the database files rather than retrieving data through a connection
   - b. mysqlhotcopy -u username -p password db_name backup_location
   - c. To use thses backups, you should stop the MySQL server and replace the datafiles in the MYSQL data directory with the backed-up files.

4. Backing Up and Restoring Manually.
   - a. Open a mysql session and issue the following commands:

     ```
     use db_name;
     lock tables table_name1 read, ...;
     flush tables;
     ```

   - b. WITHOUT EXISTING FROM YOUR SESSION IN PART (a), copy the datafiles over.
   - c. In the session from part (a), issue: unlock tables;

5. Using BACKUP TABLE and RESTORE TABLE (MyISAM Tables type ONLY!!!)
   - a. backup table t1 to '/path/to/backup';
   - b. restore table t1 from '/path/to/backup';

6. Restoring from the Binary Log
   - a. When you restore from a backup, some inserts and updates will have been made since the backup was taken. These changes are stored in your binary log or you update log. You can extract a list of operations from the binary log using:

     ```
             mysqlbinlog logfile > updates.sql.
     ```

7. Checking and Repairing Tables

a. {CHECK | REPAIR} TABLE - can be used when the server is up.
b. myisamchk - do not use on tables currently in use. Bring server down.
c. mysqlcheck - can be used when the server is up.

```
CHECK TABLE table_name;
REPAIR TABLE table_name;
```

If this doesn't work, then try myisamchk:

```
myisamchk filename_of_MyISAM_table_name;
```

Note: the filename has a .myi extension.

```
myisamchk -q -r filename_of_MyISAM_table_name;
```

Note: the filename has a .myi extension.

QUICK RECOVERY

```
myisamchk -m filename_of_MyISAM_table_name;
```

Note: the filename has a .myi extension. The rows are scanned as well as indexes.

```
myisamchk --safe-recover filename_of_MyISAM_table_name;
```

Can use mysqlcheck to check MyISAM and InnoDB tables and to repair MyISAM tables safely when the server is up and running.

```
mysqlcheck -u username -p password {db_name | --all-databases | --databases db_name1 db_name2...}
```

Use the -r option to repair any corrupted MyISAM tables.

Chapter Fifteen - Securing Your MySQL Installation

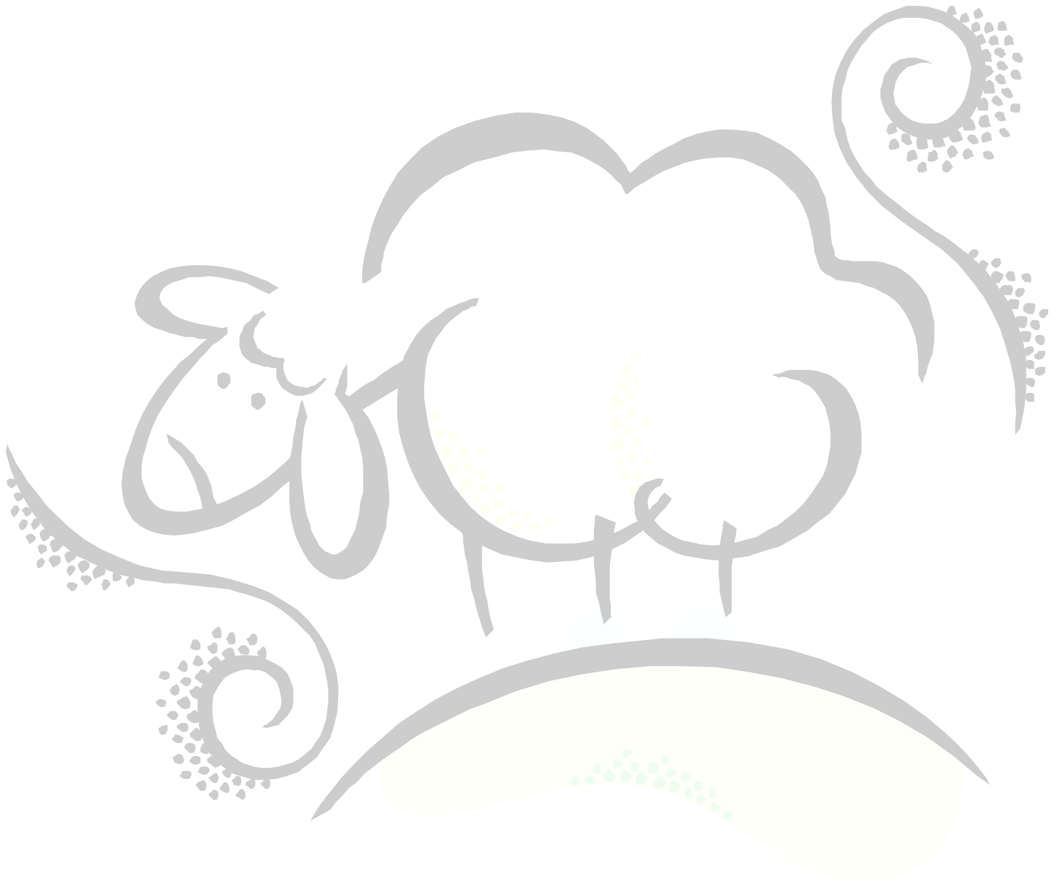
1. Set the password to the root account immediately

