CpSc 462/662: Database Management Systems (DBMS) (TEXNH Approach)

SQL and MySQL

James Wang

SQL - Structured Query Language

- **Structured Query Language, is a computer language designed for**
  - retrieval and management of data in relational database management systems
  - database schema creation and modification
  - database object access control management.
- **History:**
  - The first version of SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s.
  - Standardized in 1986 by ANSI.
  - Subsequent versions of the SQL standard have been released as ISO standards.

2

SQL Standards

- **SQL-86: First published by ANSI. Ratified by ISO in 1987. Alias: SQL-87.**
- **SQL-89: Minor revision, adopted as FIPS (Federal Information Processing Standard) 127-1. Alias: FIPS 127-1.**
- **SQL-92: Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2. Alias: SQL2, FIPS 127-2.**
- **SQL:1999: Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features. Alias: SQL3.**

3

Latest SQL Standards

- **SQL:2003: Introduced XML-related features, *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).**
- **SQL:2006: ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML.**
  - It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form.
  - It provides facilities that permit applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.

4

Why MySQL?

- **Open source databases are showing the highest growth rate in the database market, according to a new study by analyst firm Gartner.**
- **MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), a fast growing open source enterprise software stack.**
- **MySQL runs on more than 20 platforms including Linux, Windows, OS/X, HP-UX, AIX, Netware.**
- **More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from lock-in.**
- **Implies Job Opportunities.**

5

MySQL Features

- **Internals and Portability: well designed and portable.**
  - Written in C and C++.
  - Tested with a broad range of different compilers.
  - Works on many different platforms.
  - More …
- **Data Types: Support many data types.**
  - Many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, and OpenGIS spatial types.
  - Fixed-length and variable-length records.
- **Statements and Functions:**
  - Full operator and function support in the SELECT list and WHERE clause of queries.
  - Full support for SQL GROUP BY and ORDER BY clauses.
  - Support for LEFT OUTER JOIN and RIGHT OUTER JOIN with both standard SQL and ODBC syntax.
  - More …

6

Database Server

- **pheonix.cs.clemson.edu**
  - You can use SSH to access the server from anywhere.
  - User name:
    - uX or gX where u and g represent undergraduate and graduate teams respectively, X is your team number, e.g., the username for graduate team #1 is g1, for undergraduate team #10 is u10.
  - Password:
    - Initial password is given in Website: http://phoenix.cs.clemson.edu/2011Spring/u0/. Please change your password the first time logging to the server and tell your teammates about the new password.
  - Database access:
    - Use http://phoenix.cs.clemson.edu/phpMyAdmin, or use sell command that we will discuss later.
    - Use the same username and initial password as for those used for SSH to log into the database.

How to Use MySQL?

- **Connecting to the server:**

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 25338 to server version: 5.1.21-beta-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

- **Disconnecting from the server: After you have connected successfully, you can disconnect any time by typing QUIT (or \q) at the mysql> prompt:**

```
mysql> QUIT
Bye
```


How to Use MySQL (cont.)

- **Entering Queries:**

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----------------+--------------+
| VERSION()       | CURRENT_DATE |
+-----------------+--------------+
| 5.1.2-alpha-log | 2005-10-11   |
+-----------------+--------------+
1 row in set (0.01 sec)

mysql>
```

- **Change password:**

```
mysql> set password = password("XXXXXX");

Query OK, 0 rows affected (0.00 sec)


mysql> quit

Bye
```


Creating and Using a Database

- **Suppose that you have several pets in your home (your menagerie) and you would like to keep track of various types of information about them. To do so, you need to know how to:**
  - Create a database.
  - Create tables to hold your data.
  - Load data into tables.
  - Retrieve data from the tables to answer different questions about your pets.
- **Example databases:**
  - http://www.cs.clemson.edu/~jzwang/1101462/menagerie-db.tar.gz

Creating and Selecting a Database

- **Create database:**

```
mysql> CREATE DATABASE menagerie;
```

Note: you may not have the privileges to create a database. In such a case, ask your administrator to create one for you and grant necessary rights for you.

- **Use database:**

```
mysql> USE menagerie
Database changed
```

- **Select the database when connecting to MySQL:**

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

Note: "menagerie" is the database name not your password. If you want to connect to MySQL without having to wait for password prompt, you need to use "*-pmypassword*", where "*mypassword*" is your password. However, somebody may steal your password if you work in a public lab.

Creating a Table

- **Show databases:**

```
mysql> SHOW DATABASES;
+------------+
| Database   |
+------------+
| mysql      |
| menagerie  |
| tmp        |
+------------+
```

- **Show tables:**

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

- **Create a table pet:**

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
-> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);

mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| pet                 |
+---------------------+
```

