SQL (2)

Contents

- INSERT/ UPDATE/ DELETE
- Stored procedure
- Primary key and Foreign key
- View
- Data encryption
- Trigger
- SQL - Exist
- Query Execution Plan
- Types of Databases
- Normalizations

INSERT/ UPDATE/ DELETE

INSERT

- Simple Syntax: **INSERT INTO "tablename" ("column1", ...) VALUES ("value1", ...);**

```
INSERT INTO Store_Information (Store_Name, Sales, Txn_Date)
VALUES ('Los Angeles', 900, '1999-01-10');
```

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Los Angeles	900	1999-01-10

Exercise: more than one rows inserted

```
INSERT INTO Store_Information_1
        (Store_Name, Sales, Txn_Date)
 VALUES ('Lowell', 800, '1998-01-10'),
        ('Lowell', 700, '1998-01-11');
```

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
Los Angeles	300	1999-01-08
Boston	700	1999-01-08

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
Los Angeles	300	1999-01-08
Boston	700	1999-01-08
Lowell	800	1998-01-10
Lowell	700	1998-01-11

INSERT Multiple Rows

```
INSERT INTO Store_Information (Store_Name, Sales, Txn_Date)
  SELECT Store_Name, Sales, Txn_Date
  FROM store_Information_1
  WHERE Year(Txn_Date) = 1998;
```

store_information

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Los Angeles	900	1999-01-10

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Los Angeles	900	1999-01-10
Lowell	800	1998-01-10
Lowell	700	1998-01-11

UPDATE

- Simple syntax: **UPDATE "TableName"**
  SET "column1" = [new value]
  WHERE "condition";

```
UPDATE Store_Information
  SET Sales = 555
  WHERE Store_Name = 'Los Angeles'
```

Store_Name	Sales	Txn_Date
Los Angeles	1500	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Los Angeles	900	1999-01-10
Lowell	800	1998-01-10
Lowell	700	1998-01-11

Store_Name	Sales	Txn_Date
Los Angeles	555	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Los Angeles	555	1999-01-10
Lowell	800	1998-01-10
Lowell	700	1998-01-11

Exercise

```
UPDATE Store_Information
  SET Sales = 777,
      store_name = 'SunnyVale'
  WHERE Store_Name = 'Los Angeles'
```

Store_Name	Sales	Txn_Date
Los Angeles	555	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Los Angeles	555	1999-01-10
Lowell	800	1998-01-10
Lowell	700	1998-01-11

Store_Name	Sales	Txn_Date
SunnyVale	777	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
SunnyVale	777	1999-01-10
Lowell	800	1998-01-10
Lowell	700	1998-01-11

DELETE

- Simple syntax: **DELETE FROM "Tablename" WHERE "condition";**

```
DELETE FROM Store_Information
      WHERE sales=777;
```

Store_Name	Sales	Txn_Date
SunnyVale	777	1999-01-05
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
SunnyVale	777	1999-01-10
Lowell	800	1998-01-10
Lowell	700	1998-01-11

Store_Name	Sales	Txn_Date
San Diego	250	1999-01-07
San Francisco	300	1999-01-08
Boston	700	1999-01-08
Lowell	800	1998-01-10
Lowell	700	1998-01-11

Stored Procedure

Stored Procedure

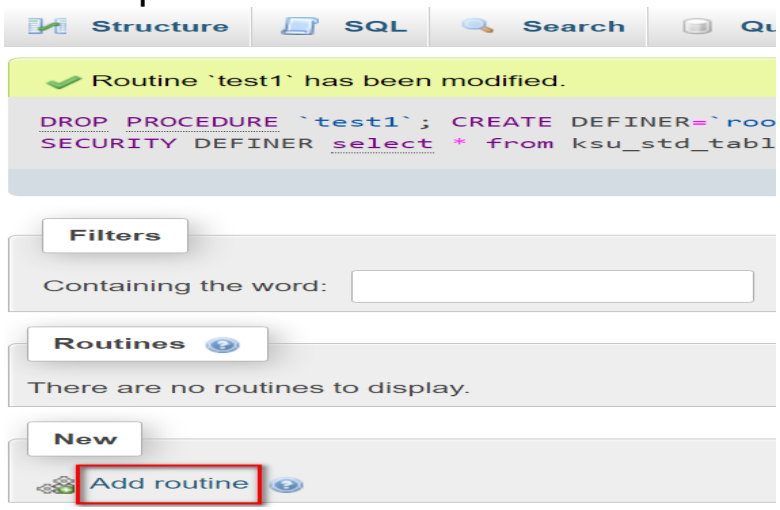
- What?
  - A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system (RDBMS) as a group, so it can be reused and shared by multiple programs, ex. php, PL/SQL , Java, Python, and so on.
- Purpose?
  - The main purpose of stored procedures to hide direct SQL queries from the code and improve performance of database operations such as select, update, insert and delete data.
- Why?
  - A stored procedure is a set of prepared SQL code that you can save, so the code can be reused over and over again.
  - a stored procedure allows them to be executed with a single call.

