

PRAKTEK DBMS LANJUT

MINGGU 4-STORED PROCEDURE

A. TUJUAN

- Memahami konsep dasar stored procedure, kelebihan dan kekurangannya.
- Memahami implementasi stored procedure di dalam basis data.
- Mampu menyelesaikan operasi – operasi data spesifik dengan memanfaatkan stored procedure

B. PETUNJUK

- Awali setiap aktivitas dengan doa, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan – latihan praktikum dengan baik dan benar.
- Kerjakan tugas – tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal – hal yang kurang jelas.
-

C. DASAR TEORI

1. Stored Procedure

Stored Procedure adalah sebuah prosedur layaknya subprogram (subrutin) di dalam bahasa pemrograman reguler yang tersimpan di dalam katalog basis data.

Beberapa kelebihan yang ditawarkan stored procedure antara lain : meningkatkan performa, mereduksi trafik jaringan, reusable, dan meningkatkan kontrol sekuriti.

Di balik kelebihan tersebut, stored procedure juga memiliki kekurangan. Di antaranya adalah berpotensi meningkatkan beban server dan penulisnya tidak mudah (memerlukan pengetahuan yang spesifik).

Contoh sintaks stored procedure :

```
CREATE PROCEDURE sp_name ([proc_parameter[,...]])  
    [characteristic ...] routine_body
```

Untuk memanggil stored procedure, digunakan perintah CALL (beberapa DBMS ada yang menggunakan EXECUTE).

```
CALL sp_name
```

Dalam Implementasinya, penggunaan stored procedure sering melibatkan parameter. Di MySQL, parameter stored procedure dibedakan menjadi tiga mode : IN, OUT, dan INOUT.

IN

Parameter yang merupakan mode default ini mengindikasikan bahwa sebuah parameter dapat di-pass ke dalam stored procedure tetapi nilainya tidak dapat diubah (dari dalam stored procedure

OUT

Mode ini mengindikasikan bahwa stored procedure dapat mengubah parameter dan mengirimkan kembali ke program pemanggil

INOUT

Mode ini pada dasarnya merupakan kombinasi dari mode IN dan OUT.

Sintaks pendefinisian parameter diperlihatkan sebagai berikut :

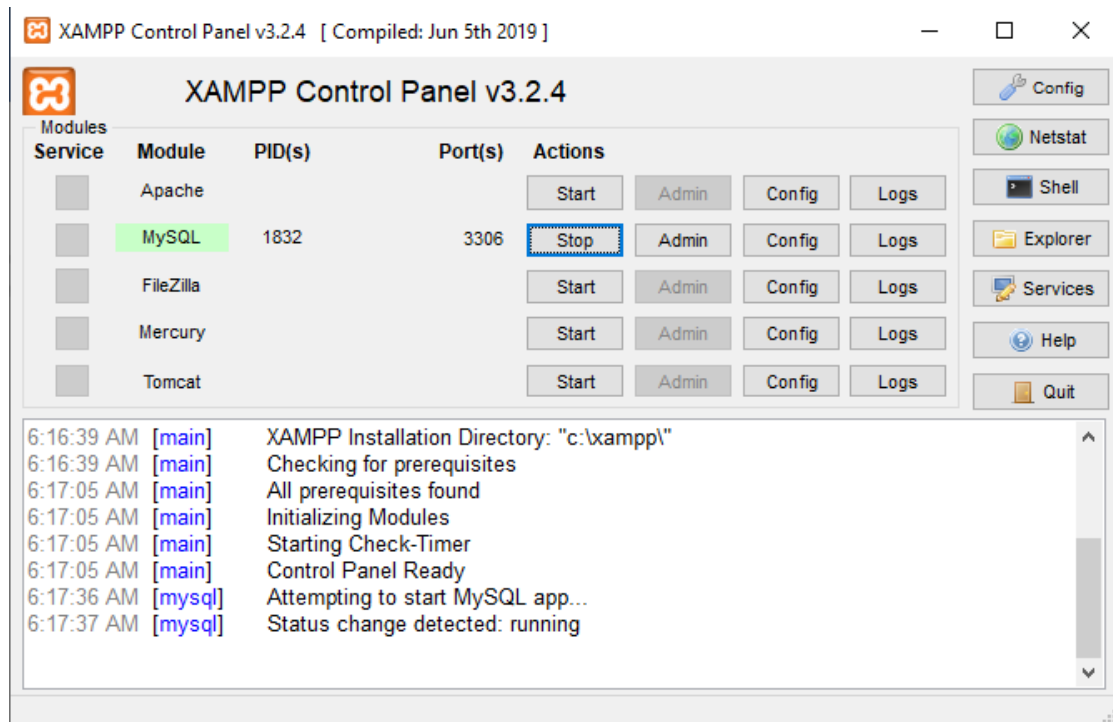
```
MODE param name param type (param size)
```

