DEADLOCK
&
RECOVERY SYSTEM

Sistem Basis Data

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

Penyelesaian masalah dengan Locking

- Latihan!
- Inconsistent Analysis Problem

Transaksi A	Waktu	Transaksi B
-	↓	-
Baca nilai 1(40) juml = 40	t1	-
-	↓	-
Baca nilai 2 (50) juml = 90	t2	-
-	↓	-
-	t3	Baca nilai 3 (30)
-	↓	-
-	t4	Modifikasi nilai 3 30 → 20
-	↓	-
-	t5	Baca nilai 1 (40)
	↓	
-	t6	Modifikasi nilai 1 40 → 50
-	↓	
-	t7	Commit
-	↓	
Baca nilai 3 (20) juml = 110 (bukan 120)	t8	
-	↓	

Nilai 1 = 40

Nilai 2 = 50

Nilai 3 = 30

Transaksi A menjumlahkan nilai 1, 2 dan 3

Transaksi B nilai3 dikurangi 10 dan nilai1 ditambah 10

Waktu	Transaksi A	Transaksi B	Nilai 1	Nilai 2	Nilai 3	Jumlah
t1	Begin_transaction					-
t2	jumlah=0					0
t3	read_lock (nilai1,2,3)	Begin_transaction				0
t4	Read (nilai1)	Write_lock (nilai3,nilai1)	**40**			0
t5	jumlah=jumlah+nilai1	Wait	40			40
t6	Write(jumlah)	Wait	40			40
t7	Read (nilai2)	Wait	40	**50**		40
t8	jumlah=jumlah+nilai2	Wait	40	50		90
t9	Write(jumlah)	Wait	40	50		90
t10	Read (nilai3)	Wait	40	50	**30**	90
t11	jumlah=jumlah+nilai3	Wait	40	50	30	120
t12	Write(jumlah)	Wait	40	50	30	120
t13	Commit/unlock (nilai1,2,3)	Wait	40	50	120	120
t14		Read (nilai3)	40	50	**30**	120
t15		Nilai3=nilai3-10	40	50	**20**	110
t16		Read (nilai1)	**40**	50	20	110
t17		Nilai1=nilai1+10	**50**	50	20	120
t18		Commit/unlock	50	50	20	120

Deadlock

- Deadlock merupakan kebuntuan (impasse) yang mungkin dihasilkan ketika dua atau lebih transaksi saling menunggu kunci yang disimpan oleh transaksi lain agar dilepaskan.

Deadlock

Time	T_{17}	T_{18}
t_1	begin_transaction	
t_2	write_lock($\mathbf{bal_x}$)	begin_transaction
t_3	read($\mathbf{bal_x}$)	write_lock($\mathbf{bal_y}$)
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	read($\mathbf{bal_y}$)
t_5	write($\mathbf{bal_x}$)	$\mathbf{bal_y} = \mathbf{bal_y} + 100$
t_6	write_lock($\mathbf{bal_y}$)	write($\mathbf{bal_y}$)
t_7	WAIT	write_lock($\mathbf{bal_x}$)
t_8	WAIT	WAIT
t_9	WAIT	WAIT
t_{10}	⋮	WAIT
t_{11}	⋮	⋮

Deadlock

- Teknik yang umum dilakukan untuk mengatasi deadlock :
  - Timeout
  - Deadlock prevention
  - Deadlock detection and recovery

Timeout

- Suatu transaksi yang meminta kunci hanya akan menunggu sistem mendefinisikan periode waktu.

- Jika kunci belum diberikan dalam periode ini, maka permintaan kunci kehabisan waktu (times out).

- Dalam kasus ini, DBMS mengasumsikan transaksi terjadi deadlock, walaupun mungkin tidak terjadi, dan transaksi tersebut **digagalkan** dan secara otomatis mengulang dari awal transaksi yang bersangkutan.