Stored Procedure Creation & Execution

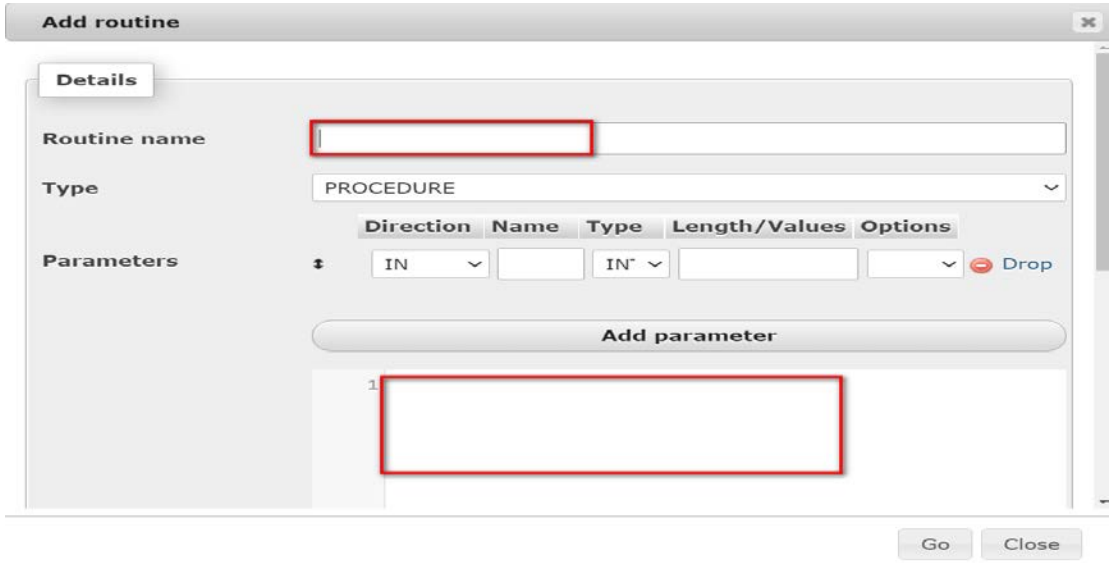
- Summarize the ideas:
  - Open PHP My Admin and select the database to create stored procedure
  - Go to Routines menu & Click on Add routine.
  - By Clicking on Add Routine Link, PHPMyAdmin will open Pop-up.
  - Follow the steps and create stored procedure. Create stored procedure to get data from users table (Without Parameters).

p.s. The name “routine” is the stored procedure we are talking about.

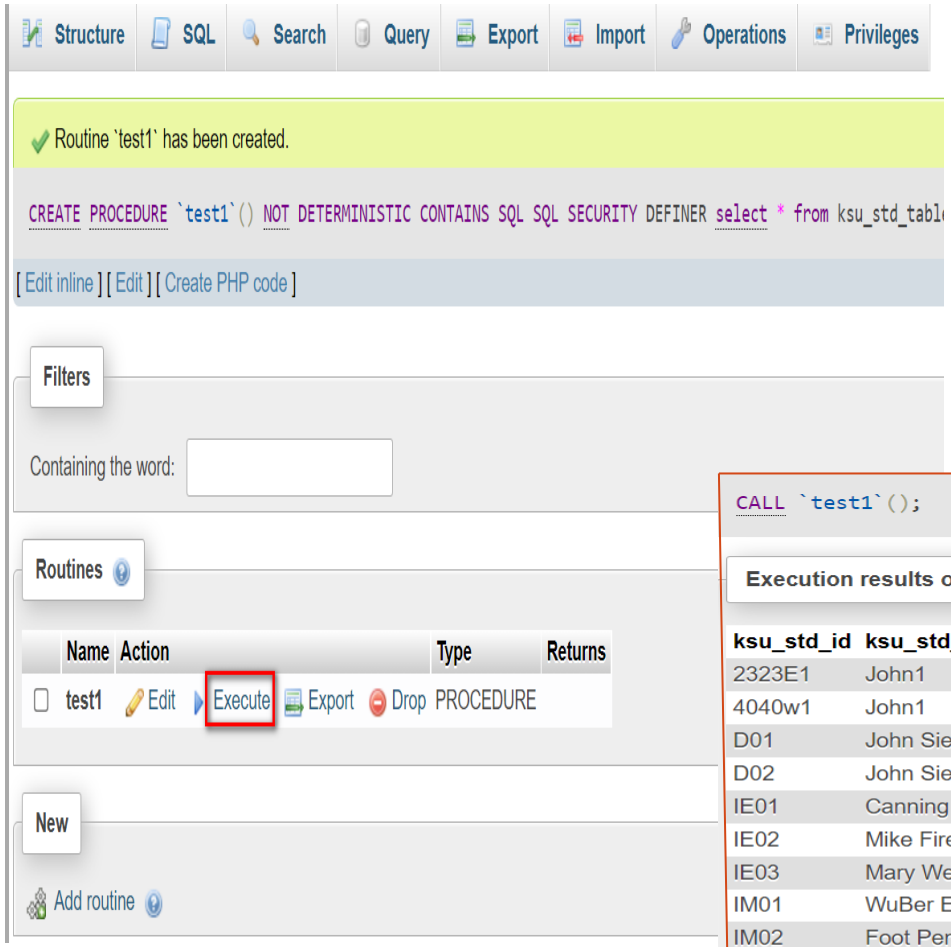
- Creation without parameters
  - Step 1: click on the Database you want to operate it
  - Step 2: click on the routines (預存程序) tab on the top bar
  - Step 3: click on the add routine



