



# Sistem Basis Data

INTEGRITAS DAN KEAMANAN BASIS DATA

Alif Finandhita, S.Kom

# INTEGRITAS DATA

- Informasi yang disimpan pada basis data hanya akan bagus jika DBMS turut membantu mencegah adanya informasi yang salah yang masuk ke basis data.
- Batasan Integritas data (Data Integrity Constrain) adalah syarat yang dispesifikasikan pada basis data untuk membatasi data yang dapat disimpan di dalam basis data.
- Jika basis data memenuhi semua batasan integritas yang dispesifikasikan pada skema basis datanya, maka basis data tersebut sudah bisa dikatakan legal.

# INTEGRITAS DATA (2)

- DBMS “memaksakan” batasan integritas data, sehingga hanya mengizinkan basis data yang legal lah yang dapat disimpan oleh DBMS.
- Batasan integritas menjamin bahwa perubahan – perubahan yang dilakukan oleh orang yang memang mempunyai otorisasi tidak akan melanggar konsistensi data.
- Artinya bahwa data akan tetap terjaga, tetap akurat, konsisten, dan handal.

# INTEGRITAS DATA (3)

- Batasan integritas menjaga terjadinya kerusakan terhadap database yang dapat terjadi dengan memastikan bahwa perubahan terhadap database tidak menyebabkan terjadinya inkonsistensi data
- Integritas data mengacu ke konsistensi dan akurasi data yang disimpan di dalam basis data.
- Batasan integritas dispesifikasikan dan “dipaksakan” pada waktu yang berbeda, yaitu :
  - Ketika DBA mendefinisikan skema basis data melalui DDL, DBA menspesifikasikan batasan / konstrain integritas yang harus selalu dipenuhi.

# INTEGRITAS DATA (4)

- Ketika aplikasi basis data dijalankan, DBMS melakukan pemeriksaan untuk mencegah terjadinya pelanggaran dan mencegah perubahan – perubahan yang melanggar konstrain integritas yang telah ditentukan sebelumnya.
- Dalam kondisi tertentu, DBMS tidak melarang suatu aksi yang dapat menimbulkan pelanggaran, namun kemudian DBMS membuat tindakan – tindakan otomatis untuk tetap memenuhi konstrain integritas, dengan demikian dijamin perubahan tidak akan mengganggu integritas data.

# JENIS INTEGRITAS DATA

- Integritas data dapat dikelompokkan menjadi dua bagian :
  - Integritas data yang berada di dalam relasi, yaitu Integritas entitas (Entity Integrity) dan Integritas Domain (Domain Integrity).
  - Integritas data yang berada di luar relasi, yaitu Integritas referensial (Referential Integrity).
- Selain itu juga terdapat aturan integritas untuk memenuhi aturan – tertentu yang ditentukan sendiri di dalam suatu perusahaan (Enterprise), disebut dengan Integritas perusahaan (Enterprise Integrity/User Defined Integrity)

# JENIS INTEGRITAS DATA (2)

- Secara garis besar, di dalam model relasional Integritas data meliputi :
  - Integritas Entitas (Entity Integrity)
  - Integritas Domain (Domain Integrity)
  - Integritas Referensial (Referential Integrity)
  - Integritas Enterprise (Enterprise/User Defined Integrity)

# Integritas Entitas

- Integritas entitas mendefinisikan sebuah baris sebagai sebuah entitas yang unik untuk suatu tabel.
- Integritas entitas memaksa integritas dari *column* atau *primary key* dari suatu tabel melalui *index, unique, constrains, primary key*, dimana *primary key* tidak boleh null.
- Dalam Integritas entitas, tidak ada baris yang duplikat di dalam satu tabel.



