

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, July 18. 2005

All Together Now, a Workshop on MySQL and phpMyAdmin

The second of a series in a hands-on workshop focused on MySQL and phpMyAdmin is scheduled for this Monday, July 18 at 7:00pm.

Date/Time: Monday, July 18 at 7:00pm

Location: Caltek.net on Hoover in L.A.

Cost: Free

Seating: limited, first-come first served, and New Attendees Welcome!!

Instructor: David Benjamin (david_AT_peterbenjamin_(dot)_com)

The evening's goal is to gain skills with MySQL, SQL, and phpMyAdmin. This evening we will be ... designing database tables for Graffiti (guestbook with a twist);creating and populating tables with phpMyAdmin;working a many-to-many relationship example;working intermediate multi-table updates;backing up a database.

workshop schedule continued in further reading section.

The agenda for this week (and last) came from specific input from attendees. This workshop can only continue with your input and participation.

This will be workshop format with the goal of learning by doing. There will be brief introductions followed by "now you do it" exercises. The instructor will give individual help and share answers to the most common questions.

Instructor Notes Workshop 2

SQL file of example Graffiti database for the typing adverse. Note that the plan is to allow time to enter these tables and rows via phpMyAdmin

Workshop Schedule

6:45 - 7:45

Start up and log in with phpMyAdmin (Available on the LAMP SIG/class boot disk: enter "localhost/phpmyadmin" in the address of the browser; a few CDROMS are available free at the class).

Create "Graffiti" database and tables using phpMyAdmin

7:45 - 8:00

Discussion of relational database representation of many-to-many relationships with the Graffiti example. Discussion of foreign keys, referential integrity, and cascading deletes (not available with MyISAM tables). We will write the SQL for a Graffiti web application, such as recording a guest visit to the Graffiti home page.

8:00 - 8:30

We will work several exercises of multi-table SELECTs and UPDATES in the World database -- more hypothetical geopolitical upsets.

8:30 - 8:50

Export from MySQL using phpMyAdmin for backup and for moving a development database to production. Export your Graffiti database to email yourself. Import (restore) using phpMyAdmin.

Discussion of the steps required to make a website using the Graffiti database and PHP. As time permits, we will start building a simple home page for Graffiti.

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Posted by David Benjamin in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, July 11. 2005

All Together Now, a Workshop on MySQL and phpMyAdmin

New! A hands-on workshop focused on MySQL and phpMyAdmin is scheduled for Monday evenings at CalTek starting this Monday, July 11 at 7:00pm.

Date/Time: Monday, July 11 at 7:00pm
Location: Caltek.net on Hoover in L.A.
Cost: Free
Seating: limited, first-come first served
Instructor: David Benjamin (david_AT_peterbenjamin_(dot)_com)

This will be workshop format with the goal of learning by doing. There will be brief introductions followed by "now you do it" exercises. The instructor will give individual help and share answers to the most common questions.

The evening's goal is to do a few SELECTs and a few INSERTs and UPDATEs. To make this a little more interesting, we'll use phpMyAdmin, a web-based visual tool that saves a lot of typing and syntax lookups.

workshop schedule and additional information in further reading section.

Instructor Notes Workshop 1

Workshop Schedule

7:00 - 7:30

start up and log in with phpMyAdmin (already on the LAMP SIG boot disk: enter localhost/phpmyadmin" in the address of the browser) and browse database tables using phpMyAdmin

7:30 - 8:00

Common SELECT usage -- start simple and using the editor in phpMyAdmin, grow the SELECT to use WHERE, ORDER BY, and LIMIT. Then, learn how to do filtered searches in phpMyAdmin and view the SQL used.

8:00 - 8:30

Common INSERT and UPDATE usage -- add/modify database rows in phpMyAdmin and view the SQL used. Cut-and-paste that SQL into the SQL editor to make customizations.

8:30 - 8:50

Simplest multi-table SELECT usage -- a short review and refresher and then a couple simple hands-on exercises.

Follow-on workshops will be paced by attendee's progress. Expect topics like (time and interest permitting): SELECT with SUM/MAX/MIN/COUNT and GROUP BY how to generate HTML table rows with SELECT creating databases and setting user permission with phpMyAdmin creating and altering tables with phpMyAdmin saving database creation/restore scripts/backups design of tables for guestbook or reservations

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, June 27, 2005

LAMPsig MySQL Class :: Class #8 (The last class)

Class #8 (the last class) of the MySQL class is scheduled for this Monday, June 27 at 7:00pm at CalTek and will cover Importing and Exporting Data Statements.

Complete syllabus for Class #8 are continued in the further reading section.

Instructor:

Christopher Thompson :: cxtompson_AT_charter_(dot)_net

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

CLASS #8 SYLLABUS

Class 8 – Importing and Exporting Data (Chapter 9, MySQL MM 1.7.5.2, 2.3.16, 8.8, 8.10, 13.1.5, 13.6.2.3)
5% exam material

Review of mysqlimport and mysqldumpmysqlimport Command line interface to LOAD DATA INFILE.mysqldump

