Welcome back

SQL Essentials

CODE YOUR LIFE

Roadmap

RDBMS
ER Model

SQL Server
DDL

DML
Operator

Function
SQL Clause

Join data

Sub Query

Best
Practice

Previous lecture

- Sub queries
- Advanced operators
- Rules of sub query
- Practice

What will we explore today?

- Paging data
- Backup the database
- Stored procedure
- Trigger
- Index demo
- SQL injection
- Best practice

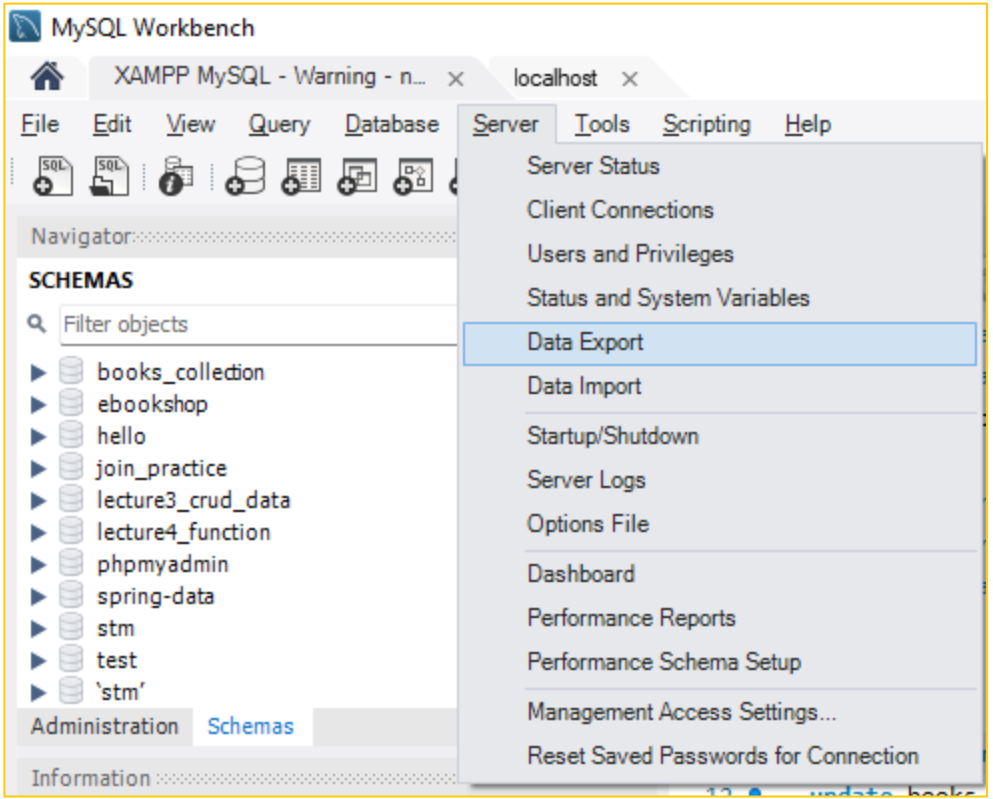
```sql
SELECT ID_KhachHang, FullName
FROM Customer
ORDER BY ID_KhachHang
```

ID_KhachHang	FullName
1	NGUYỄN HUỆ
2	PHÙNG ĐẠO
3	TRỊNH HOÀN
4	TRƯƠNG THÍ
5	HOÀNG HUÂN
6	LÊ HẬU
7	VÕ TÀI
8	VŨ NAM
9	NGION HÒA
10	VUAIS ĐÀO
11	TINKS HOÀNG
12	TRUNG THI
13	HUIAN HÀO
14	LIANG HIẾU
15	VIỄN TOÀN
16	LONG PHONG
NULL	NULL

```sql
SELECT ID_KhachHang, FullName
FROM Customer
ORDER BY ID_KhachHang
LIMIT 5 OFFSET 5;
```

