

## 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 &ndash; 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 '&quot;'; LINES TERMINATED BY '\r';  
To write:  
SELECT \* INTO OUTFILE 'file.txt' FIELDS TERMINATED BY ',' ENCLOSED BY '&quot;'; 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 '&quot;zero&quot;'; 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

Saturday, June 18, 2005

### **June Meeting - Open for Business: Under the Hood**

Main Topic: Open for Business: Under the Hood - Si Chen

In this second talk on Open For Business, Si will take us "under the hood" and show us how to build applications with Open For Business:

- philosophy behind OFBiz applications
- the OFBiz data model
- how to access the database and business logic
- how web applications are built

Background material for the talk can be found at <http://www.opensourcestrategies.com/ofbiz/tutorials.php>.

Opening Topic: MySQL Certification - Solomon Chang

Solomon recently passed both the Pearson VUE MySQL Core certification exam and the Professional exam. He will give us the benefit of his experience. I'm sure if we ask nicely, he will tell where he hid his crib notes.

Posted by Jim Workman at 20:34

## 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 &ndash; 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

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

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 &ndash; (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, &ldquo;Automatic Type Conversion and Value Clipping.&rdquo;

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