LOAD DATA INFILEAlternative to mysql INSERT statementLOAD DATA INFILE 'file_name' INTO TABLE 'table_name';file name is a string and must be quotedlocation is defaulted to localhostdefault file format (tab delimited, newline-terminated, a value for each column in table)LOAD DATA INFILE syntaxLOAD DATA [LOCAL] INFILE 'file_name' [IGNORE | REPLACE] INTO TABLE table_name format_specifiers [IGNORE n LINES] [(column_list)];Specifying the Datafile Location without using LOCALDefault location assumed to be local to the server as MySQL reads the file directly.Absolute path on server: LOAD DATA INFILE '/path/to/file.txt' INTO TABLE table_name;If using a default database then relative path is relative to the default database: LOAD DATA INFILE 'file.txt' INTO TABLE table_name;If not using a default database then the relative path is relative the the specific database data directory: LOAD DATA INFILE './data_directory/file.txt' INTO TABLE table_name;Using LOCAL means the file is local to the MySQL client host. The MySQL client reads the file and sends to the server.Absolute path on clientRelative path on client is relative to the current directory.Special note on Windows ... the path separator is \, but MySQL treats backslash as escape. You can either use the / (forward) slash, i.e., LOAD DATA INFILE 'C:/path/to/file.txt' INTO TABLE table_name; or escape the backslash character, i.e., LOAD DATA INFILE 'C:\\path\\to\\file.txt' INTO TABLE table_name>Loading into Specific ColumnsLOAD DATA INFILE 'file.txt' INTO TABLE table_name (col1, col2);Note, if order in text file doesn't match order in table columns, you can switch the the insertion text order via specifying columns, i.e., LOAD DATA INFILE 'file.txt' INTO TABLE table_name (col2, col1);Skipping datafile lines or header row(s) with IGNORELOAD DATA INFILE 'file.txt' INTO TABLE table_name IGNORE n LINES;Dealing with duplicate records.Default behavior on a duplicate Key violation is to stop loading the file at the point of error. All records previously processed remain.IGNOREkeyword will load the entire file, but discard the records causing duplicate key violations.REPLACE keyword will also load the entire file, but will replace the records with duplicate key violations with the new information.Interpreting LOAD DATA INFILE statement resultRecords: Number of tables read from the file, but not necessarily the number of records input into the table.Deleted: Number of records replaced in the table when using the REPLACE keyword.Skipped: Number of input records ignored in the data file when using the IGNORE keyword.Warnings: Indicates possible problems in the input file, i.e., missing values, data conversions, etc. Can be a number greater than the number of records input as more than 1 error can be generated per record.PrivilegesYou need the INSERT privilege for the table in which you want to LOAD DATA INFILE for LOCAL files.You also need FILE privilege for data files which are located on the server.LOAD DATA INFILE is considered an efficient insert method

SELECT INTO OUTFILEAdding the INTO OUTFILE clause into a SELECT statement creates a file on the server with the SELECT result.Since the file is created on the server, the user must have FILE privileges.The file is created with MySQL server as the owner but world-readable. Also since it is owned by MySQL you might not be able to remove the file without server admin privileges. Also, since the file is world-readable, careful about sensitive data.Default format is

one line per row, delimited by tab characters, and lines terminated with newlines. Since the file is created on the server, to access the file you must have a login account on the server host. Not an issue if you only want to read the file back in via a LOAD DATA INFILE as the server has access to the file ... even if you don't.

Data Format Specifiers LOAD DATA INFILE format specifiers are listed after the table name. SELECT ... INTO OUTFILE format specifiers are listed after the output filename. Syntax for format specifiers is the same for both statements and can be used in any order. Default values are supplied if missing. FIELDS TERMINATED BY 'string' ENCLOSED BY 'char' / OPTIONALLY ENCLOSED BY 'char' ESCAPED BY 'char' LINES TERMINATED BY 'string'; Default values TERMINATED BY: tab ('\t') ENCLOSED BY / OPTIONALLY ENCLOSED BY: unquoted LINES TERMINATED BY: newline ('\n') Special Note to platform. Unix line terminator are usually '\n' (newline), MAC OS and OSX line terminators are usually '\r' (carriage return), and Windows line terminators are usually '\r\n' (carriage return and newline) ESCAPE BY: backslash ('\'). Escape sequences as understood by MySQL: \N (NULL) \0 (ASCII NUL byte) \b (backspace) \n (newline) \r (carriage return) \s (space) \' (single quote) \" (double quote) \\ (backslash) Example. To load a file with comma-separated values, with values quoted by double quote, and line terminated by carriage returns. To insert: LOAD DATA INFILE 'file.txt' INTO TABLE table_name FIELDS TERMINATED BY ',' ENCLOSED BY '"'; LINES TERMINATED BY '\r'; To write: SELECT * INTO OUTFILE 'file.txt' FIELDS TERMINATED BY ',' ENCLOSED BY '"'; LINES TERMINATED BY '\r' FROM table_name;

Importing and Exporting NULL Values LOAD DATA INFILE a \N appearing unquoted by itself as a column value is interpreted as NULL. MySQL converts empty values to a 0, empty string, or a '"zero"'; temporal value depending of the column type. SELECT ... INTO OUTFILE writes NULL values as \N

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, June 20, 2005

LAMPsig MySQL Class :: Class #7

Class #7 of the MySQL class is scheduled for this Monday, June 20 at 7:00pm at CalTek and will continue with the topic of database design theory and joins.

Additional information and links to class diagrams are continued in the further reading section.

Instructor:

Solomon Chang :: skevin521_AT_yahoo_(dot)_com

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

Syllabus is the same as in class #6 with the topic of database theory and joins being continued. We will go over some of the rules of normalization and create a 3NF database from data supplied in a spreadsheet.

Class materials are a spreadsheet and several diagrams in progressively greater states of database normalization.

Example data in spreadsheet (.doc) format, Example data in spreadsheet (.sxc) format

Database Design Modeling diagram examples(0NF) Zero Normal Form diagram(1NF) 1st Normal Form diagram(2NF) 2nd Normal Form diagram(2NF) 2nd Normal Form diagram improved for contact and ssn number handling(3NF) 3rd Normal Form diagram

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, June 13, 2005

LAMPsig MySQL Class :: Class #6

Class #6 of the MySQL class is scheduled for this Monday, June 13 at 7:00pm at CalTek and will cover some Theory and Joins. This is a two-part class and the topic will be continued next week.

