



DATABASE TRANSACTION



UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@unikom.ac.id)

Definisi Transaction

- Sebuah unit logik dari suatu pekerjaan yang terdiri dari 1 atau banyak statement SQL
- Merupakan sebuah unit yang atomic, sehingga hanya ada 2 kemungkinan yang terjadi yaitu :
  - Data ter-commit (semua perubahan data tersimpan secara permanen)
  - Data dibatalkan (semua perubahan data dibatalkan)

Contoh Kasus

- Transaksi transfer antar rekening. Ada 3 SQL yang harus dieksekusi yaitu :
  - Menyimpan data transaksi ke tabel transaksi (INSERT)
  - Melakukan pengurangan saldo ke rekening pengirim (UPDATE)
  - Melakukan penambahan saldo ke rekening penerima (UPDATE)
- Semua SQL harus tereksekusi semua atau dibatalkan semua (atomic).

Statement dalam Transaksi

- COMMIT : Melakukan penyimpanan perubahan data.
- ROLLBACK : Melakukan pembatalan perubahan data.
- SAVEPOINT : Memberi tanda suatu posisi perubahan data.
- ROLLBACK TO savepoint : Melakukan pembatalan perubahan data dengan kembali ke kondisi savepoint tertentu.

Awal Transaksi

- Suatu transaksi database diawali dengan adanya pemanggilan statement Data Manipulation Language (DML) seperti
  - INSERT
  - DELETE
  - UPDATE

Akhir Transaksi

- Transaksi berakhir ketika terjadi salah satu hal berikut :
  - Pemanggilan statement COMMIT
  - Pemanggilan statement ROLLBACK
  - Pemanggilan statement Data Definition Language (DDL) seperti CREATE, ALTER, DROP. Hal ini akan mengakibatkan COMMIT.
  - Keluar dari SQL*PLUS secara normal. Hal ini akan mengakibatkan COMMIT
  - Keluar dari SQL*PLUS secara abnormal atau kegagalan sistem (crash, restart). Hal ini akan mengakibatkan ROLLBACK.

Kondisi Data Sebelum COMMIT/ROLLBACK

- Perubahan data dapat dikembalikan (restore) ke kondisi sebelum DML pertama dieksekusi (commit terakhir).
- User yang login akan dapat melihat data hasil perubahan, tetapi user lain tidak.
- Jika user melakukan DML terhadap baris data, maka data tersebut akan di-lock. User lain tidak bisa mengupdate data tersebut.

Kondisi data setelah COMMIT

- Semua perubahan data tersimpan secara permanen.
- Semua SAVEPOINT dihapus
- Semua lock dilepas. Sehingga user lain dapat mengupdate data tersebut.

Kondisi data setelah ROLLBACK

- Semua perubahan data dibatalkan. Data kembali ke kondisi sebelum awal transaksi.
- Semua SAVEPOINT akan dihapus
- Semua lock akan dilepas

Demo Transaksi

- Login ke oracle
- Buat sebuah tabel. Misalnya tabel Rekening

```
CREATE TABLE rekening(
    NoRek INTEGER PRIMARY KEY,
    Nama VARCHAR(10),
    Saldo NUMBER
)
```

Demo Transaksi

- Beri beberapa contoh data

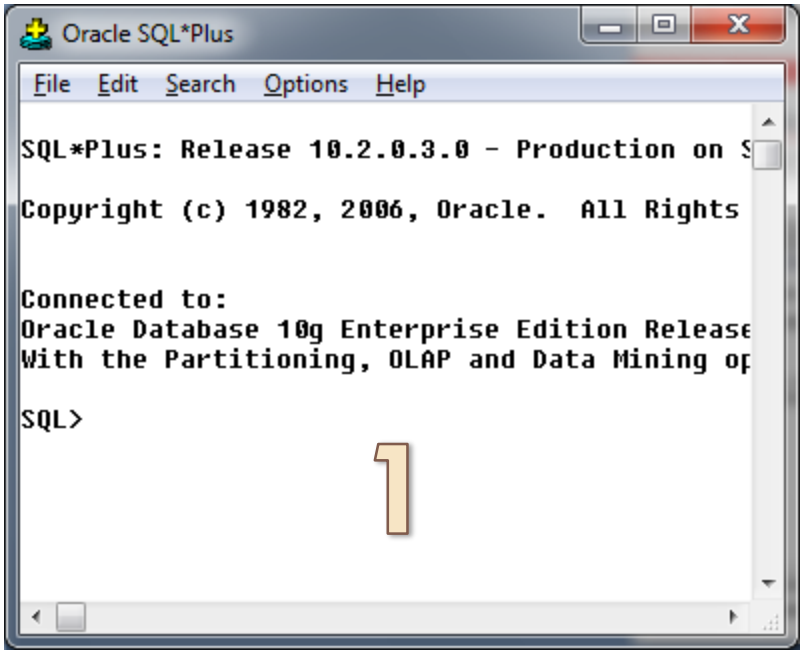
```
INSERT INTO rekening VALUES(1,'Ade',1000);
INSERT INTO rekening VALUES(2,'Badra',2000);
INSERT INTO rekening VALUES(3,'Cecep',3000);
COMMIT;
```

