Sistem Basis Data

RECOVERY SYSTEM

Alif Finandhita, S.Kom

KLASIFIKASI KEGAGALAN

- Crash adalah suatu kegagalan dari suatu sistem

- Penyebab dari kegagalan adalah :
  - **Disk Crash**, yaitu informasi yang ada di dalam disk akan hilang
  - **Power Failure**, yaitu informasi yang disimpan pada memori utama dan register akan hilang
  - **Software Error**, yaitu output yang dihasilkan tidak benar dan sistem databasenya sendiri akan memasuki suatu kondisi yang tidak konsisten (inkonsisten)

KLASIFIKASI KEGAGALAN (2)

- Kegagalan juga bisa terjadi karena diakibatkan oleh adanya force majour seperti misalkan terjadinya bencana alam (banjir, kebakaran) dan juga adanya sabotase terhadap sistem komputer

- Akibat dari adanya kejadian – kejadian tersebut maka informasi yang berkenaan dengan sistem database akan menjadi hilang

KLASIFIKASI KEGAGALAN (3)

- Berbagai kegagalan/failure yang mungkin muncul dalam sebuah sistem, masing-masing perlu ditangani dengan serius untuk mencegah terjadinya kerugian yang lebih besar.
- Salah satu kegagalan yang tidak terlalu sulit untuk ditangani adalah kegagalan yang disebabkan oleh hilangnya informasi.
- Kegagalan yang disebabkan oleh hilangnya informasi dapat ditangani dengan menggunakan teknik recovery

KLASIFIKASI KEGAGALAN (4)

- **Logical Error**, program tidak dapat lagi dilaksanakan karena kesalahan input, sehingga data tidak ditemukan dan dapat terjadi overflow.
- **System Error**, sistem berada pada keadaan yang tidak diinginkan (seperti terjadi deadlock), akibatnya program tidak dapat dilanjutkan namun setelah beberapa waktu program dapat dijalankan kembali.
- **System Crash**, kegagalan fungsi perangkat keras, menyebabkan hilangnya data pada volatile storage, tetapi data pada non-volatile storage masih tetap ada.
- **Disk Failure**, hilangnya data dari sebuah blok disk disebabkan oleh kerusakan head atau kesalahan pada waktu pengoperasian transfer data.

KLASIFIKASI STORAGE (5)

Jenis-jenis storage berdasarkan kecepatan relatif dan ketahanan terhadap kegagalan, antara lain :

- **Volatile Storage**, biasanya informasi yang terdapat pada volatile akan hilang, jika terjadi kerusakan sistem (*system crash*).
- **Non-Volatile Storage**, biasanya informasi yang terdapat pada non volatile strorage tidak akan hilang jika terjadi kerusakan sistem.
- **Stable Storage**, informasi yang terdapat dalam stable storage tidak pernah hilang. Untuk menjadikan stable storage, maka informasi direplikasi pada beberapa non-volatile storage.

RECOVERY TRANSAKSI

- Recovery dari kegagalan transaksi artinya database dikembalikan ke kondisi yang terdahulu, mendekati waktu terjadinya kegagalan.

- Untuk melakukan hal tersebut, perlu disimpan informasi mengenai perubahan terhadap data selama pelaksanaan transaksi di luar database. Informasi tersebut disebut sebagai sistem log.

RECOVERY TRANSAKSI (2)

- Teknik recovery berhubungan erat dengan mekanisme kontrol konkurensi yang digunakan pada sistem bersangkutan.

- Bila suatu transaksi dieksekusi oleh DBMS, maka untuk menjaga konsistensi dari database :
  - Semua operasi pada suatu transaksi harus selesai dilaksanakan dengan sukses (mencapai titik commit), dan hasilnya disimpan secara permanen pada database.
  - Suatu transaksi tidak mempunyai efek apapun terhadap database dan transaksi lain. Hal ini perlu, bila transaksi gagal setelah melakukan beberapa operasi, tapi sebelum semua lengkap dilaksanakan.

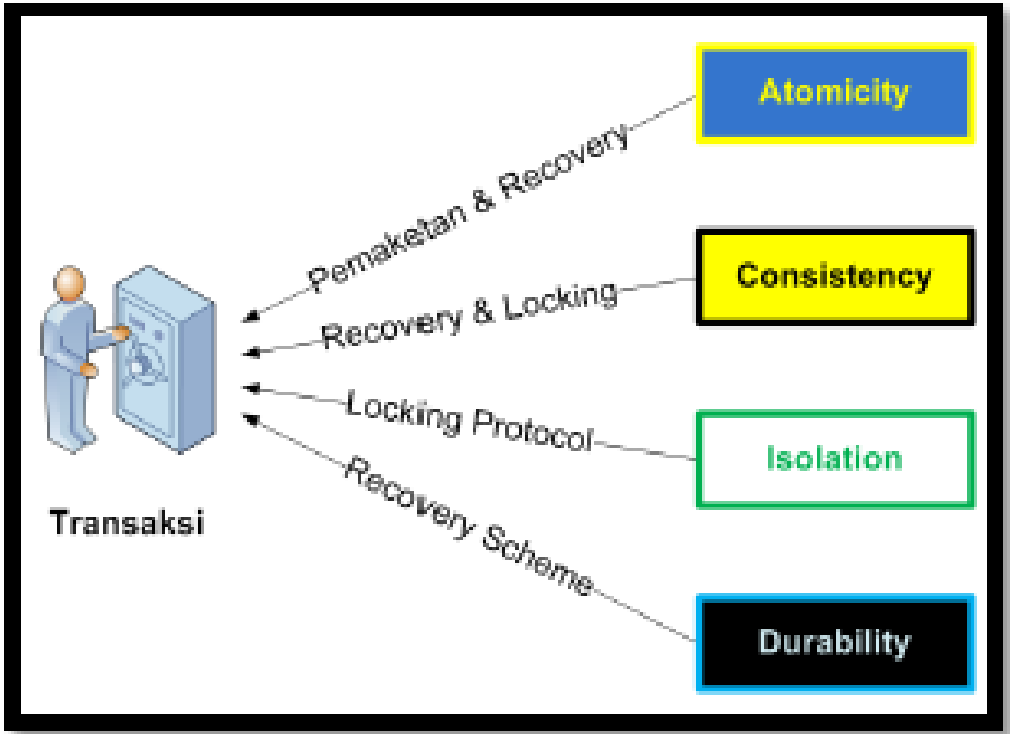
RECOVERY TRANSAKSI (3)