Complete syllabus for Class #6 are continued in the further reading section.

Instructor:

Solomon Chang :: david_AT_peterbenjamin_(dot)_com

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

NOTE: This is a draft syllabus. The final syllabus will be posted very shortly :)

Class 6 – Relational Database structure and theory and Joins (Chapter 8) 15% exam material Introduction to Relational Database Structure Rules of normalization in table structure First Normal Form (1NF) Second Normal Form (2NF) Third Normal Form (3NF) Unique vs. Primary Keys Using Multiple Columns as Primary Keys

A Join between tables is an extension of a SELECT statement but involves the following complexities. FROM clause must list all tables needed to produce the query Must specify how to match up the records. The displayed columns can include (or not include) columns from the joined tables. Avoidance of ambiguous column names by ensuring that column names are either unique, aliased, or fully qualified. Order of tables with Inner Join doesn't matter, but does matter with an Outer Join (either Left or Right Join). With an Outer Left Join the reference table is on the left, and the table from which rows might be missing is on the right. An Outer Right Join corresponds the reference table is switched to the right and the table with expected empty rows on the left. If you have the ability to choose between an Inner Join and an Outer Join, the Inner Join allows the MySQL optimizer to choose the most efficient order for processing the tables.

Inner Join with Comma Operator Language associated with specific country(ies) would be easier to understand if the country name were included. SELECT CountryCode, Language FROM CountryLanguage; The country(ies) names are in a separate table. Language and Country name need to be JOINED. SELECT Code, Name FROM Country; Result is Country name and all the Languages spoken within that country. SELECT Name, Language FROM Country, Country WHERE CountryCode = Code; For an Inner Join the order in which the FROM clause names the tables doesn't matter. The column list display one column from each table. SELECT Code, Name, Continent, Language FROM CountryLanguage, Country WHERE CountryCode = Code; Output isn't sorted unless used with ORDER BY clause SELECT Name, Language FROM Country, Country WHERE CountryCode = Code ORDER BY Name; Using WHERE in Joins / Cartesian Join Limiting Join Output with AND SELECT Name, Language FROM CountryLanguage, Country WHERE CountryCode = Code and Language = 'Swedish'; (Countries where Swedish is spoken) SELECT Name, Language FROM CountryLanguage, Country WHERE CountryCode = Code and Language = 'Sweden'; (Languages spoken in Sweden) Functions in Joined SELECT statements. Using COUNT() and HAVING to restrict output to include only those countries where more than 10 languages are spoken. SELECT COUNT(*), Name FROM CountryLanguage, Country WHERE CountryCode = Code GROUP BY Name HAVING COUNT(*) > 10;

Inner Joins with INNER JOIN Keyword INNER JOIN with ON SELECT Name, Language FROM CountryLanguage INNER JOIN Country ON CountryCode = Code; INNER JOIN with USING(). Needed if joining

column name is the same in both tables. If joining more than three like named columns then list the column names
SELECT Name, Language FROM CountryLanguage INNER JOIN Country USING (Code);

Outer Joins / LEFT JOIN. Written with the LEFT JOIN operator using either ON or USING () Select output using the INNER JOIN and LEFT JOIN: SELECT Name, Language FROM Country INNER JOIN CountryLanguage ON Code = CountryCode; (Matching Records) SELECT Name, Language FROM Country LEFT JOIN CountryLanguage ON Code = CountryCode; (Matching Record plus Country Names with no Language -- NULL). SELECT Name, Language FROM Country LEFT JOIN CountryLanguage ON Code = CountryCode WHERE CountryCode IS NULL; (Only those Countries where no Language is specified).

Outer Joins / RIGHT JOIN. Same syntax as with LEFT JOIN with table position in statement reversed. To compose an Outer LEFT JOIN: SELECT Name, Language FROM Country LEFT JOIN CountryLanguage ON CountryCode = Code WHERE CountryCode IS NULL; To compose the same query as a RIGHT JOIN SELECT Name, Language FROM CountryLanguage RIGHT JOIN Country ON CountryCode = Code WHERE CountryCode IS NULL;

Converting Subqueries to Inner Joins. MySQL has sub-selects available starting from version 4.1. Prior versions requires some hoops to duplicate the function(s) of a sub-select. Some examples below. Inner Joins are used to find matches between tables. Sub-select: Select Name FROM Country WHERE Code IN (SELECT CountryCode FROM CountryLanguage); Convert to Inner Join (Note the duplicate Country Names): SELECT Name FROM Country, CountryLanguage WHERE Code = CountryCode; Convert to Inner Join using DISTINCT (Note no duplicate Country Names): SELECT DISTINCT Name FROM Country, CountryLanguage WHERE Code = CountryCode;

Converting Subqueries to Outer Joins. Same information as above, except that Outer Joins are used to find mismatches between tables. Sub-select: Select Name FROM Country WHERE Code NOT IN (SELECT CountryCode FROM CountryLanguage); Convert to Outer LEFT Join: SELECT Name FROM Country LEFT JOIN CountryLanguage ON Code = CountryCode WHERE CountryCode IS NULL; Convert to Outer RIGHT Join: SELECT Name FROM CountryLanguage RIGHT JOIN Country ON Code = CountryCode WHERE CountryCode IS NULL;

Sub-Selects and Subqueries (available in 4.1 and 5) (BEING WORKED ON)

Resolving Ambiguous Name Clashes. When joining tables there are occasional clashes in column or table names. For column names you can either fully qualify the column name with the table name, i.e., Qualifying Column Names SELECT Name, Name FROM Country, City WHERE Code = CountryCode; SELECT Country.Name, City.Name FROM Country, City WHERE Code = CountryCode; Qualifying and Aliasing Table names

Resolving Many-to-Many Relationships (BEING WORKED ON)

Multiple Table UPDATE and DELETE Statements (BEING WORKED ON)

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, June 6, 2005

LAMPsig MySQL Class :: Class #5

Class #5 of the MySQL class is scheduled for this Monday, June 6 at 7:00pm at CalTek and will cover Insert and Replace Statements.

