Manajemen Transaksi

A. Konsep Transaksi

Transaksi adalah sebuah unit dari eksekusi program yang mampu mengakses dan mengupdate berbagai data yang memiliki kaitan logika transaksi itu sendiri dimana dalam menjaga konsitensi data secara terintegrasi dipertahankan. Sebuah transaksi harus menghasilkan konsistensi database jika sebuah transaksi tersebut sudah dijalankan, atau dengan kata lain setelah transaksi selesai maka database harus kembali konsisten.

Konsistensi suatu database berkaitan erat dengan integritas data, sehingga untuk menjamin integritas tersebut suatu database dalam menjalakan sebuah transaksi harus memiliki sifat –ACID :

1. ATOMIK : Dimana semua operasi dalam transaksi harus bekerja secara utuh/total atau tidak bekerja sama sekali (artinya pekerjaan transaksi tidak boleh dikerjakan sebagian).
2. CONSISTEN : Dampak eksekusi dari sebuah trnasaksi harus menjamin keadaan data yg konsisten.
3. ISOLASI : Pada sejumlah Transaksi yang terjadi secara bersamaan , setiap transaksi tidak boleh terpengaruh transaksi lainnya yang juga sedang berjalan walaupun berhubungan atau menujun pada database yang sama. Hasil transaksi sementara harur terjaga dan terlindungi dari eksekusi-eksekusi yang lain.
4. DURABILITY : Setelah terjadinya transaksi maka akan diikuti update atau perubahan data pada database, maka perubahan tersebut harus tetap bertahan dan dianggap paling valid dan konsisten.

PRAKTEK :

Dalam Basis Data Transaksi merupakan satuan unit kerja yang berisi perintah-perintah, yang dimulai dengan klausa BEGIN TRANSACTION dan diakhiri dengan COMMIT atau ROLLBACK.

1. Membuat Tabel account dengan type Innodb

Type table InnoDB adalah salah satu tipe table dalam MySql yang mengadopsi Enggine Oracle, (beberapa type table dalam my saql adalaha Heap, ISAM, MERGE, MyISAM, BDB dll) namun beberapa kemampuannya terbatas. Type Tabel

InnoDB memungkinkan kita memakai transaksi yang mendukung ACID dan memiliki fasilitas row level locking. Type Tabel InnoDB ini telah mendukung kemampuan-kemampuan dalam pengolahan data yg besar dengan sifat “Transaction-Safe” diantara memiliki kemampuan Commit, Rollback dan Crash Recovery. Kemampuan – kemapuan ini dirancang untuk kinerja yang maksimum ketika memproses data data dengan volume besar. Type table ini memiliki kemampuan automatic crash recovery, namun tidak dapat disetup “dipoint” mana recovery ini dilakukan layaknya chek point recovery pada Oracle.

Untuk mengechek apakah InnoDb ada pad MySql:

Mysql> show variables li ‘%have%’
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.22-community-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show variables like '%have%';

Variable_name	Value
have_archive	YES
have_bdb	NO
have_blackhole_engine	NO
have_compress	YES
have_crypt	NO
have_csv	NO
have_example_engine	NO
have_federated_engine	NO
have_geometry	YES
have_innodb	YES
have_isam	NO
have_ndbcluster	NO
have_openssl	DISABLED
have_query_cache	YES
have_raid	NO

| have_rtree_keys | YES |
| have_symlink | YES |
+----------------------+-----------------+
17 rows in set (0.12 sec)

MySql yang didukung Type InnoDb memiliki variabel lingkungan yang disebut autocmmit. Secara default, autocommit diatur ke 1. Jika autocommit di setup ke, kita tidak dapat menjalankan transaksi dengan type table apapun. Dalam Hal ini, Mysql menganggap setiap pernyataan SQL sebagai transaksi tersendiri. Agar dapat memakai “sekumpulan” perintah/pernyataan transaksi dan dianggap masing-masing tidak berdiri sendiri, maka kita harus meng-off kan autocommit. Nilai autocommit diatur atau diset ke 0;

MySql> SET autocommit=0;

Contoh Praktek Seder hana tabel dibawah ini :

Tabel berisi dua kolom yaitu number dengan tipe data integer,merupakan primary key, autoincrement dan tidak diijinkan nilai Null. Kolom balance bertipe float, perintahnya sebagai berikut :

```
create table account ( number int not null auto_increment primary key, balance float ) type = InnoDB;
```

2. Insert Data Pada tabel account

Perintah di bawah dipergunakan untuk melakukan insert data pada tabel account, karena kolom number autoincrement maka data yang dimasukkan hanya pada kolom balance:

```
insert into account (balance) values (0.0);
insert into account (balance) values (1000.0);
insert into account (balance) values (2000.0);
```

Hasil dari insert data tersebut adalah :

```
mysql> select * from account;
+--------+---------+
| number | balance |
+--------+---------+
|      1 |       0 |
|      2 |    1000 |
|      3 |    2000 |
+--------+---------+
3 rows in set (0.05 sec)
```

3. Operasi Update data pada tabel account

- Seleksi balance pada tabel account dengan number 2

```
mysql> select balance from account where number=2;
+---------+
| balance |
+---------+
|    1000 |
+---------+
```

- Update balance menjadi 1100

```
mysql> update account set balance = 1100 where number = 2;
```

- Update balance dengan menambahkan 500

```
mysql> update account set balance = balance + 500 where
number = 2;
```

Setiap Query tunggal seperti di atas bersifat atomik, sebelum satu perintah selesai dieksekusi tidak ada query lain yang dapat melakukan interupsi terhadap perintah tersebut.

