MODUL
STMIK WIDYA PRATAMA

PENGELOLAAN DBMS

TRIANSACTION

DESKRIPSI MATAKULIAH

Matakuliah Pengelolaan DBMS merupakan matakuliah yang akan membahas tentang bagaimana Mengaplikasikan, mengelola, dan memanipulasi data dalam database

CAPAIAN PEMBELAJARAN

Mahasiswa mampu mengimplementasikan, mengelola, dan memanipulasi data dalam database

DOSEN PENGAMPU

Risqiati, M. Kom
Nur Ika Royanti, M.Kom
Mosses Aidjilli, M. Kom
Nurul Amalia, M. Kom
Eny Jumiaty, M. Kom

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

(STMIK) WIDYA PRATAMA

TRANSACTION

CAPAIAN PEMBELAJARAN

Mahasiswa mampu mengimplementasikan transaction pada mysql

MATERI PEMBELAJARAN

Transaction

- Pengertian transacation
- Contoh kasus transaction

MATERI

Transaction

Pengertian dan perbedaan truncation dengan Trigger

Transaction adalah fitur sistem manajemen basis data (DBMS) yang membantu pemulihan data pada kesalahan internal. Transaction pada basis data adalah kumpulan kueri yang harus dieksekusi sehingga jika salah satu eksekusi kueri gagal, maka eksekusi kueri akan kembali ke awal. Transaction adalah mekanisme yang memungkinkan untuk menginterpretasikan beberapa perubahan pada database dengan satu operasi

Transaction dalam pemrogramman database digunakan untuk menangani terjadinya ***error*** dalam memanipulasi database. Sedangkan T***rigger*** merupakan tipe special dari store Procedure yang berjalan secara ***event Driven*** saat user melakukan modifikasi khusus terhadap table tertentu. SQL Server akan menjalankan Trigger Ketika ada perintah **INSERT, UPDATE DELETE** pada suatu table. Trigger biasanya untuk menjaga integritas data, menangani penggantian data serta dapat melakukan aksi lain saat table dimodifikasi dengan pernyataan **INSERT, UPDATE** dan **DELETE**.

Contoh kasus transaction

Nasabah X mempunyai saldo di Bank sebesar Rp 10.000.000 ingin mentransfer uang ke rekening si Y sebesar Rp 2.000.000. Maka dapat diilustrasikan secara sederhana proses yang terjadi adalah sebagai berikut :

1. Saldo si X **dikurangi** Rp 2.000.000 sehingga saldo akhir si X menjadi sebesar Rp.8.000.000
2. Saldo si Y **bertambah** sebesar Rp 2.000.000

Jika di antara langkah (1) dan (2) terjadi hal-hal buruk yang tidak diinginkan, misalnya saja mesin ATM crash, terjadi pemadaman listrik, UPS gagal up, disk pada server penuh, atau ada cracker yang merusak infrastruktur jaringan. Maka dapat dipastikan bahwa saldo si X berkurang, tetapi saldo si Y tidak bertambah. Hal tersebut merupakan sesuatu yang tidak diharapkan untuk terjadi. Dari ilustrasi di atas, maka dapat disimpulkan bahwa solusi yang tepat adalah memperlakukan perintah -perintah sebagai satu kesatuan operasi.

Sederhananya, lakukan semua operasi atau tidak sama sekali. Uniknya, konsep penyelesaian di atas sudah dikemukakan oleh para ahli sejak puluhan tahun silam. Konsep yang disebut transaksi basisdata (database transaction) ini sebenarnya cukup sederhana, antara lain:

- ✓ Tandai bagian awal dan akhir himpunan perintah,

- ✓ Putuskan di bagian akhir untuk mengeksekusi **(COMMIT)** atau membatalkan **(ROLLBACK)** semua perintah.

Properti Transaksi Basis Data

Dalam transaksi basis data, terdapat properti-properti yang menjamin bahwa transaksi dilaksanakan dengan baik. Properti-properti ini dikenal sebagai ACID (*Atomicity, Consistency, Isolation, Durability*).

1. **Atomicity**

   Transaksi dilakukan sekali dan bersifat atomic, artinya merupakan satu kesatuan tunggal yang tidak dapat dipisah, baik pekerjaan yang dilaksanakan secara keseluruhan atau yang tidak satupun, dan hal ini dikenal dengan istilah "All Or Nothing".
2. **Consistency**

   Jika basis data pada awalnya dalam keadaan konsisten, maka pelaksanaan transaksi dengan sendirinya juga harus meninggalkan basis data tetap dalam status konsisten.
3. **Isolation**

   Isolasi memastikan bahwa secara bersamaan (konkuren) eksekusi transaksi terisolasi dari yang lain.
4. **Durability**

   Begitu transaksi telah dilaksanakan (*di-commit*), maka perubahan yang diakibatkan tidak akan hilang atau tahan lama (*durable*), sekalipun terdapat kegagalan sistem.

