PENGONTROLAN BERBASIS KOMPUTER

1. Security Database

Authorization
Pemberian hak akses yang mengizinkan sebuah subyek mempunyai akses secara legal terhadap sebuah sistem atau obyek.

Subyek → *user* atau program
Obyek → *database table*, *view*, *application*, *procedure*, atau obyek lainnya yang dibuat di dalam sebuah sistem

Jenis-jenis hak akses (*privileges*)
- Penggunaan nama database yang spesifik
- *Select* (*retrieve*) data
- Membuat tabel (obyek lainnya)
- *Update* data, *delete* data, *insert* data (bisa untuk kolom-kolom tertentu)
- Menghasilkan *output* yang tidak terbatas dari operasi *query* (*user* tidak dibatasi untuk mengakses *record* tertentu)
- Menjalankan prosedur khusus dan utilitas program
- Membuat database
- Membuat (dan memodifikasi) DBMS *user identifiers* dan *authorized identifiers* jenis lainnya
- Anggota dari sebuah kelompok atau kelompok-kelompok user

Views (Subschemas)
Hasil yang dinamik dari satu atau lebih operasi relasi yang beroperasi pada relasi dasar untuk menghasilkan relasi lainnya. *View* merupakan *virtual relation* yang tidak secara nyata ada di dalam sebuah database, tetapi dihasilkan atas permintaan *user* secara khusus.

Backing Up
Proses yang secara periodik menyalin database dan menjurnal (dan memprogram) ke dalam media penyimpanan *offline*

Journaling
Proses penyimpanan dan pemeliharaan sebuah jurnal atau *log* seluruh perubahan terhadap database agar dapat me*recover* secara efektif jika terjadi kegagalan.

Checkpointing
Titik temu sinkronisasi antara database dan transaksi *log file*. Seluruh data yang disimpan di tempat sementara akan disimpan di media penyimpanan kedua.

Integrity
Pengontrolan integritas juga membantu memelihara sistem database yang aman dengan mencegah data dari invalid

Encryption
Penyandian (*encoding*) data dengan menggunakan algoritma khusus yang merubah data menjadi tidak dapat dibaca oleh program apapun tanpa mendeskripsikannya.

2. Concurrency

Hampir semua DBMS adalah sistem multi user. Sistem seperti ini memerlukan mekanisme pengontrolan konkuren. Tujuan dari mekanisme ini adalah untuk menjamin bahwa transaksi-transaksi yang konkuren tidak saling menggangu operasinya masing-masing.

Terdapat beberapa masalah yang akan timbul dalam menjalankan transaksi-transaksi yang konkuren. Tiga masalah yang umum adalah :

1. Masalah kehilangan modifikasi
2. Masalah modifikasi sementara
3. Masalah analisis yang tidak konsisten

Masalah Kehilangan Modifikasi

Transaksi A	Waktu	Transaksi B
=	↓	=
Baca R	t1	=
=	↓	=
=	t2	Baca R
=	↓	=
Modifikasi R	t3	=
=	↓	=
=	t4	Modifikasi R
=	↓	=

- Transaksi A membaca R pada t1, transaksi B membaca R pada t2. Transaksi A memodifikasi R pada t3.
- Transaksi B memodifikasi record yang sama pada t4.
- Modifikasi dari transaksi A akan hilang karena transaksi B akan memodifikasi R tanpa memperhatikan modifikasi dari transaksi A pada t3.

Masalah Modifikasi Sementara

Masalah ini timbul jika transaksi membaca suatu record yang sudah dimodifikasi oleh transaksi lain tetapi belum terselesaikan (uncommited), terdapat kemungkinan kalau transaksi tersebut dibatalkan (rollback).

Transaksi A	Waktu	Transaksi B
-	↓	-
-	t1	modifikasi R
-	↓	-
Baca R	t2	-
-	↓	-
-	t3	rollback
-	↓	-

- Transaksi B memodifikasi record R pada t1
- Transaksi A membaca R pada t2
- Pada saat t3 transaksi B dibatalkan
- Maka transaksi A akan membaca record yang salah

Transaksi A	Waktu	Transaksi B
-	↓	-
-	t1	Modifikasi R
-	↓	-
Modifikasi R	t2	-
-	↓	-
-	t3	Rollback
-	↓	-

- Pada waktu t2 transaksi A memodifikasi R
- Karena transaksi B dibatalkan pada waktu t3, maka transaksi A memodifikasi record yang salah.