- Sistem Basis Data harus melaksanakan tindakan tertentu untuk menjamin sifat transaksi yang Atomik dan Durable.

- Bagian terintegrasi dari sebuah Sistem Basis Data adalah sebuah skema recovery yang dapat memulihkan Basis Data kepada keadaan konsisten sebelum kegagalan sistem.

- Kegagalan sistem saat melakukan transaksi merupakan hal yang harus diperhatikan secara seksama karena terkait dengan 4 sifat yang harus dimiliki oleh suatu transaksi, ACID (Atomicity, Consistency, Isolation, dan Durability)

RECOVERY TRANSAKSI (4)



Ilustrasi Hubungan Fitur & Solusi Transaksi

RECOVERY TRANSAKSI (5)

- Seperti dideskripsikan pada gambar di atas, Recovery merupakan solusi dari hampir semua fitur ACID yang harus dimiliki oleh setiap transaksi.
- Oleh karena itu, transaksi tidak hanya disebut sebagai suatu unit pekerjaan, namun juga sebagai suatu unit recovery.
- Pada DBMS modern, konsep recovery sangat penting, bahkan konsep Backup pada DBMS modern tidak akan dapat dimengerti sepenuhnya tanpa pengetahuan yang cukup mengenai konsep recovery.
- Hampir seluruh DBMS modern menggunakan konsep recovery berbasis ***Log.***

ILUSTRASI 1

1. Bank Hemat adalah Bank yang memiliki banyak nasabah, Bank ini menggunakan Basis Data untuk menampung seluruh kegiatan transaksi dan data keuangan nasabah.

2. Bob adalah Administrator Basis Data Bank Hemat yang bertanggung jawab akan System Basis Data Bank tersebut.

3. Pada suatu hari, teknisi *maintenance* menemukan bahwa umur baterai pada UPS server Basis Data telah sampai pada batas pemakaian. Perbaikan UPS membutuhkan waktu selama 6 jam dan akan dilaksanakan pada keesokan harinya jam 06:00.

ILUSTRASI 2 – (2)

4. Malamnya Bob melakukan Backup DATAFILE saldo nasabah sebelum perbaikan.

5. Bob melakukan Backup seluruh DATAFILE nasabah kedalam media Optik DVD, sedangkan LOG FILE tidak di backup karena telah di replikasi pada beberapa Hard Disk.

6. Proses Backup ini baru selesai tepat saat Tim teknisi datang untuk melakukan perbaikan UPS (Jam 06:00).

7. Saat teknisi melakukan perbaikan UPS, server Basis Data di nonaktifkan sebentar untuk mencabut sambungan listrik dari UPS ke server.

8. Kepala operasional memutuskan bahwa selama perbaikan UPS, Server akan berjalan tanpa menggunakan UPS.

ILUSTRASI 2 – (3)

9. Pada hari yang sama, seorang nasabah bernama Yuni pada jam 08:00 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.

10. Pada jam 09:00 di hari yang sama, Perusahaan listrik memutuskan bahwa aliran listrik seluruh kota akan di putus selama 15 menit untuk *maintenance* di karenakan Banjir. Praktis aliran listrik seluruh kota terputus, termasuk aliran listrik ke server basis data bank Hemat.

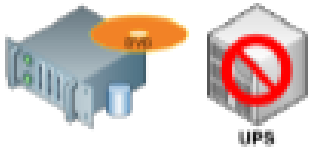
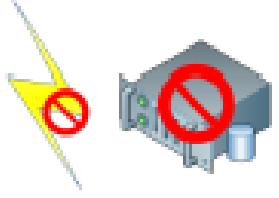
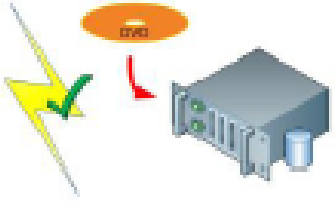
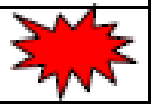
ILUSTRASI 3 – (3)

9. 15 menit kemudian listrik kembali menyala, tetapi saat Bob mengaktifkan kembali server Basis Data, ternyata server tidak mau menyala dengan benar dan Bob menemukan bahwa Hard Disk yang menyimpan DATAFILE nasabah telah rusak permanen di karenakan putusnya aliran listrik secara tiba-tiba.