# Integritas Entitas (2)

```
create table Pembelian  
  (IDPembelian smallint ,  
   IDModel smallint,  
   DeskripsiModel varchar(40),  
   primary key (IDPembelian));
```

```
create table Pembelian  
  (IDPembelian smallint ,  
   IDModel smallint,  
   DeskripsiModel varchar(40),  
   primary key (IDPembelian)  
   unique (ID Pembelian, ID Model));
```

# Integritas Domain

- Domain adalah nilai – nilai yang dimungkinkan diasosiasikan dengan setiap atribut.
- Integritas domain merupakan validasi dari masukan untuk sebuah kolom.
- Kita dapat memaksa integritas domain dengan membatasi tipe (melalui *data types*), format (melalui *check constraints* dan *rules*), atau range nilai – nilai yang mungkin (melalui *Foreign Key Constrains*, *Check Constraints*, *Default Definitions* dan *Rules*)
- Dengan integritas domain, tidak ada data yang melanggar jangkauan nilai di tiap kolom data.

# Integritas Domain (2)

- Pada saat membuat tabel ("Create Table"), kita bisa mencegah kolom bernilai Null dengan menggunakan konstrain "Not Null" yang berlaku untuk kolom.
- Bila tidak dinyatakan, SQL akan mengasumsikan kondisi Null diijinkan, kecuali bila suatu kolom dispesifikasikan sebagai bagian dari kunci utama yang dinyatakan dengan Primary Key.
- Supaya integritas entitas tetap terjaga, maka lebih baik jika konstrain "Not Null" dinyatakan.

# Integritas Domain (3)

- Jenis domain yang dapat dimiliki oleh suatu atribut :
  - Karakter bebas
  - Alphanumerik
  - Alphabet
  - Numerik
- Pemeliharaan integritas domain :
  - Pendefinisian skema / struktur tabel
  - Pemanfaatan properti field
  - Penerapan proses validasi pada pemasukan data

# Integritas Domain (4)

**create table** *biografi*

*(idpenulis* smallint unsigned not null,

*tahunlahir* year not null,

*kotalahir* varchar(40) not null default

'kosong');

**create domain** *nilai* numeric(3,2)

**constraint** *value-test* check(*value* >= 0.00)

# Integritas Referensial

- Integritas Referensial memastikan bahwa seluruh nilai dari *foreign key* cocok dengan nilai *primary key* yang dihubungkannya.
- Integritas Referensial adalah dasar relasi antar tabel yaitu antara *foreign key* dengan *primary key*.
- Data pada *foreign key* harus sesuai dengan *primary key*. Artinya :
  - Tipe data dan ukuran sama
  - Konsistensi tetap terjaga ketika ada penghapusan, pergantian data dan penambahan data pada tabel

# Integritas Referensial (2)

- Ketika integritas referensial ini dilaksanakan maka akan mengecek :
  - Penambahan record apakah record yang ditambahkan pada *foreign key* ada dalam *primary key* atau tidak.
  - Perubahan data pada *primary key* apakah akan mempengaruhi terhadap *foreign key* atau tidak.

# Integritas Referensial (3)

- Opsi ketika suatu record pada tabel yang direferensi oleh suatu *foreign key* dihapus atau diganti nilainya :
  - [ ON DELETE { CASCADE | NO ACTION } ]
  - [ ON UPDATE { CASCADE | NO ACTION } ]
- ON DELETE, merupakan tindakan pada tabel yang direferensi terjadi penghapusan record.
- ON UPDATE, merupakan tindakan apabila data tabel yang direferensi mengalami perubahan nilai record.



# Integritas Referensial (4)

- Tindakan yang dapat diatur pada ON DELETE maupun ON UPDATE ada dua, yaitu :
  - CASCADE
  - NO ACTION
- ON UPDATE CASCADE
  - Jika nilai *primary key* pada tabel yang direferensi diganti maka *foreign key* pada tabel yang mereferensi akan disamakan nilainya dengan *primary key* pada tabel yang direferensi.

# Integritas Referensial (5)

- ON DELETE CASCADE

- Jika nilai *primary key* pada tabel yang direferensi dihapus maka semua record yang nilai *foreign key*-nya=*primary key* pada tabel yang direferensi dimana recordnya yang dihapus akan turut terhapus.

