



Manajemen Transaksi

Praktikum Sistem Basis Data

Gentisya Tri Mardiani, S.Kom., M.Kom

Konsep Transaksi



- Transaksi adalah sebuah aksi atau serangkaian aksi, yang dilakukan oleh user atau aplikasi yang mengakses atau mengubah isi dari database.
- Dua hasil transaksi adalah ***commit*** atau ***rollback***.
- Jika transaksi berjalan sukses maka dikatakan *commit*, sebaliknya jika transaksi tidak berjalan sukses maka transaksi dibatalkan dan kembali ke keadaan semula dikatakan *rollback*.

Transaksi dasar



Database di awal transaksi

START TRANSACTION;

```
INSERT INTO buku VALUES ('M-005', 'Manajemen Transaksi', 1, 'CP');  
INSERT INTO buku VALUES ('B-001', 'Sistem Basis Data', 1, 'BP');
```

Sukses dieksekusi

Gagal dieksekusi

ROLLBACK ;

COMMIT;



Database di akhir transaksi

Contoh Transaksi Dasar



```
START TRANSACTION;  
INSERT INTO buku  
VALUES ('M-006', 'Manajemen Transaksi', 1, 'BP');  
  
INSERT INTO buku  
VALUES ('B-001', 'Sistem Basis Data', 1, 'AO');  
COMMIT;  
  
SELECT * FROM buku;
```

Contoh Transaksi Dasar



START TRANSACTION;

INSERT INTO buku

VALUES ('M-007', 'Manajemen Keuangan', 1, 'GM');

INSERT INTO buku

VALUES ('P-003', 'Pemrograman Dasar', 1, 'AO');

ROLLBACK;

Contoh Transaksi Dasar



```
SELECT * FROM buku;
```

Data dengan Kode_Buku M-007 dan P-003 tidak masuk, seolah tidak pernah diinputkan.

AUTOCOMMIT STATEMENTS



- CREATE DATABASE
- DROP DATABASE
- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- CREATE INDEX
- DROP INDEX
- RENAME TABLES
- LOCK TABLES
- UNLOCK TABLES

SAVEPOINT



Database di awal transaksi

START TRANSACTION;

INSERT INTO buku VALUES ('M-006', 'Manajemen Transaksi', 1, 'CP');
INSERT INTO buku VALUES ('B-001', 'Sistem Basis Data', 1, 'BP');

Sukses

Gagal

ROLLBACK ;

SAVEPOINT sp1;

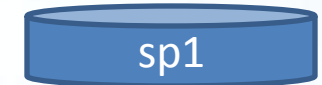
INSERT INTO buku VALUES ('M-007', 'Manajemen Keuangan', 1, 'GM');
INSERT INTO buku VALUES ('P-003', 'Pemrograman Dasar', 1, 'AO');

Sukses

Gagal

**ROLLBACK TO
SAVEPOINT SP1;**

COMMIT;



Database di akhir transaksi

SAVEPOINT



START TRANSACTION;

INSERT INTO buku VALUES ('M-007', 'Manajemen Keuangan', 1, 'GM');

INSERT INTO buku VALUES ('P-003', 'Pemrograman Dasar', 1, 'AO');

SAVEPOINT sp1;

INSERT INTO buku VALUES ('M-008', 'Manajemen 1', 1, 'CP');

INSERT INTO buku VALUES ('M-009', 'Manajemen 2', 1, 'CP');

ROLLBACK TO SAVEPOINT sp1;

INSERT INTO buku VALUES ('P-004', 'Pemrograman 1', 2, 'AO');

INSERT INTO buku VALUES ('P-005', 'Pemrograman 2', 2, 'AO');

COMMIT;

SAVEPOINT



SELECT * FROM buku;

```
mysql> select * from buku;
```

Kode_Buku	Judul	Edisi	Kode_Penerbit
A-001	Algoritma dan Pemrograman	2	IF
M-001	Manajemen Database	1	GM
M-002	Manajemen Pemasaran	1	CP
M-003	Manajemen Database	2	GM
M-004	Manajemen Keuangan	1	GM
M-005	Manajemen Waktu	1	CP
M-006	Manajemen Transaksi	1	BP
M-007	Manajemen Keuangan	1	GM
P-001	Pengenalan DBMS	1	AO
P-002	Pemrograman Dasar	2	IF
P-003	Pemrograman Dasar	1	AO
P-004	Pemrograman 1	2	AO
P-005	Pemrograman 2	2	AO
S-001	Sistem Basis Data	1	AO

14 rows in set (0.00 sec)

Data dengan Kode_Buku M-008 dan M-009 tidak disimpan karena telah di-*rollback*.