10. Akhirnya Bob memutuskan untuk mengembalikan kondisi Basis Data menggunakan backup terakhir yang ada pada DVD (Backup jam 06:00) ke Hard Disk baru, tetapi Bob **tidak melakukan proses recovery menggunakan Data Log Transaksi.**

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

ILUSTRASI 3 – (4)

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	 Pengambilan 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 

RECOVERY BERBASIS LOG

- Log
- Log Dalam Durability Transaksi
- Log Dalam Keatomikan dan Konsistensi Transaksi
- Log Pada Volatile Storage

LOG

- *Log* adalah catatan transaksi secara mendetail. setiap transaksi yang berjalan pada server basis data di catat secara mendetail pada *Log* ini.

- Tiap record pada *Log* menggambarkan operasi tunggal transaksi yang menuliskan data, *Record Log* tersebut berisi :

1. Nama transaksi : Nama unik dari transaksi yang menjalankan operasi write
2. Nama data item : Nama unik dari data item yang di tuliskan
3. Nilai lama : Nilai data item sebelum operasi write
4. Nilai baru : Nilai yang akan dimiliki data item setelah operasi write

- Sebagai tambahan, ada *Record Log* khusus yang menandakan awal dan akhir transaksi.

LOG (2)

- Contoh Isi Log :

Record No.	Isi Redo Log Record	Status
Record 1	<T_0, begin>	Berhasil
Record 2	<T_0, Saldo Yuni, 5.000.000, 3.000.000>	Berhasil
Record 3	<T_0, commit>	Berhasil

- Proses Recovery menggunakan *Log harus memiliki sifat* ***Idempoten,*** *yaitu pengeksekusian suatu operasi berulang kali, menghasilkan hasil yang sama dengan eksekusi satu kali.*

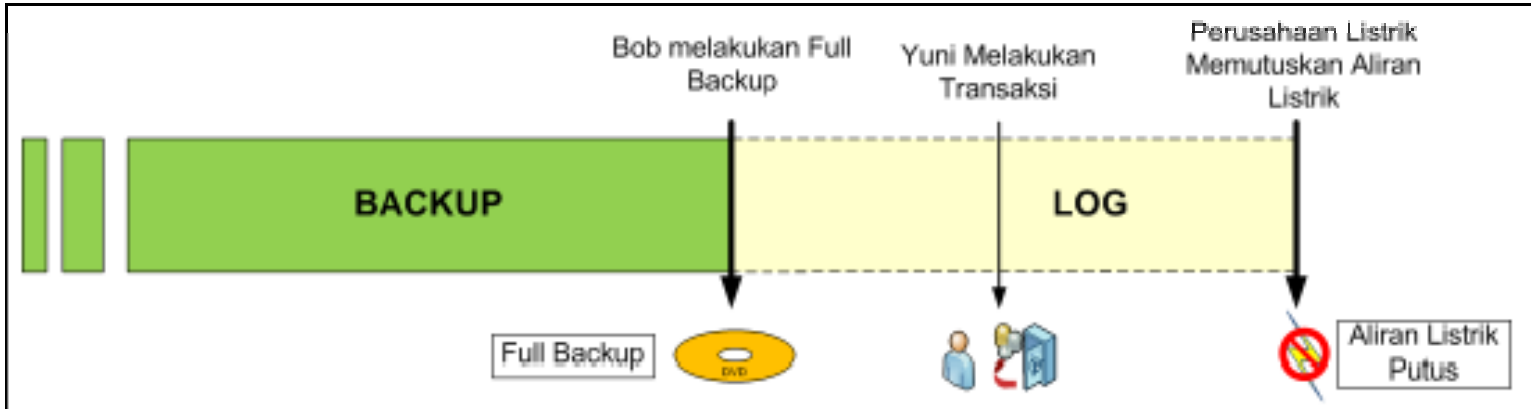
LOG DALAM DURABILITY TRANSAKSI

- Sifat Durability adalah sifat yang harus dimiliki transaksi, sifat durability ini adalah "*setelah suatu transaksi COMMIT, update harus bertahan di basis data".*
- Sifat durability ini harus dipenuhi apapun yang terjadi, untuk itu, digunakan *Log* untuk memastikan terpenuhinya sidat Durability ini.
- Pada ilustrasi 1 diatas, setelah Yuni melakukan transaksi pengambilan saldo sebesar Rp. 2.000.000 dan kemudian melihat bahwa saldonya tidak berkurang sama sekali adalah salah satu contoh terabaikannya sifat durable yang harus dimiliki transaksi. Untuk mencegah kejadian seperti ini, dilakukan penelusuran catatan *Log transaksi dengan melakukan operasi Redo.*

LOG DALAM DURABILITY TRANSAKSI (2)

Operasi Redo Transaksi

- Mekanisme proses ini adalah dengan Me-redo(ne) seluruh transaksi yang terekam didalam *Log.* Operasi ini juga biasa disebut "Forward Recovery*".*



- System melakukan penelusuran LOG FILE dari backup terakhir sampai terjadinya kegagalan System. Seluruh transaksi yang terekam pada LOG FILE tersebut di eksekusi ulang, sehingga tidak ada lagi kejadian hilangnya data suatu transaksi.

