

MySQL

Overview

- ▶ MySQL
- ▶ DDL
 - Create
 - Alter
- ▶ DML
 - Insert
 - Select
 - Update
 - Delete
- ▶ DDL(again)
 - Drop
 - Truncate

Overview

► Joins

- Cross Join
- Inner Join
- Natural Join
- Outer Join
 - Left Outer Join
 - Right Outer Join
 - Full Outer Join

MySql

- ▶ Are Sql and MySQL are different?
- ▶ SQL – Sequential Query Language
- ▶ Database
 - What ?
 - Why ?

Create

- ▶ Create Database
 - Create DATABASE company_DB
- ▶ Create Tables
 - **CREATE TABLE** department (DepartmentID INT, DepartmentName VARCHAR(20));
 - **CREATE TABLE** employee (LastName VARCHAR(20), DepartmentID INT);

Alter

▶ Alter

- `ALTER TABLE employee ADD FirstName VARCHAR(20);`
- `ALTER TABLE employee DROP COLUMN FirstName;`

Insert

- ▶ Insert a row into table
 - `INSERT INTO department VALUES(31, 'Sales');`
`INSERT INTO department VALUES(33, 'Engineering');`
 - `INSERT INTO employee VALUES('Rafferty', 31);`

Select

▶ Select

- `SELECT` DepartmentName from department;
- `SELECT` * `FROM` employee;
- `SELECT` * `FROM` employee `where` DepartmentID = 31;

Update

► Update

- **UPDATE** department **SET** DepartmentName = 'logistics' **WHERE** DepartmentID = '31'
- **UPDATE** employee **SET** DepartmentID = 31 **WHERE** LastName = 'Jones'

Delete

▶ Delete

- `DELETE FROM employee where LastName = 'Jones'`
- `DELETE FROM department where DepartmentName = 'Logistics' OR DepartmentID = '31'`

Drop

- ▶ Drop
 - **DROP** employees;

Truncate

- ▶ Truncate

- ▶ `TRUNCATE` employees;

- ▶ Does same work as “unconditional Delete *” statement but differs in action.

- `DELETE * FROM` employees

Joins

- ▶ Join is Joining tables
- ▶ Why joins ?
 - To combines the information from two or more tables

Two Example Tables

Employee table		Department table	
LastName	DepartmentID	DepartmentID	DepartmentName
Rafferty	31	31	Sales
Jones	33	33	Engineering
Heisenberg	33	34	Clerical
Robinson	34	35	Marketing
Smith	34		
John	NULL		

NOTE : DepartmentID column of the Department table is the primary key, while Employee.DepartmentID is a foreign key.

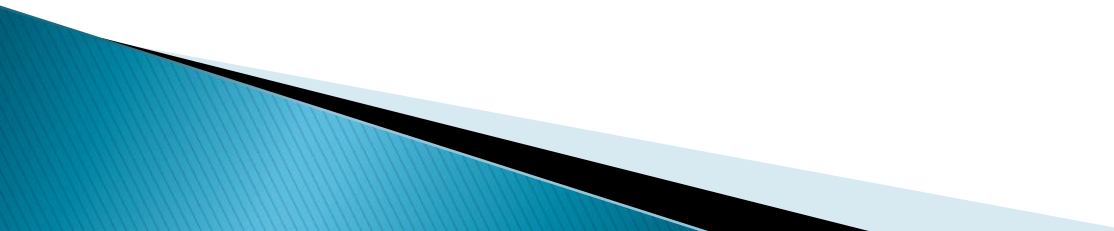
Cross Join

- ▶ CROSS JOIN returns the Cartesian product of rows from tables in the join. It will produce rows which combine each row from the first table with each row from the second table.
- ▶ `SELECT * FROM employee CROSS JOIN department;`
or
`SELECT * FROM employee, department;`

Cross Join

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Rafferty	31	Sales	31
Jones	33	Sales	31
Heisenberg	33	Sales	31
Smith	34	Sales	31
Robinson	34	Sales	31
John	NULL	Sales	31
Rafferty	31	Engineering	33
Jones	33	Engineering	33
Heisenberg	33	Engineering	33
Smith	34	Engineering	33
Robinson	34	Engineering	33
John	NULL	Engineering	33
Rafferty	31	Clerical	34
Jones	33	Clerical	34
Heisenberg	33	Clerical	34
Smith	34	Clerical	34

Inner Join

- ▶ Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate.
 - ▶ The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate
- 

Inner Join

▶ **SELECT * FROM** employee **INNER JOIN** department **ON** employee.DepartmentID = department.DepartmentID;