Penanganan Kesalahan

Fasilitas penanganan kesalahan (*error handling*) biasa diperlukan untuk mengantisipasi terjadinya kesalahan pada suatu proses transaksi, sehingga programmer bisa mengatur skenario jika suatu operasi gagal sebagian atau seluruhnya. Secara default skenario dari transaksi adalah AUTO COMMIT dimana seluruh proses yang berhasil dilaksanakan akan secara otomatis secara fisik disimpan dalam database. Jika diinginkan mulai dari posisi tertentu, maka AUTO COMMIT tidak berfungsi, dapat digunakan perintah **START TRANSACTION**.

Selanjutnya suatu perintah sesudah pernyataan **START TRANSACTION** akan ditunda untuk disimpan sampai bertemu pernyataan **COMMIT** yang akan menyimpan seluruh proses yang tertunda, atau bertemu pernyataan **ROLLBACK** yang akan membatalkan seluruh proses yang tertunda. Akan tetapi perlu diingat bahwa terdapat beberapa perintah yang tidak dapat di ROLLBACK karena mengandung fungsi COMMIT secara implisit. Perintah–peritah tersebut adalah sebagia berikut :

- ➢ ALTER TABLE

- BEGIN
- CREATE TABLE
- CREATE DATABASE
- CREATE INDEX
- DROP DATABASE
- DROP TABLE
- DROP INDEX
- LOAD MASTER DATA
- LOCK TABLES
- SET AUTOCOMMIT = 1
- START TRANSACTION
- TRUNCATE TABLE
- UNLOCK TABLES

Perilaku transaction dapat dikontrol dengan menetapkan variabel sesi yang disebut **AUTOCOMMIT**. Jika **AUTOCOMMIT diatur ke 1** (default), maka setiap pernyataan SQL (dalam suatu transaction atau tidak) dianggap sebagai transaction yang lengkap dan dilakukan secara default ketika selesai.

Ketika **AUTOCOMMIT diatur ke 0**, dengan mengeluarkan perintah SET AUTOCOMMIT = 0, rangkaian pernyataan selanjutnya bertindak seperti transaction dan tidak ada aktivitas yang dilakukan sampai transaction pernyataan COMMIT yang eksplisit dilakukan

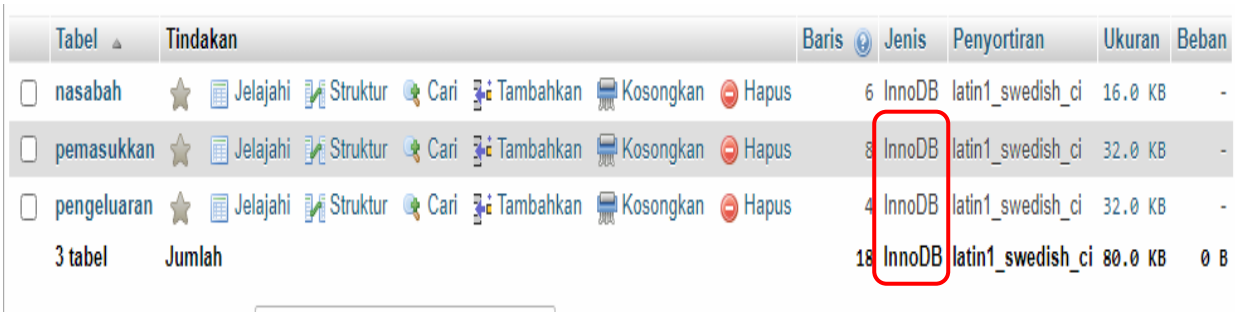
Langkah praktikum :

1. Transaksi di MySql

   MySQL memiliki dukungan untuk transaksi melalui **storage engine InnoDB (full ACID compliance)** dan **BDB (BerkeleyDB)** sejak versi 4.0. Oleh sebab itu, untuk dapat mengimplementasikan transaksi, DBMS MySQL harus mendukung salah satu atau kedua engine transactional. Untuk mengetahui apakah table yang kita gunakan sudah menggunakan engine InnoDB maka kita dapat mengeceknya melalui struktur tabel.