ID_KhachHang	FullName
6	LÊ HẬU
7	VÕ TÀI
8	VŨ NAM
9	NGION HÒA
10	VUAIS ĐÀO
NULL	NULL

Backup the database



Dump file / SQL file

Stored Procedures

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- You can also pass parameters to a stored procedure

Stored Procedures syntax

```
CREATE PROCEDURE procedure_name(IN input1 datatype1, OUT result datatype2)
BEGIN
sql_statement
END;
```

```
DELIMITER //

CREATE PROCEDURE MyFirstStoreProcedure()
BEGIN
SELECT ID_KHACHHANG,  FullName
FROM Customer
END //

DELIMITER ;
```

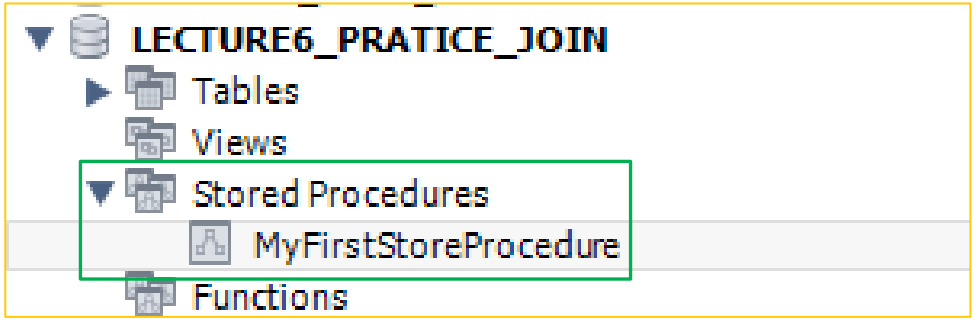
“Run” Stored Procedures

```
CALL procedure_name();
```

```
CALL MyFirstStoreProcedure()
```

	ID_KhachHang	FullName
▶	1	NGUYỄN HUỆ
	2	PHÙNG ĐẠO
	3	TRỊNH HOÀN
	4	TRƯƠNG THÍ
	5	HOÀNG HUÂN
	6	LÊ HẬU
	7	VÕ TÀI
	8	VŨ NAM
	9	NGION HÒA
	10	VUAIS ĐÀO
	11	TINKS HOÀNG
	12	TRUNG THI
	13	HUIAN HÀO
	14	LIANG HIẾU
	15	VIỄN TOÀN
	16	LONG PHONG

Where is it?


LECTURE6_PRATICE_JOIN
Tables
Views
Stored Procedures
MyFirstStoreProcedure
Functions

With parameters

```
DELIMITER //
CREATE PROCEDURE CongHaiSo(IN a INT, IN b INT, OUT tong INT)
BEGIN
                SET tong = a + b;
END //
DELIMITER ;
```

2

	@tong
▶	8

1

```
CALL CongHaiSo(5, 3, @tong);
SELECT @tong;
```

TRIGGER

- A special kind of stored procedure, which “reacts” to certain actions we make in the database.
- The main idea behind triggers is that they always perform an action in case some events happen.

Add LastModifyDate column

```
ALTER TABLE Customer
ADD LastModifyDate datetime
```

