UNIVERSITAS
GADJAH MADA

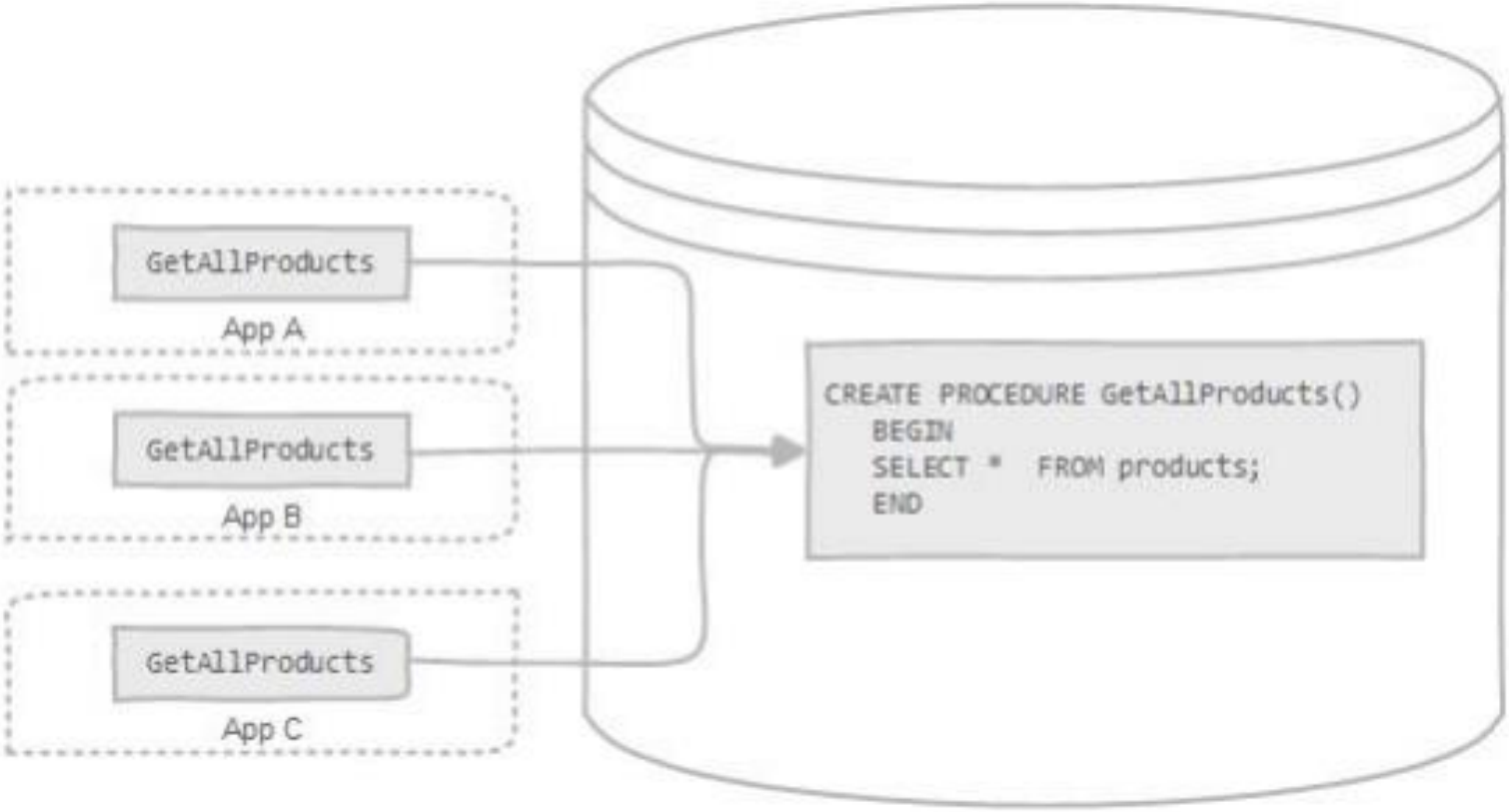
PROCEDURE

Diyah Utami Kusumaning Putri

Praktikum Basis Data

Definisi *Stored Procedure*

Suatu **prosedur tersimpan** ***(stored procedure)*** adalah bagian *statement* SQL deklaratif yang disimpan di dalam *database catalog*



UNIVERSITAS
GADJAH MADA

Keuntungan *MySQL Stored Procedure*

- Biasanya prosedur tersimpan membantu meningkatkan kinerja aplikasi/sistem
- Prosedur tersimpan membantu mengurangi *traffic* antara aplikasi dan *database server* karena aplikasi hanya mengirim nama dan parameter dari prosedur tersimpan dan tidak perlu mengirimkan *statement SQL* yang Panjang
- Prosedur tersimpan dapat digunakan kembali dan transparan untuk aplikasi apapun
- Prosedur tersimpan aman