Masalah Analisis yang Tidak Konsisten

Nilai 1 = 40 | Nilai 2 = 50 | Nilai 3 = 30

Transaksi A	Waktu	Transaksi B
-	↓	-
Baca nilai 1(40)	t1	-
juml = 40		-
-	↓	-
Baca nilai 2 (50)	t2	-
juml = 90		-
-	↓	-
-	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)	t8	
juml = 110 (bukan 120)		
-	↓	

- Transaksi A menjumlahkan nilai 1, nilai 2 dan nilai 3
- Transaksi B → nilai 1 +10 ; nilai 3 - 10
- Pada waktu t8, transaksi A membaca nilai yang salah karena nilai 3 sudah dimodifikasi menjadi 20 (transaksi B sudah melakukan commit sebelum transaksi A membaca nilai 3)

Ket :

- *Commit adalah operasi yang menyatakan bahwa suatu transaksi sudah terselesaikan/ sukses (successfull end-of-transaction).*

- *Rollback adalah operasi yang menyatakan bahwa suatu transaksi dibatalkan (unsuccessfull end-of-transaction).*

Locking

Locking adalah salah satu mekanisasi pengontrol konkuren. Konsep dasar : pada saat suatu transaksi memerlukan jaminan kalau record yang diinginkan tidak akan berubah secara mendadak, maka diperlukan kunci untuk record tersebut. Fungsi kunci (lock) adalah menjaga record tersebut agar tidak dimodifikasi transaksi lain.

Cara kerja dari kunci :

1. Pertama kita asumsikan terdapat 2 macam kunci :
   - Kunci X : kunci yang eksklusif.
   - Kunci S : kunci yang digunakan bersama-sama.

2. Jika transaksi A menggunakan kunci X pada record R, maka permintaan dari transaksi B untuk suatu kunci pada R ditunda, dan B harus menungggu sampai A melepaskan kunci tersebut.

3. Jika transaksi A menggunakan kunci S pada record R, maka :\
   a. Bila transaksi B ingin menggunakan kunci X, maka B harus menunggu sampai A melepaskan kunci tersebut.
   b. Bila transaksi B ingin menggunakan kunci S, maka B dapat menggunakan kunci S bersama A.

Tabel Kunci :

	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

X = kunci X
S = kunci S

N = No

Y = Yes

4. Bila suatu transaksi hanya melakukan pembacaan saja, secara otomatis ia memerlukan kunci S → baca (S)

   Bila transaksi tersebut ingin memodifikasi record maka secara otomatis ia memerlukan kunci X → memodifikasi (X)

   Bila transaksi tersebut sudah menggunakan kunci S, setelah itu ia akan memodifikasi record, maka kunci S akan dinaikan ke level kunci X.

5. Kunci X dan kunci S akan dilepaskan pada saat synchpoint (synchronization point). Synchpoint menyatakan akhir dari suatu transaksi dimana basis data berada pada state yang konsisten. Bila synchpoint ditetapkan maka :
   - Semua modifikasi program menjalankan operasi commit atau rollback.
   - Semua kunci dari record dilepaskan.

Masalah Kehilangan Modifikasi

Transaksi A	Waktu	Transaksi B
-	↓	-
Baca R	t1	-
(kunci S)	↓	-
-	t2	Baca R
-	↓	(kunci S)
Modifikasi R	t3	-
(kunci X)	↓	-
Tunggu	↓	-
:	t4	Modifikasi R
:	↓	(kunci X)
:	↓	Tunggu
:	↓	:
:	↓	:
Tunggu	↓	Tunggu

- Pada waktu t3, transaksi A memerlukan kunci X, maka transaksi A harus menunggu sampai B melepaskan kunci S.
- Transaksi B juga harus menunggu pada t4
  Maka tidak akan ada yang kehilangan modifikasi, tetapi terdapat keadaan baru yaitu DEADLOCK.

Masalah Modifikasi Sementara.

Transaksi A	Waktu	Transaksi B
-	↓	-
-	t1	modifikasi R
-	↓	(kunci X)
Baca R	t2	-
kunci (S)	↓	-
Tunggu	↓	-
:	t3	Synchpoint
:	↓	(kunci X dilepas)
Tunggu	↓	-
Baca R kembali	T4	-
(kunci S)	↓	-

Transaksi A	Waktu	Transaksi B
	↓	
-	t1	Modifikasi R
-	↓	(kunci X)
Modifikasi R	t2	-
kunci (X)	↓	-
Tunggu	↓	-
:	t3	Synchpoint
:	↓	(kunci X dilepas)
Tunggu	↓	-
Modifikasi R	t4	-
(kunci X)	↓	-

- Transaksi A pada t2 tidak dapat dijalankan langsung, tetapi harus menunggu sampai B melepas kunci X.
- Bila B sudah mencapai synchpoint, maka kunci X dilepaskan dan A dapat meneruskan prosesnya.
- Maka transaksi A tidak akan terjasi kesalahan dalam membaca, karena sudah mencapai synchpoint.

Masalah Analisa yang Tidak Konsisten

Nilai 1 = 40 Nilai 2 = 50 Nilai 3 = 30

Transaksi A	Waktu	Transaksi B
-	↓	-
Baca nilai 1 (40)	t1	-
(kunci S)	↓	-
jml = 40	↓	-
-	↓	-
Baca nilai 2 (50)	t2	-
(kunci S)	↓	-
jml = 90	↓	-
-	t3	Baca nilai 3 (30)
-	↓	(kunci S)
-	↓	-
-	t4	Modifikasi nilai 3
-	↓	(kunci X)

-	↓	30 → 20
-	↓	-
-	t5	Baca nilai 1 (40)
-	↓	(kunci S)
-	↓	-
-	t6	Modifikasi nilai 1
-	↓	(kunci X)
-	↓	Tunggu
Baca nilai 3 (20)	t7	:
(kunci S)	↓	:
Tunggu	↓	Tunggu
		:

- Transaksi B pada t6 tidak diijinkan, karena memerlukan kunci X maka B harus menunggu sampai A melepaskan kunci S kepada nilai 1.
- Pada t7 transaksi A juga tidak dapat langsung dilaksanakan, karena B menggunakan kunci X pada nilai 3. Maka A harus menunggu B melepaskan kunci X pada nilai 3.
- Transaksi A akan membaca nilaiyang benar, tapi timbul masalah baru yaitu DEADLOCK.