   Contoh:

   Cek struktur dari **Database Tabungan**



	Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
☐	nasabah	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	6	InnoDB	latin1_swedish_ci	16.0 KB	-
☐	pemasukkan	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	8	InnoDB	latin1_swedish_ci	32.0 KB	-
☐	pengeluaran	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	4	InnoDB	latin1_swedish_ci	32.0 KB	-
	3 tabel	Jumlah	18	InnoDB	latin1_swedish_ci	80.0 KB	0 B

Catatan:

- ✓ engine non-transactional (seperti MyISAM) tidak dapat digunakan untuk mengimplementasikan transaksi basis data
- ✓ Tipe atau storage engine-nya, HARUS InnoDB, jika belum innoDB maka harus diubah terlebih dahulu dengan perintah alter.

Misal :

Alter table nasabah engine="InnoDB";

2. Implementasi Transaction

Transaksi di MySQL diinisiasi dengan menggunakan pernyataan START TRANSACTION atau BEGIN dan diakhiri dengan COMMIT untuk menerapkan semua transaksi. Untuk membuat suatu kegiatan transaksi dalam PhpMyAdmin dapat menggunakan Stored Procedure.

a. Commit

1) Buat Procedure dengan nama **transa_1** yang berisikan transaksi penambahan 2 record pada table transaksi

Sintak:

```
DELIMITER //
CREATE PROCEDURE TransKeluar()
BEGIN
        START TRANSACTION;
        INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening, Saldo_Keluar)
        VALUES
        ('6', '2024-08-28 13:30:00', '001-01-0007', '500000'),
        ('7', '2024-08-29  11:22:48', '001-01-0006', '100000');
        SELECT * FROM pengeluaran;
END//
DELIMITER ;
```

Atau:

```
DELIMITER //
CREATE PROCEDURE TransKeluar()
BEGIN
    START TRANSACTION;
    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu,
    No_Rekening, Saldo_Keluar) VALUES ('6', '2024-08-28
    13:30:00', '001-01-0007', '500000'), ('7', '2024-08-29
    11:22:48', '001-01-0006', '100000');
    SELECT * FROM pengeluaran;
END//
DELIMITER ;
```

Jika berhasil maka satu procedure tercipta yaitu TransKeluar tanpa nilai parameter.

2) Panggil Procedure dengan perintah:

Sintak : Call TransKeluar

```
Call TransKeluar
```

Hasilnya:

Muncul 2 baris yang baru ditambahkan

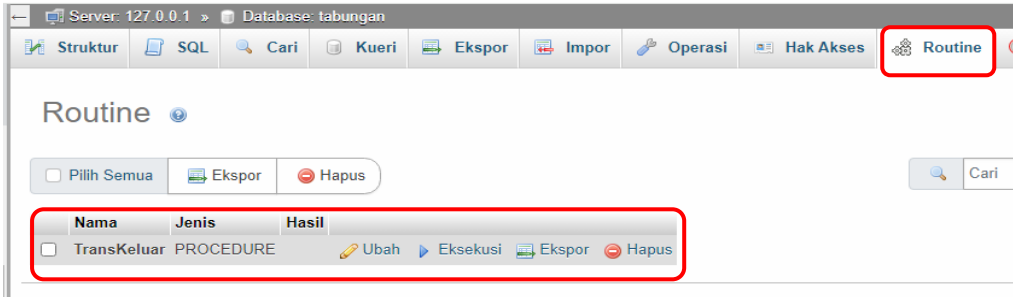
Id_Pengeluaran	Tanggal_Waktu	No_Rekening	Saldo_Keluar
1	2024-08-22 06:41:53	001-01-0001	100000
2	2024-08-22 06:41:53	001-01-0002	300000
3	2024-08-22 06:42:33	001-01-0004	250000
4	2024-08-22 06:42:33	001-01-0005	350000
5	2024-08-29 04:53:00	001-01-0006	1000000
6	2024-08-28 13:30:00	001-01-0007	500000
7	2024-08-29 11:22:48	001-01-0006	100000

3) Periksa/cek tabel Pengeluaran

Id_Pengeluaran	Tanggal_Waktu	No_Rekening	Saldo_Keluar
1	2024-08-22 06:41:53	001-01-0001	100000
2	2024-08-22 06:41:53	001-01-0002	300000
3	2024-08-22 06:42:33	001-01-0004	250000
4	2024-08-22 06:42:33	001-01-0005	350000
5	2024-08-29 04:53:00	001-01-0006	1000000

Gambar di atas, memperlihatkan bahwa tabel pengeluaran tidak ada penambahan data. Hal ini dikarenakan tidak diterapkannya transaksi dengan memanggil perintah COMMIT. Sehingga mengakibatkan pengeluaran di-rollback secara implisit.

Selanjutnya buka kembali **Procedure TransKeluar** dan tambahkan statement **COMMIT** sehingga perintahnya menjadi sbb:



Klik Ubah, untuk mengubah query.