UNIVERSITAS
GADJAH MADA

Kerugian *MySQL Stored Procedure*

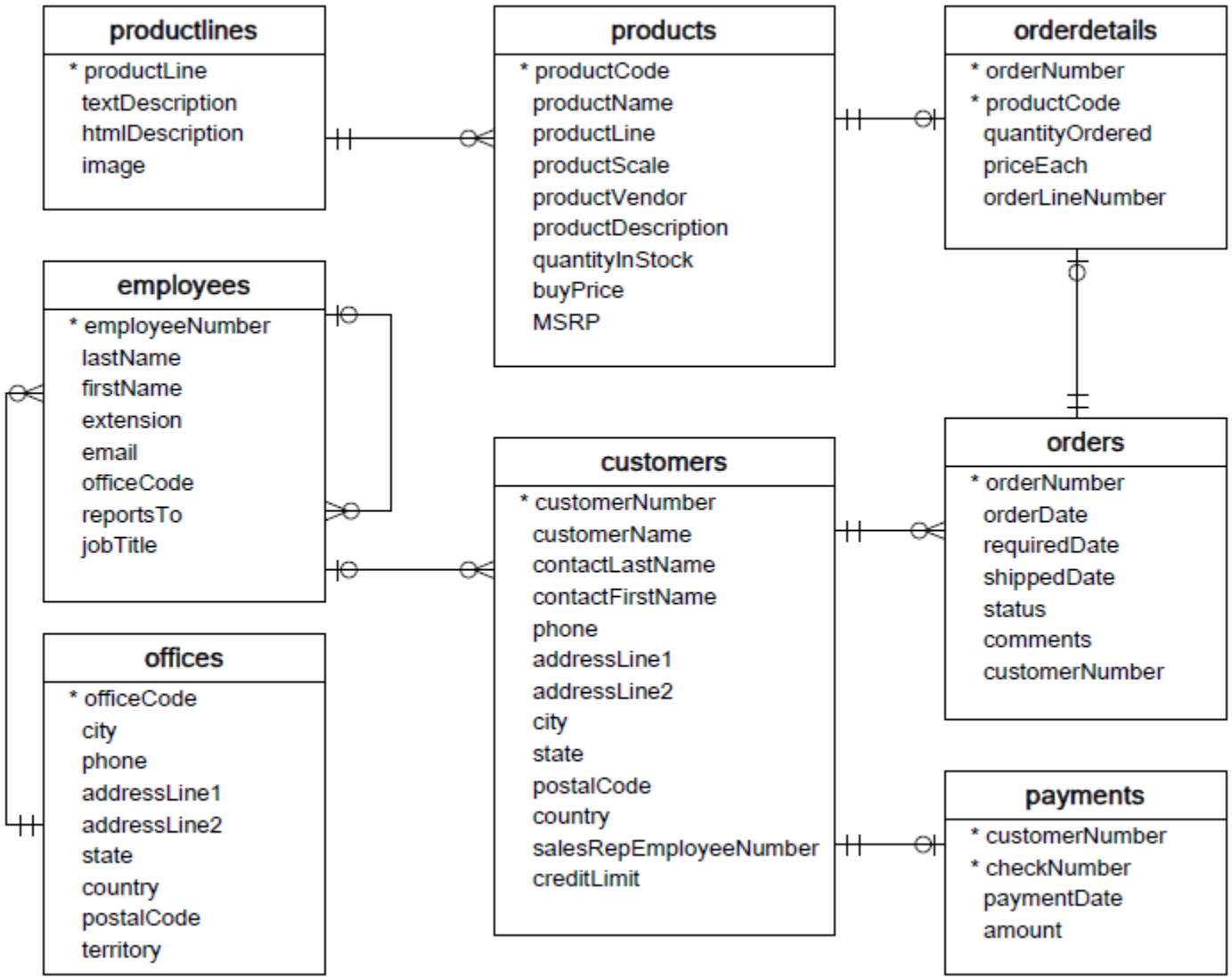
- Penggunaan memori setiap koneksi akan meningkat secara substansial apabila menggunakan banyak prosedur tersimpan
- Apabila terlalu banyak menggunakan operasi logika di dalam prosedur tersimpan, penggunaan CPU akan meningkat
- Susunan prosedur tersimpan tidak dirancang untuk mengembangkan logika bisnis yang kompleks dan fleksibel
- Sulit men-*debug* prosedur tersimpan. MySQL tidak menyediakan fasilitas tersebut
- Tidak mudah mengembangkan dan memelihara *(maintain)* prosedur tersimpan, diperlukan keahlian khusus

UNIVERSITAS
GADJAH MADA

Kelebihan dan Kekurangan *MySQL Stored Procedure*

- Prosedur tersimpan MySQL memiliki kelebihan dan kekurangan tersendiri.
- Ketika mengembangkan suatu aplikasi, *developer* harus memutuskan apakah akan menggunakan prosedur tersimpan atau tidak berdasarkan arsitektur aplikasi yang akan dibuat.

ClassicModels Database Schema



UNIVERSITAS
GADJAH MADA

Load sample database into MySQL Server

Ikuti Langkah-Langkah pada link berikut untuk memasukkan database ke *MySQL Server*

https://www.mysqltutorial.org/how-to-load-sample-database-into-mysql-database-server.aspx

UNIVERSITAS GADJAH MADA

Contoh *MySQL Stored Procedure*

- Prosedur tersimpan ***GetAllProducts()*** untuk menampilkan *(select)* semua *products* dari table *products*
- Gunakan ***mysql client tool*** dan ketikkan *command*

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
  BEGIN
  SELECT * FROM products;
  END //
DELIMITER ;
```

- Cara memanggil prosedur tersimpan *GetAllProducts()*