Stored procedure dapat mencerminkan beragam operasi data, misalnya seleksi, penambahan, pengubahan, penghapusan, dan juga operasi – operasi DDL.

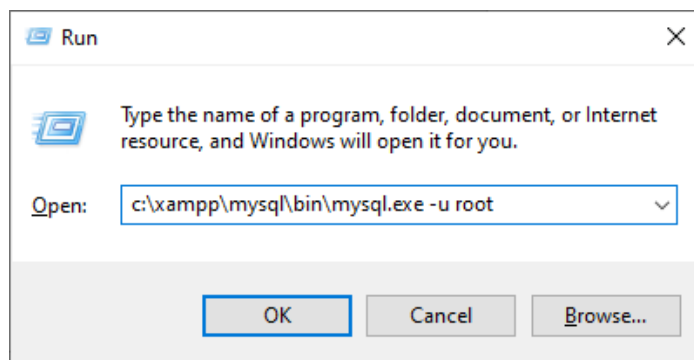
Seperti halnya procedure di dalam bahasa pemrograman, stored procedure juga dapat melibatkan variabel, pernyataan kondisional, dan pengulangan.

D. TAHAPAN PRAKTIKUM

Aktifkan server database MySQL melalui XAMPP Control Panels



Aktifkan MySQL Clients



```
C:\xampp\mysql\bin\mysql.exe
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.13-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)]> show databases;
+-----+
| Database |
+-----+
| classicmodels |
| db_desa |
| db_portal_berita |
| information_schema |
| moodledb |
| mysql |
| performance_schema |
| phpmyadmin |
| simdes |
| test |
| webservice_nira |
+-----+
11 rows in set (0.165 sec)

MariaDB [(none)]>
```

Aktifkan database classicmodels

```
MariaDB [(none)]> use classicmodels;
Database changed
MariaDB [classicmodels]> show tables;
+-----+
| Tables_in_classicmodels |
+-----+
| customers |
| employees |
| offices |
| orderdetails |
| orders |
| payments |
| productlines |
| products |
+-----+
8 rows in set (0.001 sec)

MariaDB [classicmodels]> 
```

```
MariaDB [classicmodels]> desc customers;
```

Field	Type	Null	Key	Default	Extra
customerNumber	int(11)	NO	PRI	NULL	
customerName	varchar(50)	NO		NULL	
contactLastName	varchar(50)	NO		NULL	
contactFirstName	varchar(50)	NO		NULL	
phone	varchar(50)	NO		NULL	
addressLine1	varchar(50)	NO		NULL	
addressLine2	varchar(50)	YES		NULL	
city	varchar(50)	NO		NULL	
state	varchar(50)	YES		NULL	
postalCode	varchar(15)	YES		NULL	
country	varchar(50)	NO		NULL	
salesRepEmployeeNumber	int(11)	YES	MUL	NULL	
creditLimit	decimal(10,2)	YES		NULL	

```
13 rows in set (0.103 sec)
```

Membuat stored procedure untuk menampilkan data customers, tuliskan perintah berikut:

```
MariaDB [classicmodels]> delimiter //
MariaDB [classicmodels]> create procedure getCustomers()
-> begin
-> select customerNumber,customerName,country from customers;
-> end //
Query OK, 0 rows affected (0.209 sec)

MariaDB [classicmodels]> delimiter ;
MariaDB [classicmodels]>
```

Menjalankan stored procedure yang baru saja dibuat.

```
MariaDB [classicmodels]> call getCustomers();
```

customerNumber	customerName	country
103	Atelier graphique	France
112	Signal Gift Stores	USA
114	Australian Collectors, Co.	Australia
119	La Rochelle Gifts	France
121	Baane Mini Imports	Norway
124	Mini Gifts Distributors Ltd.	USA
125	Havel & Zbyszek Co	Poland
128	Blauer See Auto, Co.	Germany
129	Mini Wheels Co.	USA
131	Land of Toys Inc.	USA

