PROTEKSI DATA (DATABASE CONTROL)

DBMS pada umumnya memiliki fasilitas proteksi data, yaitu fasilitas yang bertujuan untuk melindungi data dari berbagai resiko yang mungkin terjadi dan membawa dampak dalam basis data

Berbagai kemungkinan yang diantisipasi oleh fasilitas proteksi data adalah :

- Gangguan listrik
- Kerusakan disk
- Kesalahan perangkat lunak yang akan menyebabkan data dalam kondisi tidak konsisten
- Pengaksesan oleh user yang tidak berwenang. Untuk menghindari sabotase terhadap basis data
- Akses yang konkuren oleh user maupun aplikasi pada waktu yang bersamaan sehingga dapat menyebabkan data tidak konsisten

Untuk memproteksi data terhadap segala macam kemungkinan, DMBS menyediakan kontrol untuk :

1. Security
2. Integrity
3. Recovery
4. Concurrency

❖ SECURITY DATA

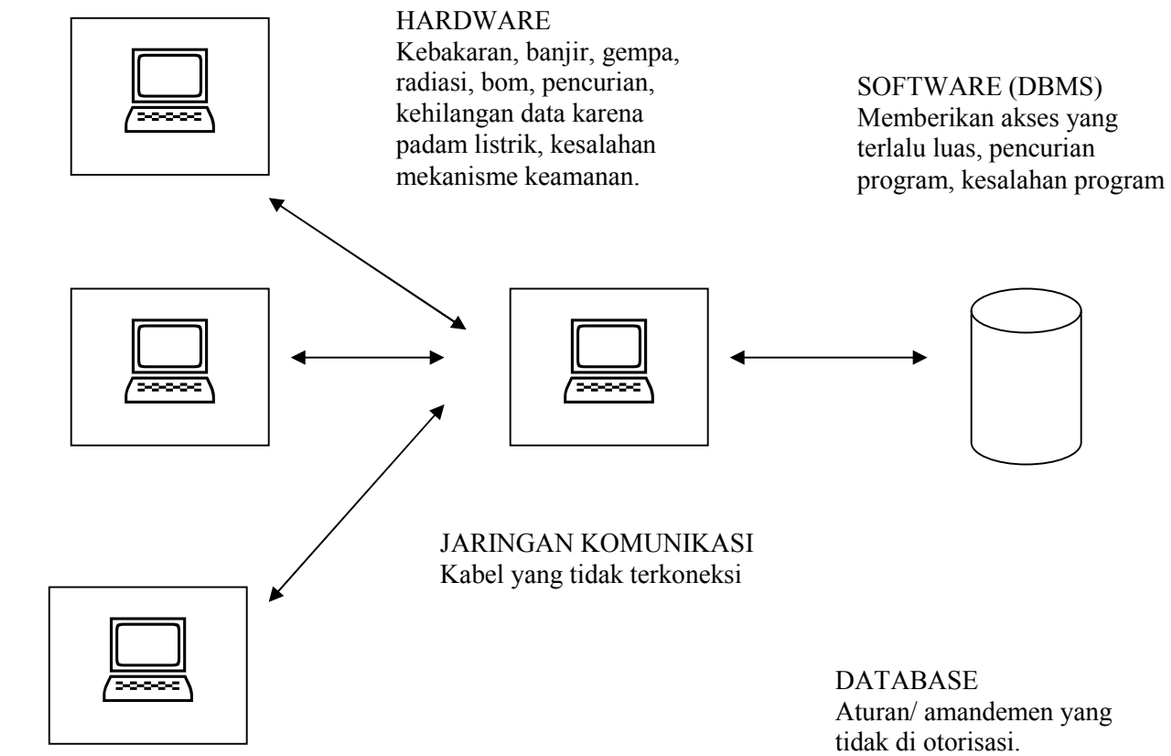
Security merupakan suatu proteksi terhadap pengrusakan data dan pemakaian data oleh user yang tidak berwenang. Organisasi harus dapat mengidentifikasi masalah keamanan yang mungkin mengganggu jalan operasional basis data.
Penyalahgunaan basis data dapat dikategorikan sebagai tindakan yang disengaja maupun yang tidak sengaja.

Untuk kategori yang tidak disengaja dapat disebabkan oleh :

1. Kerusakan selama proses transaksi
2. Anomali yang disebabkan oleh akses basis data yang konkuren
3. Anomali yang disebabkan oleh pendistribusian data pada beberapa komputer
4. Kesalahan logika yang dapat mengancam kemampuan transaksi untuk mempertahankan konsistensi basis data

Untuk kategori kedalam tindakan yang disengaja antara lain disebabkan oleh :

1. Pengambilan data/pembacaan data oleh user yang tidak berwenang
2. Pengubahan data oleh user yang tidak berwenang
3. Penghapusan data oleh user yang tidak berwenang



END USER
Menggunakan hak akses user alin
Melihat dan menutup data yang tidak diotorisasi
Staff yang tidak di training
Pemasukan data yang dilakukan oleh hacker
Virus, pemerasan

PROGRAMMER/ OPERATORS
Membuat password
Membuat program yang tidak aman
Staff yang tidak ditraining
Kebijakan keamanan dan prosedur
Pemogokan staff

DBA
Kebijaksanaan Keamanan dan prosedur

Gambar 1
Identifikasi Masalah

Pada gambar 1, menjelaskan identifikasi masalah yang dapat menggangu kinerja sistem komputer pada suatu organisasi.