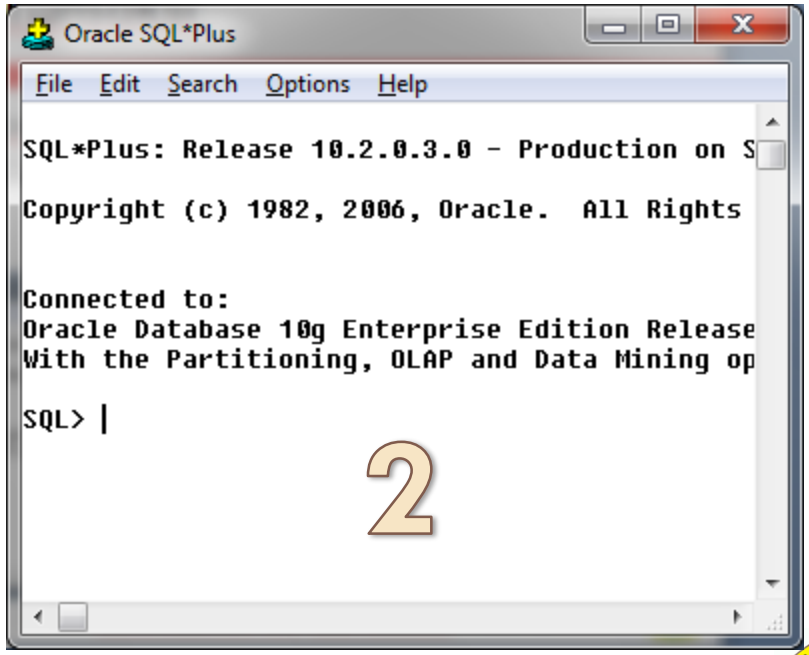
```
SELECT * FROM rekening;
```

NOREK	NAMA	SALDO
1	Ade	1000
2	Badra	2000
3	Cecep	3000

Demo Transaksi

- Buka 2 koneksi ke database dengan cara membuka 2 SQL*PLUS



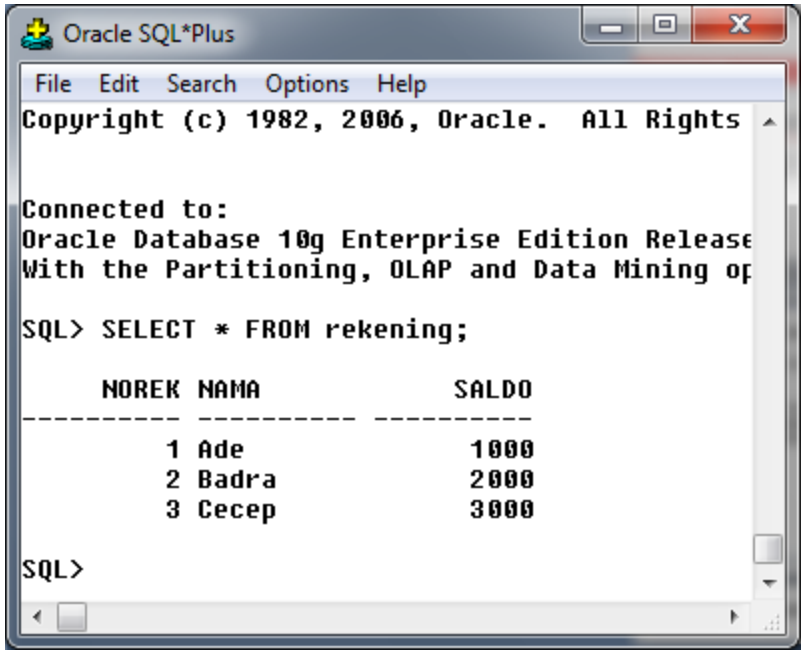


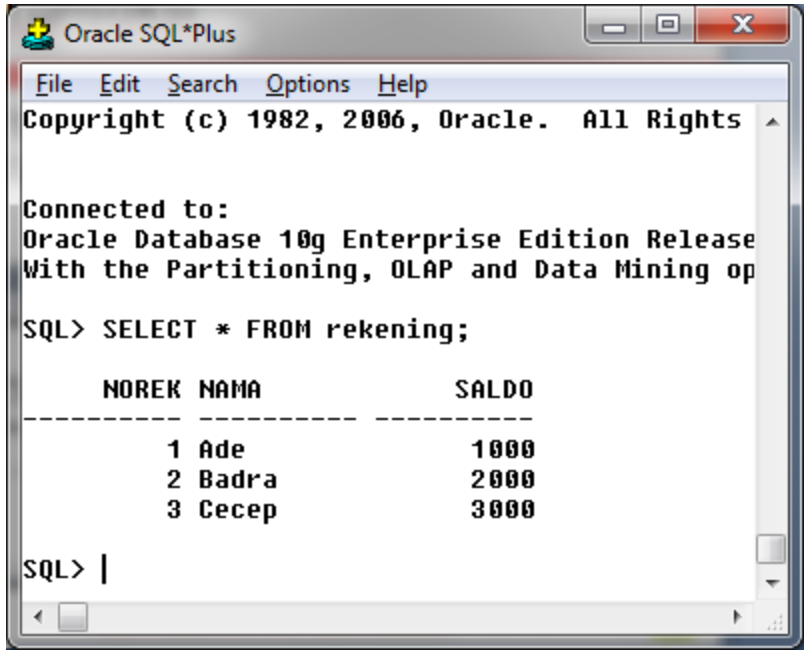
Demo Transaksi

- Eksekusi SQL pada kedua SQL*PLUS.

```
SELECT * FROM rekening
```

Demo Transaksi





Kedua layar akan menampilkan data yang sama (hasil commit terakhir).

Demo Transaksi (1)

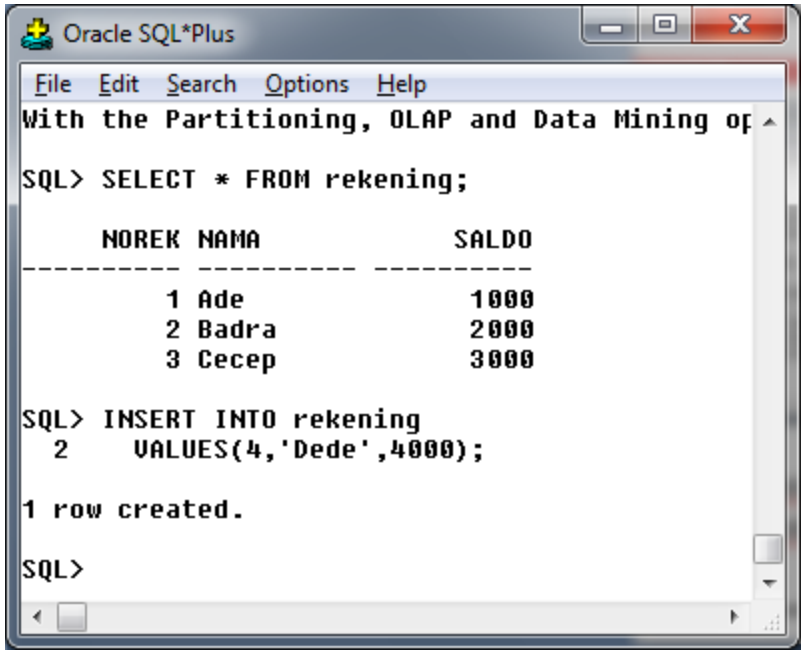
Mendemonstrasikan transaksi database dimana suatu perubahan data yang dilakukan oleh seorang user tidak dapat diakses oleh user lain sebelum melakukan COMMIT.

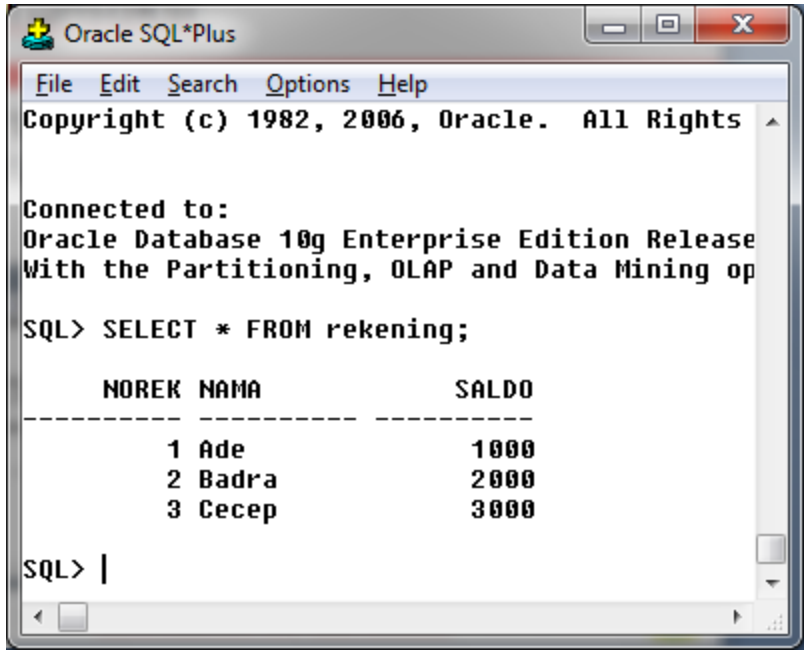
Demo Transaksi (1)

- Lakukan penambahan data baru pada SQLPlus 1

```
INSERT INTO rekening
  VALUES(4,'Dede',4000);
```

Demo Transaksi (1)