Check a table structure

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```


Loading Data into a Table

- **Suppose that your pet records can be described as following table:**

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

- **You can use INSERT to populate the data into your table.**

```
mysql> INSERT INTO pet
-> VALUES ( 'Fluffy',' Harold' , 'cat','f','1993-02-04',NULL);
```

- **Is this an efficient way if you have a lot of data?**

Load a Set of Records

- **Create a flat text file pet.txt first:**

```
Fluffy	Harold	cat	f	1993-02-04	\N
Claws	Gwen	cat	m	1994-03-17	\N
Buffy	Harold	dog	f	1989-05-13	\N
Fang	Benny	dog	m	1990-08-27	\N
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	\N
Whistler	Gwen	bird	\N	1997-12-09	\N
Slim	Benny	snake	m	1996-04-29	\N
```

- **One record per line, with values separated by tabs, and given in the order in which the columns were listed in the CREATE TABLE statement.**
- **For missing values (such as unknown sexes or death dates for animals that are still living), you can use NULL values. To represent these in your text file, use \N (backslash, capital-N).**
- **Load data:**

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet;
```


Load a Set of Records (cont.)

- **Delimiter:** Instead of the default TAB, you can use any special character as the delimiter. Some common delimiters include a space, a comma, etc. Make sure your data values do not contain the same character as the delimiter.

```
mysql> LOAD DATA LOCAL INFILE '<path>/pet.txt'
    -> INTO TABLE pet FIELDS terminated by '<delimiter>';
```

- **End of line:** Different operating systems may use different control characters at the end of each line. For instance, "\r\n" is used in Windows systems, while "\n" is used in Unix. Apple OS X uses "\r". To deal with this heterogeneity, we may specify the end of line symbols when loading the data if your text file was created on another OS.

```
mysql> LOAD DATA LOCAL INFILE '<path>/pet.txt'
    -> INTO TABLE pet FIELDS terminated by ', '
    -> LINES TERMINATED BY '\r\n';
```


Retrieving Information from a Table

- **The SELECT statement is used to retrieve information from a table. The general form of the statement is:**

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy ;
```

```
mysql> SELECT * FROM pet;
+----------+--------+---------+------+------------+------------+
| name     | owner  | species | sex  | birth      | death      |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | cat     | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | cat     | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | dog     | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | dog     | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | dog     | m    | 1979-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | bird    | f    | 1998-09-11 | NULL       |
| Whistler | Gwen   | bird    | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | snake   | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```


Selecting Particular Rows

```
mysql> SELECT * FROM pet WHERE name = 'Bowser';
+--------+-------+---------+------+------------+------------+
| name   | owner | species | sex  | birth      | death      |
+--------+-------+---------+------+------------+------------+
| Bowser | Diane | dog     | m    | 1979-08-31 | 1995-07-29 |
+--------+-------+---------+------+------------+------------+

mysql> SELECT * FROM pet WHERE birth >= '1998-1-1';
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Puffball | Diane | hamster | f    | 1999-03-30 | NULL  |
+----------+-------+---------+------+------------+-------+

mysql> SELECT * FROM pet WHERE species = 'dog' AND sex = 'f';
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Selecting Particular Columns

```
mysql> SELECT DISTINCT owner FROM pet;
+---------+
| owner   |
+---------+
| Benny   |
| Diane   |
| Gwen    |
| Harold  |
+---------+

mysql> SELECT name, species, birth FROM pet
    -> WHERE species = 'dog' OR species = 'cat';
+--------+---------+------------+
| name   | species | birth      |
+--------+---------+------------+
| Fluffy | cat     | 1993-02-04 |
| Claws  | cat     | 1994-03-17 |
| Buffy  | dog     | 1989-05-13 |
| Fang   | dog     | 1990-08-27 |
| Bowser | dog     | 1989-08-31 |
+--------+---------+------------+
```


Sorting Rows

```
mysql> SELECT name, species, birth FROM pet
    -> ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | species | birth      |
+----------+---------+------------+
| Chirpy   | bird    | 1998-09-11 |
| Whistler | bird    | 1997-12-09 |
| Claws    | cat     | 1994-03-17 |
| Fluffy   | cat     | 1993-02-04 |
| Fang     | dog     | 1990-08-27 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | snake   | 1996-04-29 |
+----------+---------+------------+
```