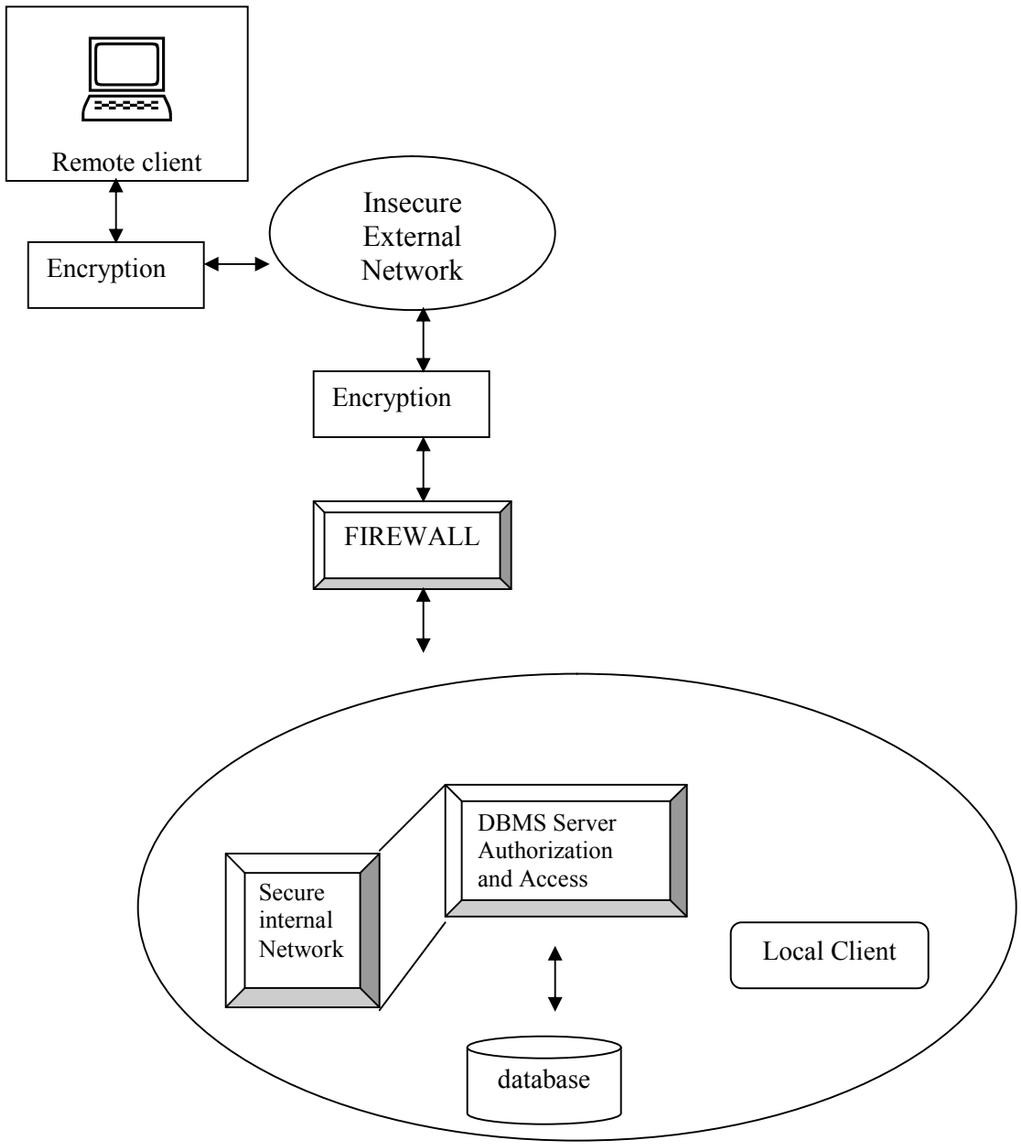
Penyalahgunaan database dapat dikategorikan sebagai tindakan yang disengaja maupun tidak disengaja.

Kategori Tindakan yang tidak disengaja	Tindakan yang disengaja
- Kerusakan selama proses transaksi - Anomali yang disebabkan oleh akses database yang konkuren - Anomali yang disebabkan oleh pendistribusian data pada beberapa komputer - Logika Error yang mengancam kemampuan transaksi untuk mempertahankan konsistensi data	- Pengambilan data / pembacaan data oleh user yang tidak berwenang - Pengubahan data oleh user yang tidak berwenang - Penghapusan data oleh user yang tidak berwenang

Untuk mengatasi masalah ini, security harus dilakukan pada beberapa tingkatan :

1. FISIKAL
   Menempatkan sistem komputer pada lokasi yang aman secara fisik dari serangan yang dapat merusak.
2. MANUSIA
   Wewenang pemakai harus dilakukan dengan hati-hati untuk mengurangi kemungkinan adanya manipulasi oleh pemakai yang tidak berwenang
3. SISTEM OPERASI
   Walaupun sistem databasenya aman, kelemahan pada sistem operasi memungkinkan pengaksesan data oleh pihak tidak berwenang, karena hampir seluruh jaringan sistem database menggunakan akses jarak jauh melalui terminal/jaringan.
4. SISTEM DATABASE
   Beberapa pemakai yang berwenang dalam sistem database mungkin hanya boleh mengakses sebagian databasenya, yang lainnya hanya boleh melihat dan menggunakan tanpa boleh mengubahnya. Hal ini harus dapat dilakukan dalam sistem tersebut.

Jika dilihat dari lingkungan komputer yang multi user, apa saja yang harus dilakukan oleh sebuah organisasi untuk mencegah hal mengenai keamanan datanya.



Keamanan data dapat dilakukan dengan otorisasi, tabel view, backup data dan recovery, integritas data, enkripsi.