2. Delete anonymous users:

   ```
   delete from user where User='';
   delete from db where User='';
   flush privileges;
   ```

3. Don't run mysqld as root.

4. Use SSL and use "require ssl" when you grant privs to a user.



Chapter Seventeen - Optimizing Your MySQL Server Configuration

1. Compile mysql for your platform and only include the character sets you need.

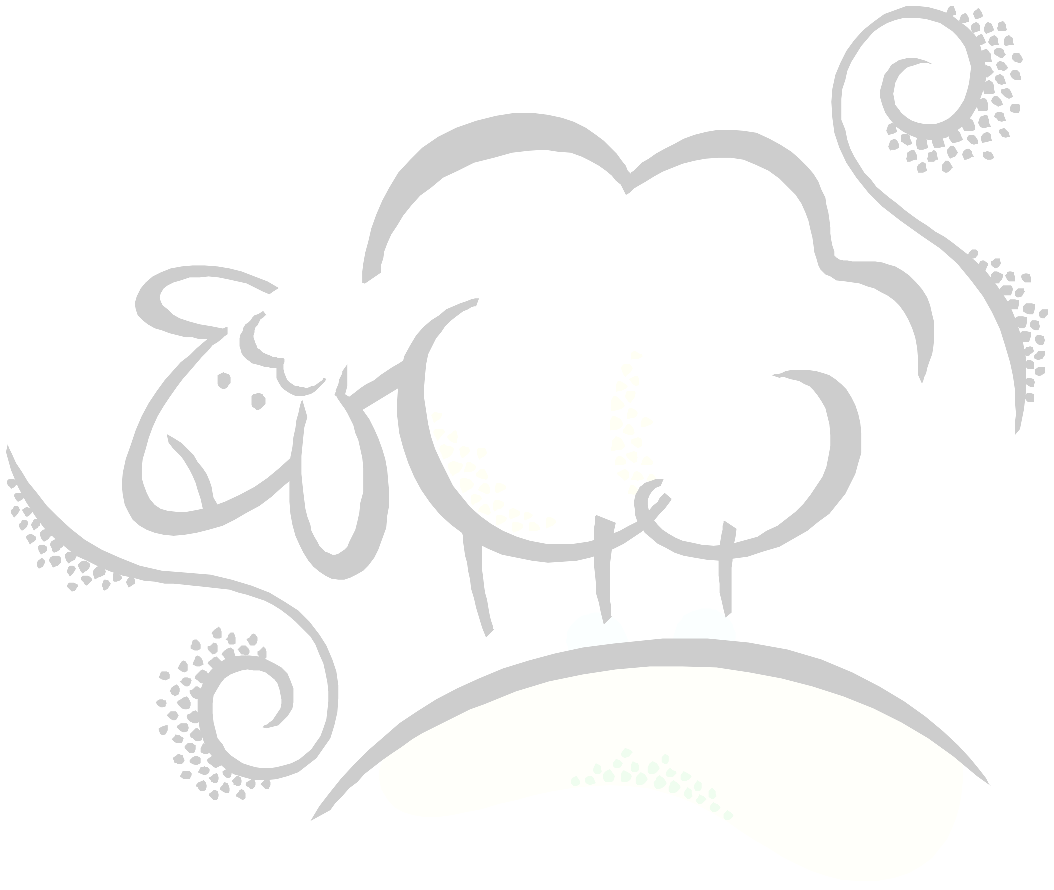
2. Tune the server parameters. See the files my-huge.cnf, my.large.cnf, my-medium.cnf, and my-small.cnf to get an idea on server parameter settings.