Complete syllabus for Class #5 are continued in the further reading section.

Instructor:

David Benjamin :: david_AT_peterbenjamin_(dot)_com

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

CLASS #5 SYLLABUS

Class 5 – (Chap 7 / MySQL-M 13.1.1, 13.1.4, 13.1.6, 13.1.9, 13.1.10 Additional references on optimization (not covered in class): 7.2.16, 7.2.17)

10% exam material

INSERT Statement. Primary difference between INSERT and REPLACE is how duplicate records are handled. Violations of unique key values in INSERT are ignored and not inserted, but REPLACE will first delete the record containing the duplicate value, and then insert a new record. **Inserting Single Record** INSERT INTO table_name (column_list) VALUES (value_list); INSERT INTO table_name SET column_name1 = value1, column_name2 = value2; INSERT INTO table_name VALUES (values_list); value_list must match column(s) number and column(s) order INSERT INTO table_name () VALUES (); Creates a row into table_name using the default values **Inserting Multiple Records with a Single INSERT Statement** INSERT INTO table_name (column1, column2) VALUES (value1a, value1b), (value2a, value2b); MySQL will return extra information with multiple-row inserts. **Records:** number of rows inserted **Duplicates:** how many records were ignored because they contained duplicate unique key values. **Value** can be non-zero if statement includes the IGNORE keyword **Warnings:** number of problems found in data values ... can occur if values are converted. **Single Record and Multiple Record** are handled somewhat differently for purposed of error-handling. See Section 4.10.6, “Automatic Type Conversion and Value Clipping.”

REPLACE statement **Inserting Single Record** REPLACE INTO table_name (column_list) VALUES(value_list); REPLACE INTO table_name SET column_name1 = value1, column_name2 = value2; **Inserting Multiple Records with Single REPLACE Statement** REPLACE INTO table_name (column1, column2) VALUES (value1a, value1b), (value2a, value2b); MySQL will return extra information with multiple-row inserts **Query OK, X rows affected** X may be greater than the number of rows inserted as duplicate unique key rows are first deleted, and then inserted. **Replaces into tables with multiple columns with unique values may cause unexpected row deletion.**

UPDATE Statement. DANGER WILL ROBINSON!! Issuing an UPDATE statement without a WHERE clause updates every row in the table! As a safety you can start MySQL with the --safe-updates option UPDATE table_name SET column_name = value WHERE some_expression; UPDATE table_name SET column_name1 = value1, column_name2 = value2 WHERE some_expression; Using UPDATE with ORDER BY and LIMIT

Handling Illegal Values **Numeric:** out of range values are clipped to nearest value in range **String:** long strings are truncated to fit in column **Invalid values are converted to column default** NULL

DELETE and TRUNCATE Statement `DELETE FROM table_name;` / `DELETE FROM table_name WHERE some_expression;` Can be used to either delete all the rows from a table, or just selected rows when used with a WHERE clause Usually executed more slowly than TRUNCATE Returns true row count indicating number of records deleted. `TRUNCATE table_name;` / `TRUNCATE TABLE table_name;` Always completely empties a table Executes quickly Might return row count of zero rather than actual number of rows deleted Using DELETE with ORDER BY and LIMIT

Recap of Some Theories: A look at transactions **Transactions: What are they and why do they matter** A transaction is an isolated sequence of queries that can either all be saved to the database or all canceled and ignored. When a transaction is committed any changes within a transaction are made permanent. When a transaction is rolled back all changes are lost and the database reverts back to the state of the last successfully committed transaction. **ACID Compliance** **Atomicity:** database modifications must follow an all or nothing rule. Each transaction is said to be atomic. If one part of the transaction fails, the entire transaction fails. It is critical that the database management system maintain the atomic nature of transactions in spite of any DBMS, operating system or hardware failure. **Consistency:** only valid data will be written to the database. If, for some reason, a transaction is executed that violates the database's consistency rules, the entire transaction will be rolled back and the database will be restored to a state consistent with those rules. **Isolation:** requires that multiple transactions occurring at the same time not impact any other execution. **Durability:** ensures that any transaction committed to the database will not be lost. Durability is ensured through the use of database backups and transaction logs that facilitate the restoration of committed transactions in spite of any subsequent software or hardware failures.

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, May 23, 2005

LAMPsig MySQL Class :: Class #4

Class #4 of the MySQL class is scheduled for this Monday, May 23 at 7:00pm at CalTek and will cover Select Statements, Expressions and Functions, Misc SQL.

Complete syllabus for Class #4 are continued in the further reading section.

Instructor:

David Benjamin :: david_AT_peterbenjamin_(dot)_com

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

CLASS #4 SYLLABUS

Class 4 – Select Statements, Expressions and Functions, Misc SQL
(Chap 5 & 6 / MySQL-M 9.2, 12, 13.1.7, 13.1.8, A.5.4)
20% exam material (10% Chapter 5, 10% Chapter 6)

Example Sorting database sortingOrderExamples. Sql file will create three tables 'sortChar', 'sortEnum', and 'sortSet'. Please create a database called sortingOrderExamples and import. These tables will be used to demonstrate sorting options when dealing with Char vs. Binary Char, ENUM, and SET column types.

```
SELECT Select Syntax mysql> SELECT values_to_display
FROM table_name
WHERE expression
GROUP BY how_to_group
HAVING expression
ORDER BY sort_options
LIMIT row_count;
```

```
mysql> SELECT 2+2, REPEAT('x', 5), DATE_ADD('2001-01-01', INTERVAL 7 DAY), 1/0;
```

Using Aliases for Column Names AS is optional keyword Aliases *may* be quoted, multiple word Alias *must* be quoted You can refer to an Alias elsewhere in a query, e.g., GROUP BY, HAVING, or ORDER BY, but it *cannot* be used as part of the WHERE clause.