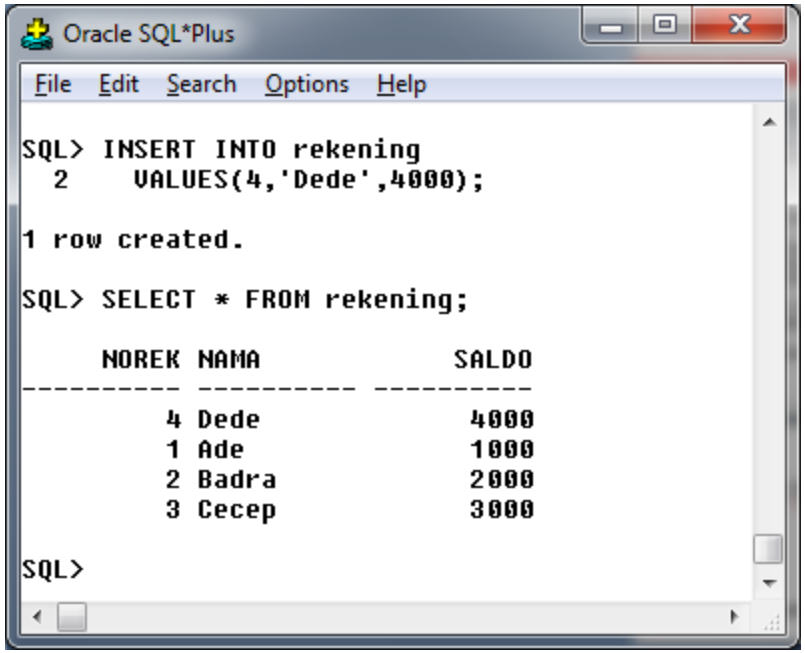
Data telah ditambahkan pada koneksi pertama

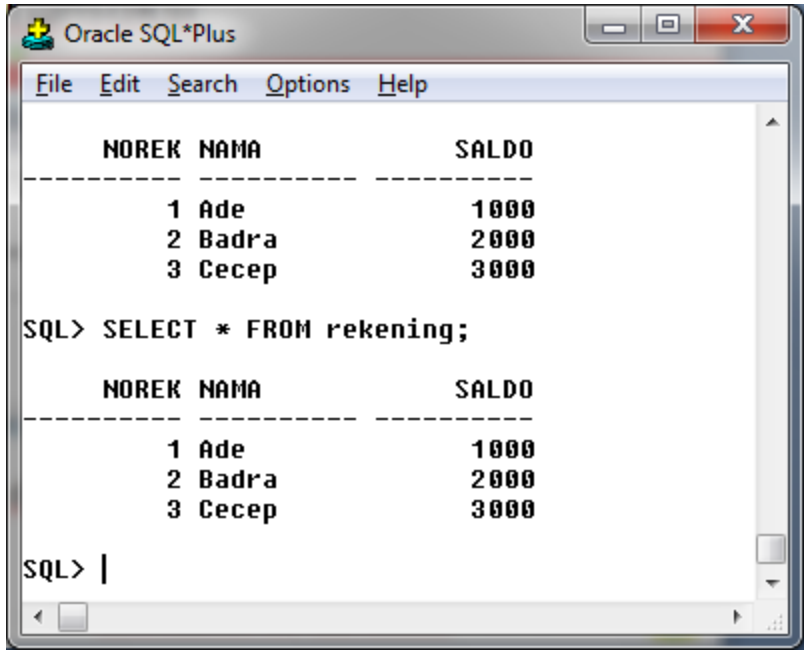
Demo Transaksi (1)

- Eksekusi SQL pada kedua SQL*PLUS.

```
SELECT * FROM rekening
```

Demo Transaksi (1)





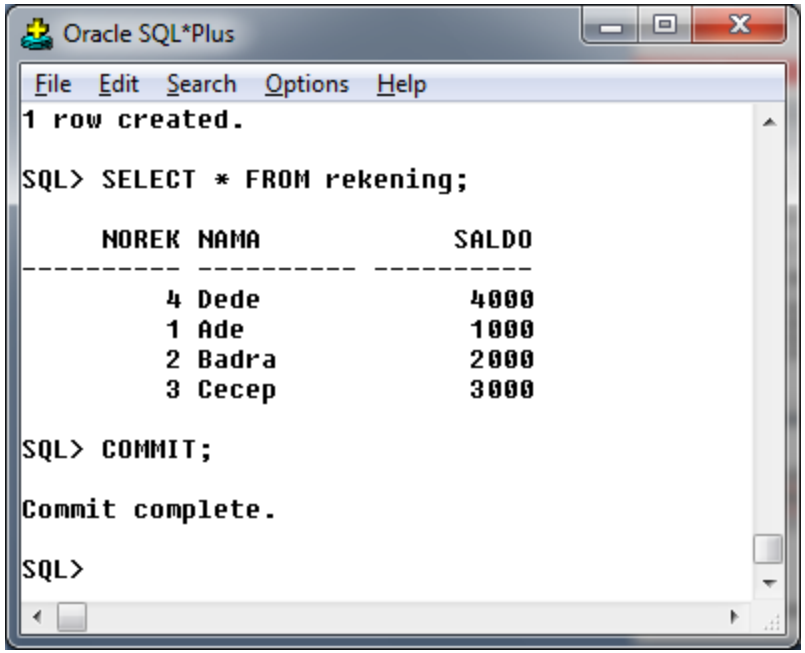
Data pada koneksi 1 ada 4 baris, sedangkan data pada koneksi 2 hanya ada 3. Ini menunjukkan perubahan data di koneksi pertama tidak bisa diakses/dilihat oleh koneksi 2. Hal ini dikarenakan perubahan data belum di-COMMIT.

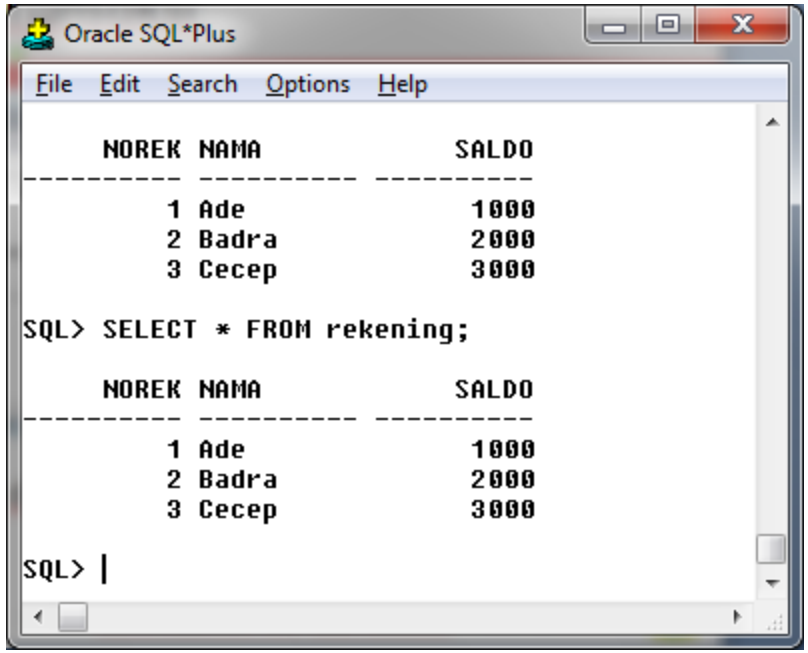
Demo Transaksi (1)

- Lakukan COMMIT di koneksi pertama.

```
COMMIT
```

Demo Transaksi (1)