Time Stamping

Salah satu alternatif mekanisme pengawasan konkuren yang dapat menghilangkan masalah deadlock adalah TIME STAMPING. Dalam skema ini tidak ada kunci yang digunakan sehingga tidak ada deadlock yang muncul. Time stamping untuk sebuah transaksi aksi merupakan suatu tanda pengenal yang unik yang menunjuk waktu mulai relatif dari transaksi.

Time stamp dapat berupa pembacaan pada kinci internal pada waktu transaksi dimulai, dapat berupa nilai dari suatu penunjuk logical yang dapat bertambah setiap kali suatu transaksi baru dimulai. Dalam hal ini nilai time stamp dari setiap transaksi adalah unik dan menunjukkan bagaimana lamanya transaksi tersebut . Pengaruh dari time stamping adalah menentukan suatu urutan serial transaksi.

Setiap item data terdiri dari sebuah lead time stamp yang memberikan time stamp transaksi terakhir untuk membawa item dan sebuah write time stamp yang memberikan transaksi terakhir untuk menuliskan / memperbaharui item.

Masalah dapat timbul dengan time stamping :
1. Suatu transaksi memerintahkan untuk membaca sebuah item yang sudah di-update oleh transaksi yang belakangan
2. Suatu transaksi memerintahkan untuk menulis sebuah item yang nilainya sudah dibaca / ditulis oleh transaksi yang belakangan

3. Recovery

Recovery Facilities
Sebuah DBMS sebaiknya menyediakan fasilitas-fasilitas berikut ini untuk membantu recovery :

- *Backup mechanism*
  melakukan backup secara periodik terhadap database yang ada
- *Logging facilities*
  Mencatat transaksi-transaksi dan perubahan-perubahan yang terjadi terhadap database. DBMS memelihara file khusus yang disebut Log (Journal) yang menyediakan informasi mengenai seluruh perubahan yang terjadi pada database.
- *Checkpoint facility*
  Mengizinkan update terhadap database yang akan menjadi database yang permanen
- *Recovery manager*
  Mengizinkan sistem untuk *restore* database ke keadaan sebelum terjadi kerusakkan

Recovery Techniques
Prosedur recovery yang digunakan tergantung dari kerusakkan yang terjadi pada database. Terdapat 2 kasus kerusakkan :

1. Jika database rusak secara fisik seperti : *disk head crash* dan menghancurkan database, maka yang terpenting adalah melakukan *restore backup database* yang terakhir dan mengaplikasikan kembali operasi-operasi *update* transaksi yang telah *commit* dengan menggunakan *log file*. Dengan asusmsi bahwa *log file*-nya tidak rusak.
2. Jika database tidak rusak secara fisik tetapi menjadi tidak konsisten, sebagai contoh : sistem *crashed* sementara transaksi dieksekusi, maka yang perlu dilakukan adalah membatalkan perubahan-perubahan yang menyebabkan database tidak konsisten. Mengulang beberapa transaksi sangat diperlukan juga untuk meyakinkan bahwa perubahan-perubahan yang dilakukan telah disimpan di dalam *secondory storage*. Disini tidak perlu menggunakan salinan *backup* database, tetapi dapat me-*restore database* ke dalam keadaan yang konsisten dengan menggunakan *before-* dan *after-image* yang ditangani oleh *log file*.

Teknik *recover* berikut ini dilakukan terhadap situasi dimana database tidak rusak tetapi database dalam keadaan yang tidak konsisten.

1. *Deferred Update*
   Update tidak dituliskan ke database sampai sebuah transaksi dalam keadaan *commit*. Jika transaksi gagal sebelum mencapai keadaan ini, transaksi ini tidak akan memodifikasi database dan juga tidak ada perubahan-perubahan yang perlu dilakukan.

   Penulisan dilakukan secara initial hanya terhadap *log* dan *log record* yang digunakan untuk *actual update* terhadap database. Jika sistem gagal, sistem akan menguji log dan menentukan transaksi mana yang perlu dikerjakan ulang, tetapi tidak perlu membatalkan semua transaksi.

2. *Immediate Update*
   Update diaplikasikan terhadap database tanpa harus menunggu transaksi dalam keadaan commit. Update dapat dilakukan terhadap database setiap saat setelah *log record* ditulis. Log dapat digunakan untuk membatalkan dan mengulang kembali transaksi pada saat terjadi kerusakkan.