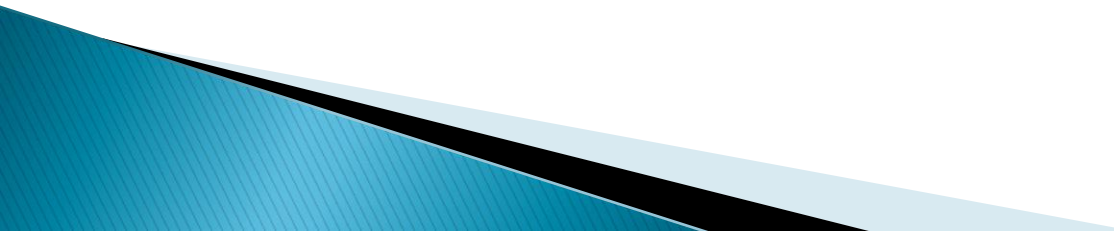
Or

▶ **SELECT * FROM** employee, department **WHERE** employee.DepartmentID = department.DepartmentID;

Inner Join

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Robinson	34	Clerical	34
Jones	33	Engineering	33
Smith	34	Clerical	34
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31

Inner Join and NULL

- ▶ Since NULL will never match any other value (not even NULL itself), unless the join condition explicitly uses the IS NULL or IS NOT NULL predicates.
 - ▶ The Inner join can only be safely used in a database that enforces referential integrity or where the join fields are guaranteed not to be NULL.
 - ▶ A left outer join can usually be substituted for an inner join when the join field in one table may contain NULL values.
- 

Equi-join

- ▶ An **equi-join** is a specific type of comparator-based join, that uses only equality comparisons in the join-predicate.
- ▶ **SELECT * FROM** employee **JOIN** department
ON employee.DepartmentID =
department.DepartmentID;
Or
- ▶ **SELECT * FROM** employee, department
WHERE employee.DepartmentID =
department.DepartmentID;

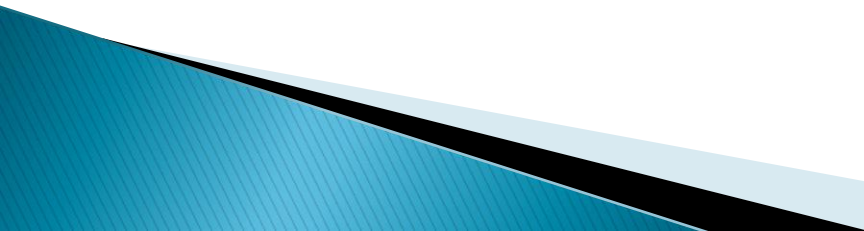
Natural Join

- ▶ A natural join is a type of equi-join where the join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables.
- ▶ `SELECT * FROM employee NATURAL JOIN department;`

Natural Join

DepartmentID	Employee.LastName	Department.DepartmentName
34	Smith	Clerical
33	Jones	Engineering
34	Robinson	Clerical
33	Heisenberg	Engineering
31	Rafferty	Sales

Outer Join

- ▶ An **outer join** does not require each record in the two joined tables to have a matching record. The joined table retains each record even if no other matching record exists.
 - ▶ Depending on which table's rows are retained (left, right, or both).
 - Left outer joins
 - Right outer joins
 - Full outer joins
- 

Left outer join

- ▶ “A Left Outer Join B” always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B)
- ▶ `SELECT * FROM employee LEFT OUTER JOIN department ON employee.DepartmentID = department.DepartmentID;`

Left Outer Join

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
<i>John</i>	NULL	NULL	NULL
Heisenberg	33	Engineering	33

Right outer join

- ▶ Every row from the "right" table (B) will appear in the joined table at least once.
- ▶ If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.
- ▶ `SELECT * FROM employee RIGHT OUTER JOIN department ON employee.DepartmentID = department.DepartmentID;`

Right Outer Join

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	Marketing	35

Full Outer Join

- ▶ Effect of applying both left and right outer joins
- ▶ `SELECT * FROM employee LEFT OUTER JOIN department ON employee.DepartmentID = department.DepartmentID`
`UNION`
`SELECT * FROM employee RIGHT OUTER JOIN department ON employee.DepartmentID = department.DepartmentID;`

Full Outer Join

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
<i>John</i>	NULL	NULL	NULL
Heisenberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	<i>Marketing</i>	35

References

- ▶ <http://en.wikipedia.org/wiki/MySQL> [Online]
- ▶ <http://dev.mysql.com/doc/refman/5.6/en/sql-syntax.html> [online]
- ▶ <http://www.w3schools.com/sql/> [online]
- ▶ “Learning MySQL” by by Seyed M.M. (Saied) Tahaghoghi, Hugh Williams (O'Reilly' Publication) [Book]
- ▶ “Learning PHP, Mysql, JavaScript and CSS” by Robin Nixon, (O'Reilly' Publication) [Book]

Thank You