PRAKTIKUM SISTEM BASIS DATA

Modul 13: Stored Procedure (Variable & Flow Controls)

Deskripsi

Modul ini berisi pengenalan tentang Stored Procedure (SP) bagian pertama pada basis data MySQL. SP merupakan program berupa prosedur yang digunakan sebagai suatu fungsi yang dijalankan oleh pengguna dengan cara dipanggil secara manual/eksplisit untuk melakukan sesuatu yang berhubungan dengan manipulasi basis data. Materi pada modul ini terdiri dari pengertian, keuntungan, kelemahan, pembuatan SP, variables, percabangan dan perulangan.

Tujuan dan Kompetensi Praktikum:

- Mahasiswa dapat menjelaskan konsep SP pada Basis Data MySQL
- Mahasiswa dapat membuat SP sederhana
- Mahasiswa dapat menggunakan SP untuk meningkatkan kemampuan SQL biasa.
- Mahasiswa dapat menampilkan SP pada MySQL.
- Mahasiswa dapat membuat dan menggunakan variabel pada MySQL SP
- Mahasiswa dapat menggunakan IF pada MySQL SP
- Mahasiswa dapat menggunakan case pada MySQL SP
- Mahasiswa dapat menggunakan loop pada MySQL SP

Alat dan Bahan:

1. Modul Praktikum
2. Komputer
3. MySQL dan phpmyadmin, HeidiSQL, DataGrip atau sejenisnya.
   Silakan membuat database terlebih dahulu, misal **[nama]_p13**
4. Gunakan file SQL ini, import pada database anda.
5. Viewer LCD

Materi dan Pembagian waktu:

Materi dan pembagian waktu pelaksanaan perkuliahan meliputi:

No.	Materi	Waktu
1	Pre-test	15 menit
2	Materi SP	45 menit
3	Latihan unguided	45 menit
4	Post-test	45 menit
	Total waktu	150 menit

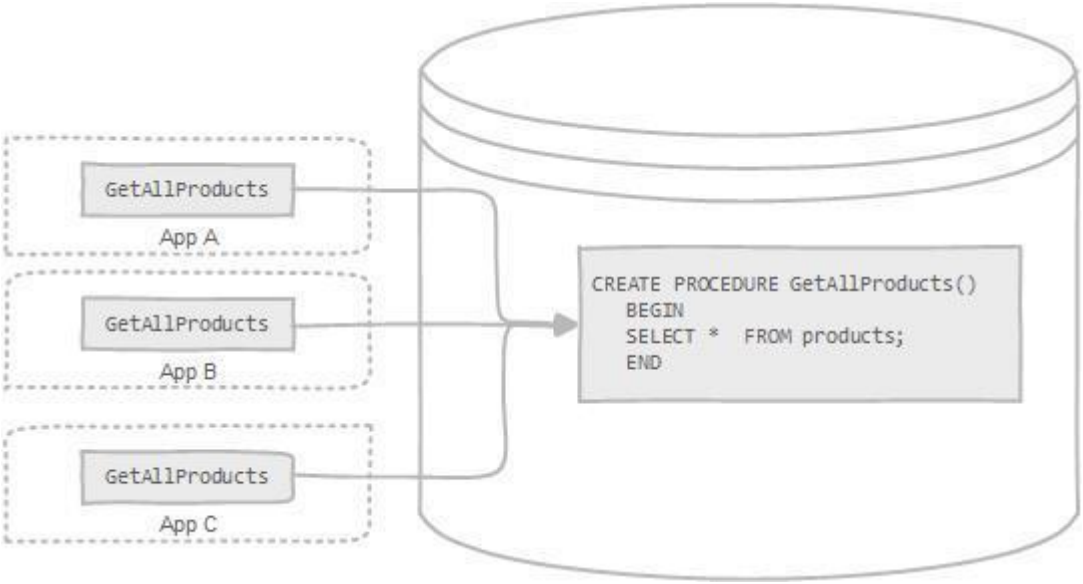
Penilaian

Pada modul dilakukan pengambilan nilai pada pre dan post-test dan bonus pengerjaan latihan.

