

# 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