  - Step 4: follow the steps and fill out the form to finish the settings.



- Step 5: click on the execute tab for the stored procedure invocation

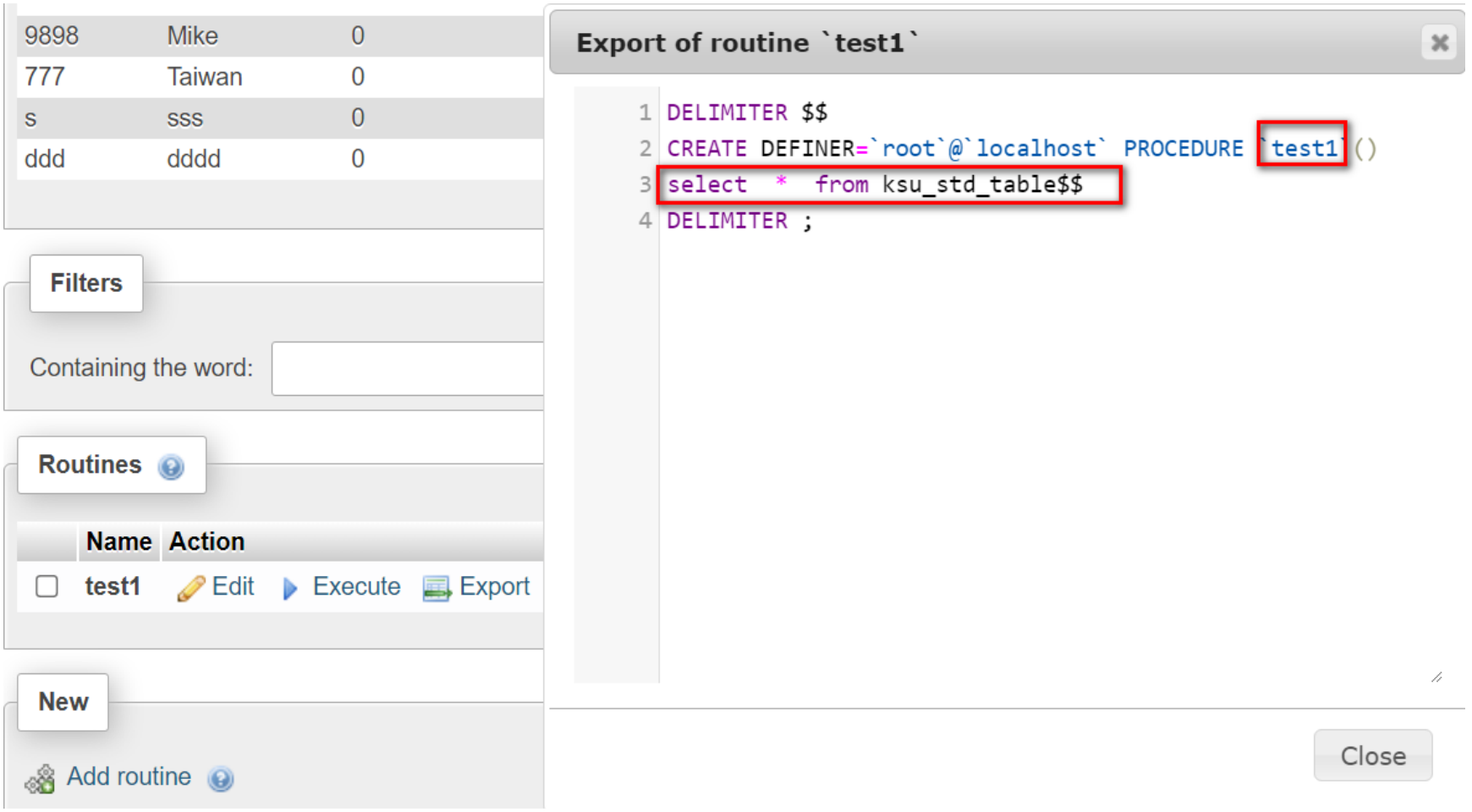



```
CALL `test1`();
```

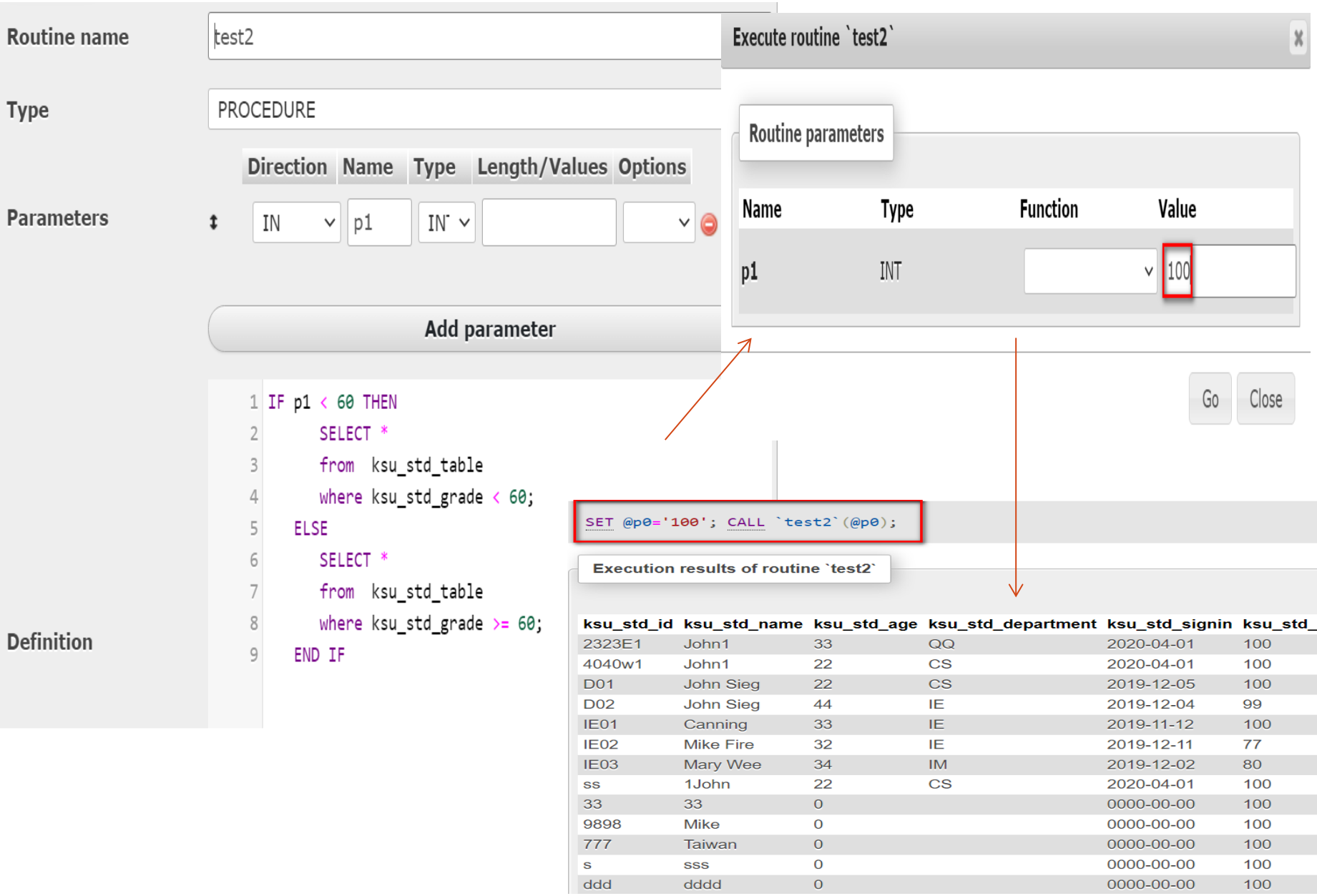
Execution results of routine `test1`

ksu_std_id	ksu_std_name	ksu_std_age	ksu_std_department	ksu_std_signin	ksu_std_grade
2323E1	John1	33	QQ	2020-04-01	100
4040w1	John1	22	CS	2020-04-01	100
D01	John Sieg	22	CS	2019-12-05	100
D02	John Sieg	44	IE	2019-12-04	99
IE01	Canning	33	IE	2019-11-12	100
IE02	Mike Fire	32	IE	2019-12-11	77
IE03	Mary Wee	34	IM	2019-12-02	80
IM01	WuBer Eat	22	IM	2019-11-12	33
IM02	Foot Penny	27	CS	2019-10-10	44
IM05	John Sieg	24	CS	2019-12-16	55
ss	1John	22	CS	2020-04-01	100
33	33	0		0000-00-00	100
9898	Mike	0		0000-00-00	100
777	Taiwan	0		0000-00-00	100
s	sss	0		0000-00-00	100
ddd	dddd	0		0000-00-00	100

Export the store procedure



- Creation with parameters