141	Euro+ Shopping Channel	Spain
144	Volvo Model Replicas, Co	Sweden
145	Danish Wholesale Imports	Denmark
146	Saveley & Henriot, Co.	France
148	Dragon Souvenirs, Ltd.	Singapore
151	Muscle Machine Inc	USA
157	Diecast Classics Inc.	USA
161	Technics Stores Inc.	USA
166	Handji Gifts& Co	Singapore
167	Herkku Gifts	Norway
168	American Souvenirs Inc	USA
169	Porto Imports Co.	Portugal
171	Daedalus Designs Imports	France
172	La Corne D'abondance, Co.	France
173	Cambridge Collectables Co.	USA
175	Gift Depot Inc.	USA
177	Osaka Souvenirs Co.	Japan
181	Vitachrome Inc.	USA
186	Toys of Finland, Co.	Finland
187	AV Stores, Co.	UK
189	Clover Collections, Co.	Ireland
198	Auto-Moto Classics Inc.	USA
201	UK Collectables, Ltd.	UK
202	Canadian Gift Exchange Network	Canada
204	Online Mini Collectables	USA
205	Toys4GrownUps.com	USA
206	Asian Shopping Network, Co	Singapore
209	Mini Caravy	France
211	King Kong Collectables, Co.	Hong Kong
216	Enaco Distributors	Spain
219	Boards & Toys Co.	USA
223	Natürlich Autos	Germany
227	Heintze Collectables	Denmark
233	Québec Home Shopping Network	Canada
237	ANG Resellers	Spain
239	Collectable Mini Designs Co.	USA
240	giftsbymail.co.uk	UK
242	Alpha Cognac	France
247	Messner Shopping Network	Germany
249	Amica Models & Co.	Italy
250	Lyon Souvenirs	France
256	Auto Associés & Cie.	France
259	Toms Spezialitäten, Ltd	Germany
260	Royal Canadian Collectables, Ltd.	Canada
273	Franken Gifts, Co	Germany
276	Anna's Decorations, Ltd	Australia
278	Rovelli Gifts	Italy
282	Souvenirs And Things Co.	Australia
286	Marta's Replicas Co.	USA
293	BG&E Collectables	Switzerland
298	Vida Sport, Ltd	Switzerland

299	Norway Gifts By Mail, Co.	Norway
303	Schuyler Imports	Netherlands
307	Der Hund Imports	Germany
311	Oulu Toy Supplies, Inc.	Finland
314	Petit Auto	Belgium
319	Mini Classics	USA
320	Mini Creations Ltd.	USA
321	Corporate Gift Ideas Co.	USA
323	Down Under Souvenirs, Inc	New Zealand
324	Stylish Desk Decors, Co.	UK
328	Tekni Collectables Inc.	USA
333	Australian Gift Network, Co	Australia
334	Suominen Souvenirs	Finland
335	Cramer Spezialitäten, Ltd	Germany
339	Classic Gift Ideas, Inc	USA
344	CAF Imports	Spain
347	Men 'R' US Retailers, Ltd.	USA
348	Asian Treasures, Inc.	Ireland
350	Marseille Mini Autos	France
353	Reims Collectables	France
356	SAR Distributors, Co	South Africa
357	GiftsForHim.com	New Zealand
361	Kommission Auto	Germany
362	Gifts4AllAges.com	USA
363	Online Diecast Creations Co.	USA
369	Lisboa Souvenirs, Inc	Portugal
376	Precious Collectables	Switzerland
379	Collectables For Less Inc.	USA
381	Royale Belge	Belgium
382	Salzburg Collectables	Austria
385	Cruz & Sons Co.	Philippines
386	L'ordine Souvenirs	Italy
398	Tokyo Collectables, Ltd	Japan
406	Auto Canal+ Petit	France
409	Stuttgart Collectable Exchange	Germany
412	Extreme Desk Decorations, Ltd	New Zealand
415	Bavarian Collectables Imports, Co.	Germany
424	Classic Legends Inc.	USA
443	Feuer Online Stores, Inc	Germany
447	Gift Ideas Corp.	USA
448	Scandinavian Gift Ideas	Sweden
450	The Sharp Gifts Warehouse	USA
452	Mini Auto Werke	Austria
455	Super Scale Inc.	USA
456	Microscale Inc.	USA
458	Corrida Auto Replicas, Ltd	Spain
459	Warburg Exchange	Germany
462	FunGiftIdeas.com	USA
465	Anton Designs, Ltd.	Spain
471	Australian Collectables, Ltd	Australia
473	Frau da Collezione	Italy