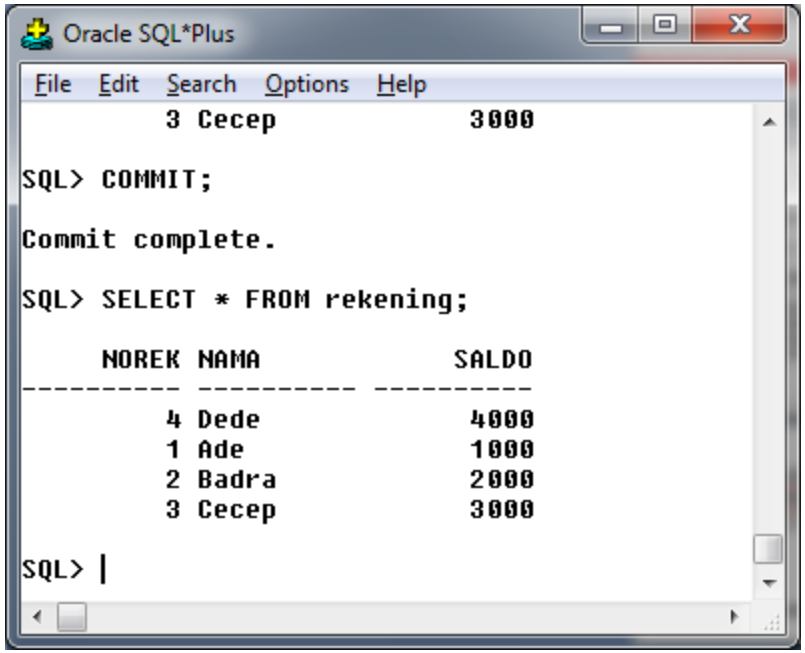
Data pada koneksi 1 ada 4 baris, sedangkan data pada koneksi 2 hanya ada 3. Ini menunjukkan perubahan data di koneksi pertama tidak bisa diakses/dilihat oleh koneksi 2. Hal ini dikarenakan perubahan data belum di-COMMIT.

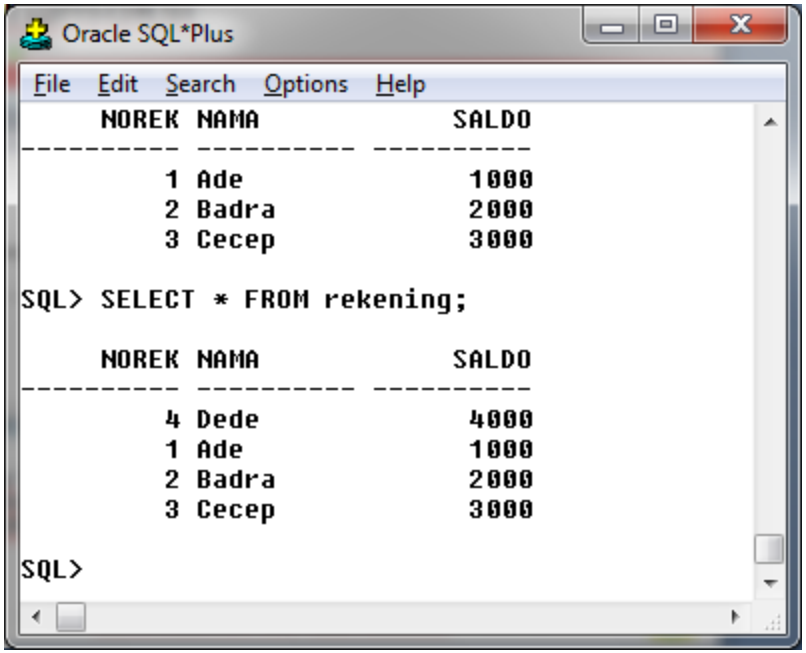
Demo Transaksi (1)

- Eksekusi SQL pada kedua SQL*PLUS.

```
SELECT * FROM rekening
```

Demo Transaksi (1)





Setelah COMMIT pada koneksi 1, maka semua perubahan yang telah dilakukan jadi dapat dilihat oleh user lain. Semua koneksi menampilkan baris sebanyak 4.

Demo Transaksi (2)

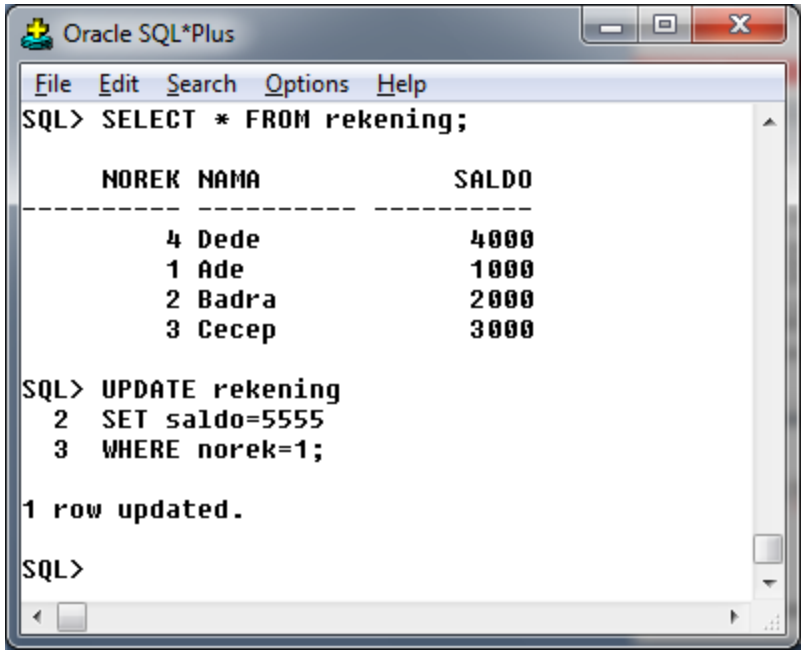
Mendemonstrasikan transaksi database dimana suatu perubahan data yang dilakukan oleh seorang user akan mengunci (lock) baris-baris yang berubah sehingga user lain tidak bisa memanipulasi baris-baris tersebut.

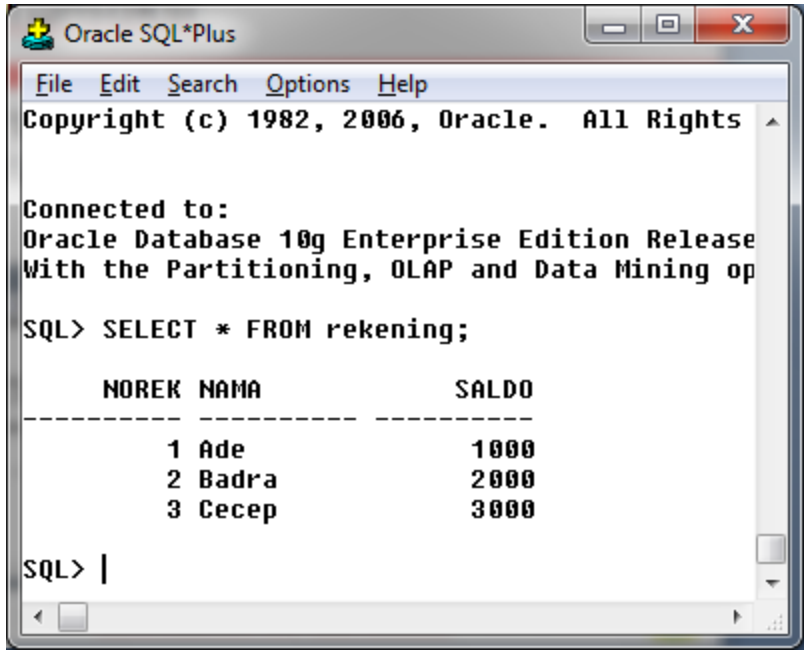
Demo Transaksi (2)