- Update data dengan operasi transfer data 1000 dari account number 2 ke account number 1.

```
update account set balance = balance - 1000 where number = 2;
update account set balance = balance + 1000 where number = 1;
```

Apabila terjadi kasus gangguan setelah proses update data pada account number 1, maka update data account number 1 gagal. Sehingga data pada account 2 tidak valid. Karena perintah tersebut terdiri dari dua perintah sehingga tidak atomik.

ALTERNATIFNYA :

Alternatif yang lain adalah dengan membuat dua perintah tersebut menjadi perintah tunggal (perintah tunggal bersifat atomik).

```
mysql>update account as source, account as dest
set source.balance = source.balance - 1000,
dest.balance = dest.balance + 1000
where source.number = 2 and dest.number = 1;
```

Pada perintah tersebut dipergunakan dua buah alias pada tabel account yaitu source dan dest. Dengan menggunakan dua buah alias, dua perintah dapat dijadikan menjadi satu buah perintah sehingga apabila satu perintah gagal maka semua gagal .

4. Update menggunakan manajemen transaksi

Namun Dalam banyak kasus metode alternative diatasdi atas tidak bisa diterapkan untuk kasus yang lain, artinya peluang kegagalan masih sangat terbuka lebar.

Solusinya adalah dengan menggunakan Manajemen Ttransaksi:

```
begin transaction;
update account set balance = balance - 1000 where number = 2;
update account set balance = balance + 1000 where number = 1;
commit;
```

Atau dengan mengganti perintah BEGIN dengan START.

```
Start transaction;
update account set balance = balance - 1000 where number = 2;
update account set balance = balance + 1000 where number = 1;
commit;
```

Operasi Update data dapat dibatalkan dengan memberikan perintah ROLLBACK pada transaksi tersebut :

```
start transaction;
update account set balance = balance - 1000 where number = 2;
update account set balance = balance + 1000 where number = 1;
select balance from account where number = 2;
rollback;
```

Pada perintah tersebut karena account number nilainya menjadi -1000 setelah operasi update dilakukan maka operasi dibatalkan dengan perintah ROLLBACK.

5. Autocommit mode

Secara default mysql mendukung autocommit mode. Setiap perintah manipulasi(update,insert dll) akan dianggap sebagai perintah yang diakhiri dengan commit. Akan tetapi autocommit mode dapat dihilangkan dengan oleh user. Untuk menghilangkannya berikan perintah:

```
mysql> set autocommit=0;
```

Jika perintah ini telah diberikan maka perintah-perintah tunggal harus diakhiri dengan commit secara manual jika akan disimpan. Apabila akan dibatalkan bisa diberikan perintah Rollback secara manual pula.
Setelah autocommit di set menjadi 0 berikan perintah berikut :

```
update account set balance = balance - 1000 where number = 2;
update account set balance = balance + 1000 where number = 1;
```

Proses di atas tidak akan disimpan apabila belum ada klausa COMMIT yang mengakhiri perintah tersebut. Apabila akan disimpan maka perintah tersebut adalah :

```
begin transaction;
update account set balance = balance - 1000 where number = 2;
commit;
begin transaction
update account set balance = balance + 1000 where number = 1;
commit;
```

Untuk mengembalikan autocommit ke default mysql perintah yang diberikan adalah :

```
mysql> set autocommit=1;
```

6. Menggunakan Lock

Untuk mencegah interferensi antar pengguna/program aplikasi yang melakukan manipulasi pada database digunakan metode isolasi menggunakan klausa Lock.

Contoh :

```
lock tables account write;
select balance from account where number = 2;
update account set balance = 1500 where number = 2;
unlock tables;
```

Lakukan operasi manipulasi melalui terminal lain atau komputer lain.
Sebelum ada perintah unlock diberikan user/program lain tidak dapat melakukan pengubahan pada tabel account.

7. Implementasikan Manajemen transaksi pada aplikasi tabel penjualan dan tabel Stok, apabila ada barang yang dijual maka akan berakibat pada pengurangan jumlah barang pada tabel stok.

Tabel Stok

Kode_Barang	Nama_Barang	Jumlah

Tabel Penjualan

Kode_Barang	Nama_Barang	Jumlah

a. Buat tabel di atas dengan type innodb
b. Insert data pada masing-masing tabel
c. Update data pada masing-masing tabel dengan menggunakan manajemen transaksi.
d. Buat mekanisme Lock pada tabel
e. Seleksi stok barang yang jumlahnya noll .