Berkaitan dengan security data, pada DBMS sering dijumpai istilah privilege (hak istimewa) dan Authority (Wewenang)

AUTHORITY
(Pemberian Wewenang)

Merupakan pemberian wewenang atau hak istimewa untuk mengakses sistem ataupun objek dari sistem.
Kontrol otorisasi dapat dibangun pada perangkat lunak dan tidak hanya mengontrol sistem atau obyek yang dapat diakses oleh pengguna, tapi juga bagaimana pengguna menggunakanny. Kontrol otorisasi sama dengan kontrrol akses.

Sistem administrasi biasanya bertanggung jawab untuk memberikan hak akses dengan membuat account penggunanya untuk penggunaan sistem komputernya, tetapi pengguna tidak mempunyai otorisasi untuk menggunakan DBMS sebelum mendapatkan hak akses yang diberikan oleh DBA.

Pada saat pengguna telah diberikan hak akses untuk menggunakan DBMS, bermacam-macam hak istimewa akan dapat digubnakan secara otomatis. Hak istimewa / privileges adalah pemberian hak akses kepada pengguna untuk menyelesaikan tugas-tugas mereka.

VIEW

Konsep view adalah cara yang diberikan pada seseorang pemakai untuk mendapatkan model database yang sesuai dengan kebutuhan perorangan. View dapat menyembunyikan data yang tidak digunakan/tidak perlu di;ihat oleh pemakai tsb.

Kemampuan ini menguntungkan dalam penangan data yang semakin mudah dan meningkatkan keamanan.

Database relasional membuat pengamanan pada level :

- Relasi
  Seseorang pemakai diperbolehkan atau tidak diperbolehkan mengakses langsung suatu relasi.
- View
  Seseorang pemakai diperbolehkan atau tidak diperbolehkan mengakses data yang terdapat pada view

Seseorang pemakai dapat memiliki beberapa wewenang atas beberapa bagian dari database yaitu :

- Read Authorization :
  Data dapat dibaca tapi tidak boleh dimodifikasi.
- Insert Authorization :
  Pemakai boleh menambahan data baru, tetapi tidak dapat memodifikasi data yang sudah ada.
- Update Authorization :
  Pemakai boleh memodifikasi tetapi tidak dapat menghapus data
- Delete Authorization :
  Pemakai boleh menghapus data

Sebagai tambahan dari otoritas untuk mengakses data, seorang pemakai juga diberikan wewenang untuk memodifikasi database anatara lain :

- Index Authorization :
  Pemakai boleh membuat dan menghapus index
- Resource Authorization :
  Mengizinkan pembuatan relasi-relasi baru.
- Alteration Authorization :
  Mengizinkan penambahan/penghapusan atribut dalam satu relasi.
- Drop Authorization :
  Pemakai boleh menghapus relasi yang ada

Perintah untuk pemberian dan pengambilan wewenang dengan SQL, berturut - turut adalah : GRANT dan REVOKE

Wewenang pemakai untuk dapat terlibat atau masuk kedalam Sistem Basis Data :

```
GRANT (CONNECT| RESOURCE | DBA) TO user_name | PUBLIC
```

Untuk mencabut hak

```
REVOKE (CONNECT| RESOURCE | DBA) FROM user_name | PUBLIC
```

Terdapat 3 Tingkat otoritas user dalam mengakses database

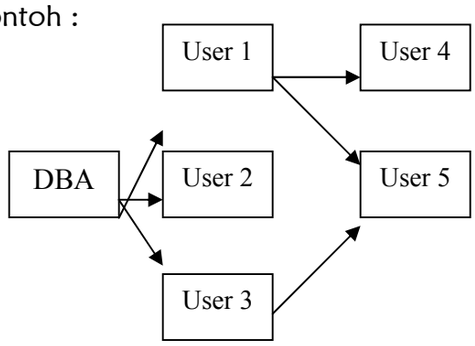
1. CONNECT
   - Dapat menampilkan dan merubah data
   - Membuat View dari table/ralasi yang diijinkan
   - Dapat menggunakan pernyataan ALTER, DROP TABLE, INDEX
   - Membeikan ijin kepada user lain untuk menggunakan table/relasi yang diijinkan.
2. RESOURCE
   - Merubah struktur table
   - Membuat table baru, view baru
   - Hak yang sama dengan CONNECT
3. DBA
   - Memberikan grant dengan level privilege kepada user
   - Melakukan DROP DATABASE
   - Hak yang sama dengan RESOURCE

Perintah pemberian wewenang user untuk menggunakan table dalam database
GRANT {privilege list} ON { nama_relasi atau nama_view} TO user_list [WITH GRANT OPTION]

WITH GRANT OPTION menyebabkan user dapat memberi wewenang kepada user lain atas table yang diijinkan

Perintah untuk menghapus wewenang user dalam menggunakan database :
REVOKE {privilege list} ON { nama_relasi atau nama_view} FROM user_list

Untuk memudahkan dalam menentukan user yang akan diberikan wewenang dapat dibuat terlebih dahulu GRAPH OTORISASI :
Contoh :



Skema Database Pelayanan Akademik :
MHS {npm,nama,alamat,kelas,sks,ipk}
MKUL {kdmk, namamk,ket}
NILAI{npm,kdmk,uts,uas,prak}

User yang terlibat :
Tingkat Resource : User 1 → Indah, User 2 → Tuti, User 3 → Ali
Tingkat Connect : User 4 → Budi, User 5 → Adi

Pernyataan SQL :
DBA :
GRANT RESOURCE TO indah, tuti, ali;
GRANT CONNECT TO budi, adi;