```
CALL GetAllProducts()
```

UNIVERSITAS
GADJAH MADA

Contoh *MySQL Stored Procedure*

Command Prompt - mysql -u root -p

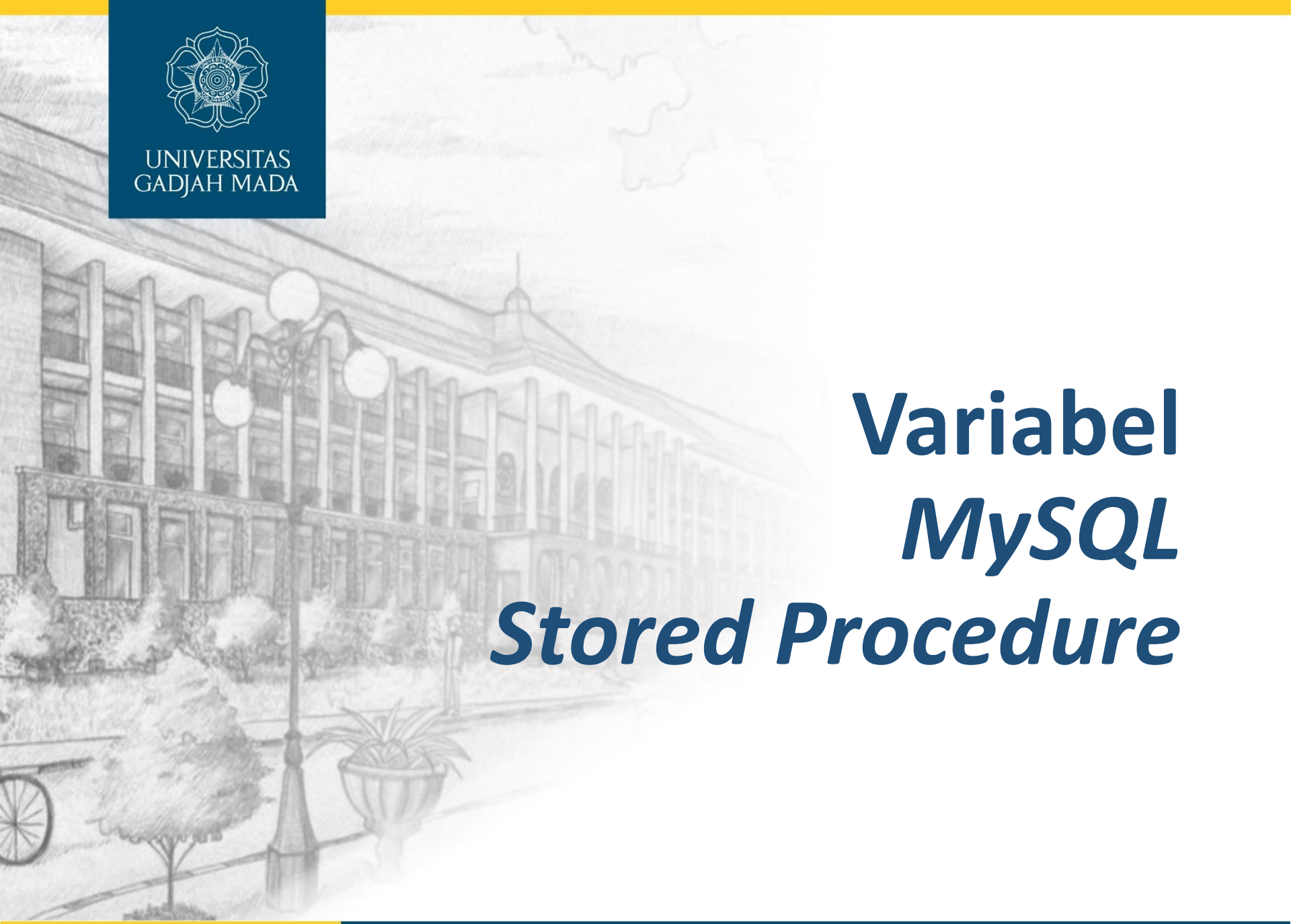
```
E:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 80
Server version: 10.4.11-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE classicmodels;
Database changed
MariaDB [classicmodels]> DELIMITER //
MariaDB [classicmodels]> CREATE PROCEDURE GetAllProducts()
    -> BEGIN
    -> SELECT * FROM products;
    -> END //
Query OK, 0 rows affected (0.716 sec)

MariaDB [classicmodels]> DELIMITER ;
MariaDB [classicmodels]> CALL GetAllProducts();
```



UNIVERSITAS
GADJAH MADA

Variabel *MySQL* *Stored Procedure*

UNIVERSITAS
GADJAH MADA

Variabel *MySQL Stored Procedure*

- Variabel adalah suatu objek data bernama yang nilainya dapat berubah saat menjalankan prosedur tersimpan.
- Biasanya variabel dalam prosedur tersimpan digunakan untuk menyimpan hasil secara langsung.
- Variabel-variabel ini bersifat lokal untuk prosedur tersimpan.
- Variabel harus dideklarasikan sebelum menggunakannya

Deklarasi Variabel

- Untuk mendeklarasikan suatu variabel pada prosedur tersimpan, gunakan *statement* ***DECLARE***

```
DECLARE variable_name
datatype(size)DEFAULT default_value;
```

- Contoh: mendeklarasikan suatu variabel ***total_sale*** dengan tipe data **INT** dan nilai default **0**

```
DECLARE total_sale INT DEFAULT 0;
```

- MySQL dapat mendeklarasikan 2 variabel atau lebih yang mempunyai tipe data yang sama menggunakan satu *statement DECLARE*

```
DECLARE x,y INT DEFAULT 0;
```

Menetapkan Nilai Variabel

- Untuk menetapkan nilai lain pada suatu variabel, gunakan *statement* SET

```
DECLARE variable_name
datatype(size)DEFAULT default_value;
SET variable_name = value;
```

- Contoh: menetapkan nilai pada variabel total_count = 10

```
DECLARE total_count INT DEFAULT 0;
SET total_count = 10;
```

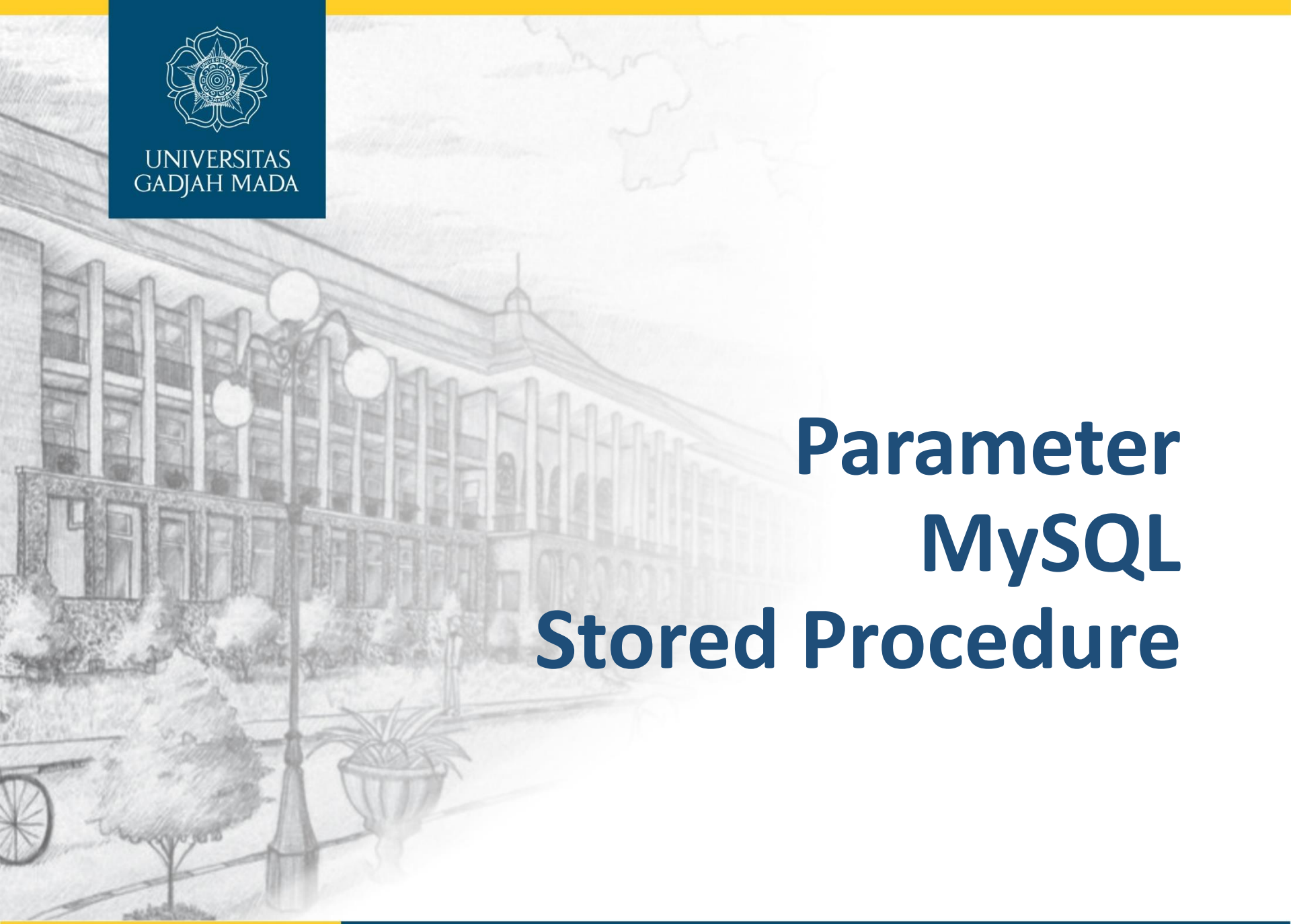
Menetapkan Nilai Variabel

- Selain *statement SET*, bisa juga menggunakan *statement* ***SELECT INTO*** untuk menetapkan nilai variabel berdasarkan hasil kueri yang mengembalikan nilai *scalar*.
- Contoh

