

Basic CRUD in MySQL Server

Create, Retrieve, Update, Delete – Using SQL Queries



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>



sli.do
#java-db

Table of Contents

1. Query Basics
2. Retrieving Data
3. Writing Data in Tables
4. Modifying Existing Records





Query Basics

SQL Introduction

SQL Queries – Few Examples

- Select first, last name and job title about employees:

```
SELECT first_name, last_name, job_title FROM employees;
```

- Select projects which start on 01-06-2003:

```
SELECT * FROM projects WHERE start_date='2003-06-01';
```

- Inserting data into table:

```
INSERT INTO projects(name, start_date)  
VALUES('Introduction to SQL Course', '2006-01-01');
```

SQL Queries – Few Examples

- Update end date of specific projects:

```
UPDATE projects  
  SET end_date = '2006-08-31'  
 WHERE start_date = '2006-01-01';
```

- Delete specific projects:

```
DELETE FROM projects  
WHERE start_date = '2006-01-01';
```



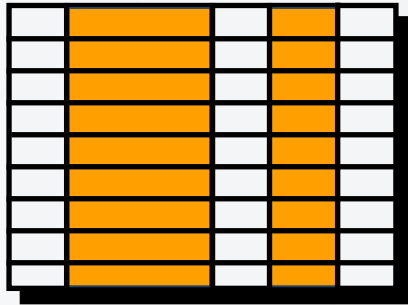
Retrieving Data

Using SQL SELECT

Capabilities of SQL SELECT

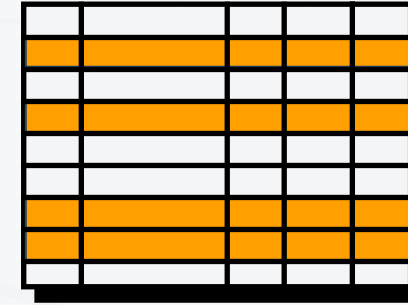
Projection

Take a subset of the columns



Selection

Take a subset of the rows



Join

Combine tables by
some column

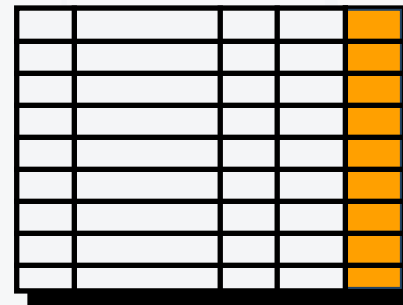


Table 1




Table 2

SELECT – Examples

- Selecting all columns from the "employees" table

id	first_name	last_name	job_title	department_id	salary
1	John	Smith	Manager	1	900
2	John	Johnson	Customer Service	1	880
3	Smith	Johnson	Porter	2	1100
...

```
SELECT * FROM employees;
```

List of columns (* for all)

Table name

Problem: Select Employee Information

- Write a query to **select** all employees from "Hotel" database
 - **Retrieve** information about their **id**, **first_name**, **last_name** and **job_title**
 - **Ordered** by id
- Note: Query **Hotel** database

id	first_name	last_name	job_title
1	John	Smith	Manager
2	John	Johnson	Customer Service
3	Smith	Johnson	Porter
...

Solution: Select Employee Information

```
SELECT id, first_name, last_name, job_title
```

List of columns

```
FROM employees
```

Table name

```
ORDER BY id;
```

- **Aliases** rename a table or a column heading:

```
SELECT e.id AS 'No.',  
       e.first_name AS 'First Name',  
       e.last_name AS 'Last Name',  
       e.job_title AS 'Job Title'  
FROM employees AS e ORDER BY id;
```

- **concat()** - returns the string that results from concatenating the arguments
 - String literals are enclosed in ['](**single quotes**)
 - Table and column names containing special symbols use [`](**backtick**)

```
SELECT concat(`first_name`, ' ', `last_name`) AS 'full_name',  
       `job_title` as 'Job Title',  
       `id` AS 'No.'  
FROM `employees`;
```

- Another function of concatenation is **concat_ws()** - stands for concatenate with **separator** and is a special form of CONCAT()

Separator

Arguments

```
SELECT concat_ws(' ', `first_name`, `last_name`, `job_title`)
AS 'full_name',
`job_title` AS 'Job Title',
`id` AS 'No.'
FROM `employees`;
```

- Skip any **NULL** values after the separator argument.