Restricting a selection Using WHERE

Using ORDER BY to Sort Natural (or default) Sort Order of Column Types Numeric: Ascending
Numeric Decimal: sorted numerically, even though DECIMAL values are stored in string form Date/Time: Ascending
Temporal (oldest values first) String Binary: Numeric Value of Bytes that make up the String String Non binary:
Lexical Enum: Internal Numeric Value based on Enum Set Order Set: Internal Number Value based on a more
complex Set Order

Limiting a Selection using LIMIT

Aggregate Functions. Using both with and without WHERE clause. Aggregate values for Empty Set

```
SUM() AVG() MIN() MAX() COUNT()
GROUP BY
```

```
HAVING
```

Using DISTINCT to Eliminate Duplicates
Concatenating SELECT Results with UNION

Pattern Matching with LIKE and WILDCARD Characters

Dealing with NULL NULL Values and Column Definitions NULL Values and NOT NULL Columns NULL Values in Expressions and Comparisons ISNULL(), IFNULL() NULL Values in Aggregate Functions are ignored except for the COUNT() Function. New in mysql 4.1 / 5 SUB-SELECTS. Will also be covered Class #6 & 7 (Chap. 8 Joins).

Using Reserved Words as Identifiers

Commentating in SQL files

Functions. Per the study guide, you are not expected to every detail about each function, but you are expected to know it's general behavior ABS() AES_DECRYPT()and AES_ENCRYPT() / DES_DECRYPT() and

DES_ENCRYPT() BIN() CEILING() CHAR() CHAR_LENGTH() CHARACTER_LENGTH() CONCAT()and

CONCAT_WS() CONV() CURRENT_DATE() and

CURRENT_TIME() DATE_ADD() DATE_FORMAT() DATE_SUB() DAYNAME() and DAYOFMONTH() and

DAYOFWEEK() and DAYOFYEAR() DECODE and

ENCODE ELT() EXPORT_SET() FIELD() FIND_IN_SET() FLOOR() FROM_DAYS() FROM_UNIXTIME() GREATER() HEX() HOUR() IF() IFNULL()and

ISNULL() IN() INSERT() INSTR() LCASE() LEAST() LEFT() LENGTH() LOAD_FILE() LENGTH() LOCATE() LOWER() LPAD() LTRIM() MAKE_SET() MD5() MID() MINUTE() MOD() MONTH() MONTHNAME() NOW() OCT() PA

SSWORD() PERIOD_ADD() PERIOD_DIFF() POSITION() POW() and

POWER() QUARTER() QUOTE() RAND() REPLACE() REVERSE() RIGHT() ROUND() RPAD() RTRIM() SEC_T

O_TIME() SECOND() SIGN() SUBSTRING()and

SUBSTRING_INDEX() TIME_FORMAT() TIME_TO_SEC() TO_DAYS() TRIM() TRUNCATE() UCASE() UNIX_TIM

ESTAMP() UPPER() WEEK() WEEKDAY() YEAR() YEARWEEK() VERSION()

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, May 16, 2005

LAMPsig MySQL Class :: Class #3

Class #3 of the MySQL class is scheduled for this Monday, May 16 at 7:00pm at CalTek and will cover the Storage Engines, DDL (Data Definition Language), Primary Keys, Columns Types, and Indexes.

Complete syllabus for Class #3 are continued in the further reading section.

Instructor:

David Rolston :: david_AT_gizmola_(dot)_com

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

CLASS #3 SYLLABUS

Class 3 – Data Definition Language.

(Chapter 4 / MySQL-M 1.2.4, 1.5.6.1, 3.3.2, 3.6.9, 11, 13.2, 13.5.4, 14, 15)

[NOTE: MySQL-M 7 covers optimization and how to use INDEXES effectively. Optimization techniques are not covered in this class.]

20% exam material

General Database and Table Properties MySQL associates each database on the server with a directory under the data directory. The directory has the same name as the database it represents. The directory contains all the files associated with the database, i.e., indexes and tables. Databases cannot be nested, i.e., one database cannot contain another. Tables consist of rows and columns. A table can be empty, i.e., it can have 0 rows, but it must have at least one column. Every table is associated with a format file in the database directory. The format file name is the same as the table name followed with a .frm extension. Depending on the storage engine MySQL might create additional files. MyISAM storage engine creates data and index file(s) named tablename.MYD and tablename.MYI respectively. InnoDB storage engine creates the .frm file, but stores the data and index information in an InnoDB tablespace.

Storage Engine and Table Types MyISAM Tables represented on disk with .frm format file, an .MYD datafile and an .MYI index file. most flexible AUTO-INCREMENT column handling can be set up to handle MERGE tables can be converted to compressed, read-only tables supports FULL TEXT searching uses table level locking for query contentions and write queries. InnoDB Tables represented on disk with .frm format file in database directory, as well as data and index storage in the InnoDB tablespace. The table space is shared by all InnoDB tables. Supports transactions with full ACID compliance provides auto-recovery after server or host crash supports foreign keys and referential integrity, including cascaded deletes and updates uses multi-versioning and row-level locking for query contentions. MERGE Tables a MERGE table is a collection of identically structured MyISAM tables represented on disk by an .frm format file and an .MRG file located in the database directory a query on a MERGE table acts as a query on all the MyISAM tables of which it consists. A MERGE table creates a logical entity that can exceed the maximum MyISAM table size BDB (Berkeley DB) Tables (Note: BDB storage engine not enabled in LAMPsig/Knoppix LiveCD installation) represented on disk by an .frm format file and a .db file that stores data and index information located in database directory. Supports transactions with full ACID compliance uses page-level locking for query contention. HEAP (Memory) Tables represented on disk by a .frm format file in database directory. Table data and indexes are stored in memory In-memory storage results in fast performance HEAP table contents do not survive a restart of the server. The structure survives, but it contains 0 data rows after a restart.