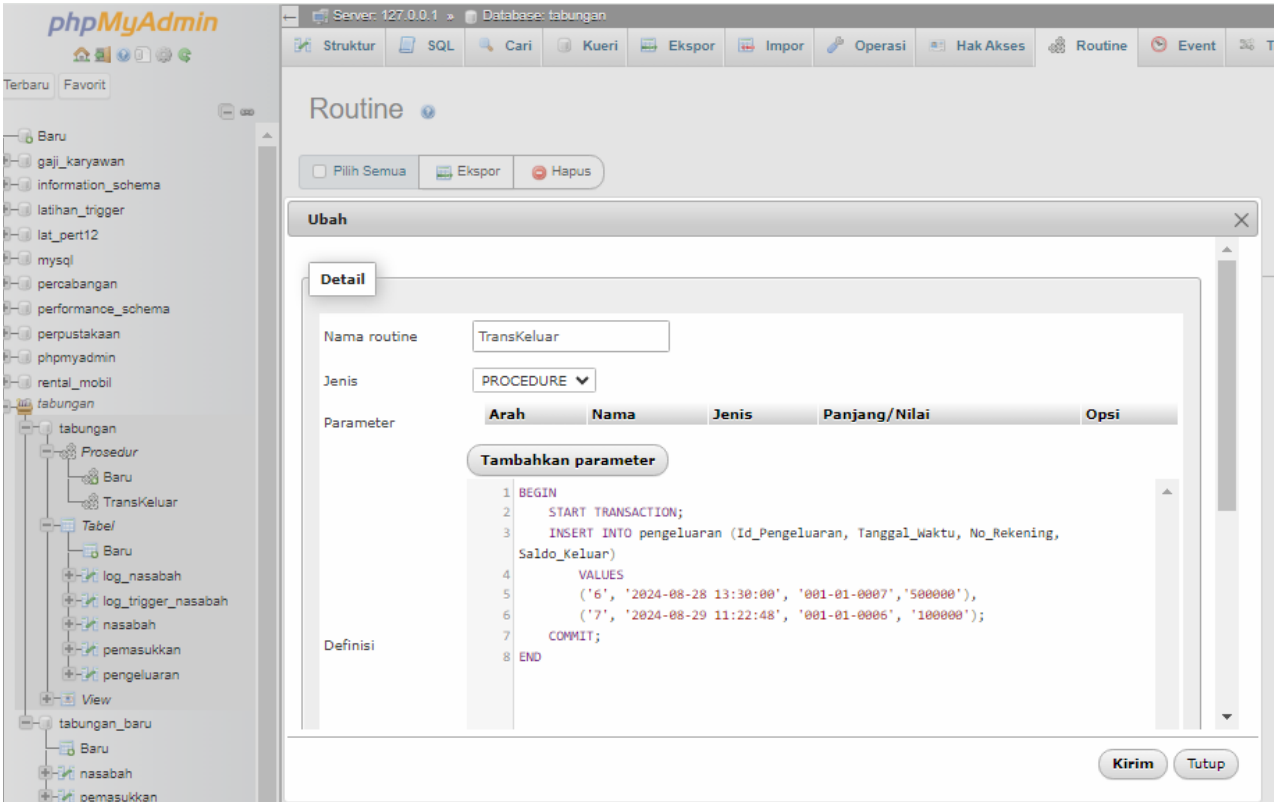
Atau

Klik "Prosedur" kemudian klik "TransKeluar".

Hapus perintah "SELECT * FROM Pengeluaran" dan diganti dengan perintah " COMMIT; " seperti gambar dibawah ini

jika sudah selesai, klik Kirim

```
BEGIN
        START TRANSACTION;
        INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening,
        Saldo_Keluar)
        VALUES
        ('6', '2024-08-28 13:30:00', '001-01-0007','500000'),
         ('7', '2024-08-29 11:22:48', '001-01-0006', '100000');
        COMMIT;
END
```