- Lakukan update pada koneksi 1.

```
UPDATE rekening
SET saldo=5555
WHERE norek=1;
```

Demo Transaksi (2)





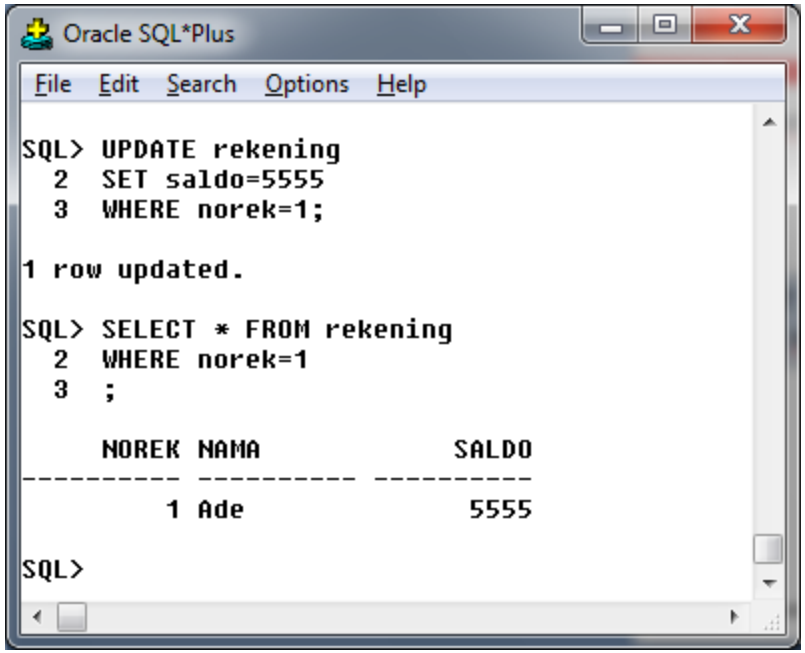
Data rekening dengan norek 1 telah diupdate.

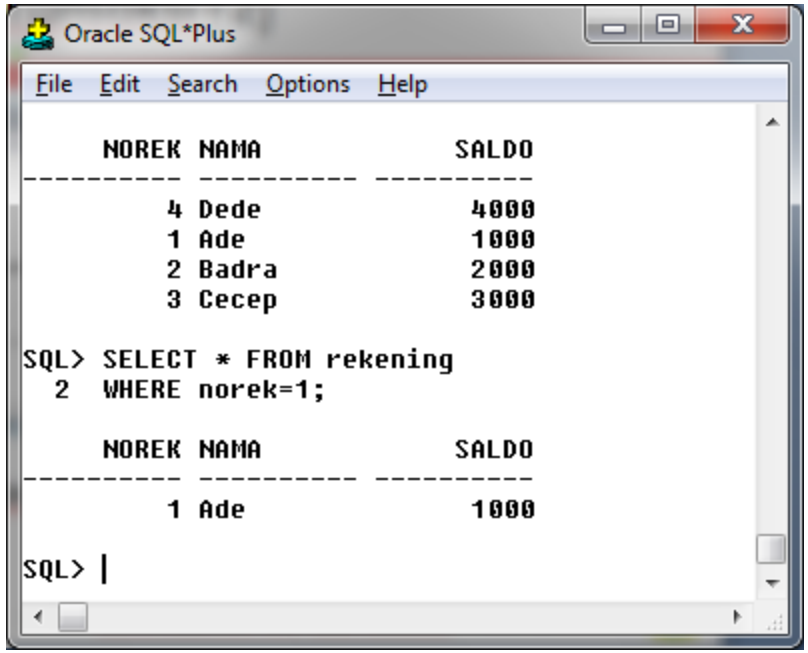
Demo Transaksi (2)

- Eksekusi SQL berikut di kedua koneksi.

```
SELECT * FROM rekening
WHERE norek=1;
```

Demo Transaksi (2)





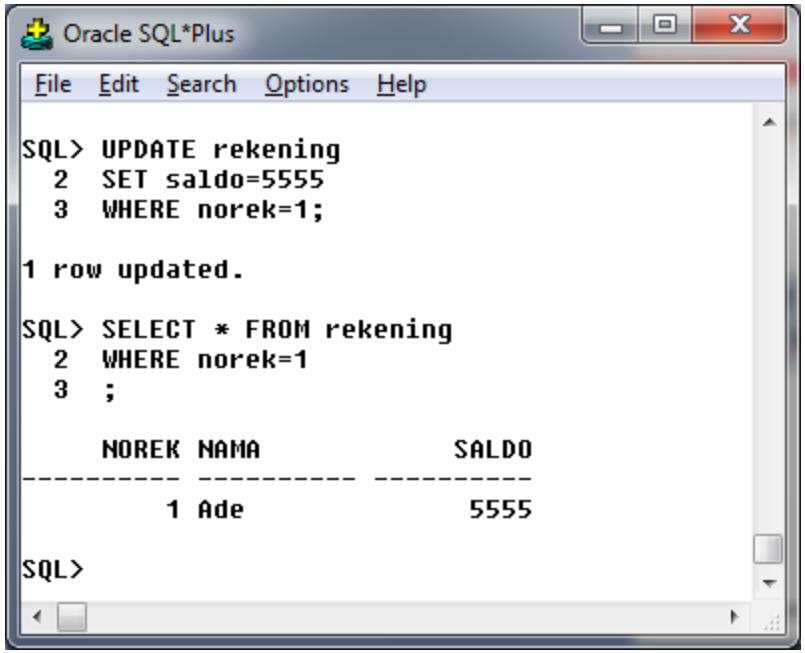
Koneksi pertama dapat melihat perubahan, tetapi user kedua tidak melihat perubahan tersebut.

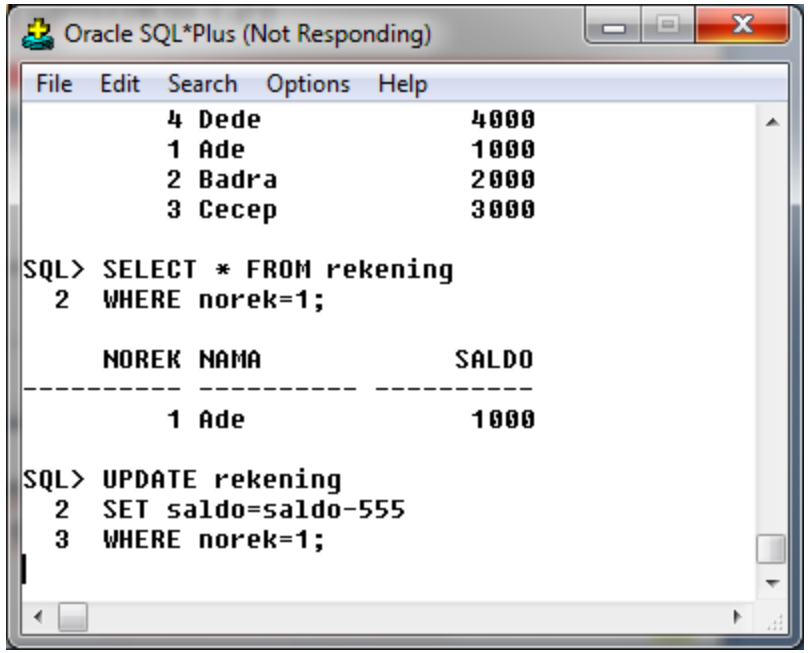
Demo Transaksi (2)