```
DECLARE total_products INT DEFAULT 0;
SELECT COUNT (*) INTO total_products
FROM products;
```

Variables Scopes

- Suatu variabel memiliki cakupan *(scope)* yang menentukan masa pemakaiannya
- Apabila mendeklarasikan suatu variabel di dalam prosedur tersimpan, maka batas *scope* adalah ketika telah mencapai *statement END*
- Deklarasi dua variabel atau lebih dengan nama yang sama dapat dilakukan dengan *scope* yang berbeda karena variabel hanya efektif dalam *scope*-nya sendiri. Namun hal ini sebaiknya tidak dilakukan
- Variabel yang namanya diawali dengan tanda @ adalah *session variable*. Variabel ini tersedia dan dapati diakses hingga akhir sesi



Parameter MySQL Stored Procedure

UNIVERSITAS GADJAH MADA

Parameter *MySQL Stored Procedure*

- Parameter membuat prosedur tersimpan lebih fleksibel dan berguna
- MySQL memiliki beberapa parameter, yaitu IN, OUT dan INOUT
  - IN adalah mode *default*. Saat menentukan parameter IN dalam prosedur tersimpan, *calling program* harus memberikan argumen ke prosedur tersimpan. Nilai pada parameter IN terlindungi/*protected* (nilai asli tetap dipertahankan). Prosedur hanya bekerja dengan salinan nilai
  - OUT. Nilai pada parameter OUT dapat diubah di dalam prosedur tersimpan dan nilai baru dapat dikembalikan ke *calling program*. Prosedur tersimpan tidak dapat mengakses *initial value* dari parameter OUT ketika mulai
  - INOUT merupakan kombinasi dari parameter IN dan OUT. *Calling program* dapat memberikan *argument* dan prosedur tersimpan dapat mengubah parameter INOUT dan memberikan nilai baru kembali ke *calling program*

UNIVERSITAS
GADJAH MADA

Sintaks untuk mendefinsikan Parameter

```
MODE param_name param_type(param_size)
```

- MODE dapat berupa IN, OUT atau INOUT, tergantung pada tujuan parameter dalam prosedur tersimpan.
- param_name adalah nama parameter. Nama parameter harus mengikuti aturan penamaan nama kolom di MySQL.
- param_name diikuti dengan param_type(param_size) yang merupakan tipe data dan ukurannya.

UNIVERSITAS GADJAH MADA

Contoh Penggunaan Parameter IN pada *Stored Procedure*

- Penggunaan parameter IN pada prosedur tersimpan *GetOfficeByCountry* untuk menampilkan *office* yang lokasinya terletak di negara tertentu

```
DELIMITER //
CREATE PROCEDURE GetOfficeByCountry(IN countryName VARCHAR(255))
  BEGIN
  SELECT *
  FROM offices
  WHERE country = countryName;
  END //
DELIMITER ;
```

- Cara menggunakan prosedur tersimpan, gunakan CALL *GetOfficeByCountry* dan masukkan nilai *countryName* untuk menampilkan *office* di negara tersebut

```
CALL GetOfficeByCountry('USA')
```

UNIVERSITAS
GADJAH MADA

Contoh Parameter IN pada *Stored Procedure*

Command Prompt - mysql -u root -p

```
MariaDB [classicmodels]> DELIMITER //
MariaDB [classicmodels]> CREATE PROCEDURE GetOfficeByCountry(IN countryName VARCHAR(255))
    -> BEGIN
    -> SELECT *
    -> FROM offices
    -> WHERE country = countryName;
    -> END //
Query OK, 0 rows affected (2.076 sec)

MariaDB [classicmodels]> DELIMITER ;
MariaDB [classicmodels]> CALL GetOfficeByCountry('USA');
+------------+---------------+-----------------+----------------------+--------------+-------+---------+------------+-----------+
| officeCode | city          | phone           | addressLine1         | addressLine2 | state | country | postalCode | territory |
+------------+---------------+-----------------+----------------------+--------------+-------+---------+------------+-----------+
| 1          | San Francisco | +1 650 219 4782 | 100 Market Street    | Suite 300    | CA    | USA     | 94080      | NA        |
| 2          | Boston        | +1 215 837 0825 | 1550 Court Place     | Suite 102    | MA    | USA     | 02107      | NA        |
| 3          | NYC           | +1 212 555 3000 | 523 East 53rd Street | apt. 5A      | NY    | USA     | 10022      | NA        |
+------------+---------------+-----------------+----------------------+--------------+-------+---------+------------+-----------+
3 rows in set (0.422 sec)

Query OK, 0 rows affected (0.439 sec)