Deadlock Prevention

- Pendekatan mencegah terjadi deadlock menggunakan transaksi timestamps:
  - **Wait-Die** – jika transaksi baru membutuhkan data pada transaksi lama, **maka transaksi baru harus menunggu** dengan syarat memiliki **timestamp** yang **lebih kecil** dari transaksi lama, selain itu transaksi digagalkan (dies) dan diulang dengan timestamps yang sama.

  - **Wound-Wait** – Jika transaksi baru meminta kunci yang dimiliki oleh transaksi lama, maka **transaksi baru** harus **memiliki timestamp** yang **lebih besar**, jika tidak, transaksi akan digagalkan (*wounded*).

Deadlock Detection

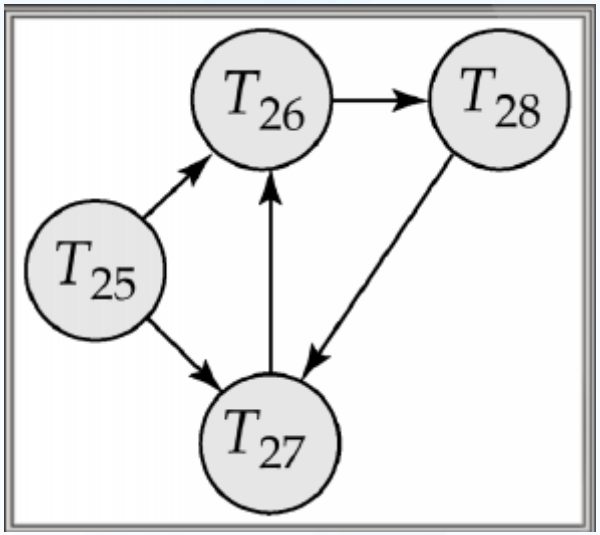
- Pendeteksian deadlock biasanya ditangani dengan membuat konstruksi *Wait For Graph* (WFG) yang memperlihatkan ketergantungan transaksi, yaitu transaksi A bergantung pada B jika transaksi B memegang kunci untuk data item yang ditunggu oleh transaksi A.

- WFG merupakan graf berarah (*directed graph*)

  G =(N, E), yang dapat dibentuk dengan cara :
  - Buatlah Node untuk setiap transaksi
  - Buatlah edge berarah T a -> T b, jika T a menunggu kunci untuk item yang sedang dikunci oleh T b.

Contoh

- Graph wait-for dengan siklus



Deadlock Recovery

- Ketika *deadlock* terdeteksi:
- Jalankan proses *rollback* pada satu atau beberapa transaksi untuk lepas dari *deadlock*. Pilih transaksi dengan resiko minimum.

- Tentukan sejauh mana transaksi harus *rollback*:
  - Batalkan transaksi mulai dari awal (total *rollback*)
  - *Rollback* dilakukan sejauh yang dibutuhkan agar terlepas dari *deadlock.*

Klasifikasi penyebab kegagalan

1. **Kesalahan logika**, program tidak dapat dijalankan karena kesalahan input, sehingga data tidak ditemukan
2. **Kesalahan sistem**, sistem berada pada kondisi yang tidak diinginkan (seperti deadlock), sehingga eksekusi transaksi tidak dapat dijalankan
3. **Kerusakan sistem**, seperti kegagalan fungsi perangkat keras, rusak, hang, menyebabkan hilangnya data pada penyimpanan *volatile*
4. **Kegagalan disk**, menyebabkan hilangnya data, jika terdapat bad sector di dalam disk akibat *read/write* dari disk rusak

Recovery Transaksi

- Recovery transaksi berarti database dikembalikan ke kondisi awal mendekati waktu terjadinya kegagalan.
- Informasi perubahan data selama transaksi harus disimpan.
- Informasi tersebut disebut **sistem log**.