Problem: Select Employees with Filter

- Find information about all employees, listing their:
 - **Full Name**
 - **Job title**
 - **Salary**
- Use **concatenation** to display first and last names as **one field**
- Note: Query **Hotel** database

Solution: Select Employees with Filter

Concatenation

```
SELECT concat(`first_name`, ' ', `last_name`) AS  
    'Full name',  
    `job_title` AS 'Job title',  
    `salary` AS 'Salary'  
FROM `employees` WHERE salary > 1000;
```

Column alias

Filtering the Selected Rows

- Use **DISTINCT** to eliminate duplicate results

```
SELECT DISTINCT `department_id`  
FROM `employees`;
```

- You can filter rows by specific conditions using the **WHERE** clause

```
SELECT `last_name`, `department_id`  
FROM `employees`  
WHERE `department_id` = 1;
```

- Other **logical operators** can be used for better control

```
SELECT `last_name`, `salary`  
FROM `employees`  
WHERE `salary` <= 20000;
```


- Conditions can be combined using **NOT**, **OR**, **AND** and brackets

```
SELECT `last_name` FROM `employees`  
WHERE NOT (`manager_id` = 3 OR `manager_id` = 4);
```

- Using **BETWEEN** operator to specify a range:

```
SELECT `last_name`, `salary` FROM `employees`  
WHERE `salary` BETWEEN 20000 AND 22000;
```

- Using **IN / NOT IN** to specify a set of values:

```
SELECT `first_name`, `last_name`, `manager_id`  
FROM `employees`  
WHERE `manager_id` IN (109, 3, 16);
```

Problem: Select Employees by Multiple Filters

- Write a query to **retrieve** information about employees, order by id
 - Who are in **department 4**
 - Have salary **higher or equal to 1000**

	id	first_name	last_name	job_title	department_id	salary
	3	Smith	Johnson	Porter	4	1100
	9	Nikolay	Ivanov	Housekeeping	4	1600



```
SELECT * FROM employees AS e
WHERE e.department_id = 4 AND e.salary >= 1000;
```

Comparing with NULL

- **NULL** is a special value that means missing value
 - Not the same as 0 or a blank space
- Checking for **NULL** values

This is always false!

```
SELECT `last_name`, `manager_id`  
FROM `employees`  
WHERE `manager_id` = NULL;
```

```
SELECT `last_name`, `manager_id`  
FROM `employees`  
WHERE `manager_id` IS NULL;
```

```
SELECT `last_name`, `manager_id`  
FROM `employees`  
WHERE `manager_id` IS NOT NULL;
```

Sorting with ORDER BY

- Sort rows with the **ORDER BY** clause
 - ASC**: ascending order, default
 - DESC**: descending order

```
SELECT `last_name`, `hire_date`  
FROM `employees`  
ORDER BY `hire_date`;
```

```
SELECT `last_name`, `hire_date`  
FROM `employees`  
ORDER BY `hire_date` DESC;
```

LastName	HireDate
Gilbert	1998-07-31
Brown	1999-02-26
Tamburello	1999-12-12
...	...

LastName	HireDate
Valdez	2005-07-01
Tsoflias	2005-07-01
Abbas	2005-04-15
...	...

- Views are **virtual tables** made from others tables, views or joins between them
- Usage:
 - To simplify writing complex queries
 - To limit access to data for certain users

Table 1		
Column 1	Column 2	Column 3

Table 2		
Column 1	Column 2	Column 3



v_table1_table2		
Column 1	Column 2	Column 3

- Get employee names and salaries, by department

```
CREATE VIEW `v_hr_result_set` AS  
SELECT  
    CONCAT(`first_name`, ' ', `last_name`) AS 'Full Name',  
    `salary`  
FROM `employees` ORDER BY `department_id`;
```

```
SELECT * FROM `v_hr_result_set`;
```

Problem: Top Paid Employee

- Create a **view** that selects all information about the **top paid employee**
 - Name the view **v_top_paid_employee**
 - Note: Query **Hotel** database



id	first_name	last_name	job_title	department_id	salary
8	Pedro	Petrov	Front Desk Super...	1	2100

Solution: Top Paid Employee

```
CREATE VIEW `v_top_paid_employee`  
AS  
  SELECT * FROM `employees`  
  ORDER BY `salary` DESC LIMIT 1;
```

Sorting column

Greatest value first

```
SELECT * FROM `v_top_paid_employee`;
```



Writing Data in Tables

Using SQL INSERT

- The SQL **INSERT** command

Values for all columns

```
INSERT INTO `towns` VALUES (33, 'Paris');
```

```
INSERT INTO projects(`name`, `start_date`)  
VALUES ('Reflective Jacket', NOW())
```

Specify columns

- **Bulk data** can be recorded in a single query, separated by comma

```
INSERT INTO `employees_projects`  
VALUES (229, 1),  
      (229, 2),  
      (229, 3), ...
```

- You can use existing records to create a **new table**

```
CREATE TABLE `customer_contacts`  
AS SELECT `customer_id`, `first_name`, `email`, `phone`  
FROM `customers`;
```

New table name

Existing source

- Or into an existing table

```
INSERT INTO projects(name, start_date)  
SELECT CONCAT(name, ' ', 'Restructuring'), NOW()  
FROM departments;
```

List of columns



Modifying Existing Records

Using SQL UPDATE and DELETE

- The SQL **UPDATE** command

New values

```
UPDATE `employees`  
SET `last_name` = 'Brown'  
WHERE `employee_id` = 1;
```

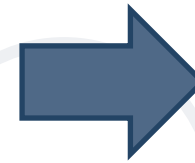
```
UPDATE `employees`  
SET `salary` = `salary` * 1.10,  
    `job_title` = CONCAT('Senior', ' ', `job_title`)  
WHERE `department_id` = 3;
```

- Note: Don't forget the **WHERE** clause!