LOG DALAM KEATOMIKAN DAN KONSISTENSI TRANSAKSI

Ilustrasi 2 (Lanjutan Ilustrasi 1)

- Tara adalah seorang mahasiswi yang kuliah di luar kota. Ayu adalah ibu dari Tara. Keduanya adalah nasabah Bank Hemat. Ayu memiliki saldo Rp 7.000.000, Tara memiliki saldo Rp 45.000.

- Pada suatu hari yang sama dengan hari perbaikan UPS pada Bank Hemat, Tara meng-SMS ibunya untuk mengirimkan uang sebesar Rp. 500.000 karena uang bulanannya hampir habis.

LOG DALAM KEATOMIKAN DAN KONSISTENSI TRANSAKSI (2)

- Ayu yang menerima SMS dari anaknya ini segera menuju ke ATM untuk mentransfer uang ke saldo Tara.

- Sesaat sebelum aliran listrik putus pada jam 09:00, Ayu melakukan transaksi mentranfer uang ke saldo Tara. Sebelum transaksi transfer uang ini "berhasil sepenuhnya", aliran listrik tiba-tiba putus.

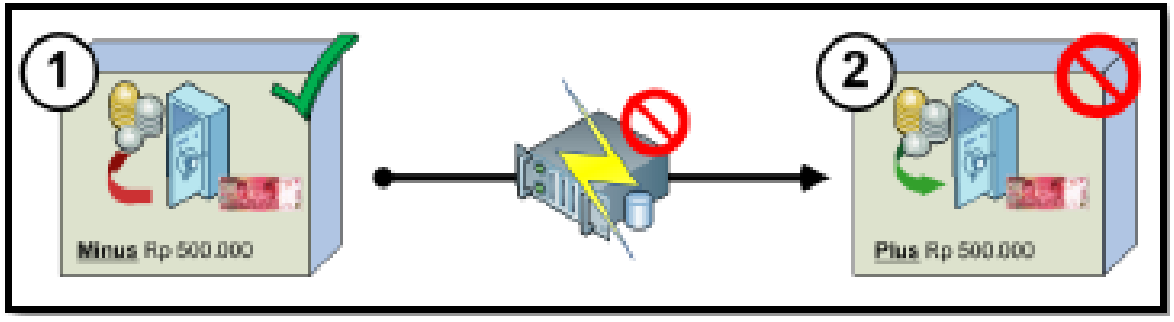
LOG DALAM KEATOMIKAN DAN KONSISTENSI TRANSAKSI (3)

- Pada ilustrasi 2 diatas, seperti yang diketahui berarti ada 2 operasi yang harus dilakukan, yaitu:
  - Mengurangi saldo pada Nasabah Ayu sebanyak Rp 500.000
  - Menambah saldo pada nasabah Tara sebanyak Rp 500.000.

- Skenario di atas menceritakan bahwa transaksi ini tidak "berhasil sepenuhnya", asumsi tidak berhasil sepenuhnya disini adalah : operasi 1 selesai di kerjakan, sedangkan operasi 2 gagal, dikarenakan putusnya aliran listrik.

LOG DALAM KEATOMIKAN DAN KONSISTENSI TRANSAKSI (4)

- Jika kondisi ini dibiarkan begitu saja, akan menyebabkan ketidak konsistenan data, yaitu : Saldo Ayu berkurang Rp 500.000 sedangkan saldo Tara tidak bertambah Rp 500.000.



- Ini berarti transaksi ini belum berhasil sempurna (COMMITED), dan transaksi ini **harus** di batalkan (ROLLBACK), sesuai dengan sifat atomic yang harus dimiliki transaksi.

- Untuk skenario seperti ini, Recovery data menggunakan *Log* ditangani dengan meng-**Undo** transaksi yang gagal tersebut.

LOG DALAM KEATOMIKAN DAN KONSISTENSI TRANSAKSI (5)