Export of routine `test2`

```
1  DELIMITER $$
2  CREATE DEFINER=`root`@`localhost` PROCEDURE `test2`(IN
   `p1` INT)
3  IF p1 < 60 THEN
4      SELECT *
5      from  ksu_std_table
6      where ksu_std_grade < 60;
7    ELSE
8      SELECT *
9      from  ksu_std_table
10     where ksu_std_grade >= 60;
11   END IF$$
12 DELIMITER ;
```

Close

Example 1

Edit

Routine name test4

Type PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Options
IN	p1	INT		Drop

Add parameter

Definition

```
1  BEGIN
2  DECLARE p1by INT;
3  SET p1by=p1 *0.6;
4
5  IF  p1by < 6 THEN
6        SELECT *
7        from  ksu_std_table
8        where ksu_std_grade < 60;
9  ELSE
10       SELECT *
11       from  ksu_std_table
12       where ksu_std_grade >= 60;
13 END IF;
14 END
```

Go

Execute routine `test4`

Routine parameters

Name	Type	Function	Value
p1	INT		100

Go Close

```
SET @p0='100'; CALL `test4`(@p0);
```

Execution results of routine `test4`

ksu_std_id	ksu_std_name	ksu_std_age	ksu_std_department	ksu_std_signin	ksu_std_grade
2323E1	John1	33	QQ	2020-04-01	100
4040w1	John1	22	CS	2020-04-01	100
D01	John Sieg	22	CS	2019-12-05	100
D02	John Sieg	44	IE	2019-12-04	99
IE01	Canning	33	IE	2019-11-12	100
IE02	Mike Fire	32	IE	2019-12-11	77
IE03	Mary Wee	34	IM	2019-12-02	80
ss	1John	22	CS	2020-04-01	100
33	33	0		0000-00-00	100
9898	Mike	0		0000-00-00	100
777	Taiwan	0		0000-00-00	100
s	sss	0		0000-00-00	100
ddd	dddd	0		0000-00-00	100

Example 2

Edit

Routine name test3

Type PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Options
IN	weight	INT		
IN	height	INT		

Add parameter

Definition

```
1  BEGIN
2  DECLARE BMI int;
3  SET BMI=weight/height*height;
4
5  IF BMI>24 THEN
6  SELECT CONCAT('Your BMI is ',BMI,' over heavy!');
7  ELSEIF  BMI>18 THEN
8  SELECT CONCAT('Your BMI is ',BMI,' Normal.');
9  ELSE
10 SELECT CONCAT('Your BMI is ',BMI,' skinny!');
11 END IF;
12 END
```