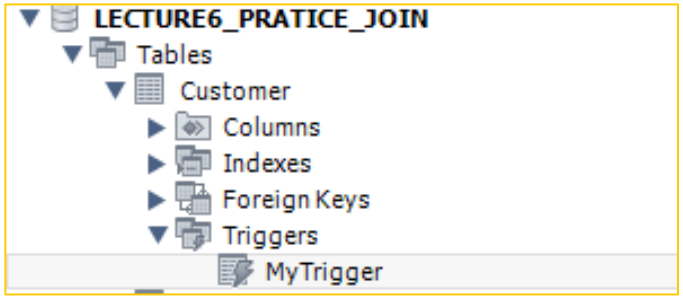
ID_KhachHang	FirstName	LastName	Gender	FullName	DateOfBirth	Address	LastModifyDate
1	NGUYỄN	HUỆ	NAM	NGUYỄN HUỆ	1992-01-10	THỦ ĐỨC - TP.HCM	NULL
2	PHÙNG	ĐẠO	NAM	PHÙNG ĐẠO	1993-02-13	THỦ ĐỨC - TP.HCM	NULL
3	TRỊNH	HOÀN	NAM	TRỊNH HOÀN	1994-02-15	THỦ ĐỨC - TP.HCM	NULL
4	TRƯƠNG	THÍ	NAM	TRƯƠNG THÍ	1995-02-17	THỦ ĐỨC - TP.HCM	NULL
5	HOÀNG	HUÂN	NAM	HOÀNG HUÂN	1995-04-13	QUẬN 9 - TP.HCM	NULL
6	LÊ	HẬU	NAM	LÊ HẬU	1994-05-19	QUẬN 10 - TP.HCM	NULL
7	VÕ	TÀI	NỮ	VÕ TÀI	1997-10-22	QUẬN 11 - TP.HCM	NULL
8	VŨ	NAM	NỮ	VŨ NAM	1990-11-21	QUẬN 12 - TP.HCM	NULL
9	NGION	HÒA	NỮ	NGION HÒA	1991-09-21	QUẬN 11 - TP.HCM	NULL
10	VUAIS	ĐÀO	NỮ	VUAIS ĐÀO	1993-08-21	BA ĐÌNH - TP.HN	NULL
11	TINKS	HOÀNG	NỮ	TINKS HOÀNG	1994-12-21	BA ĐÌNH - TP.HN	NULL
12	TRUNG	THI	NỮ	TRUNG THI	1995-03-21	BA ĐÌNH - TP.HN	NULL
13	HUIAN	HÀO	NỮ	HUIAN HÀO	1995-07-21	BA VÌ - TP.HN	NULL
14	LIANG	HIẾU	NỮ	LIANG HIẾU	1994-06-12	BA VÌ - TP.HN	NULL
15	VIỄN	TOÀN	NỮ	VIỄN TOÀN	1997-10-16	BA VÌ - TP.HN	NULL
16	LONG	PHONG	NỮ	LONG PHONG	1990-11-19	BA VÌ - TP.HN	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Create Trigger

```
DELIMITER //

CREATE TRIGGER MyTrigger BEFORE UPDATE ON Customer
FOR EACH ROW
BEGIN
    SET NEW.LastModifyDate = NOW()
END

DELIMITER ;
```



Update 1 record

```sql
UPDATE Customer
SET Address = 'TEST UPDATE ADDRESS'
WHERE ID_KhachHang = 1
```

1

```sql
SELECT Address, LastModifyDate
FROM Customer
```

Address	LastModifyDate
TEST UPDATE ADDRESS	2023-10-02 02:30:03
THỦ ĐỨC - TP.HCM	NULL
THỦ ĐỨC - TP.HCM	NULL
THỦ ĐỨC - TP.HCM	NULL
QUẬN 9 - TP.HCM	NULL
QUẬN 10 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
QUẬN 12 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL

Update 2 records

```sql
UPDATE Customer
SET Address = 'TEST'
WHERE ID_KhachHang = 3 OR ID_KhachHang = 4
```

```sql
SELECT Address, LastModifyDate
FROM Customer
```

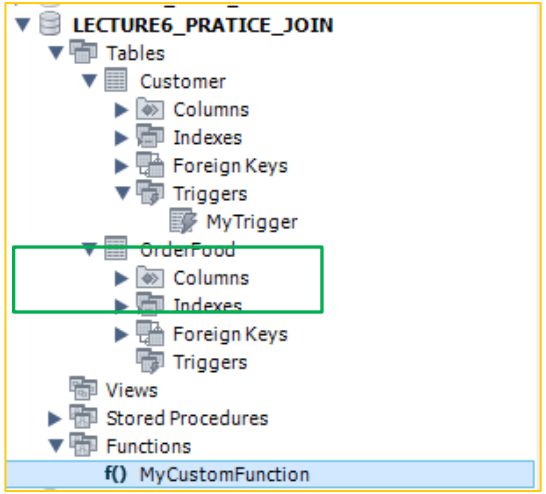
Address	LastModifyDate
TEST UPDATE ADDRESS	2023-10-02 02:30:03
THỦ ĐỨC - TP.HCM	NULL
TEST	2023-10-02 02:33:11
TEST	2023-10-02 02:33:11
QUẬN 9 - TP.HCM	NULL
QUẬN 10 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
QUẬN 12 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL

User-defined functions

User-defined functions (UDFs): allow you to create **custom functions** that extend the functionality of the MySQL database system

```
DELIMITER //

CREATE FUNCTION MyCustomFunction(YourMoney int)
RETURNS varchar(50) DETERMINISTIC
BEGIN
    DECLARE Result varchar(50);
        IF (YourMoney >= 1000000000) THEN -- 1 tỷ
            SET Result = N'bạn xứng đáng có 10 người yêu';
        ELSE
            SET Result = N'nỗ lực thì sẽ có ngày thành công';
        END IF;
    RETURN Result;
END;
//
DELIMITER ;
```



Call it

```sql
SELECT MyCustomFunction(200) AS 'Kết quả'
```

	Kết quả
1	nỗ lực thì sẽ có ngày thành công

```sql
SELECT MyCustomFunction(1000000000) AS 'Kết quả'
```

	Kết quả
1	bạn xứng đáng có 10 người yêu

Index

- Indexes are used to retrieve data from the database more quickly than otherwise.
- The users cannot see the indexes, they are just used to speed up searches/queries.

Index demo

```
CREATE TABLE MyTable (
        ID INT PRIMARY KEY AUTO_INCREMENT,
        Column1 VARCHAR(255),
        Column2 VARCHAR(255)
);
```

Index demo

```
-- Insert 1000 rows into the table
DELIMITER //
CREATE PROCEDURE Insert1000Rows()
BEGIN
	DECLARE counter INT DEFAULT 0;
	WHILE counter < 1000 DO
		INSERT INTO MyTable (Column1, Column2)
				VALUES (CONCAT('Value ', counter), CONCAT('Value ', counter));
		SET counter = counter + 1;
		END WHILE;
END //
DELIMITER ;

-- Execute the procedure to insert 1000 rows
CALL Insert1000Rows();
```

Index demo

```
-- Tạo index trên bảng MyTableIndexCREATE INDEX
IDX_HELLO_INDEX ON MyTable (Column1);

-- Enable query profilingSET profiling = 1;
-- Query without the index
SELECT * FROM MyTable WHERE Column2 =  'Value 999';
-- Query with the index
SELECT * FROM MyTable WHERE Column1 = 'Value 999';
-- Show query profiling results
SHOW PROFILES;
```

Query_ID	Duration	Query
2106	0.00289230	SET profiling = 1
2107	0.00869450	SELECT * FROM MyTable WHERE Column2 = 'Value 999' LIMIT 0, 1000
2108	0.00043840	SELECT * FROM MyTable WHERE Column1 = 'Value 999' LIMIT 0, 1000
2109	0.00018530	SET profiling = 1
2110	0.00085600	SELECT * FROM MyTable WHERE Column2 = 'Value 999' LIMIT 0, 1000
2111	0.00069970	SELECT * FROM MyTable WHERE Column1 = 'Value 999' LIMIT 0, 1000

SQL Injection

```
CREATE TABLE IF NOT EXISTS Users (  ID INT AUTO_INCREMENT PRIMARY KEY,
Username VARCHAR(50) NOT NULL,  Password VARCHAR(50) NOT NULL);

INSERT INTO Users (Username, Password) VALUES ('USER_1', '123');
```

```sql
SELECT Username
FROM Users
WHERE Username = 'USER_1' AND Password = '123'
```

```sql
SELECT Username
FROM Users
WHERE Username = 'USER_1' OR 1=1; --AND Password = '456'
```