475	West Coast Collectables Co.	USA
477	Mit Vergnügen & Co.	Germany
480	Kremlin Collectables, Co.	Russia
481	Raanan Stores, Inc	Israel
484	Iberia Gift Imports, Corp.	Spain
486	Motor Mint Distributors Inc.	USA
487	Signal Collectibles Ltd.	USA
489	Double Decker Gift Stores, Ltd	UK
495	Diecast Collectables	USA
496	Kelly's Gift Shop	New Zealand

122 rows in set (0.036 sec)

Query OK, 0 rows affected (0.521 sec)

MariaDB [classicmodels]>

E. LATIHAN

1. Stored Procedure

Seperti halnya tabel, stored procedure diciptakan dengan menggunakan perintah CREATE sebagai contoh, buat stored procedure getMahasiswa() untuk menampilkan semua data mahasiswa.

1. Ketikkan pernyataan pembuatan stored procedure berikut :

```
DELIMITER //
CREATE PROCEDURE getMhs()
BEGIN
    SELECT * FROM mahasiswa;
END //
DELIMITER;
```

Perintah DELIMITER digunakan untuk mengubah delimiter standar, misalnya di sini dari titik koma (;) menjadi slash ganda (//). Langkah ini umumnya dilakukan ketika isi stored procedure mengandung titik

koma

– yang merupakan delimiter standar di SQL.

Pernyataan di antara BEGIN dan END merupakan badan (*body*) stored procedure.

Perintah DELIMITER di akhir baris digunakan untuk mengembalikan delimiter ke karakter semula.

2. Eksekusi Query tersebut dengan memanggil procedure `getMahasiswa()`.

```
CALL getMhs();
```

nim	nama	Jenis_Kelamin	alamat
101	Arif	L	Jl. Kenangan
102	Budi	L	Jl. Jombang
103	Wati	P	Jl. Surabaya
104	Ika	P	Jl. Jombang
105	Tono	L	Jl. Jakarta
106	Sari	P	Jl. Malang

2. Parameter IN

Stored procedure di contoh sebelumnya memperlihatkan bentuk default (tanpa parameter). Di sini kita juga bisa mendefinisikan parameter yang nantinya dapat digunakan olehh pernyataan di body stored procedure.

Sebagai contoh, kita bisa mendapatkan semua data matakuliah di semester tertentu.

```
DELIMITER //  
CREATE PROCEDURE getMkBySemester(IN smt INT(2))  
BEGIN  
    SELECT * FROM matakuliah  
    WHERE semester = smt;  
END //  
DELIMITER;
```

Untuk memanggil stored procedure yang memiliki parameter, maka kita harus menspesifikasikan argumenya. Misalkan kita ingin mendapatkan data matakuliah di semester 3.

CALL getMkBySemester (3) ;

kd_mk	nama_mk	sks	semester	kode_dos
PTI447	Praktikum Basis Data	1	3	11
PTI777	Sistem Informasi	2	3	99
TIK342	Praktikum Basis Data	1	3	11

Apabila pemanggilan stored procedure di atas mengabaikan argumen, DBMS akan merespon dengan pesan kesalahan.

Bergantung kebutuhan, pendefinisian parameter pada stored procedure juga bisa lebih dari satu. Sebagai contoh, buat stored procedure dengan dua buah parameter seperti berikut :

```
DELIMITER //
CREATE PROCEDURE getMkBySemSks (
    IN Smt INT(2),
    IN Sks INT(2))
BEGIN
    SELECT * FROM matakuliah
    WHERE semester = Smt
    AND sks = Sks ;
END //
DELIMITER;
```

Pemanggilan stored procedure di atas tentunya akan memerlukan dua buah argumen.

