

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

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