Materi Praktikum (45 menit)

1. **Pengertian SP**

   SP adalah kumpulan instruksi-instruksi / program untuk suatu hal tertentu yang spesifik dan ditulis dalam bahasa SQL, disimpan dalam basis data serta dapat dipanggil secara spesifik menggunakan suatu nama tertentu dan bahkan dipanggil dari suatu bahasa pemrograman. SP dapat dipanggil oleh triggers, SP lain, dan aplikasi lain. SP juga dapat bersifat rekursif. Beberapa DBMS besar mendukung SP, seperti MySQL dan Oracle. Gambar 1 merupakan gambar ilustrasi SP.



Gambar 1. Ilustrasi SP

2. **Keuntungan dan Kelemahan SP**

   Keuntungan SP antara lain:

   a. SP sangat mengurangi overhead pemrosesan SQL pada basis data
   b. SP memiliki kecepatan yang tinggi dalam pemrosesan SQL karena disimpan di dalam basis data dan langsung dijalankan di server.
   c. SP bersifat reusable seperti layaknya fungsi / prosedur dalam bahasa pemrograman.
   d. SP bersifat Write Once and Run Everywhere. Sekali ditulis, SP akan disimpan di server dan dipanggil dengan mudah serta memiliki sintaks standar.
   e. SP digunakan dalam aspek keamanan database. Pada kasus tertentu SP menjembatani akses data terhadap suatu tabel di mana user tidak bisa melakukan query terhadap tabel. Permission select terhadap suatu tabel kadang tidak diperbolehkan karena masalah data sensitif. Untuk menjembatani hal tersebut, cukup dibuatkan priviledge EXECUTE tanpa perlu melakukan SELECT secara eksplisit terhadap user tersebut, sehingga user tersebut hanya perlu menjalankan stored procedure untuk hal-hal tertentu saja.

   Kelemahan SP antara lain:

   a. SP tidak memiliki version control, sehingga modifikasi terhadap suatu stored procedure tidak ada historinya, dan tidak bisa dikembalikan ke kondisi semula.

b. SP hanya bisa di export / backup dengan priviledge administrator database, tidak bisa pengguna biasa.
c. Jika menggunakan banyak SP, memory yang digunakan cukup besar.
d. Sulit untuk mendebug SP karena jarang ada editor yang memiliki fitur tersebut.

3. Implementasi SP

Untuk membuat SP pada MySQL, bentuk umumnya adalah sebagai berikut:

```
CREATE
  [OR REPLACE]
  PROCEDURE [IF NOT EXISTS] [nama procedure]
      ([IN|OUT|INOUT] [param1] [tipedata], .....)
  { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
BEGIN
      [perintah-perintah]
END
```

Jika klausa IF NOT EXISTS digunakan, maka procedure hanya akan dibuat jika sebuah procedure dengan nama sama belum pernah ada. Jika sudah ada, maka akan muncul peringatan.

Penjelasan parameter pada SP adalah:

- **IN** – parameter IN akan menerima nilai masukan ke dalam procedure. Procedure bisa memodifikasi nilainya, namun modifikasi ini tidak terlihat dari sisi pemanggil saat procedure selesai melakukan tugasnya.
- **OUT** – parameter OUT akan melempar nilai dari procedure kembali ke sisi pemanggil. Nilai awal di dalam procedure secara default adalah NULL. Karena sifatnya, maka nilai parameter ini otomatis terlihat dari sisi pemanggil saat procedure selesai melakukan tugasnya.
- **INOUT** – gabungan dari kedua jenis di atas. Pemanggil melemparkan suatu nilai, bisa diproses oleh procedure, kemudian hasil modifikasi nilai ini akan terlihat dari sisi pemanggil.