3. Two important parameters are:
   a. key_buffer_size - where MyISAM indexes are stored in memory. The bigger the better (20% to 50% of memory)
   b. table_cache - This limits the max number of tables that can be opened at once. The size depends on what the op/sys will allow.
   c. read_buffer_size - used when a full table scan is performed to store the table data.
   d. sort_buffer_size - used for ORDER BY

4. If you have access to multiple disks, you can improve performance by putting different databases on different disks. You can also used RAID--RAID0 will improve reading and writing performance, and RAID1 or RAID5 will improve reading performance. You will also get better performance with SCSI drives over IDE drives.

Chapter Eighteen - Optimizing Your Database

```
1. Use indexes judiciously

2. ANALYZE TABLE table_name;

3. OPTIMIZE TABLE table_name;   - use this to defrag MyISAM and BDB tables.
```



Chapter Nineteen - Optimizing Your Queries

```
1. Benchmarking your queries:  benchmark(100,'select * from table_name where....');

2. Turn on slow query logging: --log-slow-queries=filename  --log-long-format

3. Define what is a slow query with long-query-time=#

4. Use mysqldumpslow script to view the slow query log (or open as text file).

5. Use EXPLAIN to see what the database engine is doing to your query:
   explain  your-sql-query-here;
   a. look at the column marked TYPE and POSSIBLE_KEYS.
   b. TYPE takes on the following values:
      i. eq_ref - best thing to see!  (system and const are better).  A single row
                  will be read from the table for each row in the other table.
      ii. ref - all rows with matching index values will be read from the table.
      iii. range =- Rows in a particular range will be read from the table.
      iv. index - complete index scan.
      v. ALL - full scan.
   c. POSSIBLE_KEYS - if you see NULL, you may want to add an index.

6. MySQL uses an estimated number of rows (as shown in EXPLAIN) to work out the best
   order in which to join the tables.  If you notice that the estimate is off, you
   may want to try STRAIGHT JOIN to force the table order.

7. EXPLAIN Columns:
   ID - Sequence Number, one for each select (if subquery)
   SELECT_TYPE - SIMPLE means plain vanilla select.  If you are using subqueries, the
                 outer query will be marked PRIMARY and the inner queries will be
                 SUBSELECT or DEPENDENT SUBSELECT.
   TABLE - This is the table this row is about
   TYPE - see (b) above
   POSSIBLE_KEYS - see (c) above.
   KEY - Tells you which index was selected for use in the query.  It will be NULL if
          no index was selected.
   KEY_LEN - Length of the index MySQL decided to use.
   REF - This is the value being compared to the key to decide whether to select
         rows.
   ROWS - Estimate of number of rows read to generate the results.  Multiply to get
          total number of rows read.
   EXTRA - Add'l information (Using Index means MySQL can retrieve the result of the
           query completely from the index.)
```

Support sheepsqueezers.com

If you found this information helpful, please consider supporting sheepsqueezers.com. There are several ways to support our site:

- ☐ Buy me a cup of coffee by clicking on the following link and donate to my PayPal account: Buy Me A Cup Of Coffee?.
- ☐ Visit my Amazon.com Wish list at the following link and purchase an item: http://amzn.com/w/3OBK1K4EIWIR6

Please let me know if this document was useful by e-mailing me at `comments@sheepsqueezers.com`.