Date Calculations

```
mysql> SELECT name, birth, CURDATE(),
    -> (YEAR(CURDATE())-YEAR(birth))
    -> - (RIGHT(CURDATE(),5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet;
+----------+------------+------------+-----+
| name     | birth      | CURDATE()  | age |
+----------+------------+------------+-----+
| Fluffy   | 1993-02-04 | 2003-08-19 |  10 |
| Claws    | 1994-03-17 | 2003-08-19 |   9 |
| Buffy    | 1989-05-13 | 2003-08-19 |  14 |
| Fang     | 1990-08-27 | 2003-08-19 |  12 |
| Bowser   | 1989-08-31 | 2003-08-19 |  13 |
| Chirpy   | 1998-09-11 | 2003-08-19 |   4 |
| Whistler | 1997-12-09 | 2003-08-19 |   5 |
| Slim     | 1996-04-29 | 2003-08-19 |   7 |
| Puffball | 1999-03-30 | 2003-08-19 |   4 |
+----------+------------+------------+-----+
```


Pattern Matching

- SQL pattern matching allows you to use '_' to match any single character and '%' to match an arbitrary number of characters (including zero characters).
- In MySQL, SQL patterns are case-insensitive by default. Some examples are shown here.
- Note that you do not use = or <> when you use SQL patterns; use the LIKE or NOT LIKE comparison operators instead.

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```


Regular Expression Matching

- The other type of pattern matching provided by MySQL uses extended regular expressions.
- When you test for a match for this type of pattern, use the REGEXP and NOT REGEXP operators (or RLIKE and NOT RLIKE, which are synonyms).
- Some characteristics of extended regular expressions are:
  - '.' matches any single character.
  - A character class '[...]' matches any character within the brackets. For example, '[abc]' matches 'a', 'b', or 'c'.
  - To name a range of characters, use a dash. '[a-z]' matches any letter, whereas '[0-9]' matches any digit.
  - '*' matches zero or more instances of the thing preceding it. For example, 'x*' matches any number of 'x' characters, '[0-9]*' matches any number of digits, and '.*' matches any number of anything.
  - A REGEXP pattern match succeeds if the pattern matches anywhere in the value being tested. (This differs from a LIKE pattern match, which succeeds only if the pattern matches the entire value.)
  - To anchor a pattern so that it must match the beginning or end of the value being tested, use '^' at the beginning or '$' at the end of the pattern.

Regular Expression Pattern Matching Example

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+

mysql> SELECT * FROM pet WHERE name REGEXP "fy$";
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Counting Rows

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
+--------+----------+
| owner  | COUNT(*) |
+--------+----------+
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+

mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
+---------+----------+
| species | COUNT(*) |
+---------+----------+
| bird    |        2 |
| cat     |        2 |
| dog     |        3 |
| hamster |        1 |
| snake   |        1 |
+---------+----------+
```


Using More Than One Table

```
mysql> CREATE TABLE event (name VARCHAR(20), date DATE,
    -> type VARCHAR(15), remark VARCHAR(255));

mysql> LOAD DATA LOCAL INFILE 'event.txt' INTO TABLE event;