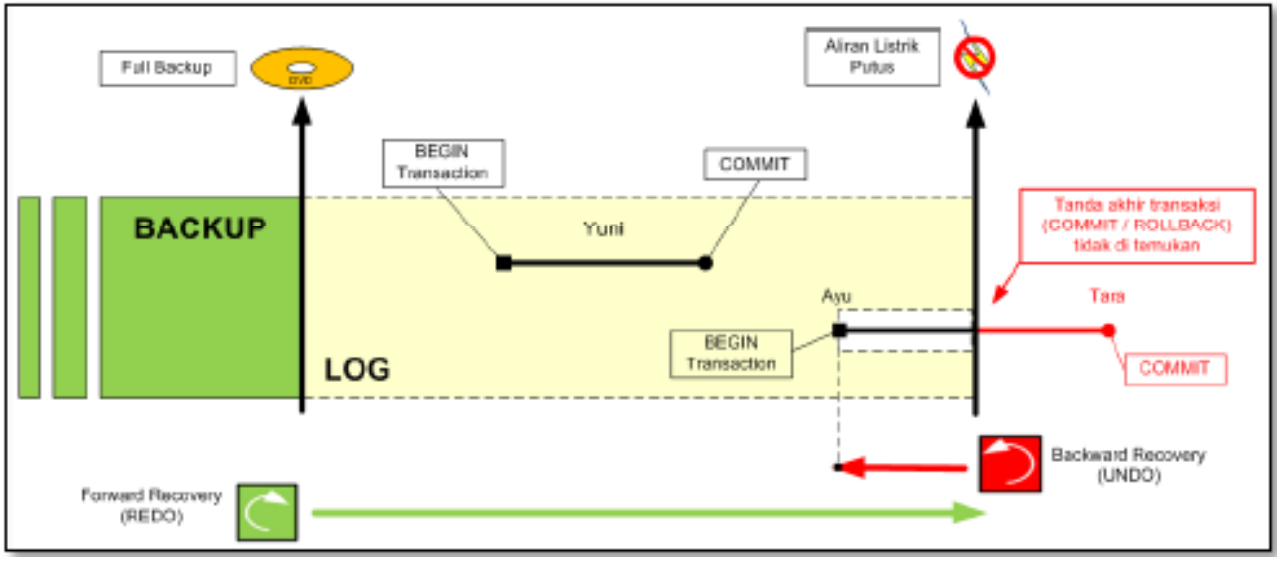
Operasi Undo Transaksi

- Seperti namanya Un-Do(ne) , operasi ini membatalkan perubahan terakhir. Sampai kepada titik ditemukannya catatan awal transaksi <BEGIN> pada *Log.* Operasi ini juga biasa disebut "Backward Recovery".

- Operasi Undo ini akan berjalan saat mekanisme Recovery RDBMS menjalankan operasi Redo dengan menelusuri catatan *Log* transaksi dari awal sampai akhir dan menemukan suatu transaksi yang tidak diakhiri dengan sempurna (tidak ditemukannya klausa COMMIT atau ROLLBACK pada Log).

LOG DALAM KEATOMIKAN DAN KONSISTENSI TRANSAKSI (6)

Ilustrasi Operasi Undo Transaksi :



- Karena sampai pada akhir *Log* tidak di temukan record *Log* yang mengakhiri transaksi transfer uang Ayu, maka secara otomatis transaksi transfer uang tersebut harus di batalkan (ROLLBACK), sehingga uang Ayu kembali utuh karena transaksi telah di batalkan.

LOG PADA VOLATILE STORAGE

Secara umum ada dua jenis media penyimpanan :

- **Media penyimpanan sementara (Volatile Storage).**
  - Data yang ada di media penyimpanan ini hanya ada selama aliran listrik mengalir ke media ini.
  - Jika aliran listrik terputus, maka data yang tersimpan akan hilang.
  - Kelebihan media penyimpanan ini adalah kecepatannya yang sangat tinggi (karena metode pengaksesannya secara langsung).
  - Contoh media penyimpanan jenis ini adalah RAM, Cache dan Register.
- **Media Penyimpanan Permanen (Nonvolatile Storage).**
  - Data yang ada di media penyimpanan ini akan tetap ada walaupun aliran listrik terputus.
  - Kecepatan akses media penyimpanan ini jauh lebih lambat di bandingkan dengan kecepatan akses Volatile Storage.
  - Contoh media penyimpanan jenis ini adalah Hard Disk, Magnetic Tape, Floppy Disk dan Optical Disc.

LOG PADA VOLATILE STORAGE (2)

- Perbedaan umum dari kedua jenis storage ini adalah, Volatile Storage cepat tetapi bersifat sementara sedangkan Non-volatile Storage bersifat tetap tetapi lambat.

- Tiap record pada *Log* menggambarkan operasi tunggal transaksi, dan tiap catatan ini harus di simpan di non-volatile storage agar saat terjadi padamnya aliran listrik datanya tidak hilang dan dapat digunakan untuk recovery.

LOG PADA VOLATILE STORAGE (3)

Ilustrasi Contoh Record Log :

Record No.	Isi Redo Log Record	Status
Record 1	<T_0 begin>	Berhasil
Record 2	<T_0, Saldo Yuni, 5.000.000, 3.000.000>	Berhasil
Record 3	<T_0, commit>	Berhasil
Record 4	<T_1 begin>	Berhasil
Record 5	<T_1, Saldo Ayu, 7.000.000, 6.500.000>	Berhasil
Record 6	<T_1, Saldo Tara, 45.000, 545.000>	Gagal
Record 7	<T_1, commit>	Gagal

- Tiap nilai dari record tersebut di simpan ke Nonvolatile Storage (Hard Disk), jadi,
  1. Record 1 -> simpan ke Hard Disk,
  2. Record 2 -> simpan ke Hard Disk,
  3. Record 3 -> simpan ke Hard Disk,
  4. Record 4 -> simpan ke Hard Disk,
  5. Record 5 -> simpan ke Hard Disk,
  6. Record 6 dan 7 -> gagal karena putusnya aliran listrik.

LOG PADA VOLATILE STORAGE (4)

- Hal ini berakibat sangat buruk dalam hal performa, karena kecepatan akses Hard Disk yang lambat.
- Selain itu hal ini juga sangat tidak efisien, karena ukuran masing-masing *Record Log* yang sangat kecil, sehingga system sangat terbebani dengan operasi tulis ke Hard Disk untuk tiap-tiap *Record Log.*
- Untuk mengatasi masalah performa dan ketidak efisienan tersebut, digunakan Volatile Storage (RAM), dengan mekanisme *mem-buffer*, yaitu mengumpulkan record-record yang berukuran sangat kecil tersebut sampai berukuran cukup besar (biasanya sampai berukuran satu *Logical Block [**box**]***),*** baru dituliskan ke Hard Disk.