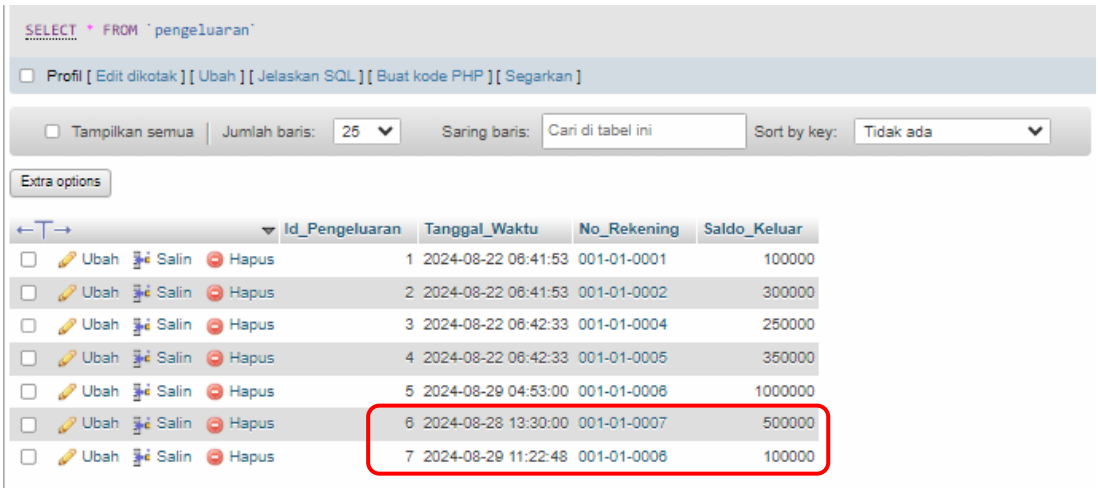


4) Panggil kembali procedure dengan :

   Call TransKeluar

5) Periksa tabel Pengeluaran dengan:

   Select * from Pengeluaran;



	Id_Pengeluaran	Tanggal_Waktu	No_Rekening	Saldo_Keluar
Ubah Salin Hapus	1	2024-08-22 06:41:53	001-01-0001	100000
Ubah Salin Hapus	2	2024-08-22 06:41:53	001-01-0002	300000
Ubah Salin Hapus	3	2024-08-22 06:42:33	001-01-0004	250000
Ubah Salin Hapus	4	2024-08-22 06:42:33	001-01-0005	350000
Ubah Salin Hapus	5	2024-08-29 04:53:00	001-01-0006	1000000
Ubah Salin Hapus	6	2024-08-28 13:30:00	001-01-0007	500000
Ubah Salin Hapus	7	2024-08-29 11:22:48	001-01-0006	100000

Ketika telah ditambahkan perintah COMMIT, maka data telah tersimpan secara permanen (tidak di-rollback).

b. Autocommit Mode

Selain menggunakan pernyataan START TRANSACTION, juga dapat menggunakan pernyataan SET untuk mengatur nilai variabel autocommit. Nilai default autocommit adalah 1 , yang

menyatakan bahwa transaksi basis data telah aktif. Dengan kata lain, setiap perintah akan langsung diterapkan secara permanen.

1) Modifikasi isi perintah procedure TransKeluar dengan mengganti START TRANSACTION dengan SET seperti di bawah ini.

```
BEGIN
        SET AUTOCOMMIT = 0;
        INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening, Saldo_Keluar)
        VALUES
        ('8', '2024-08-29 11:22:48', '001-01-0006', '200000');
        SELECT * FROM pengeluaran;
END
```

2) Panggil procedure

   Call TransKeluar

3) Periksa/cek tabel Pengeluaran

Id_Pengeluaran	Tanggal_Waktu	No_Rekening	Saldo_Keluar
1	2024-08-22 06:41:53	001-01-0001	100000
2	2024-08-22 06:41:53	001-01-0002	300000
3	2024-08-22 06:42:33	001-01-0004	250000
4	2024-08-22 06:42:33	001-01-0005	350000
5	2024-08-29 04:53:00	001-01-0006	1000000
6	2024-08-28 13:30:00	001-01-0007	500000
7	2024-08-29 11:22:48	001-01-0006	100000
8	2024-08-29 11:22:48	001-01-0006	200000

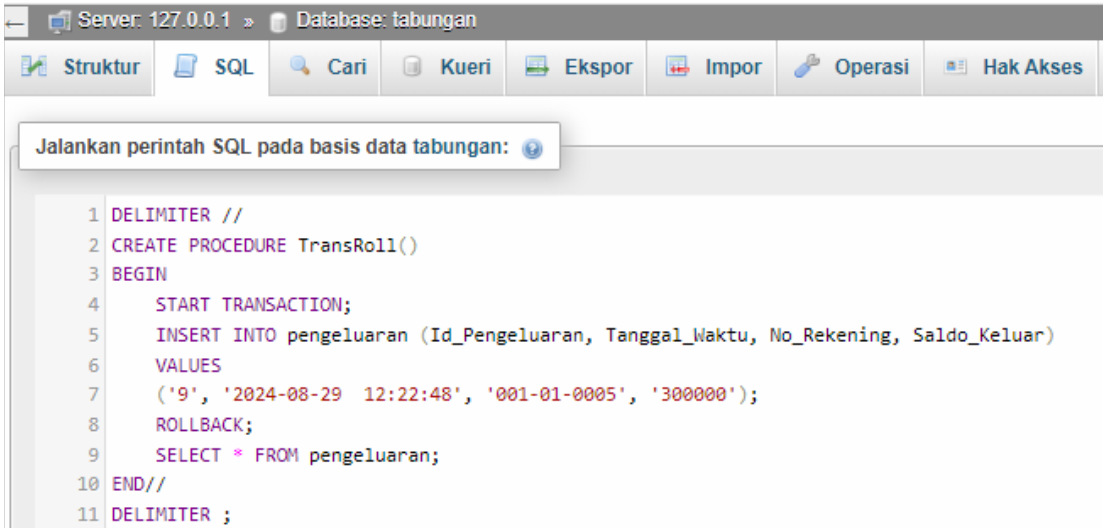
Data yang baru ditambahkan akan disimpan secara permanen, karena telah mengaktifkan transaksi basisdata.