MariaDB [classicmodels]> CALL GetOfficeByCountry('France');
+------------+-------+-----------------+--------------------------+--------------+-------+---------+------------+-----------+
| officeCode | city  | phone           | addressLine1             | addressLine2 | state | country | postalCode | territory |
+------------+-------+-----------------+--------------------------+--------------+-------+---------+------------+-----------+
| 4          | Paris | +33 14 723 4404 | 43 Rue Jouffroy D'abbans | NULL         | NULL  | France  | 75017      | EMEA      |
+------------+-------+-----------------+--------------------------+--------------+-------+---------+------------+-----------+
1 row in set (0.000 sec)

Query OK, 0 rows affected (0.341 sec)
```

UNIVERSITAS GADJAH MADA

Contoh Penggunaaan Parameter OUT pada *Stored Procedure*

- Penggunaan parameter OUT pada prosedur tersimpan *CountOrderByStatus* untuk menampilkan jumlah *order* berdasarkan *order status*

```
DELIMITER $$
CREATE PROCEDURE CountOrderByStatus(
  IN orderStatus VARCHAR(25),
  OUT total INT)
BEGIN
  SELECT count(orderStatus)
  INTO total
  FROM orders
  WHERE status = orderStatus;
END $$
DELIMITER ;
```

UNIVERSITAS GADJAH MADA

Contoh Penggunaan Parameter OUT pada *Stored Procedure*

- Cara menggunakan prosedur tersimpan untuk mendapatkan jumlah *order* yang dikirim *(shipped)*, gunakan *CALL CountOrderByStatus* dan masukkan status order *'shipped'*, dan juga argumen (@total) untuk mendapatkan nilai kembalian.

```
CALL CountryOrderByStatus('Shipped', @total);
SELECT @total;
```

UNIVERSITAS
GADJAH MADA

Contoh Parameter OUT pada *Stored Procedure*

```
MariaDB [classicmodels]> DELIMITER $$
MariaDB [classicmodels]> CREATE PROCEDURE CountOrderByStatus(
    -> IN orderStatus VARCHAR(25),
    -> OUT total INT)
    -> BEGIN
    -> SELECT count(orderNumber)
    -> INTO total
    -> FROM orders
    -> WHERE status = orderStatus;
    -> END $$
Query OK, 0 rows affected (0.574 sec)

MariaDB [classicmodels]> DELIMITER ;
MariaDB [classicmodels]> CALL CountOrderByStatus('Shipped', @total);
Query OK, 1 row affected (0.003 sec)

MariaDB [classicmodels]> SELECT @total;
+--------+
| @total |
+--------+
|    303 |
+--------+
1 row in set (0.329 sec)
MariaDB [classicmodels]> CALL CountOrderByStatus('in process', @total);
Query OK, 1 row affected (0.001 sec)

MariaDB [classicmodels]> SELECT @total AS total_in_process;
+------------------+
| total_in_process |
+------------------+
|                6 |
+------------------+
1 row in set (0.000 sec)
```

UNIVERSITAS GADJAH MADA

Contoh Penggunaan Parameter INOUT pada *Stored Procedure*

- Penggunaan parameter INOUT pada prosedur tersimpan

```
DELIMITER $$
CREATE PROCEDURE set_counter(
  INOUT count INT(4),
  IN inc INT(4))
BEGIN
  SET count=count + inc;
END $$
DELIMITER ;
```

- Cara menggunakan prosedur tersimpan dengan perintah *CALL set_counter*

```
SET @counter = 1;
CALL set_counter(@counter,1);          --1 + 1 = 2
CALL set_counter(@counter,1);          --2 + 1 = 3
CALL set_counter(@counter,5);          --3 + 5 = 8
SELECT @counter;
```

UNIVERSITAS
GADJAH MADA

Contoh Parameter INOUT pada *Stored Procedure*

```
MariaDB [classicmodels]> DELIMITER $$
MariaDB [classicmodels]> CREATE PROCEDURE set_counter(
    -> INOUT count INT(4),
    -> IN inc INT(4))
    -> BEGIN
    -> SET count = count+inc;
    -> END $$
Query OK, 0 rows affected (0.542 sec)

MariaDB [classicmodels]> DELIMITER ;
MariaDB [classicmodels]> SET @counter=1;
Query OK, 0 rows affected (0.093 sec)

MariaDB [classicmodels]> CALL set_counter(@counter,1);
Query OK, 0 rows affected (0.001 sec)

MariaDB [classicmodels]> CALL set_counter(@counter,1);
Query OK, 0 rows affected (0.001 sec)

MariaDB [classicmodels]> CALL set_counter(@counter,5);
Query OK, 0 rows affected (0.000 sec)

MariaDB [classicmodels]> SELECT @counter;
+----------+
| @counter |
+----------+
|        8 |
+----------+
1 row in set (0.001 sec)
```

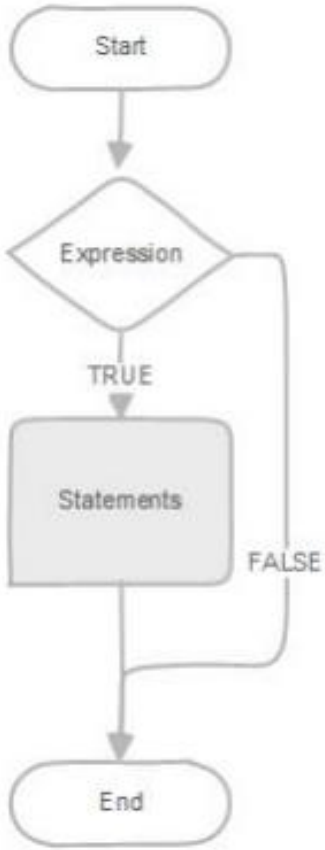
MySQL
IF Statement

MySQL IF Statement

- *Statemen IF* digunakan untuk mengeksekusi sekumpulan perintah SQL berdasarkan kondisi atau nilai ekspresi tertentu.
- Untuk membentuk suatu ekspresi pada MySQL dapat digabungkan dengan literal, variabel, operator, dan bahkan fungsi. Suatu ekspresi dapat mengembalikan nilai TRUE, FALSE, atau NULL.