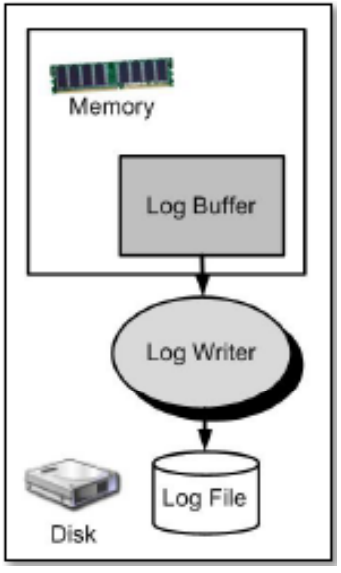
LOG PADA VOLATILE STORAGE (5)

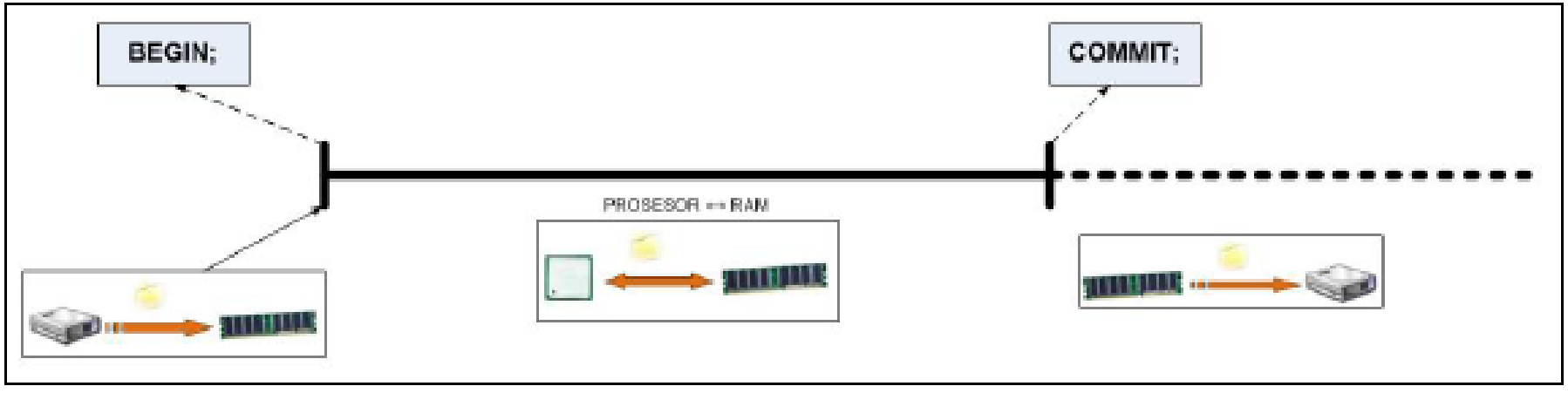
- Mekanisme membuffer *Record Log* pada RAM sampai berukuran cukup besar tersebut memiliki pengecualian, yaitu pada saat suatu transaksi COMMIT.
- Hal ini di sebabkan Sifat Durability yang harus dimiliki transaksi.
- Pengecualian ini dilakukan untuk menghindari isi Buffer Memory yang hilang saat kegagalan system (baik karena putusnya aliran listrik, System Hang atau sebab-sebab Nonfisik lainnya).

LOG PADA VOLATILE STORAGE (6)

- DBMS mengalokasikan tempat khusus di memory untuk fasilitas *pem-buffer-an Log* ini, yang dinamakan *Log Buffer.*
- DBMS membuat sebuah proses khusus yang menuliskan isi Buffer dari Memory ke Hard Disk, yang di namakan Proses *Log Writer.*
- Proses *Log Writer* ini berjalan di latar belakang (background process). Sedangkan untuk *Log* yang dituliskan ke Hard Disk dinamakan LOG FILE.
- Dapat disimpulkan bahwaTransaksi yang belum di COMMIT data-data *Lognya* akan berada pada RAM. baru setelah di-COMMIT, data-data *Log* tersebut dituliskan ke Hard Disk oleh sebuah proses.

LOG PADA VOLATILE STORAGE (7)





DATAFILE

- Pada berbagai Literatur, DATAFILE ini biasa dinamakan DATABASE, karena fungsinya yang menyimpan data sebenarnya.
- Pada System Manajemen Basis Data (DBMS) nama DATABASE sendiri bukanlah sebatas tempat penyimpanan data table, namun juga seluruh data yang dibutuhkan oleh aplikasi DBMS untuk mengelola aplikasinya.
- DATABASE yang berada di Hard Disk bisa disebut DATAFILE. Sedangkan DATABASE yang dibuffer di RAM dinamakan *Database Buffer.*