CALL getMkBySemSks (3,2) ;

kd_mk	nama_mk	sks	semester	kode_dos
PTI777	Sistem Informasi	2	3	99

Variabel

Di MySQL, kita juga bisa mendeklarasikan variabel global – ruang lingkup session – dengan menggunakan perintah SET dan notasi @. Sebagai contoh, perintah berikut

akan mendeklarasikan variabel bernama **smt** dan diinisialisasi dengan nilai **3**. Dan untuk memeriksa nilai variabel, gunakan perintah select

```
SET @smt = 3;
SELECT @smt;
```

@smt
3

Langkah selanjutnya, kita bisa memanfaatkan variabel yang telah dideklarasikan untuk operasi – operasi lain, misalnya sebagai argumen stored procedure.

```
CALL getMHSBySemester (@smt);
```

kd_mk	nama_mk	sks	semester	kode_dos
PTI447	Praktikum Basis Data	1	3	11
PTI777	Sistem Informasi	2	3	99
TIK342	Praktikum Basis Data	1	3	11

Penambahan Data

Pada operasi penambahan, data – data terkait diisikan melalui argumen. Selanjutnya, isi stored procedure akan memasukkan data ke dalam tabel.

Berikut adalah contoh stored procedure untuk menambahkan data pada tabel dosen

```
DELIMITER //
CREATE PROCEDURE AddDosen(
    IN kode_dos VARCHAR(10),
    IN nama_dos VARCHAR(50),
    IN alamat_dos VARCHAR(100)
)
BEGIN
    INSERT INTO dosen VALUES(
        kode_dos,nama_dos,alamat_dos);
END //
DELIMITER;
```

Lakukan eksekusi terhadap procedure tersebut

```
call AddDosen('212','Gunawan','Jl. Ambarawa');
```

Selanjutnya lakukan pengecekan data pada tabel dosen.

```
select * from dosen;
```

kode_dos	nama_dos	alamat_dos
10	Suharto	Jl. Jombang
11	Martono	Jl. Kalpataru
12	Rahmawati	Jl. Jakarta
13	Bambang	Jl. Bandung
14	Nurul	Jl. Raya Tidar
212	Gunawan	Jl. Ambarawa

Operasi – operasi manipulasi data lainya bisa anda coba sendiri, dan tidak jauh berbeda dengan pernyataan SQL reguler

3. Parameter OUT

Dalam konteks bahasa pemrograman, parameter OUT analog dengan *passing-by-reference*. Dengan demikian, parameter ini nilainya bisa diubah oleh stored procedure.

```
DELIMITER //
CREATE PROCEDURE JumlahDosen(
    OUT jumlah_dos INT(3)
)
BEGIN
    SELECT COUNT(kode_dos)
    INTO jumlah_dos FROM dosen;
END //
DELIMITER;
```

Untuk mengeksekusi stored procedure dengan parameter OUT, dibutuhkan argumen yang spesifik.

```
call JumlahDosen(@jumlah_dosen);
```

Perhatikan, argumen harus menggunakan notasi @, yang mengindikasikan sebagai suatu parameter OUT.

Langkah selanjutnya, untuk mendapatkan nilai variabel, gunakan pernyataan

SELECT

```
select @jumlah_dosen;
```

@jumlah_dosen

Parameter mode OUT juga bisa dikombinasikan dengan mode IN.

4. Parameter INOUT

Pada parameter dengan mode INOUT ini, kita bisa mengirimkan parameter kedalam stored procedure dan mendapatkan nilai kembalian yang baru dari stored procedure yang didefinisikan.

Sebagai contoh, definisikan stored procedure seperti berikut :

```
DELIMITER //
CREATE PROCEDURE CountBySks(
    INOUT var INT(3)
)
BEGIN
    SELECT COUNT(kd_mk)
    INTO var
    FROM matakuliah
    WHERE sks=var;
END //
DELIMITER;
```

Lakukan eksekusi pada procedure tersebut untuk mencari jumlah matakuliah yang memiliki sks = 2. Dengan mendeklarasikan variabel @sks dengan nilai 2 terlebih dahulu.

```
set @sks=2;
```



@sks
2

```
call CountBySks(@sks);
```

Lakukan pengecekan pada variabel sks setelah dilakukan eksekusi pada stored procedure tersebut.

```
select @sks;
```



@sks
2

Pendekatan INOUT juga dapat direpresentasikan dengan memisah parameter IN dan OUT