Limits on Database Components MySQL doesn't place limits on number of databases, but the OS or filesystem might due to limits on number of sub-directories allowed in filesystem tree. MySQL databases are represented on disk in sub-directories MySQL doesn't place limits on number of files in directory, tho the OS or filesystem might. MyISAM tables are represented on disk in files under the database subdirectory MySQL does place limits on size of individual tables. Techniques around this limitation would include: MERGE tables (for MyISAM storage engine) RAID setup (for datafiles only as index tables are stored in a single file) Convert MyISAM tables to InnoDB tables as InnoDB allows for

larger datafiles. For OS or filesystem limitations see if OS dependent adjustments are possible.

Identifier Syntax. Identifiers identify a specific database elements, e.g, databases, tables, table columns, aliases, and (sometimes) indexes. Legal Characters Qualifiers for Table and Column Names. Sometimes qualifiers are necessary to resolve ambiguity.

```
SELECT * FROM Country;
SELECT * FROM world.Country; SELECT Name FROM Country;
SELECT Country.Name FROM Country;
SELECT world.Country.Name FROM world.Country;
CREATE and DROP DATABASE
```

```
CREATE TABLE Syntax Temporary
DROP TABLE
```

ALTER TABLE Adding and Dropping Columns Modifying Existing Columns Renaming a Table Specifying Multiple Alterations

Creating and Dropping Indexes Four types of Indexes non-unique index UNIQUE PRIMARY KEY FULL TEXT Defining at Table-Creation Time Creating and Using Primary Keys Modifying Indexes of Existing Tables Column Types Numeric / Integer TINYINT SMALLINT MEDIUMINT INT BIGINT Numeric / Floating-Point and Fixed-Decimal FLOAT DOUBLE DECIMAL String Binary and Nonbinary String Characteristics CHAR VARCHAR BLOB TEXT ENUM SET Date and

Time DATE TIME DATETIME TIMESTAMP YEAR Column Options UNSIGNED ZEROFILL AUTO_INCREMENT BINARY NULL and NOT NULL DEFAULT PRIMARY KEY and UNIQUE Using AUTO_INCREMENT

Automatic Type Conversion and Value Clipping

DESCRIBE and SHOW to display table structures

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, May 9, 2005

LAMPsig MySQL Class :: Class #2

Class #2 of the MySQL class is scheduled for this Monday, May 9nd at 7:00pm at CalTek and will cover the various clients used to access a MySQL server.

Due to space limitations the class is now full so no more students are currently being accepted.

Complete syllabus for Class #2 are continued in the further reading section.

Instructor:

Steve Glasser :: steve_AT_fpig_(dot)_net

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Solomon Chang :: skevin521_AT_yahoo_(dot)_com

CLASS #2 SYLLABUS

Class 2 – Using the MySQL client. (Chapter 3 / MySQL-M 3.1, 4, 25)

10% exam material

Invoking Command-Line Client Programs from Shell Prompt short vs. long shell> mysql -V or -h shell> mysql --version or --host options followed by values --host=myhost.example.com -h myhost.example.com -hmyhost.example.com default values and changing default values through my.cnf Windows my.ini /my.cnf *nix global /etc/my.cnf, local /home/user/.my.cnf

Connection Parameter Options Host (default value localhost) --host=host_name / -h host_name / -hhost_name Port (default value 3306) --port=port_number / -P port_number Socket (default value *nix /tmp/mysql.sock, Windows pipe name 'MySQL') --socket=socket_name / -S socket_name Windows NT servers --enable-named-pipe option User (default value *nix system login name, Windows 'ODBC') --user=user_name / -u user_name / -uuser_name Password (no default value) --password=password_value or -ppassword_value Note spacing (-ppassword_value) OK to omit the password_value in initial connection parameter as with either span STYLE="font-weight: medium">–password or -p flag a password prompt will appear If you omit the password option then your MySQL account must be specifically set up to allow you to connect without a password. Compress --compress / -C Using Custom Option Files --defaults-file=file_location Must be first option after the command name For an option specified multiple times in the same or separate option file(s), the option found last takes precedence. Options on the command line take precedence over options found in options files

Selecting Default (or Current) Database Naming on command line when connecting shell> mysql --user=username --password database_name

shell> mysql --user=root --password world shell> mysqldump --user=username --password database_name > database_name.sql

shell> mysqldump --user=root --password world > world.sql Specifying with sql statements mysql> SELECT * FROM database_name.database_table; Select or Change default database mysql> USE database_name; mysql> SELECT * FROM database_table;

MySQL Client Program Interactive mode useful for day-to-day, one time queries, etc. Batch mode running prewritten queries, complex queries that are difficult to type, or to be run automatically by scheduler without user intervention.