```
IF expression THEN
  statements;
END IF;
```

- Jika evaluasi ekspresi bernilai TRUE, maka *statement* akan dieksekusi, jika tidak, kontrol diteruskan ke *statement* berikutnya setelah END IF.



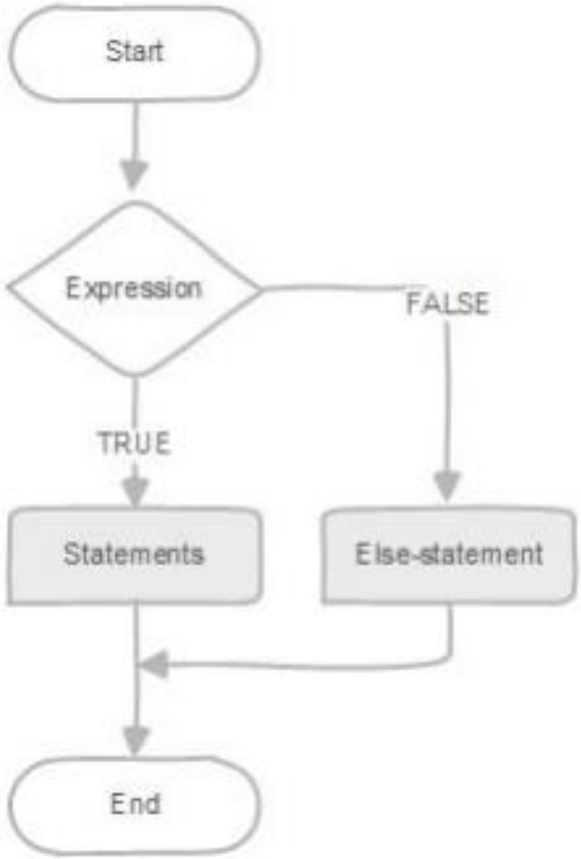
UNIVERSITAS
GADJAH MADA

MySQL IF-ELSE Statement

- *Statemen IF-ELSE* digunakan apabila kita ingin menjalankan *statement* ketika ekspresi yang dievaluasi tidak bernilai *TRUE*

```
IF expression THEN
  statements;
ELSE
  else-statements;
END IF;
```

- Jika evaluasi ekspresi bernilai TRUE maka akan dijalankan *statement* pertama, jika bernilai FALSE maka akan dijalankan *statement* lainnya

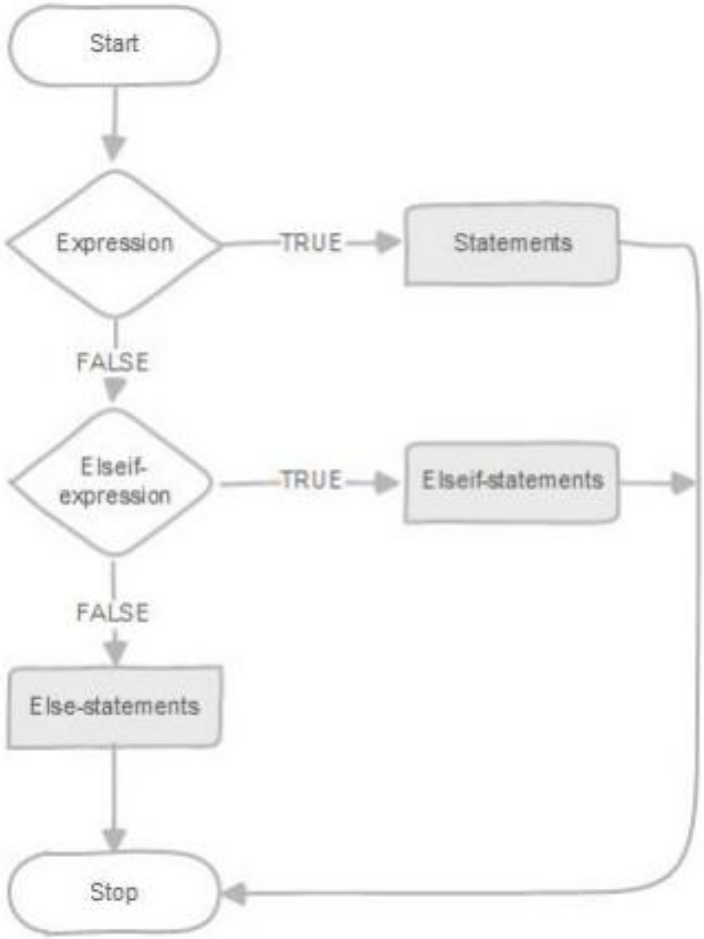


UNIVERSITAS GADJAH MADA

MySQL IF-ELSEIF-ELSE Statement

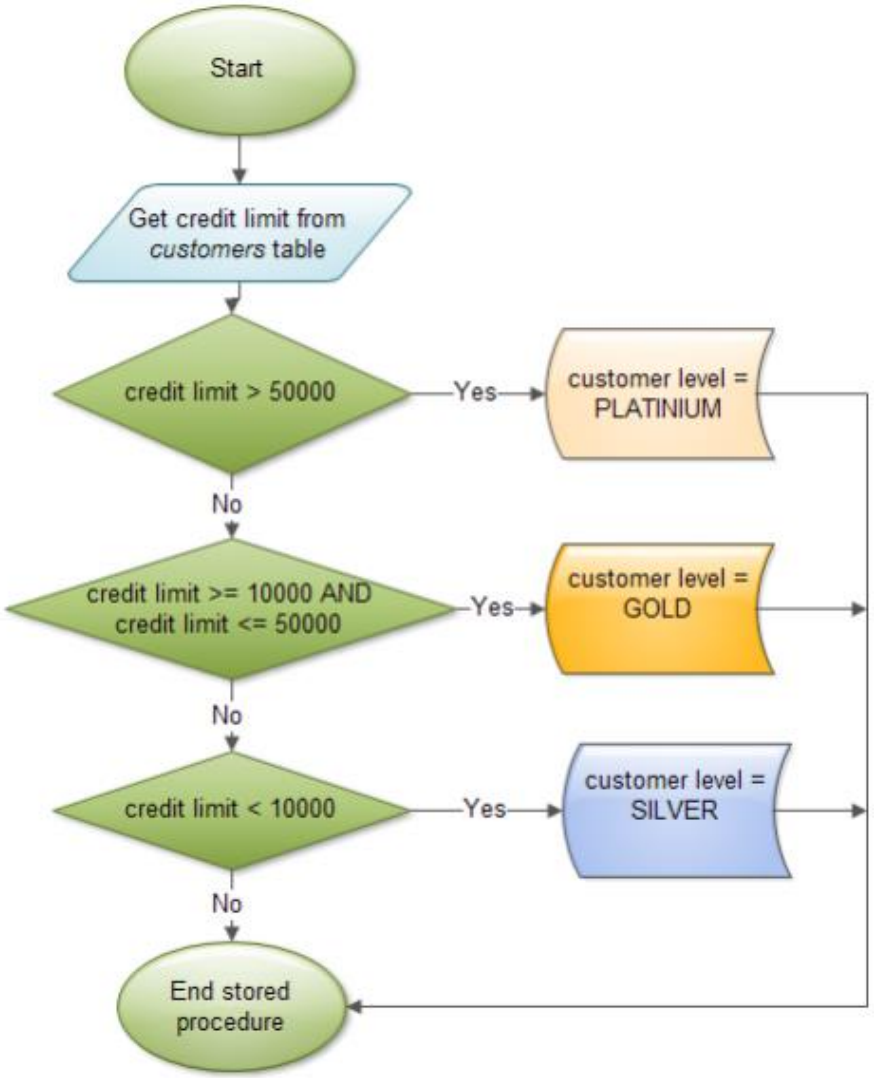
- *Statemen IF-ELSEIF-ELSE* digunakan untuk menjalalankan *statement* kondisional berdasarkan beberapa ekspresi