Go Close

Execute routine `test3`

Routine parameters

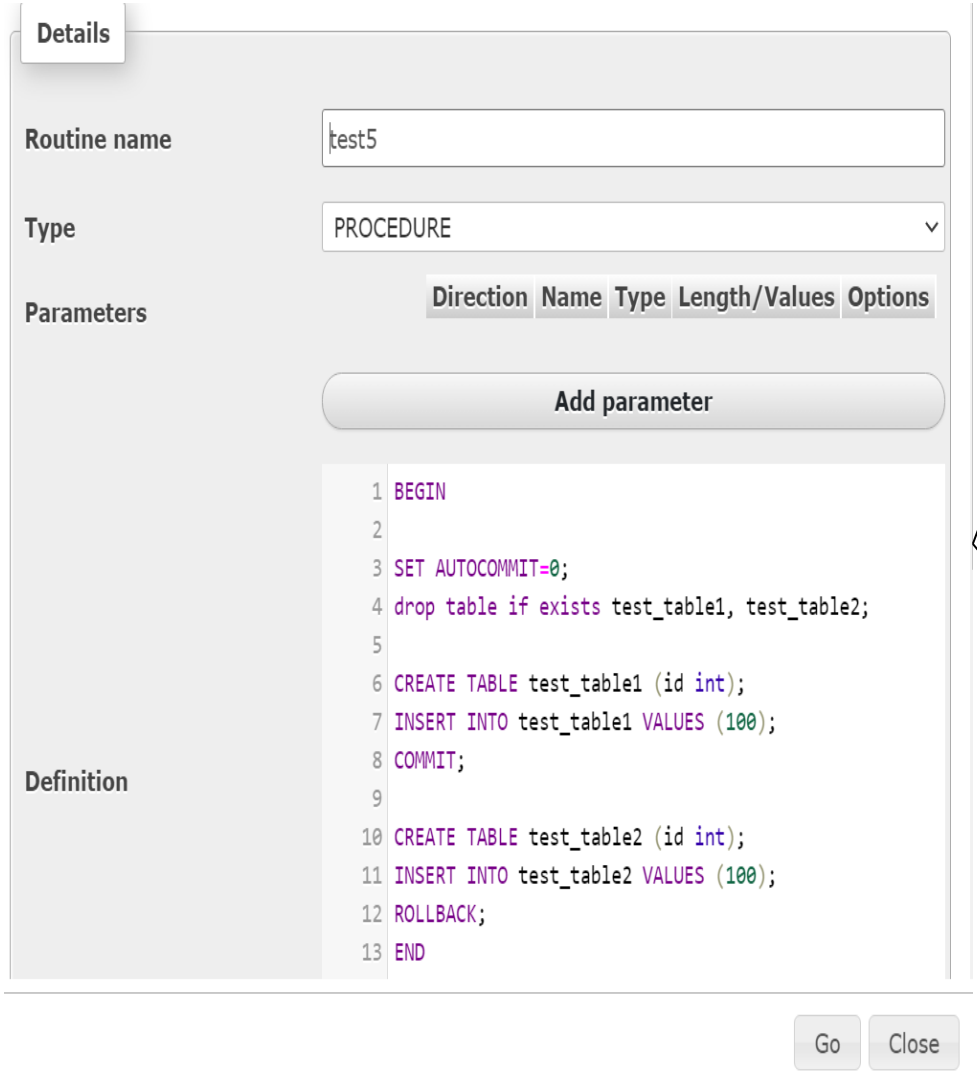
Name	Type	Function	Value
weight	INT		70
height	INT		175