Using MySQL Client Interactively Connect to server and database shell> mysql -u root -p -h localhost world A mysql> prompt will appear ready to accept sql statements mysql> is primary prompt -> generic secondary prompt: waiting for next line of statement or statement terminator '>' specific secondary prompt: waiting for end of single-quoted string '>' specific secondary prompt: waiting for end of double-quoted string '>' specific secondary prompt: waiting for end of backtick-quoted identifier. Canceling a statement with clear query \c sql statement need to be terminated in order for the MySQL server to execute. The terminators ; and \g are interchangeable mysql> SELECT * FROM world; mysql> SELECT * FROM world\g The terminator \G

(capital G) displays query results with each column value on a separate line
mysql> SELECT * FROM world\G
Executing statements from text or source file. If statement in source file end with an error, mysql ignores remainder of file. To execute entire file regardless of statement errors use the --force or -f option. A source file can contain other execute other source files, but careful of endless source loops.
mysql> SOURCE source_path/file_name;
shell> mysql database_name < source_path/file_name
MySQL output
Interactive Mode displays with bars and dashes in tabular format
Batch Mode produces tab-delimited output between data values
Output options
--batch or -B to display with tabs even if used interactively
--raw or -r option to suppress newline and carriage return escape-sequences like \n or \r
--table or -t to display tabular format in table even if used in batch mode
--html or -H to produce output in HTML format
--xml or -X for XML format
Summary of Client Commands.
Long form is case insensitive, short form case sensitive.
Long form can only be executed at the mysql> primary prompt unless you invoke with --named-commands options. Short form can be executed on any input line
mysql> sql statements: SELECT,INSERT, UPDATE, and DELETE
mysql> QUIT or \q
mysql> SOURCE or \.
mysql> STATUS or \s
mysql> HELP or \h
mysql> CLEAR or \c
Using --safe-updates option (--i-am-a-dummy) Limits UPDATE and DELETE statement to those either containing a WHERE clause or a LIMIT clause. Restricts returned rows to 1,000 unless the statement has a LIMIT clause. Multiple-table SELECT statements are allowed only if MySQL will examine no more than 1,000,000 rows to process the query.
mysqlimport

mysqldump

Reloading mysqldump output

Checking tables with mysqlcheck and myisamchk

MySQL Connectivity Drivers
MySQL Connector/ODBC
MySQL Connector/J

Posted by Sharon Lake in Classes: MySQL at 19:00

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

Monday, May 2, 2005

LAMPsig MySQL Class :: Class #1

Class #1 of the MySQL class is scheduled for this Monday, May 2nd at 7:00pm at CalTek. Address, directions, and parking information can be found [here](#).

Due to space limitations the class is now full so no more students are currently being accepted.

Class #1 will cover an introduction to the class, resources available, MySQL & MySQL AB, MySQL Software and Components, and an intro in getting information about MySQL database and table information, and data directories structure.

Complete syllabus for Class #1 are continued in the further reading section.

Instructor:

David Benjamin :: david_AT_peterbenjamin_(dot)_com

Class Contact:

Sharon Lake :: sharon_AT_linuxchixla_(dot)_org

Lab Help:

Steve Glasser :: steve_AT_fpig_(dot)_net

CLASS #1 SYLLABUS

Class 1 – Introduction. (Chapter 1 & 2 / MySQL-M 1, 2, 3.3, 3.4, 5, 8, 21, 22)
20% exam material (5% Chapter 1, 10% Chapter 2)

Class Goals

To be introduced to the basic MySQL knowledge needed to pass the MySQL Core Certification. The MySQL Core Certification includes testing the knowledge needed to maintain a basic MySQL installation and use MySQL as a backend database for applications.

MySQL Core and Professional Certifications

Certifications available

MySQL certification information can be found on the MySQL site.<http://www.mysql.com/training/certification/>
Core Certification - tests basic MySQL knowledge sufficient to main a basic MySQL installation and create application p6t5yy7rograms that use MySQL as a backend database

Professional Certification - covers installation, large table maintenance, storage engines, and optimizations.

MySQL Certification Study Guide (ISBN: 0672326329)

Study Guide Errata: <http://www.mysql.com/training/certification/studyguides/errata.html>

Starting February 1st, 2005, the MySQL Certification Exams will be updated to reflect the changes that have taken place with the release of MySQL Server version 4.1.

4.1 Test Update information (<http://www.mysql.com/training/certification/41update.html>)

MySQL Version 4.1 Exam Preparation addendum pdf
(<http://www.mysql.com/training/certification/mysql-certification-41addendum.pdf>)

MySQL Certification Self Test <http://www.mysql.com/training/certification/selftest/core/index.php>

Knoppix LiveCD. A Knoppix LiveCD will be available with the MySQL 4.1 server and client, the MySQL manual, and the sample 'World' database already pre-installed.

Installed Components

MySQL server -- user: root password: lampsig
MySQL client
MySQL Manual (pdf and html versions available). <http://dev.mysql.com/doc/>
MySQL Certificate Study Guide Sample Chapter and Errata
<http://www.mysql.com/training/certification/studyguides/errata.html>
phpMyAdmin <http://localhost/phpmyadmin>
Sample world Database installed. <http://dev.mysql.com/doc/world-setup.html>
<http://dev.mysql.com/get/Downloads/Manual/world.sql.gz/from/pick>

What is MySQL

MySQL and MySQL AB

Dual Licensing

GPLCommercial

Major Program Components (mysqld server, mysql client programs)

Server: mysqld / mysqld-max) & mysqld-nt / mysql-max-nt)

Client(s)

mysql

MySQL Control Center (MySQLcc) -- No longer in development.

MySQL Administrator -- Server administration

MySQL Query Browser -- SQL browser

mysqlimport

mysqladmin

mysqlcheck

Difference Between Major MySQL Distributions

States: pre-alpha, alpha, beta, gamma, production

Production Versions: MySQL 4.0, 4.1, and 5)

MySQL APIs

Interfaces provided within MySQL (C client library, ODBC connector, JDBC connector)

Third party interfaces (PERL DBD::mysql, PHP, Python, etc.) While these APIs are downloadable from the MySQL site, they do not receive official support.

