LAB 3: DML Queries

JOINS and AGGREGATION

1

What is a join?

- SQL allows you to cross reference information in tables.
- There are two methods to do this:
  - **Compare & Contrast:** Compare values in a given table against another and get a matching pair result table from the data.
  - **JOIN:** Join the two tables into a temporary table based on some rules.

2

Example Movie DB

- The Movie DB schema…

```
Inventory (TapeID,MovieID)
Movies (MovieID, MovieName)
Suppliers (SupplierID, SupplierName)
MovieSupplier (SupplierID, MovieID, Price)
Orders (OrderID, SupplierID, MovieID, Copies)
```

This database is for a movie company. This database keeps track of the possible movies available on video. This database also keeps track of different suppliers in the company and the videos they have ordered.

A. Compare & Contrast Tables

- This is done by putting a statement in the WHERE clause of a query.
- This will simply compare the two tables and add the matching pairs to the result set.

Example: Give me the list of all the movies in the inventory.

```
SELECT DISTINCT M.MovieName
FROM Movies M, Inventory I
WHERE M.MovieID = I.MovieID
```

More like using the Cartesian Product operator and then looking to compare values using the '=' condition

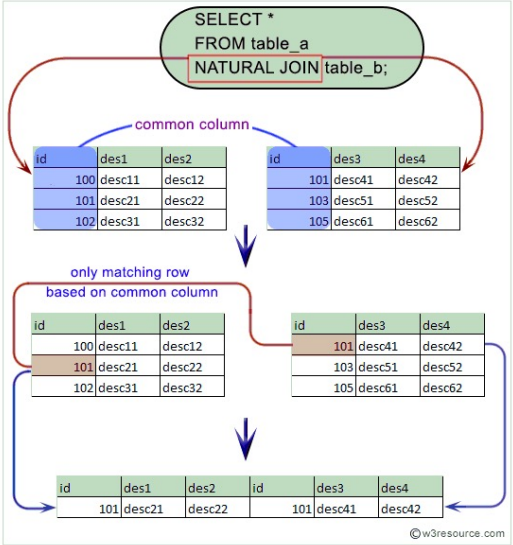
B. SQL Command: JOIN

- We introduce the types of joins SQL offers
  - `NATURAL JOIN`
  - `JOIN … ON (or INNER JOIN … ON)`
  - `LEFT JOIN … ON`
  - `RIGHT JOIN … ON`

  } Outer joins of Relational Algebra

- **`NATURAL JOIN`**
  - Does the same thing as compare & contrast.
  - Syntax : `<Table1> NATURAL JOIN <Table2>`
  - Attributes with the same name of associate tables will be used to join the tables.
  - Common Attributes occur just once.

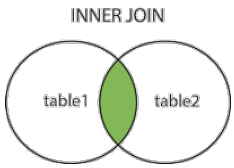
NATURAL JOIN



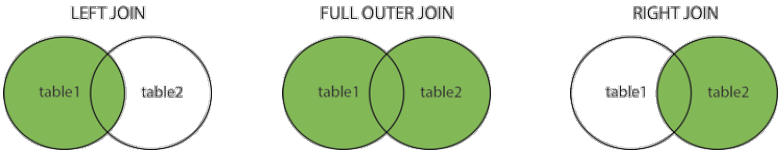
https://www.w3resource.com/mysql/advance-query-in-mysql/mysql-natural-join.php

SQL Command: JOIN

- There are two classes of JOINs in MySQL.
  - **INNER**



  - **OUTER**



7 / 12

7

C. What about JOIN or INNER JOIN?

JOIN…ON (or INNER JOIN…ON)

- Does the same thing as NATURAL JOIN.
- Syntax : `<Table1> [INNER] JOIN <Table2> ON <Condition>`
- Can handle multiple conditions for joining tables using **AND** clause
- Can handle user specified attributes using the **USING** clause

8 / 12

8

D. What about OUTER JOINS: LEFT JOIN / RIGHT JOIN

- Idea of these Joins
  - The result table comes from two tables...the table to **the LEFT** and the table to **the RIGHT** of the **JOIN**

- Syntax:
  - LEFT JOIN...ON Syntax

```
<Left Table> LEFT JOIN <Right Table> ON <Condition>
```

  - RIGHT JOIN...ON Syntax

```
<Left Table> RIGHT JOIN <Right Table> ON <Condition>
```

9 / 12

9

Using Outer JOINS

- STEP 1: Syntax is similar to an <INNER> JOIN...
  - LEFT JOIN Syntax

```
<Left Table> LEFT JOIN <Right Table> ON <Condition>
```

  - RIGHT JOIN Syntax

```
<Left Table> RIGHT JOIN <Right Table> ON <Condition>
```

- STEP 2: After results generated useful if you're looking for non-matching pairs...
  - **IS NULL**

    This can be used in the WHERE clause to check if an attribute to some record has a null value.
  - **IS NOT NULL**

    This also can be used in the WHERE clause...

10 / 12

10

Example

Given...

MOVIES

MOVIEID	MOVIENAME
1	STAR WARS
2	EMPIRE STRIKES BACK
3	RETURN OF THE JEDI

INVENTORY

TAPEID	MOVIEID
1	1
2	3
3	3

QUERY: List movie titles are not in the movie inventory

First Try...

```
SELECT *
FROM Movie M LEFT JOIN Inventory I ON M.MovieID = I.MovieID;
```

Gives the result...

MOVIEID	MOVIENAME	TAPEID
1	STAR WARS	1
3	RETURN OF THE JEDI	2
3	RETURN OF THE JEDI	3
2	EMPIRE STRIKES BACK	NULL

Answer & A New Question...

What movie titles are not in the movie inventory?

Second Query Try...only look for the null valued answers...

```
SELECT *
FROM Movie M LEFT JOIN Inventory I ON M.MovieID = I.MovieID
WHERE I.TapeID IS NULL;
```

Gives the result...

MOVIEID	MOVIENAME	TAPEID
2	EMPIRE STRIKES BACK.	NULL

Thus, just change the SELECT statement to what you want, and you're done!

Try this query...

Question: What are the names of the movies in the movie inventory?
(Hint: Do I have to watch for duplicates?)

What is aggregation?

- Aggregation is the ability to do simple mathematical functions on groups of data in the result table.
- Methods provided in MySQL...
  - SUM : Adds up any grouped values
  - MIN : Finds the smallest value in the set given
  - MAX : Finds the largest value in the set given
  - AVG : Finds the mean value of the set given
  - COUNT : Counts the number of records in the set given
- To understand how to use aggregation, you must understand how a query is executed...

13

Understanding a Query...

- Query Format

```
SELECT <Column Name(s)>
FROM <Tables/Joined Tables>
WHERE <Condition(s)>
GROUP BY <Column Name (s)> HAVING <Having Clause>
```

- Execution Order of the Query...
  - 1st: It finds the tables/joined tables it needs in the FROM clause.
  - 2nd: It evaluates row actions defined in the WHERE clause to produce a temporary result table.
  - 3rd: It uses the GROUP BY and HAVING definitions to know what further actions to take.
  - 4th: It uses the SELECT clause to know what rows to show.

14

Aggregation cont.

- This can be used in the SELECT or HAVING clause…

Question: What is the average price of a movie?

```
SELECT AVG(Price) AS Average_Price
FROM MovieSupplier;
```

Question: What movie suppliers have a movie that costs less than $9?

```
SELECT DISTINCT S.SupplierName
FROM MovieSupplier MS JOIN Supplier S ON S.SupplierID =
MS.SupplierID
GROUP BY MS.SupplierName HAVING MIN(MS.Price) < 9.00;
```