- Eksekusi SQL berikut di kedua koneksi kedua yang akan mengupdate data pada baris yang dikunci.

```
UPDATE rekening
SET saldo=saldo-555
WHERE norek=1;
```

Demo Transaksi (2)





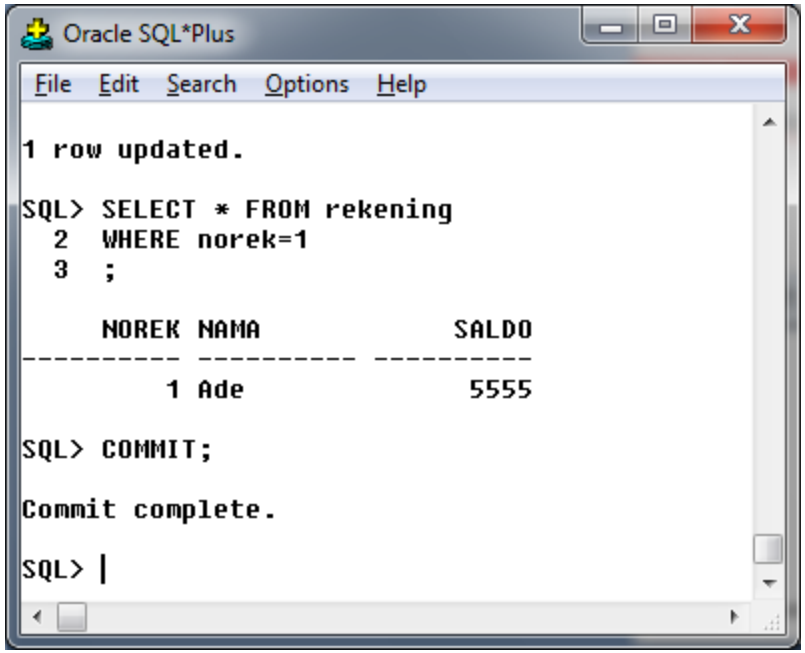
User ke-dua akan menunggu sampai baris yang dikunci dilepas oleh user pertama dengan COMMIT atau ROLLBACK.

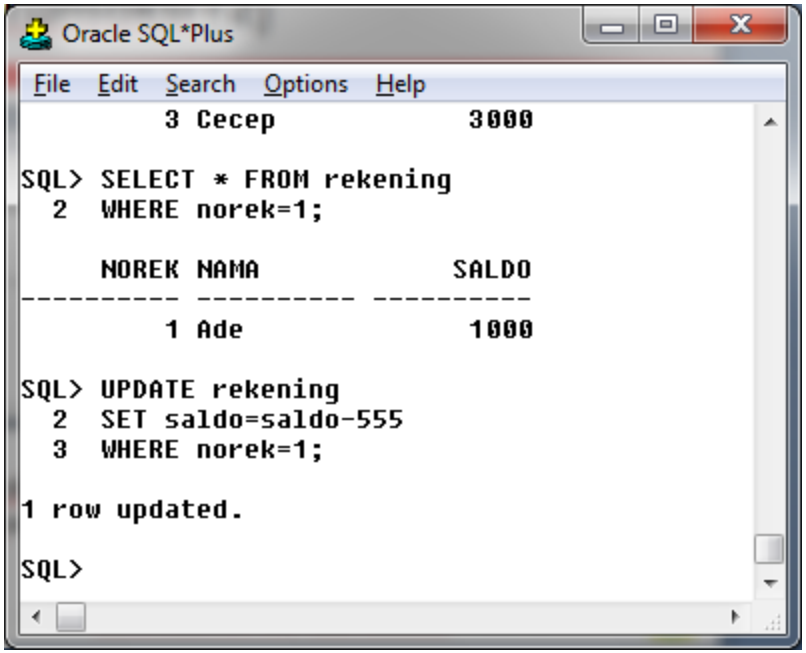
Demo Transaksi (2)

- Lakukan COMMIT di koneksi pertama.

```
COMMIT;
```

Demo Transaksi (2)





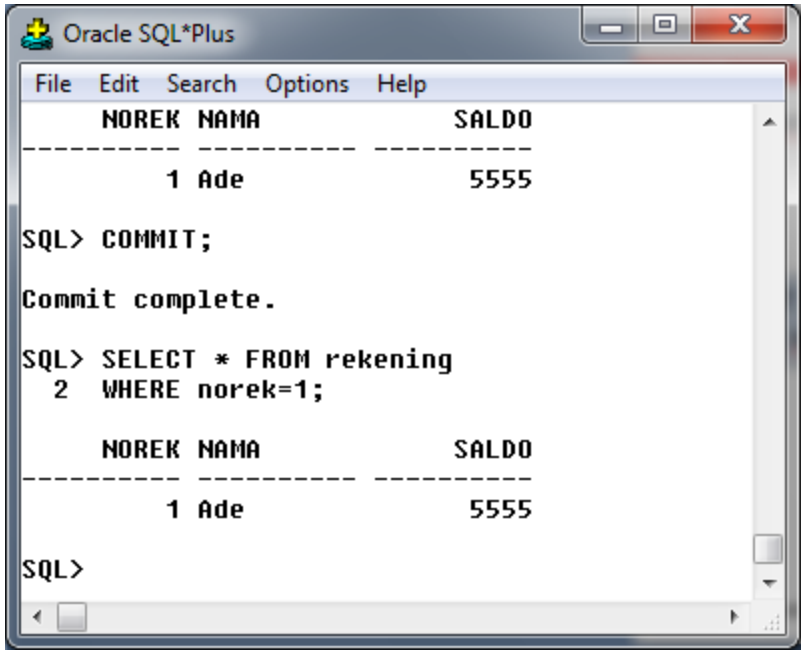
Ketika koneksi pertama melakukan COMMIT, maka lock pada baris norek 1 dilepas, sehingga akan membuat koneksi 2 melakukan update.

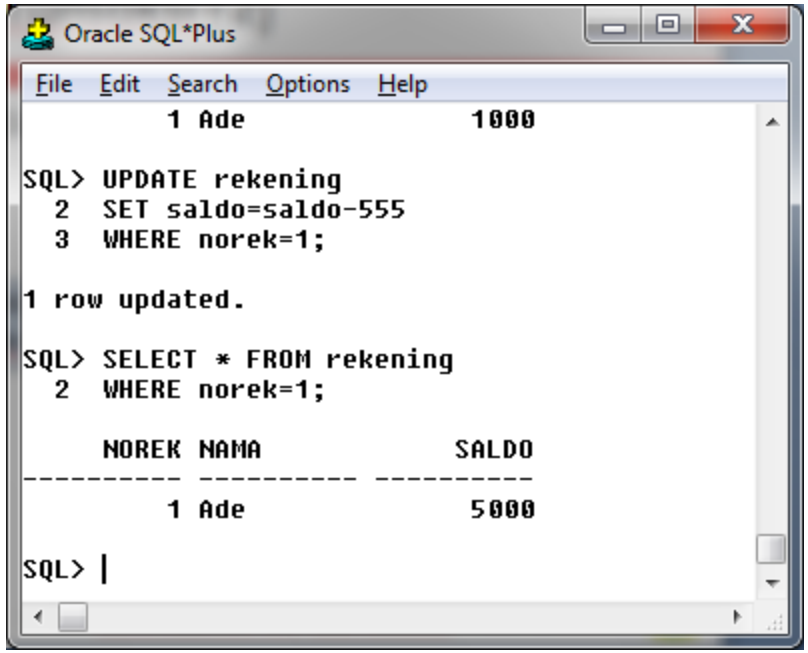
Demo Transaksi (2)