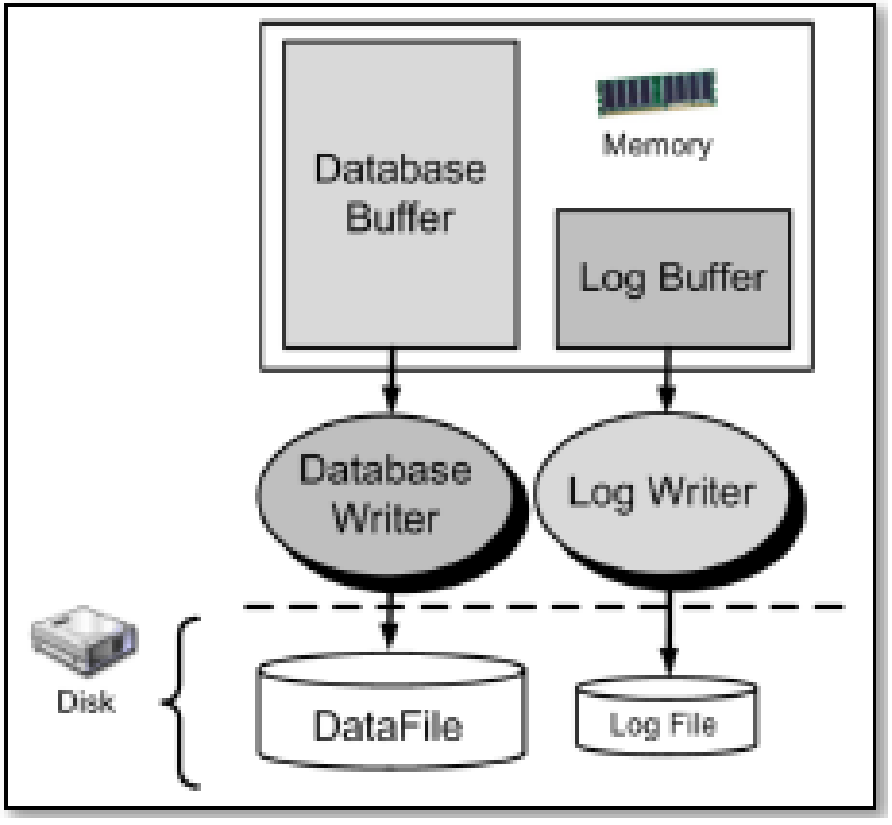
DATAFILE (2)

- **Datafile adalah file fisik yang menyimpan data yang telah di insert kedalam tiap table pada DBMS.**
  - DATAFILE adalah komponen yang berbeda dengan LOG FILE.
  - Perbedaan komponen tersebut dapat di deskripsikan seperti berikut: DATAFILE *adalah tempat menyimpan data, sedangkan* LOG FILE *adalah tempat penyimpanan catatan bahwa data tersebut telah di simpan.*
  - Karena fungsinya sebagai tempat penyimpanan data (bukan tempat penyimpanan catatan), peranan DATAFILE menjadi sangat penting, karenanya, informasi yang tersimpan di dalamnya HARUS dapat diandalkan (*Reliable).*

DATAFILE (3)

- Salah satu cara untuk menjaga ke-*Reliable*-an isi dari DATAFILE ini adalah dengan cara menggunakan catatan pada LOG FILE sebagai acuan dalam proses Recovery.
- Redundancy penyimpanan data antara DATAFILE dan LOG FILE menyebabkan akan semakin besarnya kebutuhan penyimpanan data.
- Hal ini sebanding dengan terjaminnya ke-*Reliable*-an isi data.

DATAFILE (4)


Database Buffer
Memory
Log Buffer
Database Writer
Log Writer
Disk
DataFile
Log File

DATAFILE (5)

- **Datafile menggunakan Buffer untuk mempecepat pengaksesan data oleh banyak user**
  - DATAFILE suatu perusahaan dapat berukuran ratusan, bahkan hingga ribuan gigabyte. Walaupun tidak semua data ini diakses secara bersamaan, namun biasanya pengaksesan banyak data secara bersamaan sering terjadi
  - System Basis Data harus dapat memfasilitasi jika terjadi pengaksesan banyak data secara bersamaan ini, dengan tetap menjaga kecepatan tiap operasi pengaksesan data, sehingga tidak ada user yang menunggu lama untuk melihat/mengubah datanya.

DATAFILE (6)

- Untuk kebutuhan akses multiuser DATAFILE menggunakan Buffer pada RAM.
- Tempat khusus di RAM yang disediakan untuk hal ini, dinamakan *Database Buffer.*
- Pem-*buffer*-an DATAFILE ini bukan hanya untuk operasi membaca data (SELECT) saja, namun juga penulisan data (INSERT/UPDATE/DELETE).
- Hal tersebut berarti perubahan data dilakukan di Database Buffer (di RAM), bukan langsung ke DATAFILE (di Hard Disk).

DATAFILE (7)

- **Penulisan perubahan pada Database Buffer di RAM ke DataFile di Hard Disk terjadi pada saat kapasitas Database Buffer penuh dan setelah Log telah dituliskan sebelumnya.**
  - *Database Buffer* digunakan untuk mempercepat waktu akses data.
  - Data yang tersimpan di *Database Buffer* akan hilang jika terjadi kegagalan system (baik karena putusnya aliran listrik, System Hang atau sebab-sebab Nonfisik lainnya).
  - Oleh karenanya isi *Database Buffer* harus dituliskan ke *DATAFILE* di media penyimpanan tetap (Hard Disk). Penulisan *Database Buffer* ke *DATAFILE* ditangani oleh proses latar belakang yang bernama *Database Writer.*

DATAFILE (8)

- Algoritma yang biasa di gunakan untuk penggantian isi *Database Buffer* adalah *Least Recently Used (LRU).*
- Algoritma LRU yaitu penulisan isi *Database Buffer* yang paling terakhir digunakan/diakses ke Hard Disk untuk memberi ruang kepada *Database Buffer* untuk menyimpan data baru.
- Penulisan data ke DATAFILE tidak boleh dilakukan sebelum record *Log* dituliskanke LOG FILE di media penyimpanan tetap, karena informasi yang terkandung dalam LOG FILE digunakan untuk merekonstruksi keadaan DATAFILE, jika DATAFILE bermasalah.