Pernyataan CONTAINS SQL, NO SQL, READS SQL DATA, atau MODIFIES SQL DATA hanyalah klausa untuk memberitahu server mengenai apa yang dilakukan oleh procedure. Jika kontennya tidak sesuai dengan pernyataan yang ditulis, MariaDB tidak akan mempermasalahkan. Jika tidak ditulis, defaultnya CONTAINS SQL.

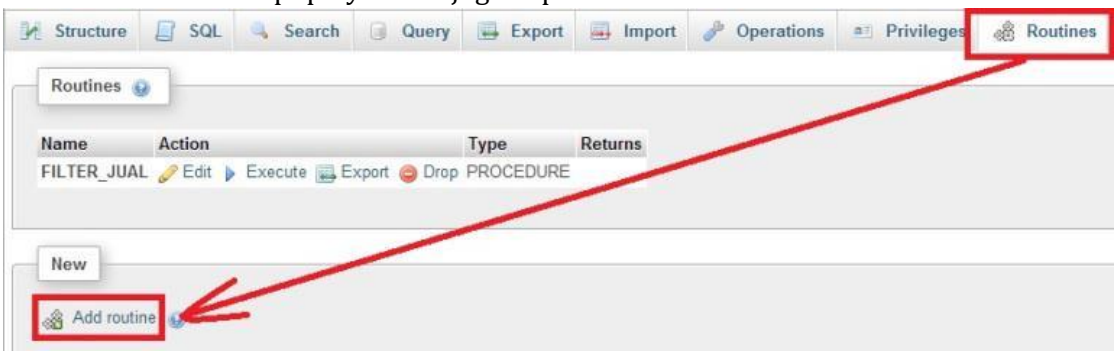
Contoh SP sederhana untuk menghitung rata-rata gaji adalah:

```
DELIMITER $$
CREATE PROCEDURE `avg_sal`(OUT avg_sal decimal)
BEGIN
      SELECT AVG(sal) INTO avg_sal FROM salary;
END$$
DELIMITER ;
```

Cara pemanggilan SP pada SQL query:

```
CALL avg_sal(@out);
SELECT @out;
```

Pembuatan SP dalam phpMyAdmin juga dapat melalui menu Routine.



4. **Cara menampilkan SP pada Database MySQL**

Cara untuk menampilkan stored procedure yang ada di database MySQL adalah menggunakan sintaks:

```
SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE expr];
```

Contoh:

```
SHOW PROCEDURE STATUS;
SHOW PROCEDURE STATUS WHERE db = 'dbmahasiswa';
SHOW PROCEDURE STATUS WHERE name LIKE '%product%';
```

Sedangkan untuk menampilkan source code dari SP adalah:

```
SHOW CREATE PROCEDURE stored_procedure_name;
SHOW CREATE FUNCTION function_name;
```

Function akan kita bahas pada pertemuan berikutnya.

Selain menjalankan dengan cara CALL, SP pada phpMyAdmin juga dapat dijalankan melalui interface EXECUTE.

5. **Penggunaan variable pada SP**

5.1 Variabel Lokal Procedure

Seperti pada pemrograman pada umumnya, kita bisa menggunakan variabel lokal pada function dan procedure. Pendeklarasian variabel memiliki sintaks sebagai berikut:

```
DECLARE var_name [, var_name] ... type [DEFAULT value]
```

Contoh:

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

Pemberian nilai ke sebuah variabel dilakukan dengan menggunakan statement SET. Hasil dari query juga dapat dimasukkan ke dalam variabel menggunakan SELECT ... INTO .... Berikut adalah beberapa contoh pemberian nilai ke variabel.

```
SET total_sale = 50;
DECLARE numPekerja INT;
SELECT COUNT(*) INTO numPekerja FROM pekerja;
```

5.2 Variabel Lokal User

Variabel lokal pada level user dibuat dengan menggunakan simbol @ diikuti dengan nama variabelnya. Pendeklarasian tersebut bisa dilakukan di console dan di baris-baris SQL. Cara membaca data dari variabel user tersebut dengan menggunakan perintah **`SELECT @nama_variabel`**.