Pernyataan ini mengakibatkan Indah, Tuti, Ali dapat terlibat dalam system basis data dengan tingkatan RESOURCE, sedangkan budi, adi dengan tingkatan CONNECT, tetapi semua user belum dapat memakai/mengakses table yang ada dalam system basis data.
Pernyataan SQL untuk memberi wewenang user untuk akses table dalam system basis data :
DBA :

GRANT select,insert,delete,update ON MHS,MKUL TO indah,ali WITH GRANT OPTION;
GRANT all ON NILAI TO tuti;
GRANT select,insert,delete,update,index ON NILAI to ali WITH GRANT OPTION;

Indah :

GRANT select,update(alamat) ON MHS TO budi;
GRANT select,insert ON MKUL TO adi;

Ali :

GRANT select,update, delete ON NILAI TO adi;

Jika user 1 mempunyai hak untuk memberikan authoritas kepada user 3 didefinisikan pada perintah GRANT yang disertai WITH GRANT OPTION maka User 1 juga berhak untuk memberikan authoriats kepada user lain, misalnya User3.

Pernyataan untuk mencabut wewenang :

REVOKE digunakan untuk mengambil wewenang yang dimiliki oleh pemakai.
Syntax :
REVOKE (privilege list> ON <nama relasi/view> FROM <pemakai>
Contoh :
REVOKE SELECT ON MKUL FROM budi;
REVOKE UPDATE(alamat) ON MHS FROM adi;

Privilege list dapat berupa bentuk tambahan authoritas seperti : READ, INSERT, DROP, DELETE, INDEX, ALTERATION, dan RESOURCE

Wewenang update dapat digunakan untuk seluruh atribut atau sebagian saja dan pemberian wewenang update selalu diikuti dengan nama atribut yang dapat diupdate.

BACKUP

Backup merupakan proses secara periodik yang mengambil duplikat dari database dan melakukan logging file (dan mungkin juga program) ke media penyimpanan eksternal.

DBMS harus memiliki fasilitas backup dengan database recovery-nya dilanjutkan dengan kesalahannya. Dengan menyimpan pada tempat yang aman sehingga jika terjadi kesalahan pada data maka data dapat diambil dari data duplikat yang terakhir.

Selain melakukan backup data penting juga melakukan pen-jurnal-an dimana proses ini menyimpan dan mengatur log file (jurnal) dari semua perubahan yang dibuat oleh database utnuk recovery yang nantinya akan efektif jika terjadinya kesalahan.

JOURNALING/LOG FILE

Untuk membuat transaksi database track pada tempatnya, DBMS menggunakan log ataupun jurnal yang berisi informasi mengenai perubahan yang terjadi pada database.

Jurnal mungkin akan berisi data :

- Record transaksi;
    - Identifikasi dari record
    - Tipe dari record jurnal (transaksi start, insert, update, delete, abort, commit)
    - Item data sebelum perubahan (operasi update dan delete)
    - Item data sesudah perubahan (operasi insert dan update)
    - Informasi manajemen jurnal seperti pointer sebelum dan record jurnal selanjutnya untuk semua transaksi.

- Record checkpoint

CHECKPOINT

Informasi pada jurnal digunakan untuk memulihkan database dari kegagalan yang dilakukan. Satu kesulitan dengan skema ini adalah ketika kegagalan terjadi, tidak diketahui seberapa jauh jurnal untuk kembali mencarinya dan berakhir dengan melakukan transaksi REDO. Untuk membatasi pencarian dan pemrosesan secara teratur digunakan tehnik checkpoint/pos pemeriksaan.

ENCRYPTION

Merupakan pengkodean data dengan algoritma khusus algoritma khusus sehingga data yang dikirim tidak dapat dibaca sebelum adanya proses deskripsi.
Salah satu algoritma yang digunakan untuk proses enkripsi ini adalah DES (Data Encryption Standard).

❖ INTEGRITAS DATA

Salah satu karakteristik sistem informasi yang baik adalah kemampuannya memberikan informasi yang akurat dan tepat waktu. Keakuratan informasi hanya dapat diperoleh jika didukung perancangan dan implementasi database yang handal. Integrity di dalam istilah basis data berarti memeriksa keakuratan dan validasi data.

Oleh karena itu database harus menjamin integritas (keutuhan) data yang disimpannya. Harus dijamin agar perubahan terhadap basis data yang dilkaukan user yang berhak tidak menghasilkan ketidakkonsistenan data. Harus dijamin pula gara database tidak mengalami kerusakan secara sengaja.
Untuk itu dalam database dikenal dengan aturan integritas (integrity constraints) yang mengatur definisi dan modifikasi terhadap database sehinggan menjamin integritas database tersebut.
Terdapat beberapa jenis aturan integritas (integrity constraints) yang menjamin konsistensi dan integritas database, yaitu :

1. Aturan integritas entitas (Entity Integrity Constraints)
2. Aturan Domain (Domain Constraints)
3. Aturan integritas refensial (Referential Integrity Constraints)
4. Aturan berbasis atribut (Attribute-based Constraints) dan Aturan berbasis Record (Tuple Based Constraints)
5. Pernyataan (Assertions)
6. Pemicu (Trigger)

- **Aturan Integritas Entitas (*Entity Integrity Constraints*)**

Aturan ini diterapkan dengan cara mendeklarasikan kunci primer (*primary key*) untuk setiap entitas agar dijamin tidak ada baris-baris dalam table yang memiliki nilai yang sama (duplikat record).