Problem: Update Employees Salary

- **Update** all employees salaries whose **job_title** is "Manager" by **100**

```
UPDATE employees
SET salary = salary + 100
WHERE job_title = 'Manager';
SELECT salary
FROM employees;
```



	salary
▶	1000
	880
	1100
	1100
	1500.23
	990
	1800
	2100
	1600

- Deleting specific rows from a table

```
DELETE FROM `employees`  
WHERE `employee_id` = 1;
```

Condition

- Note: Don't forget the **WHERE** clause!
- Delete all rows from a table (**TRUNCATE** works faster than **DELETE**)

```
TRUNCATE TABLE users;
```


Problem: Delete from Table

- **Delete** all employees from the "employees" table who are in department **2 or 1**
- **Order** by id

	id	first_name	last_name	job_title	department_id	salary
▶	3	Smith	Johnson	Porter	4	1100
	6	Ivan	Petrov	Waiter	3	990
	7	Jack	Jackson	Executive Chef	3	1800
	9	Nikolay	Ivanov	Housekeeping	4	1600

Solution: Delete from Table

Delete Data

OR Condition

```
DELETE FROM employees  
WHERE department_id = 1  
OR department_id = 2;  
SELECT * FROM employees
```

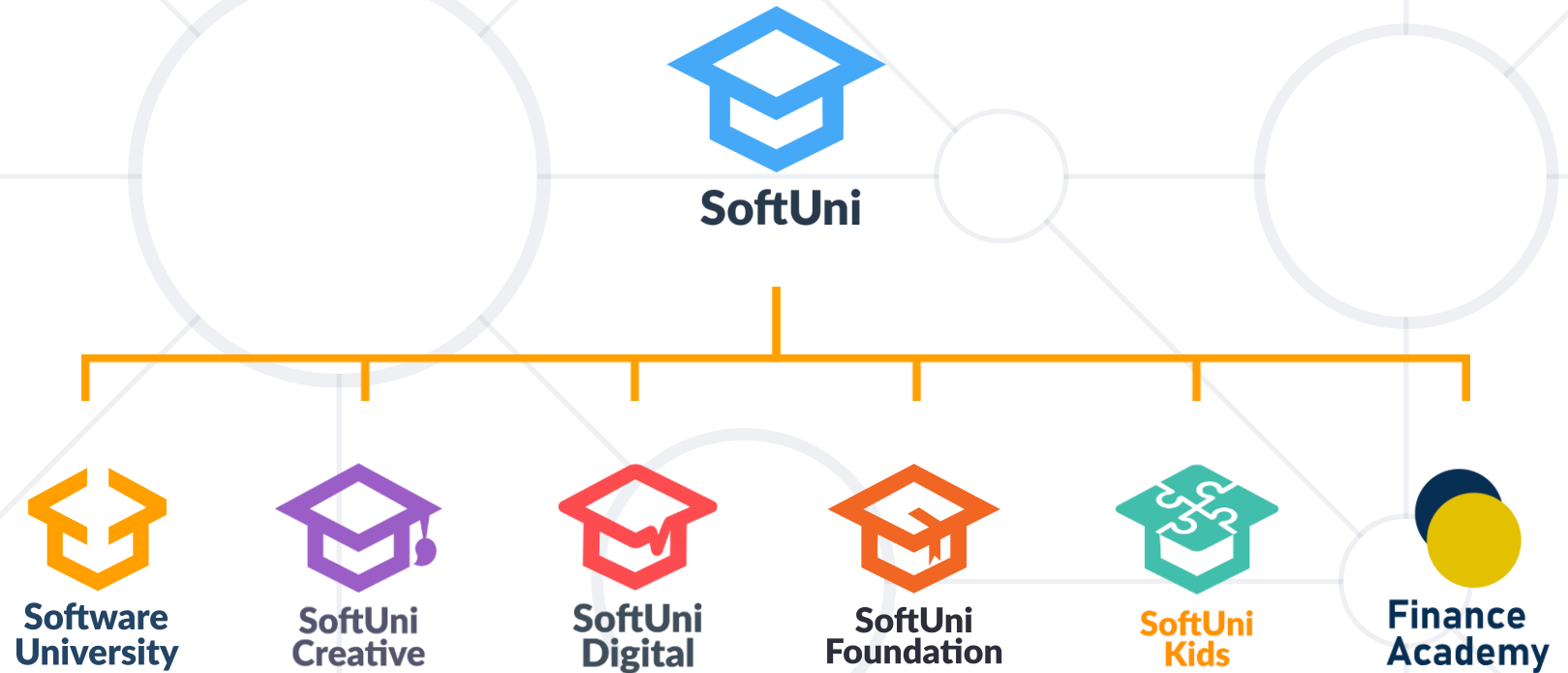
- We can easily manipulate our database with SQL queries

```
SELECT *  
FROM `projects`  
WHERE `start_date` = '2006-01-01';
```

- Queries provide a flexible and powerful method to manipulate records



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