DATAFILE (9)

- Perbedaan system penulisan DATAFILE dengan LOG FILE adalah isi DATAFILE ini tidak perlu dituliskan saat transaksi di COMMIT, karena catatan-catatan tentang transaksi telah dituliskanke LOG FILE.

- Jika terjadi data transaksi yang telah di COMMIT pada *Database Buffer* hilang (karena kegagalan system), data transaksi yang hilang tersebut masih dapat dapat dipulihkan dengan menelusuri catatannya di LOG FILE.

CHECKPOINT

- Misalkan kita mengubah sebagian jalan cerita dari *ilustrasi 1*, dimana setelah Arus Listrik padam, tidak terjadi kerusakan pada Hard Disk Server dan DATAFILE tidak hilang, yang hilang hanya Buffer pada RAM.
- Karena mekanisme penulisan DATAFILE dari Buffer RAM ke Hard Disk tidak dilakukan saat transaksi COMMIT, berarti saat Server kembali dinyalakan, DATAFILE "belum" dalam kondisi konsisten.
- Untuk memulihkan DATAFILE ke kondisi konsisten, maka dilakukan proses Recovery dengan cara *menelusuri catatannya di LOG FILE, untuk menentukan operasi mana yang harus di Redo dan operasi mana yang harus di Undo.*

CHECKPOINT (2)

- Secara prinsip, operasi Recovery dilakukan dengan metode menelusuri seluruh *Log* untuk menentukan operasi apa yang harus dilakukan. Ada dua kekurangan pada metode ini:
  - Proses penelusuran *Log* membutuhkan waktu yang lama
  - Kebanyakan transaksi yang harus di redone telah menuliskan perubahan datanya ke DATAFILE, walaupun akan mengulangi transaksi tersebut (idempoten). Pengulangan transaksi akan menyebabkan proses recovery lebih lama
- Untuk mengurangi kedua kekurangan beban kerja/waktu tambahan tersebut, digunakan konsep *Checkpoint.*

CHECKPOINT (3)

- *Checkpoint* adalah kejadian penulisan secara paksa dari Buffer di RAM ke File di Hard Disk.
- Aksi-aksi yang harus dilakukan adalah :
  - Menuliskan semua record *Log dari Log Buffer di RAM ke LOG FILE di Hard Disk.*
  - Menuliskan semua DATA dari *Database Buffer di RAM ke DATAFILE di Hard Disk.*
  - Menuliskan record *<checkpoint> ke LOG FILE di Hard Disk.*
- *Checkpoint* dilakukan pada saat tertentu. Biasanya DBMS akan menentukan waktu selang tertentu dalam pengambilan *Checkpoint*, atau jika terjadi event yang di tentukan oleh system (misalnya pada saat banyaknya transaksi sejak pengambilan Checkpoint terakhir telah bejumlah tertentu).

KEGAGALAN SISTEM DAN PROSES RECOVERY

- Pada kegagalan System, baik disebabkan oleh putusnya aliran listrik ataupun karena aplikasi DBMS harus di restart, isi Buffer pada di RAM hilang, baik *Log Buffer* maupun *Database Buffer.*

- Pada saat system restart dan melakukan proses recovery, system akan membuat dua daftar :

  - [daftar-undo] yang berisi transaksi-transaksi yang harus di Undo (Dibatalakan-ROLLBACK), dan
  - [daftar-redo] yang berisi transaksi-transaksi yang harus di Redo (dikerjakan ulang).

KEGAGALAN SISTEM DAN PROSES RECOVERY (2)

- System membuat kedua daftar tersebut dengan urutan sebagai berikut :
  1. kedua daftar tersebut kosong
  2. System menelusuri *Log* pada LOG FILE di Hard Disk dari <*checkpoint*> kedepan

- Rincian urutannya adalah sebagai berikut :
  1. Buat [daftar-redo] dan [daftar-undo] kosong
  2. Baca *Log* dari <*checkpoint*> , masukkan seluruh transaksi yang sedang berjalan pada saat <*checkpoint*> ke dalam [daftar-undo]
  3. Mulai telusuri *Log* ke depan dari <*checkpoint*>
     a. jika ditemukan record *<Ti, begin> pindahkan transaksi Ti tersebut ke [daftar-undo]*
     b. jika ditemukan record *<Ti, commit> atau <Ti, rollback> pindahkan transaksi Ti tersebut ke [daftar-redo]*

KEGAGALAN SISTEM DAN PROSES RECOVERY (3)

4. Setelah kedua daftar tersebut di buat, Ulangi lagi penelusuran *Log* dari <*checkpoint*> ke depan. Ulangi seluruh transaksi yang ada di [daftar-Redo], sampai akhir dari *Log*.

5. Setelah seluruh transaksi yang ada di [daftar-redo] telah di ulangi, telusuri *Log* dari belakang (dari akhir Log) batalkan seluruh perubahan yang terjadi dari transaksi yang berada di [daftar-undo] sampai ditemukannya record Log <begin> untuk transaksi tersebut.