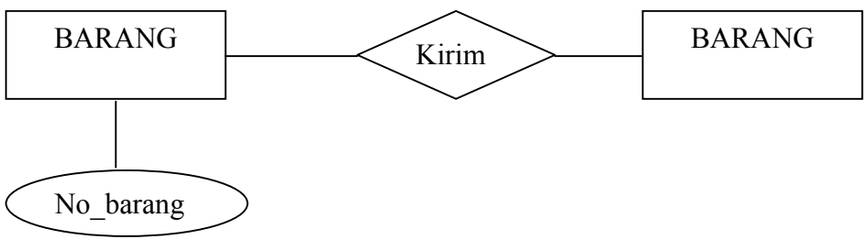
- **Aturan Domain (*Domain Integrity*)**

Domain merupakan kumpulan tipe data dan jangkauan nilai yang diperbolehkan pada atribut sebuah relasi. Untuk setiap atribut yang didefinisika dalam suatu relasi harus ditentukan definisi domain atribut tersebut. Definisi domain dari sebuah atribut akan mencakup : tipe data, panjang, format, jangkauan, nilai yang memungkinkan, keunikan dan kemungkinan data *null*. Aturan-aturan yang dirumuskan dalam definisi domain tersebut dinamakan ***aturan domain (domain constraints)***.

Atribut dari sebuah relasi perlu memiliki definisi domain dengan tujuan :

a. Sistem dapat mengcek keakuratan data yang dimasukan ke basis daa.
b. Menguji kueri apakah perbandingan dalam criteria yan dilakukan sudah sesuai.

Contoh :



Gambar …… ER Diagram

Gambar diatas merupakan ER Diagram yang belum lengkap, maka definisikan domain No_Barang.

Nama Artribut : No_barang
Arti : No barang yang ada di took
Tipe data : Numerik
Format : 0 desimal
Jangkauan : 1-99.999
Keunikan : Harus unik
Dukungan Null : Not Null

- **Aturan integritas refensial (Referential Integrity Constraints)**

Aturan Integritas Referensial adalah aturan yang mengatur kebeneran referensi dari satu obyek ke obyek lain dalam database.

Contoh :

Tabel BARANG

No_barang	Nama_barang	No_pemasok
100	Flash Disk	10
110	Printer	10
120	Monitor	20
130	Kabel data	30

Tabel PEMASOK

No_pemasok	Nama_pemasok
10	Godi Computer
20	Jaya Komputer
30	Astro

Setiap barang dalam table BARANG memiliki pemasok dan setiap pemasok dapat memasok lebih dari satu barang. Dalam table BARANG, field No_pemasok adalah kunci tamu (foreign key)

Aturan referensial akan menjamin bahwa setiap nilai dalam kolom kunci tamu dari table yang merujuk (me-refer) harus tepat sama dengan nilai dalam kolom kunci primer dari table yang dirujuk direfer), atau jika tidak akan bernilai null.

Dalam kasus ini, **table BARANG adalah table yang merujuk dan table PEMASOK adalah table yang dirujuk**. Jadi setiap nilai dala kolom no_pemasok pada table BARANG harus memiliki nilai yang tepat sama dengan kolom no_pemasok pada table PEMASOK

✓ Manipulasi data dalam database

Perubahan yang dilakukan terhadap database dapat menyebabkan pelanggaran terhadap aturan integritas referensial. Dalam melakukan operasi insert, delete, update harus didefiniskan aturan yang menjamin bahwa aturan integritas referensial diterapkan. Aturan tersebut adalah *insertion rule*, *deletion rule dan update rule*.

a. *Insertion Rule :*
   Aturan : Jika dilakukan penyisipan baris terhadap sebuah table yang merujuk maka harus terdapat nilai yang bersesuaian dalam table yang dirujuk.

b. *Deletion Rule :*
   Aturan : Jika dilakukan penghapusan baris terhadap sebuah table yag dirujuk dan memiliki nilai yang bersesuaian dengan table yang merujuk maka harus ada perlakuan tertentu untuk menjamin inegritas table database tersebut yaitu :
   (1) RESTRICT. Tidak mengijinkan penghapusan terhadap baris dalam table yang dirujuk.
   (2) Nullify. Mengeset nilai yang bersesuaian dari table yang merujuk menjadi null dan kemudian menghapus baris dalam table yang dirujuk.
   (3) Cascading deletion. Baris yang bersesuaian dalam table yang merujuk secara otomatis akan ikut terhapus.

c. *Update Rule :*
   Terdapat 2 hak yang harus diperhatikan:
   (1) Jika nilai yang di update adalah kunci tamu pada table yang merujuj, maka perlakuan seperti insertion rule.
   (2) Jika nilai yag di update pada table yang dirujuk memiliki nilai yang bersesuaian dengan table yang dirujuk maka perlakuan adalah sama dengan pada deletion rule

✓ Referential Integrity dalam SQL

Ada 2 cara untuk mengimplementasikan aturan integritas referensial yaitu :

a. Program aplikasi yang dibuat menerapkan logika yang menjamin aturan integritas tetap dipertahankan, dalam baris-baris programnya.
   Cara ini kurang handal, karena tidak setiap program aplikasi digunakan untuk permasalahan yang menggunakan aturan integritas.
b. Aturan integritas referensial dikenakan pada table pada saat dibuat. Hal ini bias dilakukan jika DBMS yang dipaki memiliki fasilitas untuk mendukung integrita referensial melalui perintah SQL.
   Contoh perintah SQL yang digunakan untuk mendefinisikan aturan integritas referensial pada saat table dibuat. Pada contoh ini perlakuan cascading diterapkan untuk delete dan update :

```
Create table BARANG
(.....
Foreign key (no_pemasok) references PEMASOK on delete cascade, on
update cascade
.....)
```

- **Attributed-Based Constraints dan Tuple-Based Constraints**

Adalah aturan yang menentukan bahwa pada saat dilakkan insert dan update, nilai atribut tertentu harus memenuhi kondisi tertentu. Jika kondisi tidak dipenuhi maka system akan menolak dilakukannya proses input atau update.