LOCKING TABLES



- Proses mengunci tabel yang sedang terlibat dalam suatu operasi.
- Penguncian tabel diharapkan dapat meningkatkan kecepatan dalam mengupdate isi tabel.
- Apabila suatu transaksi mengakses suatu data maka suatu lock (kunci) dapat mencegah pengaksesan oleh transaksi lain.

LOCKING TABLES



LOCK TABLES *tbl_name* [[AS] *alias*] *lock_type*
[, *tbl_name* [[AS] *alias*] *lock_type*] ...

lock_type:

READ [LOCAL] |
[LOW_PRIORITY] WRITE

UNLOCK TABLES

READ [LOCAL] lock



READ [LOCAL] lock:

- Transaksi dapat membaca isi dari tabel, tetapi tidak dapat menuliskan/mengupdate isi ke dalam tabel.
- Pada saat yang sama dapat menggunakan READ lock secara bersamaan.
- Transaksi lain dapat membaca isi tabel tanpa harus menggunakan perintah READ lock.
- Pada InnoDB tables, READ LOCAL sama dengan READ.

[LOW_PRIORITY] WRITE lock



[LOW_PRIORITY] WRITE lock:

- Transaksi dapat membaca dan mengupdate isi tabel.
- Hanya transaksi yang menggunakan WRITE lock yang dapat mengakses tabel. Transaksi lain tidak dapat mengakses sampai kunci dilepaskan (unlock).

Contoh



```
LOCK TABLES buku READ;  
SELECT COUNT(*) FROM buku;
```

```
mysql> select count(*) from buku;  
+-----+  
| count(*) |  
+-----+  
|      14 |  
+-----+  
1 row in set (0.05 sec)
```

```
SELECT COUNT(*) FROM penerbit;
```

```
mysql> select count(*) from penerbit;  
ERROR 1100 (HY000): Table 'penerbit' was not locked with LOCK TABLES
```

```
UNLOCK TABLES;
```

Contoh



LOCK TABLES buku READ;

SELECT * FROM penerbit;

```
mysql> select * from penerbit;  
ERROR 1100 (HY000): Table 'penerbit' was not locked with LOCK TABLES  
mysql>
```

UNLOCK TABLES;

Contoh



LOCK TABLES buku WRITE, penerbit AS pt READ;

SELECT * FROM buku;

SELECT * FROM penerbit;

SELECT * FROM penerbit AS pt;

UNLOCK TABLES;

Table Locking and Transactions



FLUSH TABLES WITH READ LOCK;

START TRANSACTION;

... do something with tables ...

UNLOCK TABLES;

Contoh



FLUSH TABLES WITH READ LOCK;

START TRANSACTION;

SELECT * FROM buku;

SELECT * FROM penerbit;

UNLOCK TABLES;



SET autocommit=0; (*not* START TRANSACTION)

LOCK TABLES t1 WRITE, t2 READ, ...;

... do something with tables t1 and t2 here ...

COMMIT;

UNLOCK TABLES;



```
SET autocommit=0;  
LOCK TABLES buku WRITE, penerbit READ;  
INSERT INTO buku VALUES ('M-010', 'Manajemen 1', 1, 'CP');  
SELECT * FROM buku;  
SELECT Nama_Penerbit, Lokasi FROM penerbit  
WHERE Lokasi='Yogya';  
COMMIT;  
UNLOCK TABLES;
```

Ketentuan LOCK TABLES



- Tentukan tabel yang akan dikunci (*lock*),
- Jika tabel akan dikunci dengan *read lock* dan *write lock*, maka tempatkan perintah *write lock* sebelum perintah *read lock*.
- Lakukan *locking* suatu tabel sampai tujuan *locking* terpenuhi.
- Perintah *commit* dan *rollback* tidak berarti proses *locking* selesai, untuk melepaskan proses *locking*, maka gunakan perintah UNLOCK TABLES.