Contoh:

```
mysql> set @jum_sks := 1+1;
Query OK, 0 rows affected (0.11 sec)

mysql> SELECT @jum_sks;
+----------+
| @jum_sks |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT * FROM mata_kuliah WHERE jumlah_SKS = @jum_sks;
+-------------+------------------+------------+----------+------------+
| kode_matkul | nama_matkul      | jumlah_SKS | semester | NIP_dosen  |
+-------------+------------------+------------+----------+------------+
| FISDAS      | Fisika Dasar     |          2 |        1 | 0480432066 |
| MIKROP      | Mikro Prosesor   |          2 |        5 | 0480432066 |
| SISOPR      | Sistem Operasi   |          2 |        4 | 0160436012 |
| TEKKOM      | Teknik Kompilasi |          2 |        6 | 0480432066 |
+-------------+------------------+------------+----------+------------+
4 rows in set (0.00 sec)
```

6. Penggunaan Flow Control IF pada SP

SP pada prinsipnya seperti program/fungsi pada pemrograman biasa. Sehingga pada MySQL kita juga dapat membuat SP dimana di dalamnya mengandung unsur percabangan atau IF. Dengan menggunakan IF maka kita bisa mengeksekusi kumpulan statement SQL berbasis pada kondisi tertentu atau nilai dari suatu ekspresi. Untuk membentuk ekspresi pada MySQL, kita dapat menggabungkan literal, variabel, operator, dan bahkan fungsi. Suatu ekspresi dapat mengembalikan hasil TRUE, FALSE, atau NULL.

Bentuk umum perintah IF pada MySQL:

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

Jika ada dua kondisi yang akan dibandingkan:

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

Jika ada lebih dari dua kondisi lain yang akan dibandingkan:

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

7. Penggunaan Flow Control CASE pada SP

Selain menggunakan model IF, SP MySQL juga mendukung CASE, mirip dengan SWITCH CASE pada bahasa pemrograman biasa. Terdapat dua model CASE: Simple CASE dan Searched CASE.

Bentuk umum perintah Simple CASE adalah:

```
CASE case_expression
      WHEN value1 THEN commands
      WHEN value2 THEN commands
      ...
      ELSE commands
END CASE;
```

Bentuk umum perintah Searched CASE adalah:

```
CASE
      WHEN expression_value1 THEN commands
      WHEN expression_value2 THEN commands
      ...
      ELSE commands
END CASE;
```

8. Penggunaan Flow Control LOOP pada SP

Perulangan pada SP ada dua macam:

– Perulangan WHILE

Bentuk umum:

```
WHILE expression DO
      statements
END WHILE;
```

Contoh:

```
DECLARE income INT;
```

```
SET income = 0;
WHILE income <= 3000 DO
      SET income = income + starting_value;
END WHILE;
```

– Perulangan REPEAT
Bentuk umum:

```
REPEAT
      statements;
UNTIL expression
END REPEAT;
```

Contoh:

```
DECLARE income INT;
SET income = 0;
REPEAT
      SET income = income + starting_value;
      UNTIL income >= 4000
END REPEAT;
```

LATIHAN 1 (VARIABEL) - Pastikan database **db_SP1.sql** sudah terimpor.

1. a. Perhatikan tabel **offices**. Buatlah SP **getOfficeByCountry** menggunakan parameter IN bernama **countryName** VARCHAR(255) untuk melihat semua kantor dari tabel **offices** berdasarkan negara yang diinputkan pada countryName.

1. b. Panggil SP yang telah dibuat, inputkan untuk negara USA.
Hasil:

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

1. c. Perhatikan tabel **orders**. Buatlah SP **countOrderByStatus** menggunakan parameter IN dan OUT untuk mendapatkan data jumlah order dari tabel **orders** berdasarkan status ordernya. Parameternya ada dua, untuk IN: **orderStatus** VARCHAR(25) dan untuk OUT: **total** INT. Clue: hasil jumlah order status harus disimpan ke dalam variabel total untuk dikeluarkan di parameter out.