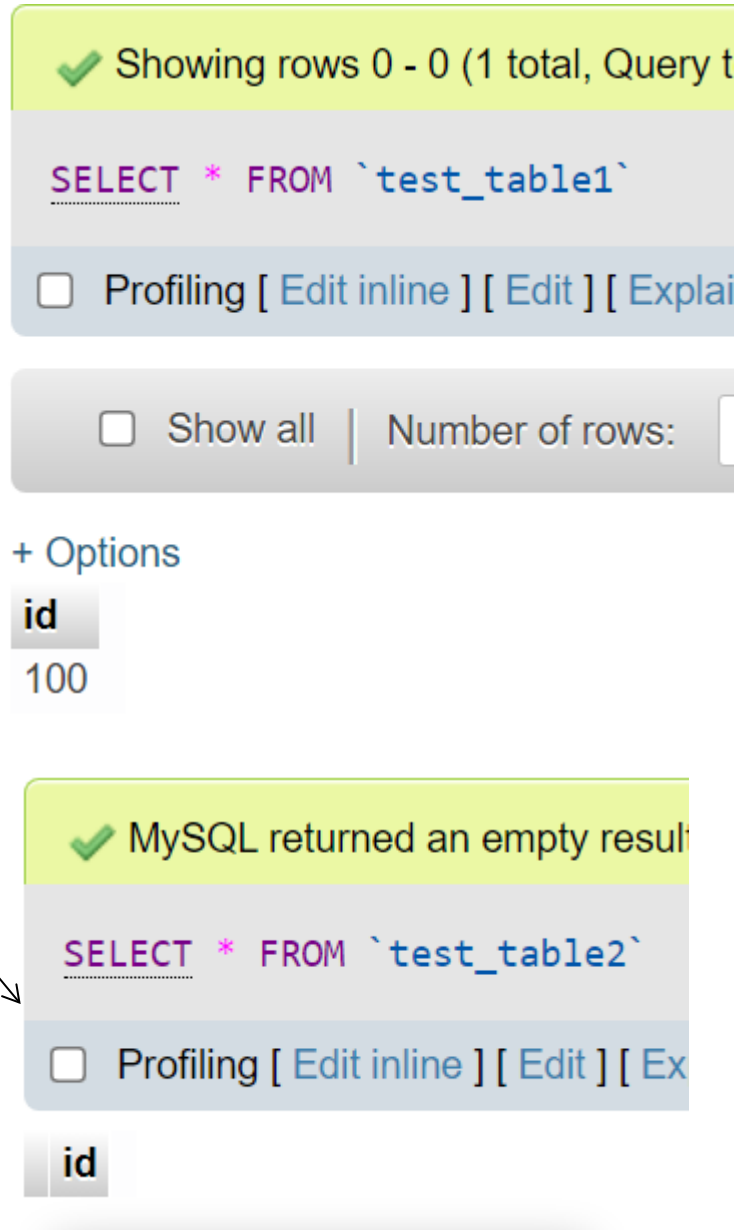
Go Close

Execution results of routine `test3`

CONCAT('Your BMI is ',BMI,' over heavy!')
Your BMI is 70 over heavy!

Example 3


Details
Routine name
test5
Type
PROCEDURE
Parameters
Direction Name Type Length/Values Options
Add parameter
Definition
1 BEGIN
2
3 SET AUTOCOMMIT=0;
4 drop table if exists test_table1, test_table2;
5
6 CREATE TABLE test_table1 (id int);
7 INSERT INTO test_table1 VALUES (100);
8 COMMIT;
9
10 CREATE TABLE test_table2 (id int);
11 INSERT INTO test_table2 VALUES (100);
12 ROLLBACK;
13 END
Go
Close


Showing rows 0 - 0 (1 total, Query t
SELECT * FROM `test_table1`
Profiling [Edit inline] [Edit] [Explai
Show all | Number of rows:
+ Options
id
100
MySQL returned an empty result
SELECT * FROM `test_table2`
Profiling [Edit inline] [Edit] [Ex
id

Commit v.s. rollback

- The most important aspect of a database is the ability to store data and the ability to manipulate data. COMMIT and ROLLBACK are two such keywords which are used in order store and revert the process of data storage.
- COMMIT and ROLLBACK are performed on transactions.
- A transaction is the smallest unit of work that is performed against a database. Its a sequence of instructions in a logical order. A transaction can be performed manually by a programmer or it can be triggered using an automated program.
- COMMIT is the SQL command that is used for storing changes performed by a transaction. When a COMMIT command is issued it saves all the changes since last COMMIT or ROLLBACK.

Commit v.s. rollback

- **Syntax for SQL Commit**

```
COMMIT;
```

- **SQL Commit Example**
  - **CASE 1**

Customer:-

CUSTOMER ID	CUSTOMER NAME	STATE	COUNTRY
1	Akash	Delhi	India
2	Amit	Hyderabad	India
3	Jason	California	USA
4	John	Texas	USA

Now let us delete one row from the above table where State is “Texas”.

```
DELETE from Customer where State = 'Texas';
```

SQL Delete without Commit

Post the DELETE command if we will not publish COMMIT, and if the session is closed then the change that is made due to the DELETE command will be lost.

Commit v.s. rollback

- **Syntax for SQL Commit**

```
COMMIT;
```

- **SQL Commit Example**
  - **CASE 2**

```
DELETE from Customer where State = 'Texas';
COMMIT;
```

SQL Commit Execution

Using the above-mentioned command sequence will ensure that the change post DELETE command will be saved successfully.

Output After Commit

CUSTOMER ID	CUSTOMER NAME	STATE	COUNTRY
1	Akash	Delhi	India
2	Amit	Hyderabad	India
3	Jason	California	USA

Commit v.s. rollback

- **Syntax for SQL Commit :** ROLLBACK is the SQL command that is used for reverting changes performed by a transaction. When a ROLLBACK command is issued it reverts all the changes since last COMMIT or ROLLBACK.

  ROLLBACK;

- **SQL Rollback Example**

Post the DELETE command if we publish ROLLBACK it will revert the change that is performed due to the delete command.

```
DELETE from Customer where State = 'Texas';
ROLLBACK;
```

Stored procedure v.s. function

- Basic Differences between Stored Procedure and Function in SQL Server. The function must return a value but in Stored Procedure it is optional. Even a procedure can return zero or n values. Functions can have only input parameters for it whereas Procedures can have input or output parameters.