```
IF expression THEN
  statements;
ELSEIF elseif-expression THEN
  elseif-statements;
...
ELSE
  else-statements;
END IF;
```



Contoh
MySQL IF-ELSEIF-ELSE Statement

- Prosedur tersimpan *GetCustomerLevel()* menggunakan dua parameter *customer number* dan *customer level*
  - *Customer number* digunakan untuk mendapatkan nilai *credit limit* dari tabel *customer*
  - Kemudian, *credit limit* digunakan untuk menentukan *customer level (platinum, gold* dan *silver)*



Contoh Penggunaan *MySQL IF-ELSEIF-ELSE Statement*


```
DELIMITER $$

CREATE PROCEDURE GetCustomerLevel(
  IN p_customerNumber INT(11),
  OUT p_customerLevel VARCHAR(10))
BEGIN
  DECLARE creditlim DOUBLE;
  SELECT creditlimit INTO creditlim
  FROM customers
  WHERE customerNumber = p_customerNumber;
  IF creditlim > 50000 THEN
    SET p_customerLevel = 'PLATINUM';
  ELSEIF (creditlim <= 50000 AND  creditlim >= 10000) THEN
    SET p_customerLevel = 'GOLD';
  ELSEIF creditlim < 10000 THEN
    SET p_customerLevel = 'SILVER';
  END IF;
END $$

DELIMITER ;
```

```
CALL GetCustomerLevel(103, @p_customerLevel);
SELECT @p_customerLevel;
```

Contoh
MySQL IF-ELSEIF-ELSE Statement


```
MariaDB [classicmodels]> DELIMITER $$
MariaDB [classicmodels]> CREATE PROCEDURE GetCustomerLevel(
    -> IN p_customerNumber INT(11),
    -> OUT p_customerLevel VARCHAR(10))
    -> BEGIN
    -> DECLARE creditlim DOUBLE;
    -> SELECT creditlimit INTO creditlim
    -> FROM customers
    -> WHERE customerNumber = p_customerNumber;
    -> IF creditlim > 50000 THEN
    -> SET p_customerLevel = 'PLATINUM';
    -> ELSEIF (creditlim <=50000 AND creditlim >= 10000) THEN
    -> SET p_customerLevel = 'GOLD';
    -> ELSEIF creditlim < 10000 THEN
    -> SET p_customerLevel = 'SILVER';
    -> END IF;
    -> END $$
Query OK, 0 rows affected (0.658 sec)
```

```
MariaDB [classicmodels]> CALL GetCustomerLevel(103, @p_customerLevel);
Query OK, 1 row affected (0.130 sec)

MariaDB [classicmodels]> SELECT @p_customerLevel;
+------------------+
| @p_customerLevel |
+------------------+
| GOLD             |
+------------------+
1 row in set (0.000 sec)
```

MySQL
CASE Statement

MySQL CASE Statement

- *CASE statement* membuat kode lebih mudah dibaca dan efisien.
- Terdapat dua bentuk *CASE statement* yaitu :
  - *Simple Case Statement*
  - *Searched Case Statement*

UNIVERSITAS GADJAH MADA

Simple Case Statement

- *Simple case statement* digunakan untuk mengecek nilai dari suatu ekspresi dari sekumpulan nilai unik.
- Sintaks *simple case statement*

```
CASE case_expression
  WHEN when_expression_1 THEN commands
  WHEN when_expression_2 THEN commands
  ...
  ELSE commands
END CASE;
```

- Apabila nilai *case_expression* dan *when_expression_n* sama maka *commands* pada cabang *WHEN* tersebut akan dijalankan.
- Apabila tidak ada *when_expression* yang cocok dengan nilai *case_expression*, *commands* dalam klausa ELSE akan dijalankan.
- Klausa ELSE opsional. Jika klausa ELSE dihilangkan dan tidak ada kecocokan yang ditemukan, MySQL akan memunculkan *error*.

Contoh Penggunaan *Simple Case Statement*

- Prosedur tersimpan *GetCustomerShipping()* menggunakan dua paramete yaitu *customer number* sebagai parameter IN dan mengembalikan *shipping period* berdasarkan *country* dari *customer*
  - *Customer number* digunakan untuk mendapatkan *country* dari tabel *customer*
  - Kemudian, *simple case statement* digunakan untuk menentukan *shipping period* berdasarkan *country*

Contoh Penggunaan *Simple Case Statement*


```
DELIMITER $$

CREATE PROCEDURE GetCustomerShipping(
  IN p_customerNumber INT(11),
  OUT p_shiping VARCHAR(50))
BEGIN
  DECLARE customerCountry VARCHAR(50);
  SELECT country INTO customerCountry
  FROM customers
  WHERE customerNumber = p_customerNumber;