mysql> SELECT pet.name,
    -> (YEAR(date)-YEAR(birth)) - (RIGHT(date,5)<RIGHT(birth,5)) AS age,
    -> remark
    -> FROM pet INNER JOIN event
    ->   ON pet.name = event.name
    -> WHERE event.type = 'litter';
+--------+------+-----------------------------+
| name   | age  | remark                      |
+--------+------+-----------------------------+
| Fluffy |    2 | 4 kittens, 3 female, 1 male |
| Buffy  |    4 | 5 puppies, 2 female, 3 male |
| Buffy  |    5 | 3 puppies, 3 female         |
+--------+------+-----------------------------+
```

* Question: *What is the difference between INNER JOIN and OUTER JOIN?*

Getting Information About Databases and Tables

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+

mysql> SHOW TABLES;
+---------------------+
| Tables_in_menagerie |
+---------------------+
| event               |
| pet                 |
+---------------------+

mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```


Common MySQL Commands

Description	Command
To login (from unix shell) use -h only if needed.	[mysql dir]/bin/mysql -h hostname -u root -p
Create a database on the sql server.	create database [databasename];
List all databases on the sql server.	show databases;
Switch to a database.	use [db name];
To see all the tables in the db.	show tables;
To see database's field formats.	describe [table name];
To delete a db.	drop database [database name];
To delete a table.	drop table [table name];

Common MySQL Commands

Description	Command
To login (from unix shell) use -h only if needed.	[mysql dir]/bin/mysql -h hostname -u root -p
Create a database on the sql server.	create database [databasename];
List all databases on the sql server.	show databases;
Switch to a database.	use [db name];
To see all the tables in the db.	show tables;
To see database's field formats.	describe [table name];
To delete a db.	drop database [database name];
To delete a table.	drop table [table name];

Common MySQL Commands (cont.)

Description	Command
Show all data in a table.	SELECT * FROM [table name];
Returns the columns and column information pertaining to the designated table.	show columns from [table name];
Show certain selected rows with the value "whatever".	SELECT * FROM [table name] WHERE [field name] = "whatever";
Show all records containing the name "Bob" AND the phone number '3444444'.	SELECT * FROM [table name] WHERE name = "Bob" AND phone_number = '3444444';
Show all records not containing the name "Bob" AND the phone number '3444444' order by the phone_number field.	SELECT * FROM [table name] WHERE name != "Bob" AND phone_number = '3444444' order by phone_number;

Common MySQL Commands (cont.)

Description	Command
Show all records starting with the letters 'bob' AND the phone number '3444444'.	SELECT * FROM [table name] WHERE name like "Bob%" AND phone_number = '3444444';
Use a regular expression to find records. Use "REGEXP BINARY" to force case-sensitivity. This finds any record beginning with a.	SELECT * FROM [table name] WHERE rec RLIKE "^a$";
Show unique records.	SELECT DISTINCT [column name] FROM [table name];
Show selected records sorted in an ascending (asc) or descending (desc).	SELECT [col1],[col2] FROM [table name] ORDER BY [col2] DESC;
Return number of rows.	SELECT COUNT(*) FROM [table name];
Sum column.	SELECT SUM(*) FROM [table name];

31

Common MySQL Commands (cont.)

Description	Command
Join tables on common columns.	select lookup.illustrationid, lookup.personid,person.birthday from lookup left join person on lookup.personid=person.personid=statement to join birthday in person table with primary illustration id;
Switch to the mysql db. Create a new user.	INSERT INTO [table name] (Host,User,Password) VALUES('%','user',PASSWORD('password'));
Change a users password.(from unix shell).	[mysql dir]/bin/mysqladmin -u root -h hostname.blah.org -p password 'new-password'
Change a users password.(from MySQL prompt).	SET PASSWORD FOR 'user'@'hostname' = PASSWORD('passwordhere');
Allow the user "bob" to connect to the server from localhost using the password "passwd"	grant usage on *.* to bob@localhost identified by 'passwd';

32

Common MySQL Commands (cont.)

Description	Command
Switch to mysql db.Give user privilages for a db.	INSERT INTO [table name] (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,Create_priv,Drop_priv) VALUES ('%','databasename','username','Y','Y','Y','Y','Y','N'); or grant all privileges on databasename.* to username@localhost;
To update info already in a table.	UPDATE [table name] SET Select_priv = 'Y',Insert_priv = 'Y',Update_priv = 'Y' where [field name] = 'user';
Delete a row(s) from a table.	DELETE from [table name] where [field name] = 'whatever';
Update database permissions/privilages.	FLUSH PRIVILEGES;
Delete a column.	alter table [table name] drop column [column name];

33

Common MySQL Commands (cont.)

Description	Command
Add a new column to db.	alter table [table name] add column [new column name] varchar (20);
Change column name.	alter table [table name] change [old column name] [new column name] varchar (50);
Make a unique column so you get no dupes.	alter table [table name] add unique ([column name]);
Make a column bigger.	alter table [table name] modify [column name] VARCHAR(3);
Delete unique from table.	alter table [table name] drop index [colmn name];
Load a CSV file into a table.	LOAD DATA INFILE '/tmp/filename.csv' replace INTO TABLE [table name] FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' (field1,field2,field3);
Dump all databases for backup. Backup file is sql commands to recreate all db's.	[mysql dir]/bin/mysqldump -u root -ppassword --opt >/tmp/alldatabases.sql

34

Common MySQL Commands (cont.)

Description	Command
Dump one database for backup.	[mysql dir]/bin/mysqldump -u username -ppassword --databases databasename >/tmp/databasename.sql
Dump a table from a database.	[mysql dir]/bin/mysqldump -c -u username -ppassword databasename tablename > /tmp/databasename.tablename.sql
Restore database (or database table) from backup.	[mysql dir]/bin/mysql -u username -ppassword databasename < /tmp/databasename.sql
Create Table Example 1.	CREATE TABLE [table name] (firstname VARCHAR(20), middleinitial VARCHAR(3), lastname VARCHAR(35),suffix VARCHAR(3), officeid VARCHAR(10),userid VARCHAR(15),username VARCHAR(8),email VARCHAR(35),phone VARCHAR(25), groups VARCHAR(15),datestamp DATE,timestamp time,pgpemail VARCHAR(255));
Create Table Example 2.	create table [table name] (personid int(50) not null auto_increment primary key,firstname varchar(35),middlename varchar(50),lastname varchar(50) default 'bato');

35

References

- **MySQL documentation, http://dev.mysql.com/doc/**
- **PHP MySQL Tutorial, http://www.php-mysql-tutorial.com/**
- **http://www.pantz.org/database/mysql/mysqlcommands.shtml**

36