Recovery Transaksi

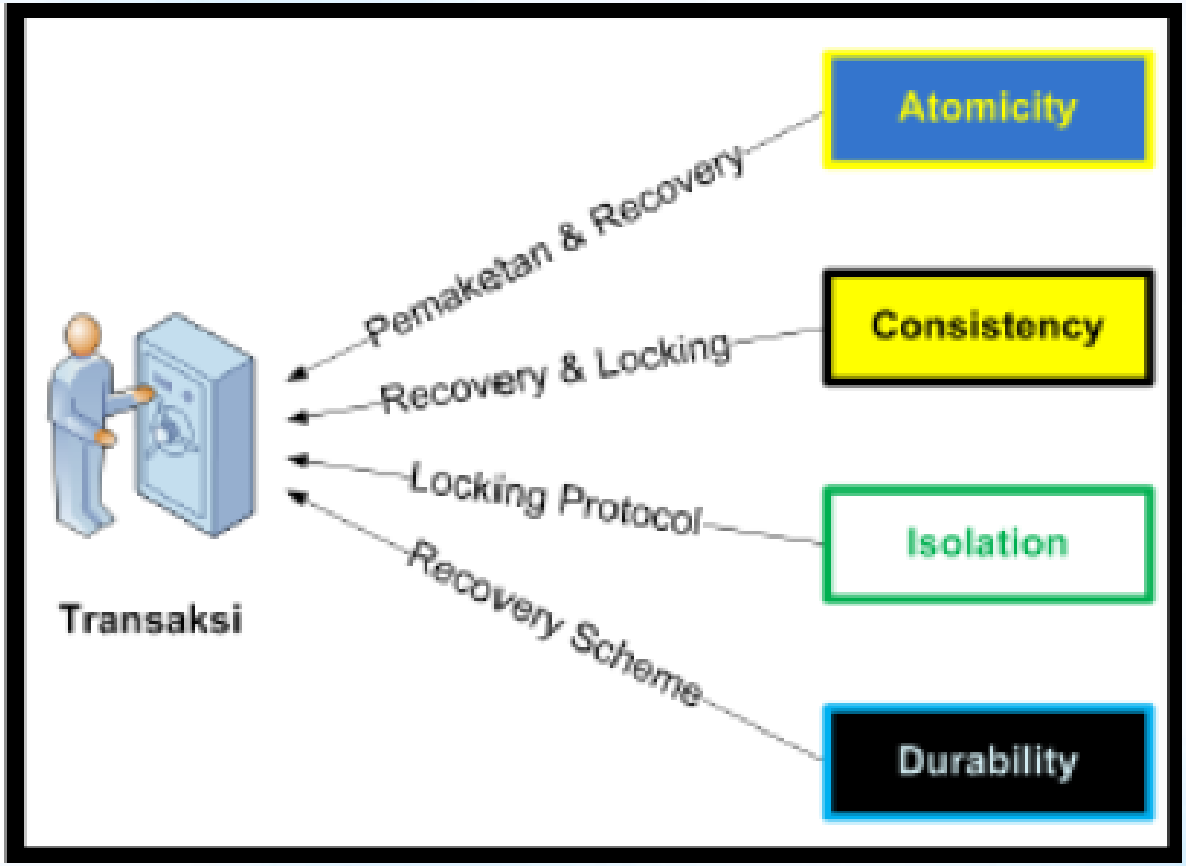
- Teknik recovery berhubungan dengan kontrol konkurensi yang digunakan pada sistem tersebut.

- Bila suatu transaksi dieksekusi oleh DBMS, maka untuk menjaga konsistensi:
  - Suatu transaksi harus selesai (mencapai commit) dan hasilnya disimpan secara permanen di database.

