The Learning House

427 S 4th Street #300

Louisville KY 40202

Relational Databases Unit

Lesson 6: MySQL Installation

Query Strategies

SOFTWARE GUILD

Objectives

- Handle building complex queries

To Complete this Lesson

- You will use the SWCCorp database to complete this lesson.

Steps to Building Queries

1. **Organize**
   - Go through the tables and make a list of all of the tables and fields that contain the data you are looking for.
2. **Join**
   - Build all of the table joins, using a select * statement.
3. **Filter**
   - Add any filter statements necessary to your where clause.
4. **Itemize**
   - We rarely need everything; go back to your select * and swap it out for just the fields required.

Example

Let's say we want to get the FirstName, LastName, City, and State for our corporate employees located in Washington

```
-- Test first table logic
SELECT *
FROM Location;

-- Test second table with join
SELECT *
FROM Location
    INNER JOIN Employee
    ON Location.LocationID = Employee.LocationID;

-- Test all tables with criteria
SELECT *
FROM Location
    INNER JOIN Employee
    ON Location.LocationID = Employee.LocationID
WHERE `State` = 'WA';

-- Choose the fields
SELECT FirstName, LastName, City, `State`
FROM Location
    INNER JOIN Employee
    ON Location.LocationID = Employee.LocationID
WHERE `State` = 'WA';
```

Table Aliasing

- Using the AS keyword, we can alias a table name to save some typing.

```
SELECT FirstName, LastName, City, `State`
FROM Location AS loc
    INNER JOIN Employee AS emp
    ON loc.LocationID = emp.LocationID
WHERE `State` = 'WA';
```

The AS is Optional

- This is also valid. Ultimately, follow the pattern your team sets.

```
SELECT FirstName, LastName, City, `State`
FROM Location loc
    INNER JOIN Employee emp
    ON loc.LocationID = emp.LocationID
WHERE `State` = 'WA';
```

Aliases Work in the Select List as Well

- It is only required when you have the same field name in both tables so the server knows which column you want. ("Ambiguous error")

```
SELECT emp.FirstName, emp.LastName, loc.City, loc.`State`
FROM Location loc
    INNER JOIN Employee emp
    ON loc.LocationID = emp.LocationID
WHERE loc.`State` = 'WA';
```

Cross joins and unmatched queries

AND NOW FOR SOMETHING DIFFERENT

CROSS JOINS

Sometimes we don't have a relatable field between tables, but we are looking for every combination between records in tables.

In this case, we can use a *Cross Join*, also known as a *Cartesian Combination.*

It basically reads as: "give me every possible combination of records between the two tables."

Try the queries to the right in SWCCorp

```
SELECT *
FROM Employee
WHERE EmpID IN (1,2);

SELECT * FROM MgmtTraining;

SELECT *
FROM Employee
CROSS JOIN MgmtTraining
WHERE EmpID IN (1,2);
```

Unmatched Records Queries

This is a common SQL pattern. We have a table with a list of elements that is optionally joined to another table.

We want to know which elements in the first table has no matches in the second.

LEFT JOIN and check for NULL!

```
-- Location without employees
SELECT *
FROM Location AS l
    LEFT OUTER JOIN Employee AS e
    ON l.LocationID = e.LocationID
WHERE e.LocationID IS NULL
```

Lab Exercises (SWCCorp)

1. Write a query to show every combination of employee and location.
2. Find a list of all the Employees who have never found a Grant.

Fin

- Next up: Data Definition Language