- Eksekusi SQL berikut di kedua koneksi.

```
SELECT * FROM rekening
WHERE norek=1;
```

Demo Transaksi (2)





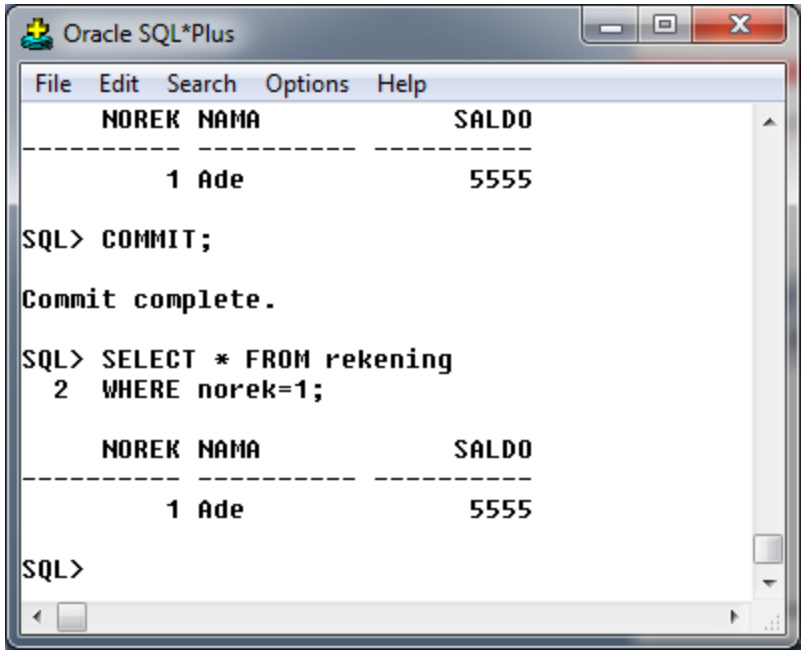
Hasil perubahan user 2 dapat dilihat di koneksi 2, tetapi belum terlihat di user 1. Ini dikarenakan user 2 belum melakukan COMMIT.

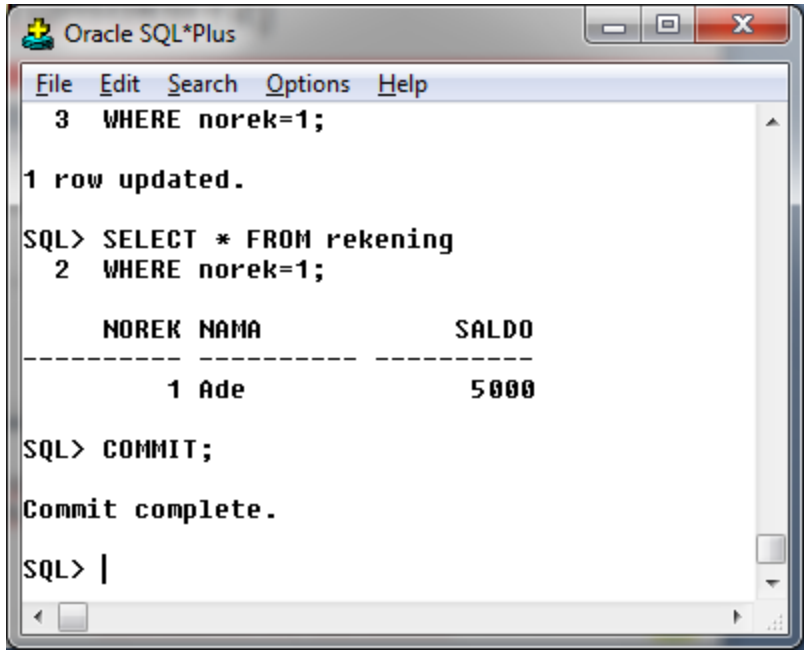
Demo Transaksi (2)

- Lakukan COMMIT di koneksi kedua.

```
COMMIT;
```

Demo Transaksi (2)





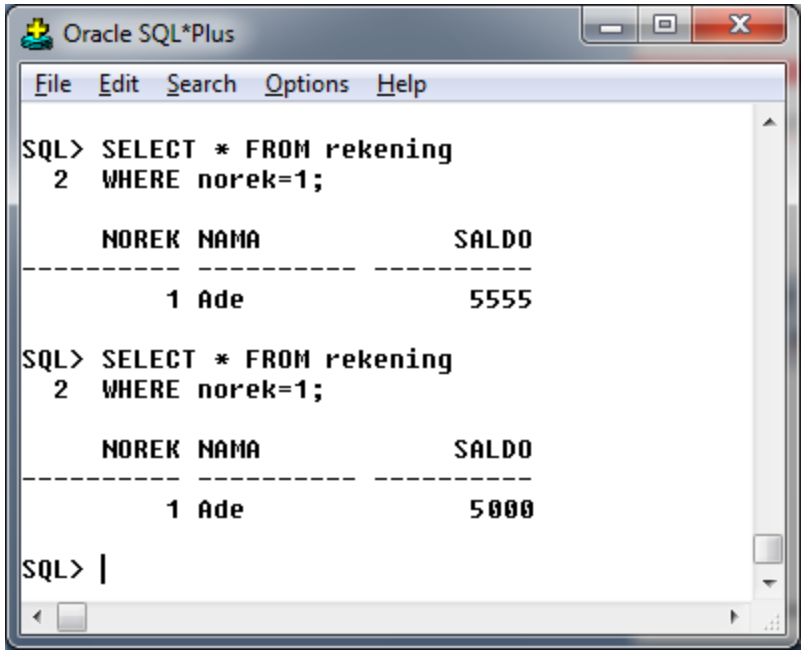
Ketika koneksi pertama melakukan COMMIT, maka lock pada baris norek 1 dilepas, sehingga akan membuat koneksi 2 melakukan update.

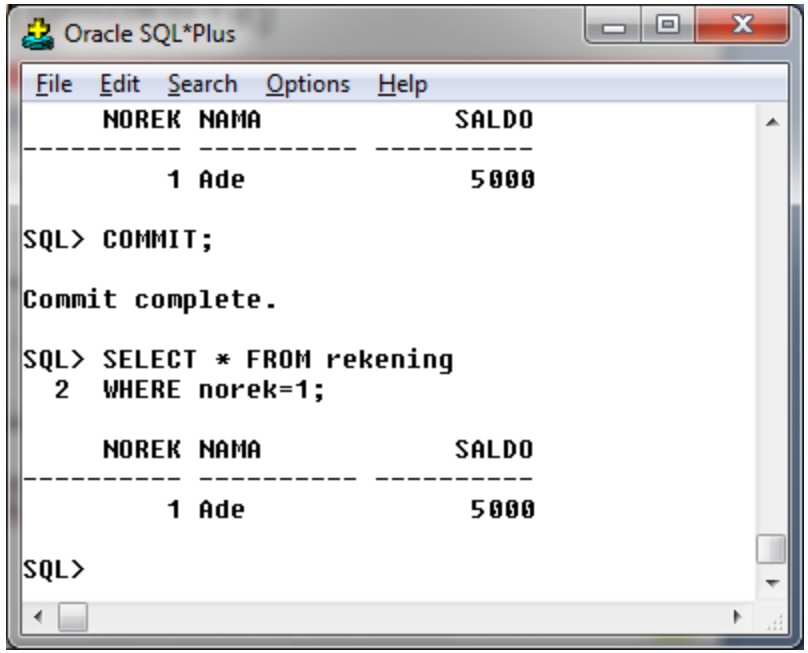
Demo Transaksi (2)

- Eksekusi SQL berikut di kedua koneksi.

```
SELECT * FROM rekening
WHERE norek=1;
```

Demo Transaksi (2)





Semua perubahan data telah COMMIT. Semua user dapat melihat data yang sama.