Recovery Transaksi

- Kegagalan sistem saat melakukan transaksi merupakan hal yang harus diperhatikan karena terkait dengan sifat suatu transaksi yaitu ACID (Atomicity, Consistency, Isolation, Durability)

Hubungan recovery dengan transaksi



Recovery Transaksi

- Recovery merupakan solusi dari hampir semua fitur ACID yang harus dimiliki oleh setiap transaksi.
- Transaksi tidak hanya disebut sebagai unit proses, melainkan juga sebagai suatu unit recovery.
- Hampir seluruh DBMS modern menggunakan konsep recovery berbasis log.

Ilustrasi

Waktu	Kegiatan dan Kejadian	Ilustrasi
06:00	Bob membackup DATAFILE ke DVD Teknisi menonaktifkan UPS untuk di perbaiki	
08:00	Yuni melakukan penarikan uang Rp 2.000.000	
09:00	Perusahaan listrik memutuskan aliran listrik Server basis data bank hemat mati secara tiba-tiba.	
09:15	Aliran listrik kembali menyala. Bob menghidupkan kembali server Basis Data dan menemukan bahwa Hard Disk rusak, lalu Bob mentransfer data backup terakhir (jam 06:00) ke Hard Disk baru.	
Hari Esok	Yuni menemukan bahwa saldo tabungannya masih Rp. 5.000.000, bukan Rp 3.000.000 seperti seharusnya.	Inkonsistensi Data 

Ilustrasi

- Bank hemat adalah bank yang memiliki banyak nasabah, bank ini menggunakan database untuk menampung seluruh kegiatan transaksi dan data nasabah.

- Bob adalah Database Administrator Bank Hemat yang bertanggung jawab akan system basis data bank tersebut.

- Pada suatu hari, teknisi maintenance menemukan bahwa UPS (Unit Power Supply) server basis data mengalami masalah dan butuh mengalami perbaikan. Perbaikan UPS dilaksanakan pada jam 06.00.

Ilustrasi (lanjutan)

- Malamnya, Bob melakukan backup DATAFILE data nasabah sebelum perbaikan UPS.
- Bob melakukan backup seluruh DATAFILE nasabah ke dalam media optik DVD, sedangkan LOG file tidak dibackup karena telah direplikasi pada harddisk.
- Proses backup ini baru selesai tepat saat tim teknisi data untuk melakukan perbaikan UPS pada jam 06.00.
- Kepala operasional memutuskan bahwa selama perbaikan UPS, server basis data akan berjalan tanpa menggunakan UPS.

Ilustrasi (lanjutan)

- Pada hari yang sama, jam 08.00, seorang nasabah bernama Yuni mengambil uang dari tabungannya sebesar Rp.2.000.000 dari ATM. Sehingga saldo tabungan Yuni berkurang dari Rp.5.000.000 menjadi Rp.3.000.000

- Pada jam 09.00, perusahaan listrik memutuskan aliran listrik di kota tersebut selama 15 menit untuk maintenance. Aliran listrik terputus, termasuk aliran listrik ke server basis data bank hemat.

Ilustrasi (lanjutan)

- Listrik kembali menyala 15 menit kemudian, namun saat Bob mengaktifkan kembali server basis data, ternyata server tidak menyala dengan benar, lalu Bob menemukan bahwa harddisk yang menyimpan DATAFILE nasabah telah rusak dikarenakan putusnya aliran listrik secara tiba-tiba.

- Akhirnya Bob memutuskan untuk mengembalikan kondisi basis data menggunakan backup terakhir yang ada pada DVD (backup pada jam 06.00) ke harddisk baru, tetapi Bob **tidak melakukan proses recovery menggunakan data log transaksi**.

- Keesokan harinya saat Yuni kembali ke ATM, dia terkejut karena menemukan saldo tabungannya masih berjumlah Rp.5.000.000.

- Pada ilustrasi, merupakan salah satu contoh terabaikannya sifat durability yang harus dimiliki oleh transaksi.

- Sifat durability berarti “setelah suatu transaksi commit, perubahan yang terjadi harus tetap bertahan di dalam basis data”.

Bagaimana cara untuk mencegah hal tersebut?

bersambung…

☺