1. d. Panggilah stored procedure yang telah dibuat dan inputkan parameter in = 'Shipped' dan gunakan @total untuk parameter outnya. Setelah itu ambil variabel total tersebut menggunakan SELECT.
Hasil:

	@total
▶	303

1. e. Buat SP **setCounter** dengan menggunakan parameter INOUT untuk menampilkan counter yang terus bertambah dari 1, 2, 3, 4 dst. Fungsi ini membutuhkan 2 parameter, yaitu **count** INT(4) berjenis INOUT, dan **inc** INT(4) berjenis IN. Count digunakan untuk mendapatkan data hasil pertambahan counter, sedangkan inc digunakan untuk bertambahnya counter.

1. f. Gunakan SP tersebut, pertama set @counter = 1, kemudian increment dengan nilai tertentu: 1, 1, dan 5, sehingga total nilai @counter = 8. Tampilkan nilai @counter dengan menggunakan perintah SELECT.

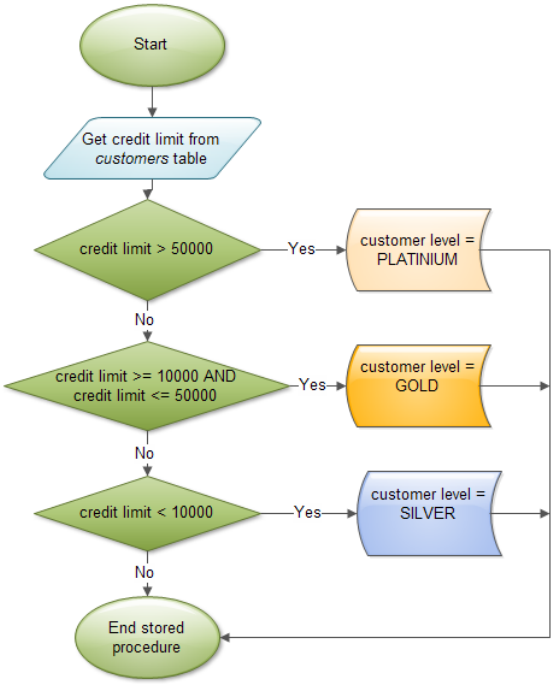
LATIHAN 2 (IF) – Perhatikan struktur tabel **customers:**

#	Name	Datatype	Length/Set	Unsign...	Allow NULL	Zerofill	Default
1	**customerNumber**	**INT**	**11**	☐	▨	☐	**No default**
2	customerName	VARCHAR	50	▨	☐	▨	No default
3	contactLastName	VARCHAR	50	▨	☐	▨	No default
4	contactFirstName	VARCHAR	50	▨	☐	▨	No default
5	phone	VARCHAR	50	▨	☐	▨	No default
6	addressLine1	VARCHAR	50	▨	☐	▨	No default
7	addressLine2	VARCHAR	50	▨	☑	▨	NULL
8	city	VARCHAR	50	▨	☐	▨	No default
9	state	VARCHAR	50	▨	☑	▨	NULL
10	postalCode	VARCHAR	15	▨	☑	▨	NULL
11	country	VARCHAR	50	▨	☐	▨	No default
12	salesRepEmploye...	INT	11	☐	☑	☐	NULL
13	creditLimit	DECIMAL	10,2	☐	☑	☐	NULL

Berikut contoh isi data tabel **customers**:

customerNumber	customerName	creditLimit
103	Atelier graphique	21,000.00
112	Signal Gift Stores	71,800.00
114	Australian Collectors, Co.	117,300.00
119	La Rochelle Gifts	118,200.00
121	Baane Mini Imports	81,700.00
124	Mini Gifts Distributors Ltd.	210,500.00
125	Havel & Zbyszek Co	0.00
128	Blauer See Auto, Co.	59,700.00
129	Mini Wheels Co.	64,600.00
131	Land of Toys Inc.	114,900.00
141	Euro+ Shopping Channel	227,600.00
144	Volvo Model Replicas, Co	53,100.00
145	Danish Wholesale Imports	83,400.00
146	Saveley & Henriot, Co.	123,900.00
148	Dragon Souveniers, Ltd.	103,800.00
151	Muscle Machine Inc	138,500.00
157	Diecast Classics Inc.	100,600.00
161	Technics Stores Inc.	84,600.00
166	Handji Gifts& Co	97,900.00
167	Herkku Gifts	96,800.00
168	American Souvenirs Inc	0.00