Demo Transaksi (3)

Mendemonstrasikan transaksi database menggunakan SAVEPOINT

Demo Transaksi (2)

- Eksekusi SQL berikut pada koneksi 1

1
```
INSERT INTO rekening
VALUES(6,'Ferdi',6000);
```
Penyisipan data baru dengan nomor rekening 6

2
```
SAVEPOINT SP1;
```
Titik aman SP1. Setelah data ditambah 1 baris

3
```
DELETE FROM rekening
WHERE NoRek<5;
```
Hapus data yang nomor rekening < 5

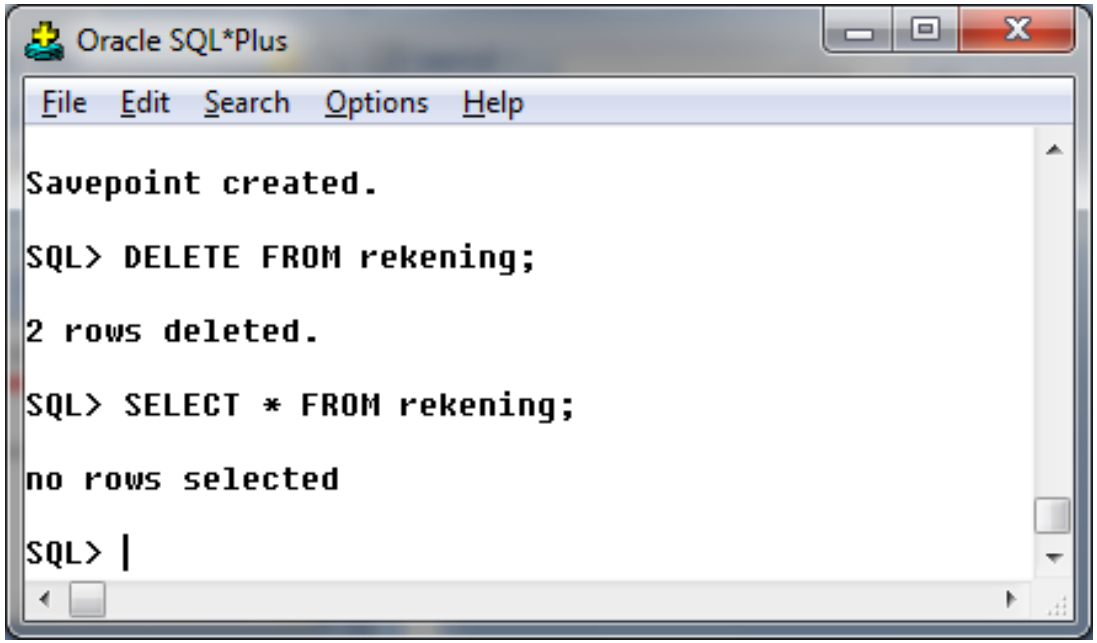
4
```
SAVEPOINT SP2;
```
Titik aman SP2. Setelah penghapusan data yang nomor rekening < 5

5
```
DELETE FROM rekening;
```
Hapus semua data rekening

Demo Transaksi (2)

- Eksekusi SQL berikut pada koneksi 1

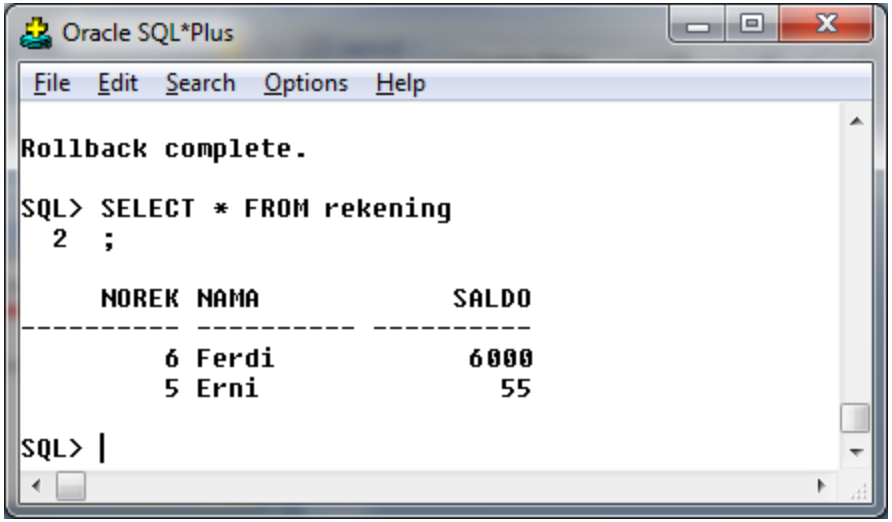
```
SELECT * FROM rekening
```



Demo Transaksi (2)

- Eksekusi SQL berikut pada koneksi 1

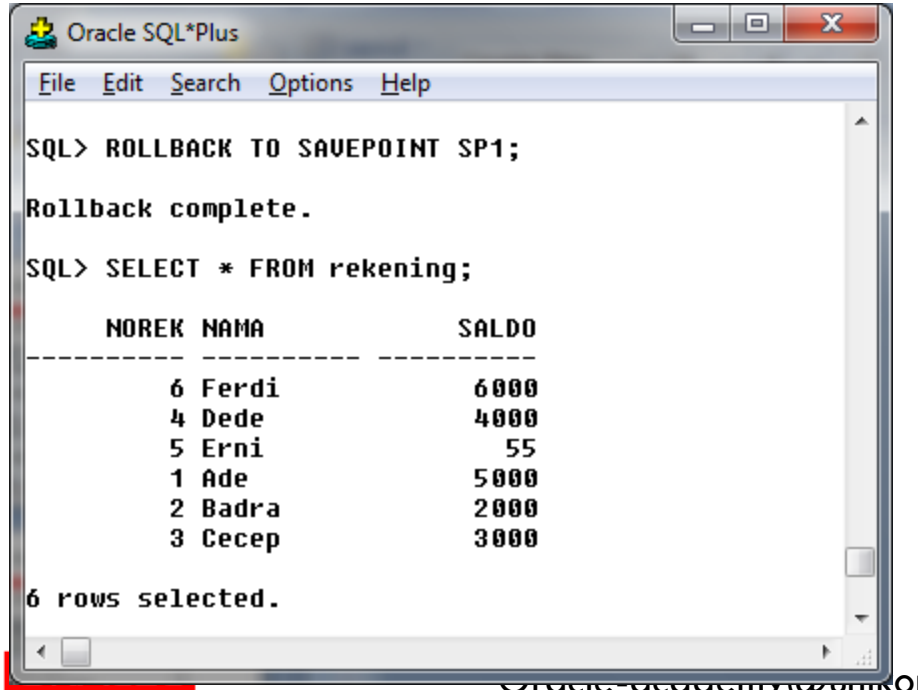
```
ROLLBACK TO SAVEPOINT SP2;
SELECT * FROM rekening;
```



Demo Transaksi (2)

- Eksekusi SQL berikut pada koneksi 1

```
ROLLBACK TO SAVEPOINT SP1;
SELECT * FROM rekening;
```



Demo Transaksi (2)

- Eksekusi SQL berikut pada koneksi 1

```
ROLLBACK;
SELECT * FROM rekening;
```