  CASE customerCountry
  WHEN 'USA' THEN
    SET p_shiping = '2-day Shipping';
  WHEN 'Canada' THEN
    SET p_shiping = '3-day Shipping';
  ELSE
    SET p_shiping = '5-day Shipping';
  END CASE;
END $$

DELIMITER ;
```

Pemanggilan
Simple Case Statement


```
SET @customerNo = 112;

SELECT country INTO @country
FROM customers
WHERE customerNumber = @customerNo;

CALL GetCustomerShipping(@customerNo, @shipping);

SELECT @customerNo AS Customer,
       @country AS Country,
       @shipping AS Shipping;
```

Contoh Penggunaan *Simple Case Statement*

UNIVERSITAS GADJAH MADA

```
MariaDB [classicmodels]> DELIMITER $$
MariaDB [classicmodels]> CREATE PROCEDURE GetCustomerShipping(
    -> IN p_customerNumber INT(11),
    -> OUT p_shiping VARCHAR(50))
    -> BEGIN
    -> DECLARE customerCountry VARCHAR(50);
    -> SELECT country INTO customerCountry
    -> FROM customers
    -> WHERE customerNumber = p_customerNumber;
    -> CASE customerCountry
    -> WHEN 'USA' THEN
    -> SET p_shiping = '2-day Shipping';
    -> WHEN 'Canada' THEN
    -> SET p_shiping = '3-day Shipping';
    -> ELSE
    -> SET p_shiping = '5-day Shipping';
    -> END CASE;
    -> END $$
Query OK, 0 rows affected (0.656 sec)
MariaDB [classicmodels]> DELIMITER ;
MariaDB [classicmodels]> SET @customerNo = 112;
Query OK, 0 rows affected (0.000 sec)

MariaDB [classicmodels]> SELECT country into @country
    -> FROM customers
    -> WHERE customernumber = @customerNo;
Query OK, 1 row affected (0.001 sec)

MariaDB [classicmodels]> CALL GetCustomerShipping(@customerNo, @shipping);
Query OK, 1 row affected (0.001 sec)

MariaDB [classicmodels]> SELECT @customerNo AS Customer,
    -> @country AS Country,
    -> @shipping AS Shipping;
+----------+---------+----------------+
| Customer | Country | Shipping       |
+----------+---------+----------------+
|      112 | USA     | 2-day Shipping |
+----------+---------+----------------+
1 row in set (0.000 sec)
```

MySQL Searched CASE Statement

UNIVERSITAS GADJAH MADA

Searched Case Statement

- *Simple case statement* hanya mencocokkan nilai ekspresi dengan sekumpulan nilai yang berbeda. Untuk melakukan pencocokan yang lebih kompleks seperti *range*, kita dapat menggunakan *searched case statement*.
- ***Searched case statement*** sama dengan *IF statement*, namun sintaksnya jauh lebih mudah dibaca.
- Sintaks *searched case statement*

```
CASE
  WHEN condition_1 THEN commands
  WHEN condition_2 THEN commands
  ...
  ELSE commands
END CASE;
```

UNIVERSITAS GADJAH MADA

Searched Case Statement

```
CASE
  WHEN condition_1 THEN commands
  WHEN condition_2 THEN commands
  ...
  ELSE commands
END CASE;
```

- MySQL akan mengevaluasi setiap kondisi dalam klausa WHEN hingga menemukan suatu kondisi yang bernilai TRUE, kemudian *commands* yg berhubungan dalam klausa THEN akan dieksekusi.
- Jika tidak ada kondisi yang bernilai TRUE, *commands* dalam klausa ELSE akan dieksekusi. Apabila klausa ELSE tidak didefinisikan dan tidak ada kondisi yang bernilai TRUE, MySQL akan memunculkan *error*.
- MySQL tidak memperbolehkan terdapat perintah kosong pada klausa THEN atau ELSE. Apabila tidak ingin menangani logika dalam klausa ELSE dan mencegah MySQL memunculkan *error*, *commands* dapat diganti dengan blok BEGIN END kosong pada klausa ELSE.

Contoh
Searched Case Statement


```
DELIMITER $$

CREATE PROCEDURE GetCustomerLevelCase(
  IN p_customerNumber INT(11),
  OUT p_customerLevel VARCHAR(10))
BEGIN
  DECLARE creditlim DOUBLE;
  SELECT creditlimit INTO creditlim
  FROM customers
  WHERE customerNumber = p_customerNumber;
  CASE
  WHEN creditlim > 50000 THEN
    SET p_customerLevel = 'PLATINUM';
  WHEN (creditlim <= 50000 AND  creditlim >= 10000) THEN
    SET p_customerLevel = 'GOLD';
  WHEN creditlim < 10000 THEN
    SET p_customerLevel = 'SILVER';
 END CASE;
END $$

DELIMITER ;
```

Pemanggilan
Searched Case Statement


```
SET @customerNo = 112;

SELECT country INTO @country
FROM customers
WHERE customerNumber = @customerNo;

CALL GetCustomerLevelCase(112, @level);

SELECT @level AS 'Customer Level';
```

Contoh Penggunaan *Searched Case Statement*


```
MariaDB [classicmodels]> DELIMITER $$
MariaDB [classicmodels]> CREATE PROCEDURE GetCustomerLevelCase(
    -> IN p_customerNumber INT(11),
    -> OUT p_customerLevel VARCHAR(10))
    -> BEGIN
    -> DECLARE creditlim DOUBLE;
    -> SELECT creditlimit INTO creditlim
    -> FROM customers
    -> WHERE customerNumber = p_customerNumber;
    -> CASE
    -> WHEN creditlim > 50000 THEN
    -> SET p_customerLevel = 'PLATINUM';
    -> WHEN (creditlim <=50000 AND creditlim >= 10000) THEN
    -> SET p_customerLevel = 'GOLD';
    -> WHEN creditlim < 10000 THEN
    -> SET p_customerLevel = 'SILVER';
    -> END CASE;
    -> END $$
Query OK, 0 rows affected (0.579 sec)

MariaDB [classicmodels]> DELIMITER ;
MariaDB [classicmodels]> CALL GetCustomerLevelCase(112,@level);
Query OK, 1 row affected (0.001 sec)

MariaDB [classicmodels]> SELECT @level AS 'Customer Level';
+----------------+
| Customer Level |
+----------------+
| PLATINUM       |
+----------------+
1 row in set (0.000 sec)
```

THANK YOU ☺