Types of Databases

Types of Databases

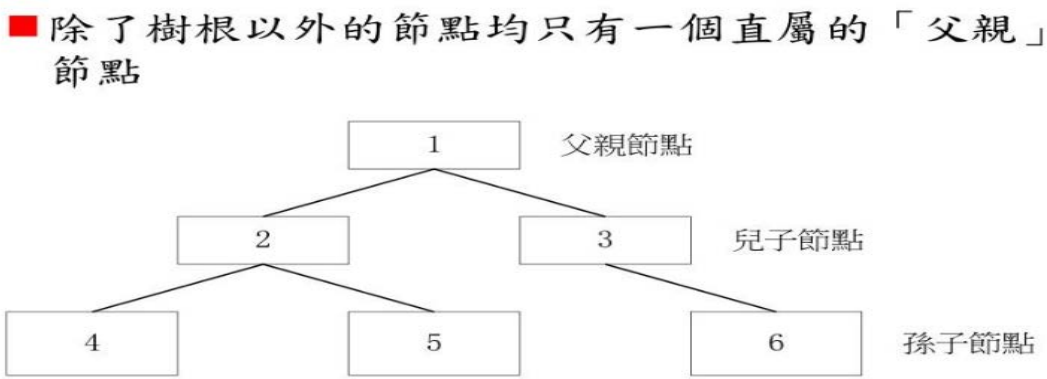
- A company should use the type of database that fit in its requirements and needs. There are various types of database structures:
  - **Relational databases-**Relational databases have been around since the 1970s. The name comes from the way that data is stored in multiple, related tables. Within the tables, data is stored in rows and columns. Organizations that have a lot of unstructured or semi-structured data should not be considering a relational database. (Examples of unstructured data are: Rich media. Media and entertainment data, surveillance data, geo-spatial data, audio, weather data. Semi Structured Data Examples: Email 、 CSV 、 XML and JSON documents 、 HTML, and so on)
    - Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL and IBM DB2

Types of Databases

- A company should use the type of database that fit in its requirements and needs. There are various types of database structures: (We just introduce few types here!)
  - **NoSQL** is a broad category that includes any database that doesn't use SQL as its primary data access language.
    - These types of databases are also sometimes referred to as non-relational databases. Unlike in relational databases, data in a NoSQL database doesn't have to conform to a pre-defined schema, so these types of databases are great for organizations seeking to store unstructured or semi-structured data.
    - One advantage of NoSQL databases is that developers can make changes to the database on the fly, without affecting applications that are using the database.
    - Apache Cassandra, MongoDB, CouchDB, and CouchBase

Types of Databases

- A company should use the type of database that fit in its requirements and needs. There are various types of database structures: (We just introduce few types here!)
  - **Hierarchical databases** use a parent-child model to store data. If you were to draw a picture of a hierarchical database, it would look like a family tree, with one object on top branching down to multiple objects beneath it.
    - **Examples:** IBM Information Management System (IMS), Windows Registry



Types of Databases

- A company should use the type of database that fit in its requirements and needs. There are various types of database structures: (We just introduce few types here!)
  - **Document databases** Document databases, also known as document stores, use JSON-like documents to model data instead of rows and columns.
    - Sometimes referred to as document-oriented databases, document databases are designed to store and manage document-oriented information, also referred to as semi-structured data. Document databases are simple and scalable, making them useful for mobile apps that need fast iterations.
    - **Examples:** MongoDB, Amazon DocumentDB, Apache CouchDB

Why SQL Statements

What is a Relational Database (RDBMS)

- A relational database is a type of database that stores and provides access to data points that are related to one another.
- Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables.
- In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

Industry’s Best RDBMS

- Oracle database products offer customers cost-optimized and high-performance versions of Oracle Database, the world's leading converged, multi-model database management system, as well as in-memory, NoSQL and MySQL databases.
- Oracle Autonomous Database enables customers to simplify relational database environments and reduce management workloads.
- Good procedural computer language: PL/ SQL

Benefits of RDBMS

- The simple yet powerful relational model is used by organizations of all types and sizes for a broad variety of information needs.
- Relational databases are used to track inventories, process ecommerce transactions, manage huge amounts of mission-critical customer information, and much more.
- A relational database can be considered for any information need in which data points relate to each other and must be managed in a secure, rules-based, consistent way.
- Relational databases have been around since the 1970s. Today, the advantages of the relational model continue to make it the most widely accepted model for databases.

Relational model and data consistency

- The relational model is the best at maintaining data consistency across applications and database copies (called instances). For example, when a customer deposits money at an ATM and then looks at the account balance on a mobile phone, the customer expects to see that deposit reflected immediately in an updated account balance. Relational databases excel at this kind of data consistency, ensuring that multiple instances of a database have the same data all the time.
- It's difficult for other types of databases to maintain this level of **timely consistency** with large amounts of data. Some recent databases, such as NoSQL, can supply only "**eventual consistency**." Under this principle, when the database is scaled or when multiple users access the same data at the same time, the data needs some time to "catch up." Eventual consistency is acceptable for some uses, such as to maintain listings in a product catalog, but for critical business operations such as shopping cart transactions, the relational database is still the gold standard.

Q&A