SQL Injection

```
-- LOGIN TO THE SYSTEM
SELECT * FROM Users WHERE Username = 'USER_1' AND Password = '123';

-- MODIFY THE QUERY WITH Username = 'USER_1' OR 1=1 --' AND Password = '456‘
SELECT * FROM Users WHERE Username = 'USER_1' OR 1=1; -- ALWAYS TRUE -> SQL injection;
```

```
-- using parameter to improve the security
SET @username = 'USER_1';
SET @password = '123';
SELECT * FROM Users WHERE Username = @username AND Password = @password;
```

Benefit of Coding Standards

- Enhanced Efficiency
- Risk of project failure is reduced
- Minimal Complexity
- Easy to Maintain
- Bug Rectification
- A Comprehensive Look
- Cost-Efficient

SQL Comment

- Always use comment to explain your code.
- Use natural/human language in comment to easy understand.
- All comments should be same format.
- Break comment line to avoid horizontal scroll bar.

Naming conventions

- Must be simple, meaningful & do not conflict with system name.
- Names must begin with a letter and may not end with an underscore.

Format code

- Always use **UPPERCASE** for the reserved keywords like SELECT and WHERE.
- Break line to avoid horizontal scroll bar. It recommended that start line with **KEYWORD**

Avoid SELECT *

```sql
-- Bad query
SELECT *
FROM table_name;
```

```sql
-- Better query
SELECT col1, col2, col3
FROM table_name;
```

DISTINCT

```sql
-- Bad query
SELECT DISTINCT FirstName, LastName
FROM Customers;
```

```sql
-- Better query
SELECT ID, FirstName, LastName
FROM Customers;
```

Careful with HAVING

- The HAVING clause is used to filter the rows after all the rows are selected and it is used like a filter.
- It works by going through the **final result table** of the query, parsing out the rows that don't meet the HAVING condition.

Careful with HAVING

```sql
USE LECTURE5_JOIN;
-- Bad query
SELECT CustomerID, COUNT(CustomerID) AS OrderCount
FROM CustomerOrder
GROUP BY CustomerID
HAVING CustomerID = 1 OR CustomerID = 3;
```

```sql
USE LECTURE5_JOIN;
-- Better query
SELECT CustomerID, COUNT(CustomerID) AS OrderCount
FROM CustomerOrder
WHERE CustomerID = 1 OR CustomerID = 3
GROUP BY CustomerID
```

COUNT, AVG, SUM

- COUNT(1) & COUNT (*) are the same
- Ignore NULL value

```
SELECT COUNT(column_name) FROM table_name;  -- Counts all rows, including NULLs.
SELECT AVG(column_name) FROM table_name;    -- Ignores NULL values.
SELECT SUM(column_name) FROM table_name;    -- Ignores NULL values.
```

Avoid using UNION

- Avoid using UNION clause whenever possible
- UNION clause causes sorting data in the table and that slows down SQL execution.
- Use UNION ALL and remove duplicates

Simplicity

```sql
-- Bad query
SELECT OrderID, FoodName, DeliveryAddressID
FROM CustomerOrder
WHERE DeliveryAddressID = 1 + 1;
```

```sql
-- Better query
SELECT OrderID, FoodName, DeliveryAddressID
FROM CustomerOrder
WHERE DeliveryAddressID = 2;
```

Big picture

```
SELECT column_data
FROM source
    JOIN source2
WHERE condition
GROUP BY
HAVING condition
ORDER BY sort [ASC|DESC]
```

Some other topics

- SQL Wildcards, Trigger
- IF ELSE, SQL CASE Expression
- SOME Operators
- SQL AUTO INCREASE ON/OFF
- SQL INJECTION
- DELETE, UPDATE CASCADE
- SQL Concurrency

And more

- SQL Transaction
- Database clusters (high availability)
- Scaling database (scale ability)
- Distribution database
- Database cluster
- No-SQL

CHEAT SHEET

- w3schools

Thank you!

*Any **questions**?*

Extra Resources

Name	Link
Became SQL god?	https://www.w3schools.com/sql/default.asp