Interpreting the DESCRIBE output

Field

Type

Null

Key

Default

Extra

Server / client architecture

Creating New Database

Root user vs. User. Brief overview of Grant tables (Note: not covered in MySQL Core Certification)

Demonstration: create Sample World Database from MySQL command line

Demonstration: populating the world database by importing Sample World Database .sql file

Where is/are the data directory(ies)

Intro to switching default databases with use (covered in depth in MySQL client)

Posted by Sharon Lake in Classes: MySQL at 19:00

Monday, April 25, 2005

LAMPsig MySQL Class :: Optional Lab

The optional lab for the upcoming MySQL class is scheduled for this Monday, April 25th at 7:00pm at CalTek. Address, directions, and parking information can be found [here](#).

Due to space limitations the class is now full so no more students are currently being accepted.

This optional pre-class is designed to help bootstrap those new to databases, MySQL, or a Linux LiveCD environment. Also, if you are planning on using an alternative MySQL server this would be the place to confirm access and trouble shoot any connection or version problems.

The lab goals are continued in further reading section.

Instructor:

David Benjamin :: [david_AT_peterbenjamin_\(dot\)_com](mailto:david_AT_peterbenjamin_(dot)_com)

Class Contact:

Sharon Lake :: [sharon_AT_linuxchixla_\(dot\)_org](mailto:sharon_AT_linuxchixla_(dot)_org)

MySQL CLASS OPTIONAL LAB GOALS

KNOPPIX LIVECD: To feel comfortable enough with Knoppix Linux LiveCD so that work in the LAMPsig MySQL class can be accomplished.

Basic

- How to boot from CD

- How to access installed MySQL manual

- How to access the mysql client

- How to restart the mysql server if it dies

- For non-Linux users some basics in getting around the file system

- Intro to some text editors

- Reasons and work arounds for non-persistent data

Optional

- How to start up misc applications
- How to save to persistent media (Note: this is dependent on hardware configurations and will be different for each user. Because of this the ability to save persistent data is not guaranteed)

- How to connect to the internet while using Knoppix CD (Note: this is dependent on hardware configurations and will be different for each user. Because of this the ability to connect to the internet not guaranteed)

NON-KNOPPIX CONFIGURATIONS: For those choosing to access other MySQL server options.

- Confirm access to MySQL server

- Confirm MySQL version

- Confirm ability to install sample 'world' database

Sample World Database

- Knoppix: How to use installed sample 'world' database

- Other: How to import world.sql into existing database (Note: import methods are dependent on specific MySQL server access. Help will be provided, but cannot be guaranteed)

Know how to access other resources

- Introduction to MySQL manual

Blog Export: LAMPsig, <http://www.lampsig.org/new/>

MySQL Certification Study Guide and other books available through
Amazon.com, Bookpool.com, Half.com, or the Library
MySQL.com
Mailing List

Have a basic introduction to MySQL

Know what is a database.
Know how MySQL fits into the family of databases
Be introduced to the server/client relationship

Posted by Sharon Lake in Classes: MySQL at 19:00

Friday, April 15. 2005

New LAMPsig Class Announcement :: MySQL

A new course is being organized to introduce the basic MySQL knowledge needed to pass the MySQL Core Certification Exam. The MySQL Core Certification Exam covers the knowledge needed to maintain a basic MySQL installation and use MySQL as a backend database for applications.

The class will start with an optional informal lab on Monday, April 25, 2005 that will introduce databases in general, the MySQL database specifically, and the LAMPsig/Knoppix learning CD. The MySQL class proper will start on Monday, May 2, 2005 and be conducted over 8 weekly sessions.

The syllabus will follow the first 9 chapters of the MySQL Certification Study Guide (ISBN 0672326329) quite closely. A sample chapter (Chapter 3) is available from MySQL here. If you are interested in pursuing the MySQL Core Certification Exam, getting this book as a study guide is highly recommended. If you are only looking for an intensive introduction to MySQL SQL, then the MySQL manual will be sufficient for class.

The primary learning tool is the MySQL client communicating directly with the MySQL server via SQL commands. This course is meant to be hands-on and a Knoppix LiveCD will be provided with the MySQL 4.1 server and client, the MySQL manual, and the sample 'World' database already pre-installed. Note that access to a MySQL server and installation issues unrelated to the Knoppix LiveCD will need to be handled via the LAMPsig user or class mailing list. MySQL installation is not part of the Core Certification Exam and this material will not be covered.

The class will be held on Monday evenings at 7:00pm - 9:00pm at CalTek (832 S. Hoover St., Los Angeles, CA 90005). RSVPs are being requested as space is a concern. Please respond to [sharon_AT_linuxchixla_\(dot\)_org](mailto:sharon_AT_linuxchixla_(dot)_org) if you would like to reserve a spot in this class.

The class schedule (subject to change):

Date	Topic
April 25, 2005	Optional informal introductory lab covering MySQL / Knoppix LiveCD / General Database concepts
May 2, 2005	Class intro / MySQL and MySQL AB / MySQL software and components (Chap 1 & 2)
May 9, 2005	Using the MySQL client (Chap 3)
May 16, 2005	Data Definition Language (Chap 4)
May 23, 2005	Select Statements / Expressions and Functions (Chap 5 & 6)(no class Memorial day)
June 6, 2005	Insert and Replace Statements (Chap 7)
June 13, 2005	Relational DB structure and theory and Joins (Chap 8)
June 20, 2005	Relational DB structure and theory and Joins cont. (Chap 8)
June 27, 2005	Importing and Exporting Data (Chap 9)

Posted by Sharon Lake in Classes: MySQL at 02:42