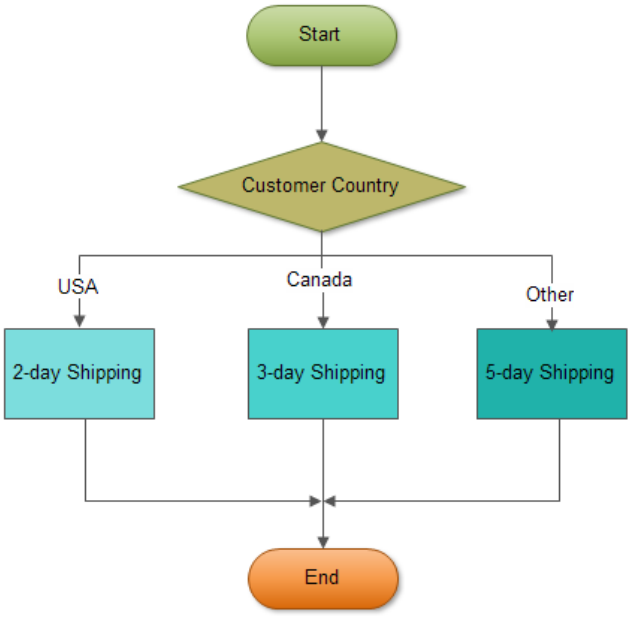
Berdasarkan tabel di atas, akan dibuat penentuan level customer berdasarkan limit kartu kreditnya. Lihat flowchart berikut:



2. a. Buatlah SP **getCustomerLevel** dengan parameter IN berupa **customerNumber**, parameter OUT berupa **customerLevel**.
2. b. Panggil prosedur tersebut dan tampilkan outputnya berdasarkan customerNumber yang dimasukkan.

LATIHAN 3 (CASE) – Perhatikan struktur tabel **customers:**

3. a. Buatlah SP pada MySQL dengan nama **getCustomerShipping** untuk menampilkan lama waktu pengiriman customer berdasarkan countrynya. Parameter IN berisi **customerNumber** untuk kemudian diketahui countrynya. Parameter OUT berupa jenis shipping. Lihat flowchart pada halaman berikut untuk lebih jelasnya.



3. b. Panggil prosedur tersebut dan tampilkan outputnya berdasarkan customerNumber yang dimasukkan.

LATIHAN 4

4. a. Buatlah tabel penerbit sebagai berikut:

Name	Type
kode_penerbit	char(4)
nama_penerbit	varchar(150)
kota_penerbit	varchar(100)
email_penerbit	varchar(200)
website_penerbit	varchar(200)
telepon_penerbit	varchar(20)

4. b. Buatlah SP untuk menambah data penerbit dengan nama **tambahPenerbit.** Parameter IN adalah semua nama field pada tabel penerbit. Kemudian gunakan SP tersebut!

4. c. Buatlah SP untuk menghitung jumlah kota penerbit yang sama dengan nama **hitungJumlahKota**. Parameter IN adalah namaKota, OUT adalah jumlah kota. Gunakan SP tersebut!

4. d. Buatlah SP untuk menghapus data penerbit dengan nama **hapusPenerbit**. Parameter input adalah email penerbit. Gunakan SP tersebut!

REFERENSI:

MySQL Stored Prcedures - https://www.sitepoint.com/stored-procedures-mysql-php/

Steven Feuerstein, Guy Harrison, "MySQL SP Programming", 2006, O'Reilly. ISBN: 978-0-59-610089-6

MySQL Stored Procedures Tutorial : http://www.mysqltutorial.org/mysql-stored-proceduretutorial.aspx