c. ROLLBACK TRANSACTION

Selain perintah COMMIT untuk akhir transaksi dapat juga digunakan pernyataan ROLLBACK, dimana Rollback berfungsi untuk menggugurkan rangkaian perintah. ROLLBACK akan dilakukan ketika ada satu atau lebih perintah yang gagal dilaksanakan. Di samping itu, ROLLBACK juga dapat dilakukan secara eksplisit dengan memanggil pernyataan ROLLBACK.

1) Membuat Procedure dengan nama TransRoll, seperti berikut:

```
DELIMITER //
CREATE PROCEDURE TransRoll()
BEGIN
    START TRANSACTION;
    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening, Saldo_Keluar)
    VALUES
    ('9', '2024-08-29  12:22:48', '001-01-0005', '300000');
     ROLLBACK;
    SELECT * FROM pengeluaran;
END//
DELIMITER ;
```



2) Panggil procedure dengan CALL :

```
CALL TransRoll
```

3) Lihat isi tabel Pengeluaran

Id_Pengeluaran	Tanggal_Waktu	No_Rekening	Saldo_Keluar
1	2024-08-22 06:41:53	001-01-0001	100000
2	2024-08-22 06:41:53	001-01-0002	300000
3	2024-08-22 06:42:33	001-01-0004	250000
4	2024-08-22 06:42:33	001-01-0005	350000
5	2024-08-29 04:53:00	001-01-0006	1000000
6	2024-08-28 13:30:00	001-01-0007	500000
7	2024-08-29 11:22:48	001-01-0006	100000

d. Checkpointing

Normalnya, ROLLBACK akan menggugurkan keseluruhan perintah dalam blok transaksi. Akan tetapi kondisi ini terkadang tidak dikehendaki, misal terdapat tiga perintah, namun kita hanya ingin menggugurkan perintah setelah perintah kedua (perintah pertama masih ada). Dalam kasus ini, kita bisa memanfaatkan ***fitur checkpointing***.

1) Buat procedure dengan CekPoint, dengan isi perintah sbb:

```
DELIMITER //
CREATE PROCEDURE CekPoint()
BEGIN
    START TRANSACTION;
    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening,
    Saldo_Keluar)
    VALUES ('10','2024-08-22 06:41:53','001-01-0001','100000');
    SAVEPOINT Point1;

    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening,
    Saldo_Keluar)
    VALUES ('11','2024-08-23 06:41:53','001-01-0001','200000');
    ROLLBACK TO SAVEPOINT Point1;

    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening,
    Saldo_Keluar)
    VALUES ('12','2024-08-24 06:41:53','001-01-0001','300000');
    COMMIT;

    SELECT * FROM pengeluaran;
```

END//

DELIMITER ;

```
DELIMITER //
CREATE PROCEDURE CekPoint()
BEGIN
    START TRANSACTION;
    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening, Saldo_Keluar)
    VALUES ('10','2024-08-22 06:41:53','001-01-0001','100000');
    SAVEPOINT Point1;

    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening, Saldo_Keluar)
    VALUES ('11','2024-08-23 06:41:53','001-01-0001','200000');
    ROLLBACK TO SAVEPOINT Point1;

    INSERT INTO pengeluaran (Id_Pengeluaran, Tanggal_Waktu, No_Rekening, Saldo_Keluar)
    VALUES ('12','2024-08-24 06:41:53','001-01-0001','300000');
    COMMIT;

    SELECT * FROM pengeluaran;

END//
DELIMITER ;
```

2) Pangggil procedure dengan perintah

```
CALL CekPoint;
```

3) Panggil/cek tabel Pengeluaran dan perhatikan hasilnya:

Id_Pengeluaran	Tanggal_Waktu	No_Rekening	Saldo_Keluar
1	2024-08-22 06:41:53	001-01-0001	100000
2	2024-08-22 06:41:53	001-01-0002	300000
3	2024-08-22 06:42:33	001-01-0004	250000
4	2024-08-22 06:42:33	001-01-0005	350000
5	2024-08-29 04:53:00	001-01-0006	1000000
6	2024-08-28 13:30:00	001-01-0007	500000
7	2024-08-29 11:22:48	001-01-0006	100000
10	2024-08-22 06:41:53	001-01-0001	100000
12	2024-08-24 06:41:53	001-01-0001	300000

Catatan :

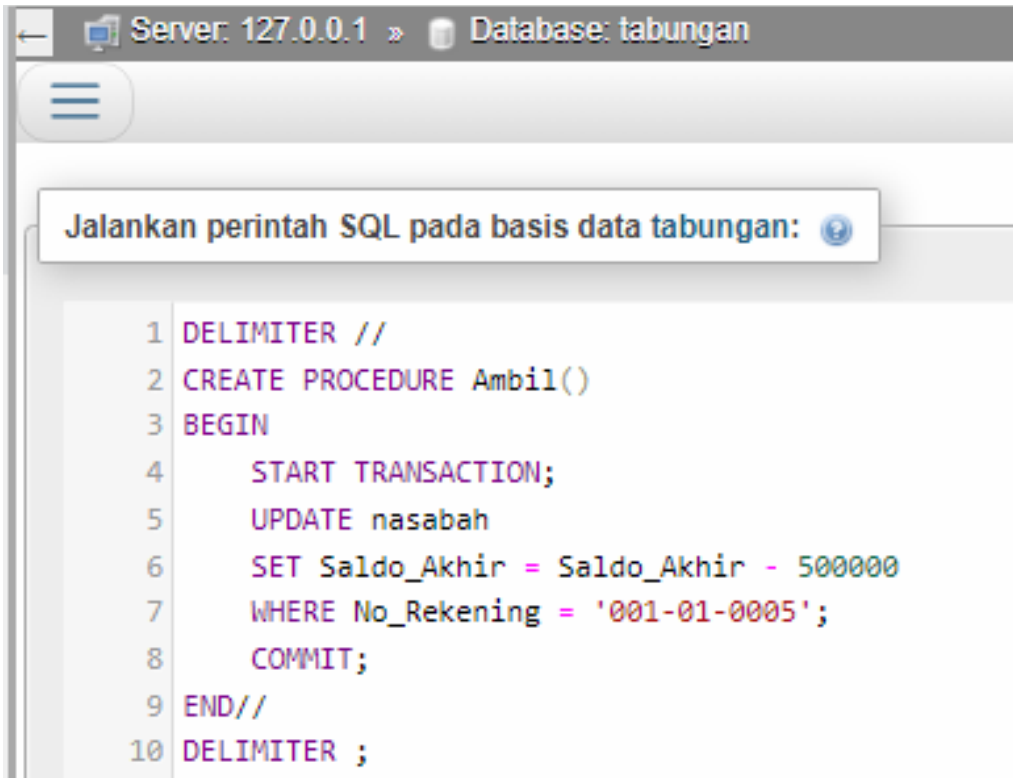
Perhatikan data yang ditambahkan sebelum ditandai dengan perintah **SAVEPOINT** akan disimpan secara permanen. Sedangkan jika ada penambahan data diantara perintah **SAVEPOINT** dan perintah **ROLLBACK**, maka data tidak akan tersimpan permanen. Data akan tersimpan secara permanen apabila sudah ada pernyataan **COMMIT**

e. Trasaksi pengambilan uang dari tabel nasabah

No_Rekening	Nama_Nasabah	Jenis_Kelamin	Alamat	Kota_Kab	Kelurahan	Tanggal_Lahir	Saldo_Akhir
001-01-0001	Usman Majid	Laki - Laki	Jl. Melati No. 25	Kota Pekalongan	Kauman	2010-08-04	100000
001-01-0002	Mawar Melati	Perempuan	Jl. Toba No. 20	Kota Pekalongan	Kauman	2004-06-10	2000000
001-01-0003	Bobi Nasution	Laki - Laki	Jl. Kenanga No. 1	Kota Pekalongan	Medono	2008-04-02	2000000
001-01-0004	Nurul Sintya	Perempuan	Jl. Kenanga No. 5	Kota Pekalongan	Medono	2006-08-09	1500000
001-01-0005	Kenan Sudrajat	Laki - Laki	Jl. Melati No. 1	Kabupaten Pekalongan	Kutosari	2007-01-03	5000000
001-01-0006	Siti Nurbaya	Perempuan	Jl. Anggrek No. 12	Kabupaten Pekalongan	Harjosari	1999-10-08	2400000
001-01-0007	Joko Susilo	Laki - Laki	Jl. Kenanga No. 10	Kabupaten Pekalongan	Kutosari	2008-08-12	0