Karakteristik dari kedua aturan ini adalah :

1. Attribute/Tuple-based constraints diberlakukan hanya pada proses insert dan update. Pada proses delete aturan ini tidak berlaku
2. Pada beberapa DBMS, attribute/tuple-based constraints hanya diperbolehkan beroperasi pada satu table , sehingga tidak boleh ada sub-kueri didalamnya. Namun ada juga DBMS yang memungkinkan aturan ini dikenakan pada beberapa table.

Contoh :

- Attribted-based constraints :
  Operasi insert/update data pada table MAHASISWA mensyaratkan nilai atribut AGAMA adalah “Islam”, “Protestan”, “Katholik”, “Budha”, “Hindu”. Jika nilai atribut AGAMA bukan salah satu dari itu maka proses insert/update akan ditolak.

- Tuple-based constraints :
  Operasi insert/update data pada table MAHASISWA mensyaratkan nilai atribut USIA < 30

Assertions

Assertions adalah aturan yang diterapkan untuk membuat agar database tetap pada kondisi yang diinginkan.

Karakteristik Assertions :

1. Berlaku pada proses insert/delelte/update
2. Bisa melibatkan beberapa table.

Contoh :
Dalam system Perbankan terdapat aturan : “Jumlah semua pinjaman pada setiap kantor cabang tidak boleh melebihi jumlah semua pinjaman pada cabang tersebut”.

Untuk mengimplementasikan aturan tersebut dibuat perintah SQL sebagai berikut :

```
Create assertion CONTROL_LOAN chek
(not exist
      (select * from CABANG where
        (select sum (jumlah) from PINJAMAN
          where  PINJAMAN.NAMA_CAB= CABANG.NAMA_CAB) >=
          (select sum(SALDO) from SIMPANAN
            Where SIMPANAN.NAMA_CAB=CABANG.NAMA_CAB)))
```

Ketika assertion dibuat, system akan menguji validitasnya. Jika assertion tersebut valid, maka setiap modifikasi dalam database akan diijinkan asalkan tidak menyebabkan assertion tersebut dilanggar. Akibatnya system akan membutuhkan waktu yang tidak sedikit untuk melakukan pengecekan apakah assertion dipenuhi atau tidak. Oleh karena itu, pemakaian assertion harus dipertimbangkan dengan hati-hati.

Trigger

Trigger adalah aturan yang nengeksekusi peritah secara otomatis sebagai akibat sampingan sari proses modifikasi (insert/delete/update) dalam database. Ruang lingkup trigger bisa mencakup atribut dalam satu table atau dapat juga atribut dari beberapa table.

Operasi trigger mencakup komponen :
1. Aturan User : statemen yang digunakan untuk menyatakan operasi trigger
2. Event : operasi manipulasi data yang memicu operasi
3. Nama Table : nama table yang diakses/dimodfikasi
4. Kondisi : kondisi yang menyebabkan suatu operasi dijalankan
5. Aksi : tindakan yang dilakukan saat operasi dijalankan

Contoh :
Dalam system aplikasi perbankan terdapat ketentuan yang menjadi aturan user : "jumlah penarikan uang oleh nasabah tidak boleh melebihi saldo terakhir"

Event yang memicu operasi adalah menambah baris (insert) dalam table PENARIKAN. Kondisi yang terjadai : Jml_penarikan > saldo

Ketika kondisi ini terjadi, aksi yang dilakukan adalah menolak operasi insert.

CONCURRENCY DATA

Konkurensi berarti bahwa sejumlah transaksi diperkenankan untuk mengakses data yang sama dalam waktu yang sama. Hal ini seperti ini menjadi titik perhatian bagi DBMS yang mendukung multiuser. Sehingga diperlukan mekanisme pengontrolan konkurensi. Tujuannya untuk menjamin bahwa transaksi-transaksi yang konkuren tidak saling mengganggu operasi masing-masing.

Dalam kasus konkurensi, terdapat 3 masalah yang dapat terjadi :
1. Masalah Kehilangan Modifikasi (*lost update problem*)
2. Masalah Modifikasi Sementara (*uncommitted dependency problem*)
3. Masalah Analisis Yang tidak Konsisten (*inconsistent analysis problem*)

Ad. 1 Masalah Kehilangan Modifikasi
Perhatikan ilustrasi berikut ini yang menggambarkan proses :
- Transaksi A membaca (retrieve) record r pada waktu t1
- Transakasi B membaca record r pada waktu t2
- Transakasi A meng-update record r (bedasarkan nilai pada waktu t1) pada waktu t3
- Transakasi B meng-update record r (bedasarkan nilai pada waktu t2, yang sama dengan nilai pada waktu t1) pada waktu 4

Hasilnya, update yang dilakukan oleh transaksi A akan hilang pada waktu t4 karena tertimpa oleh update transaksi B (yang tidak melihat hasil update transaksi A)

Transakasi A	Waktu	Transaksi B
-		-
Bacac r	t1	
-	-	-
-	t2	Baca r
Update r	t3	-
-	-	-
	t4	Update r

Transaksi A kehilangan Modifikasi pada waktu t4

Ad. 2 Masalah Modifikasi Sementara

Masalah ini muncul jika suatu transaksi A diijinkan untuk membaca atau meng-update suatu record yang telah di-update oleh transaksi B namun transaksi B tersebut belum *committed*. Karena transaksi B belum *committed* maka selalu ada kemungkinan transaksi B tidak *committed* tetapi *rolled back*.