Contoh penggunaanya, misal untuk mendapatkan jumlah mahasiswa yang jenis kelaminnya adalah L.

```

DELIMITER //
CREATE PROCEDURE CountByGender(
    IN gender VARCHAR(3),
    OUT total INT(3)
)
BEGIN
    SELECT COUNT(nim)
    INTO total
    FROM mahasiswa
    WHERE Jenis_Kelamin=gender;
END //
DELIMITER;

```

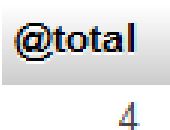
Kemudian lakukan eksekusi pada procedure tersebut.

```

CALL CountByGender('L',@total);
select @total;

```

Maka akan didapat jumlah dari mahasiswa yang berjenis kelamin L adalah :



The screenshot shows a variable named **@total** with a value of **4**.

5. Pencabangan dan Pengulangan

Penggunaan pernyataan – pernyataan percabangan ataupun pengulangan dapat dilakukan di dalam stored procedure. Sehingga dapat digunakan untuk menghasilkan suatu procedure yang lebih kompleks.

Contoh berikut merupakan penggunaan dari pernyataan IF di dalam stored procedure.

```

DELIMITER //
CREATE PROCEDURE DemoIF(
    IN bil INT(3)
)
BEGIN
    DECLARE str VARCHAR(50);
    IF(bil<0) THEN
        SET str = 'BILANGAN NEGATIF';
    ELSE
        SET str = 'BILANGAN POSITIF';
    END IF;

    SELECT str;
END //

```

Di dalam procedure tersebut membutuhkan beberapa variabel tambahan seperti str, sehingga di deklarasikan variabel str dengan tipe varchar. Variabel ini digunakan sebagai keluaran dari procedure tersebut.

Kemudian eksekusi stored procedure tersebut dengan memberikan parameter bilangan berapapun.

```
CALL DemoIF(7);
```

str
BILANGAN POSITIF

Contoh tersebut merupakan contoh untuk pernyataan kondisi didalam stored procedure. Dan berikut merupakan contoh penggunaan perulangan atau LOOPING pada stored procedure.

```
DELIMITER //
CREATE PROCEDURE DemoLOOP(
    IN bil INT(3)
)
BEGIN
    DECLARE str VARCHAR(50);
    DECLARE i INT(3);
    SET i=1;
    SET str='';
    WHILE i <= bil DO
        SET str=CONCAT(str,i,',');
        set i=i+1;
    END WHILE;

    SELECT str;
END //
DELIMITER;
```

Lakukan eksekusi pada procedure tersebut.

```
CALL DemoLOOP(9);
```

str
1,2,3,4,5,6,7,8,9,

F. TUGAS MINGGU 4

Buatlah database dengan nama **db_sp_noabsen_nama**, buat table dengan struktur seperti di bawah ini dan isikan datanya.

Screenshot setiap tahapan dan hasilnya, upload file dokumentasi ke elearning, nama file:

DBMSL-20211-REG-KELAS-M4-NOABSEN-NAMA.docx

Tabel Buku

id_buku	id_penulis	nama_buku	genre	tahun
102	190	Spring In London	Romance	2007
111	129	Good Fift	Action	2010
142	178	Kata Hati	Romance	2008
153	128	Marmut Merah Jambu	Comedy	2012
188	128	Koala Kumal	Comedy	2015
196	129	Pillow Talk	Romance	2008
322	190	In Blue Moon	Action	2007

Tabel Penulis

id_penulis	nama_penulis
128	Raditya Dika
129	Bernard Batubara
178	Christian simamora
190	Ilana Tan

Tabel detail_buku

id_buku	harga	stok
102	75000	20
142	92000	12
196	84000	27
111	63000	21
322	129000	7
153	89000	14
188	93000	26

1. Buatlah store prosedur untuk menampilkan semua data Buku yang diterbitkan sebelum tahun 2012
2. Buatlah store prosedur untuk mengetahui apakah suatu buku tersedia atau tidak. jika Tersedia, set status "BUKU TERSEDIA". jika tidak Tersedia, set status "BUKU SEDANG KOSONG"
3. Buatlah suatu stored procedure yang berfungsi untuk menambahkan data pada tabel penulis .
4. Buatlah stored procedure untuk Mengetahui Jumlah buku yang Bergenre Romance.