1) Buat procedure dengan nama **Ambil**

```
DELIMITER //
CREATE PROCEDURE Ambil()
BEGIN
        START TRANSACTION;
        UPDATE nasabah
        SET Saldo_Akhir = Saldo_Akhir - 500000
        WHERE No_Rekening = '001-01-0005';
        COMMIT;
END//
DELIMITER ;
```



2) Panggil prosedur perintah dengan Call ambil

```
1 CALL Ambil
```

3) Periksa/cek tabel Nasabah

No_Rekening	Nama_Nasabah	Jenis_Kelamin	Alamat	Kota_Kab	Kelurahan	Tanggal_Lahir	Saldo_Akhir
001-01-0001	Usman Majid	Laki - Laki	Jl. Melati No. 25	Kota Pekalongan	Kauman	2010-08-04	100000
001-01-0002	Mawar Melati	Perempuan	Jl. Toba No. 20	Kota Pekalongan	Kauman	2004-06-10	2000000
001-01-0003	Bobi Nasution	Laki - Laki	Jl. Kenanga No. 1	Kota Pekalongan	Medono	2008-04-02	2000000
001-01-0004	Nurul Sintya	Perempuan	Jl. Kenanga No. 5	Kota Pekalongan	Medono	2006-08-09	1500000
001-01-0005	Kenan Sudrajat	Laki - Laki	Jl. Melati No. 1	Kabupaten Pekalongan	Kutosari	2007-01-03	4500000
001-01-0006	Siti Nurbaya	Perempuan	Jl. Anggrek No. 12	Kabupaten Pekalongan	Harjosari	1999-10-08	2400000
001-01-0007	Joko Susilo	Laki - Laki	Jl. Kenanga No. 10	Kabupaten Pekalongan	Kutosari	2008-08-12	0

f. Trasaksi pengambilan uang dari tabel Nasabah akan terjadi error sehingga nominal tidak berkurang

1) Buat procedure dengan nama Batal

```
DELIMITER //
CREATE PROCEDURE Batal()
BEGIN
        START TRANSACTION;
        UPDATE nasabah
        SET Saldo_Akhir = Saldo_Akhir - 500000
        WHERE No_Rekening = '001-01-0005';
        ROLLBACK;
END//
DELIMITER ;
```

Server: 127.0.0.1 » Database: tabungan

Jalankan perintah SQL pada basis data tabungan:

```
DELIMITER //
CREATE PROCEDURE Batal()
BEGIN
    START TRANSACTION;
    UPDATE nasabah
    SET Saldo_Akhir = Saldo_Akhir - 500000
    WHERE No_Rekening = '001-01-0005';
    ROLLBACK;
END//
DELIMITER ;
```

2) Cek periksa table Nasabah

No_Rekening	Nama_Nasabah	Jenis_Kelamin	Alamat	Kota_Kab	Kelurahan	Tanggal_Lahir	Saldo_Akhir
001-01-0001	Usman Majid	Laki - Laki	Jl. Melati No. 25	Kota Pekalongan	Kauman	2010-08-04	100000
001-01-0002	Mawar Melati	Perempuan	Jl. Toba No. 20	Kota Pekalongan	Kauman	2004-06-10	2000000
001-01-0003	Bobi Nasution	Laki - Laki	Jl. Kenanga No. 1	Kota Pekalongan	Medono	2008-04-02	2000000
001-01-0004	Nurul Sintya	Perempuan	Jl. Kenanga No. 5	Kota Pekalongan	Medono	2006-08-09	1500000
001-01-0005	Kenan Sudrajat	Laki - Laki	Jl. Melati No. 1	Kabupaten Pekalongan	Kutosari	2007-01-03	4500000
001-01-0006	Siti Nurbaya	Perempuan	Jl. Anggrek No. 12	Kabupaten Pekalongan	Harjosari	1999-10-08	2400000
001-01-0007	Joko Susilo	Laki - Laki	Jl. Kenanga No. 10	Kabupaten Pekalongan	Kutosari	2008-08-12	0

3) Panggil procedure dengan

```
CALL Batal
```

4) Periksa Kembali table Nasabah, bagaimana hasilnya dan apa yang terjadi?

DAFTAR PUSTAKA

Kristiaono, P. (2015). *Pemrograman Stored Procedure pada My SQL.* Yogyakarta: Andi Offset

Utami, Ema. (2008). *RDBM menggunakan MS SQL Server 2000.* Yogyakarta: Graha Ilmu

https://elektro.um.ac.id