Perhatikan ilustrasi berikut ini, yang menggambarkan proses transaksi A melihat suatu update yang belum *committed* pada waktu t2. Update tersebut kemudian dibatalkan pada waktu t3. Dengan demikian transaksi A berjalan atas dasar asumsi yang salah (yaitu asumsi bahwa record r mempunyai nilai seperti pada waktu t2, padahal tidak)

Transakasi A	Waktu	Transaksi B
-		-
-		-
-	t1	Update r
-	-	-
Baca r	t2	-
-		-
-	t3	-
-		Rollback

Transaksi A tergantung pada proses update yang belum committed pada waktu t2

Ilustrasi berikutnya menggambarkan keadaan yang lebih buruk. Transaksi A tidak hanya tergantung pada transaksi yang belum committed, tetapi juga kehilangan update pada waktu t3 karena transaksi B mengalami rolled back sehingga nilai record r kembali ke nilai awal.

Transakasi A	Waktu	Transaksi B
-		-
-		-
-	t1	Update r
-		-
update r	t2	-
-		-
-	t3	Rollback
-		-

Transaksi A meng-update suatu perubahan yang belum committed pada waktu t2 dan kehilangan update tersebut pada waktu t3

Ad. 3 Masalah Analisis Yang Tidak Konsisten

Perhatikan ilustrasi dibawah ini yang menunjukkan 2 transaksi A dan B beroperasi pada suatu record rekening nasabah.
Transaksi A menjumlahkan total saldo pada rekening-rekening yang telah dibaca, sedangkan transaksi B mentransfer 10 dari rekening 3 ke rekening 1. Hasil dari transaksi A yaitu 110 adalah salah. Jika meneruskan prosesnya dengan menuliskan hasil pada basis data, akibatnya basis data akan berada dalam status inkonsisten. Dikatakan bahwa transaksi A telah melihat status inkonsisten dari basis datasehingga melakukan analisis yang inkonsisten.

REK 1 = 40	REK 2= 50	REK 3= 30
Transakasi A	Waktu	Transaksi B
-		
-		
Baca REK1	t1	
Saldo=40		
-		
Baca REK2	t2	
Saldo = 90		-
-		-
-	t3	Baca REK3
-		-
-	t4	Update REK3 30→20
-		-
-	t5	Baca REK1
-		-
-	t6	Update REK1 40→50
-		-
-	t7	Commit
-		-
Baca REK3	t8	
Saldo=110		

Transaksi A tergantung pada suatu perubahan yang tidak konsisten yang tidak committed pada waktu t2

Masalah-masalah dalam konkurensi dapat diatasi dengan teknik pengendalian konkurensi. Terdapat 2 pendekatan pengendalian konkurensi yaitu :

1. Penguncian (Locking)
2. Pembuatan versi (versioning)

Ad. 1 Penguncian (Locking)

Locking adalah salah satu mekanisme pengontrol konkuren. Dalam pendekatan ini, setiap data yang sedang diakses oleh pemakai untuk proses update dikunci hingga proses update selesai (selesai bisa berarti committed atau rolled back).
Pada saat data dikunci, pemakai lain tidak diijinkan mengakses data tersebut.

Pendekatan ini sering disebut pedekatan pesimistis karena DBMS menerapkan pendekatan sangat hati-hati dalam penguncian record sehingga program/pemakai lain tidak dapat memakainya.

Level Penguncian

Salah satu pertimbangan dalam mengimplementasikan penguncian adalah memilih level penguncian (locking level/granularity).

Beberapa level penguncian yang dikenal adalah :

- Level Basis Data
  Seluruh basis data dikunci dan tidak boleh diaksis oleh pemakai lain.
  Level penguncian ini dipakai dalam kasus-kasus tertentu, seperti ketika basis data sedang di-backup

- Level Tabel
  Penguncian dilakukan pada table yang memuat record yang dimaksud.

Level ini biasanya dipakai pada saat dilakukan update pada hamper seluruh record table, misalnya meng-update data GAJI seluruh karyawan karena ada kenaikan gaji sebesar 5%.

- Level Block atau halaman
  Blok atau halaman fisik yang memuat record tertentu dikunci.
  Level pengucian ini jarang digunakan karena suatu blok fisik bisa memuat record dari beberapa table.

- Level Record
  Hanya record yang diakses saja yang dikunci, sedangkan record lain dalamsatu tabel tetap dapat diakses oleh pemakai lain. Level ini paling banyak digunakan.

- Level Field
  Penguncian dilakukan hanya pada field tertentu dari record yang diakses. Level ini biasa digunakan jika update hanya mempengaruhi satu atau dua field dalam suatu record. Level pengucian ini jarang digunakan.

Tipe Penguncian

Terdapat 2 tipe pengucian yaitu :

1. Penguncian Berbagi *(Shared Lock/S lock/Read Lock)*
   Dalam tipe penguncian ini, transaksi lain masih dapat membaca (tetapi tidak dapat meng-update) data yang dikunci. Suatu transaksi dapat melakukan shared lock pada data yang akan dibaca, tetapi tidak akan di-update. Dengan melakukan shared lock ini pemakai lain akan dicegah dari tindakan melakukan exclusive lock.

2. Pengucian Eksklusif *(Exclusive lock/X lock/Write lock)*
   Dalam tipe penguncian ini, transaksi lain tidak dapat membaca maupun meng-update data yang dkunci, hingga data tersebuat dibuka (unlocked). Suatu transaksi sebaiknya melakukan penguncian eksklusif jika ia akan meng-update data tersebut. Dengan melakukan penguncian eksklusif, pemakai lain akan dicegah dari tindakan melakan shared lock maupun exclusive lock pada data tersebut.

Bedasarkan sifat tipe pengucian S-lock dan X-lock, maka matriks kompatibilitas adalah sebagai berikut :

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

Protokol Pengaksesan Data

Untuk menerapkan X lock dan S lock serta menjamin bahwa 3 masalah dalam konkurensi tidak terjadi, dikenal dengan protocol pengaksesan data *(data access protocol)* sebagai berikut :

- Jika transakasi A memakai kunci X pada record r, maka permintaan transaksi B untuk suatu kunci pada record r ditunda dan transaksi B harus menunggu sampai transaksi A melepaskan kunci X.
- Jika transaksi A memakai kunci S pada record r, maka :
  - Jika transaksi B, ingin memakai kunci X, maka transaksi B harus menunggu transaksi A melepaskan kunci S.
  - Jika transaksi B ingin memakai kunci S, maka transaksi B dapat menggunakan kunci S bersama transaksi A.
- Jika suatu transaksi hanya melakukan pembacaan saja, maka hanya memerlukan kunci S

- JIka suatu transaksi ingin memodifikasi record, maka diperlukan kunci X.
- Jika suatu transaksi menggunakan kunci S, setelah itu akan memodifikasi record, maka kunci S akan dinaikkan menjadi kunci X.
- Kunci X dan kunxi S akan dilepaskan pada saat synchronization point (synchpoint), yang menyatakan akhir dari suatu transaksi dimana database berada pada kondisi yang konsisten. Semua modifikasi data menjalan
- operasi Commit atau Rollback, setelah itu semua kunci dari record akan dilepaskan.

Contoh Implementasi Locking

Implementasi locking untuk mengatasi ketiga masalah konkuren :

1. Masalah Kehilangan modifikasi

Untuk menghindari ketidakkonsistenan data yang terjadi, maka akan dilakukan mekanisme locking terhadap transaksi yang ada.

Transakasi A	Waktu	Transaksi B
-		-
Bacac r	t1	
(meminta kunci S untuk r)		
-	-	-
-	t2	Baca r
		(meminta kunci S untuk r)
Update r	t3	-
(meminta kunci X untuk r)		
Wait	-	-
Wait	t4	Update r
Wait		(meminta kunci X untuk r)
Wait		Wait
Wait		Wait
Wait		Wait

Transaksi A tidak kehilangan modifikasi pada waktu t4 namun terjadi deadlock

Update r yang dilakukan transaksi A pada watu t3 tidak akan diterima sehingga berada pada kondisi wait. Hal ini terjadi karena permintaan kunci X dari transaksi A konflik dengan kunci S yang dilakukan oleh transaksi B ketika perintah baca r.

Hal yang sama juga terjadi pada transaksi B ketika akan meng-update r. Akibatnya transaksi A ataupun transaksi B berada dalam status wait dan saling menunggu untuk membuka kunci agar bisa melanjutkan transaksi. Keadaan ini disebut deadlock.

2. Masalah Modifikasi Sementara

Untuk mengatasi masalah modifikasi sementara dapat dilakukan mekaniskem locking sebagai berikut :

Transakasi A	Waktu	Transaksi B
-		-
-		-
-	t1	Update r
		(meminta kunci X pada r)
-	-	-
Baca r	t2	-
(meminta kunci S pada r)		
Wait		-
Wait	t3	Rollback/commit
Wait		(melepas kunci X pada r)
Resume : Baca r		
(meminta kunci S pada r)		

Transaksi A dicegah untuk menunggu update yang belum committed pada waktu t2

Transakasi A	Waktu	Transaksi B
-		-
-		-
-	t1	Update r
		(meminta kunci X pada r)
-		-
update r	t2	-
(meminta kunci X pada r)		-
Wait	t3	Rollback
Wait		(melepas kunci X pada r)
Wait		
Resume : update r	t4	
(meminta kunci X pada r)		

Transaksi A dicegah untuk melakukan update pada perubahan yang belum committed pada waktu t2.

3. Masalah Analisis Yang Tidak Konsisten

Untuk menghindari ketidak konsistenan data yang terjadi, maka akan dilakukan mekanisme locking terhadap transaksi yang ada. Salah sau cara yang dilakukan diilustrasikan pada gambar berikut :

Transakasi A	Waktu	Transaksi B
-		-
-		
Baca r (meminta S lock pada r)	t1	
-		
-	t2	Baca r (meminta S lock pada r)
-		
Update r (meminta X lock pada r)	t3	-
Wait		
Wait	t4	Update r (meminta X lock pada r)
Wait		Wait
Wait		Wait
Wait		Wait

Transaksi A tidak kehilangan update pada waktu t4 tapi terjadi deadlock

Update yang dilakukan oleh transaksi A pada waktu t3 tidak diterima sehingga berada pada kondisi wait. Masalah ini terjadi karena x lock dari transaksi A konflik dengan S lock yang dilakukan oleh transaksi B ketika melakukan perintah Baca

Masalah yang sama terjadi juga pada transaksi B ketika meng update r. Sehungga transaksi A maupun B akan berada dalam status wait dan saling menunggu untuk membuka lock agar dapat melanjutkan transaksi. Kondisi seperti ini disebut deadlock.