- ON UPDATE NO ACTION

- Jika nilai *primary key* pada tabel yang direferensi diganti maka *foreign key* pada tabel yang mereferensi nilainya tidak ikut berubah

- ON DELETE NO ACTION

- Jika nilai *Primary Key* pada tabel yang direferensi dihapus maka semua record yang nilai *foreign key*-nya=*primary key* tidak ikut dihapus

# Integritas Referensial (6)

```
create table customer  
  (customer-name      char(20),  
   customer-street    char(30),  
   customer-city char(30),  
   primary key (customer-name))
```

```
create table branch  
  (branch-name char(15),  
   branch-city char(30),  
   assets      integer,  
   primary key (branch-name))
```

# Integritas Referensial (7)

```
create table account  
  (account-number char(10),  
   branch-name char(15),  
   balance integer,  
   primary key (account-number),  
   foreign key (branch-name) references branch)
```

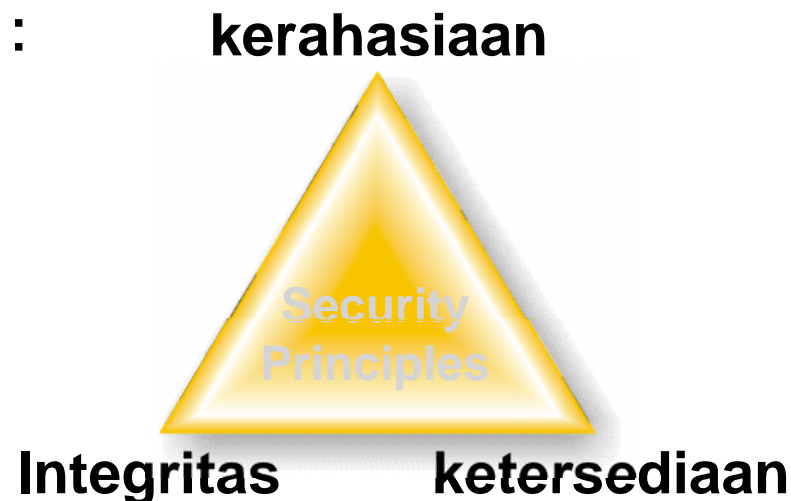
```
create table depositor  
  (customer-name char(20),  
   account-number char(10),  
   primary key (customer-name, account-number),  
   foreign key (account-number) references account,  
   foreign key (customer-name) references customer)  
  on delete cascade on update cascade
```

# Integritas Enterprise (User Defined Integration)

- Integritas Enterprise / User Defined Integration mengizinkan kita untuk menentukan *specific business rules* sendiri yang tidak sama pada katogeri integritas yang lainnya

# KEAMANAN DATA

## ■ Prinsip Pengamanan :



- Kerahasiaan menjamin perlindungan akses informasi
- Integritas menjamin bahwa informasi tidak dapat diubah dan tetap konsisten
- Ketersediaan menjamin kesiapan akses informasi

# KEAMANAN DATA (2)

- Contoh :
  - Kerahasiaan:  
catatan medis pasien harus tertutup untuk umum
  - Integritas:  
catatan medis harus benar
  - Ketersediaan:  
catatan medis pasien dapat diakses saat dibutuhkan untuk pengobatan

# KEAMANAN DATA (3)

- Pengamanan data merupakan mekanisme untuk melindungi sistem basis data dari aksi yang disengaja (mis : percobaan pencurian / modifikasi data oleh pihak yang tidak berwenang) dan tidak disengaja (mis: bencana alam, kebakaran, dll)
- Kerusakan yang biasa terjadi : kehilangan kerahasiaan, kehilangan kebebasan (privacy), kehilangan keutuhan data (integrity), kehilangan ketersediaan data (availability).



# KEAMANAN DATA (4)

Ancaman potensial terhadap sistem komputer :

- **DBMS dan Software Aplikasi :**
  - Kesalahan dalam mekanisme pengamanan dengan pemberian hak akses yang lebih besar
  - Pengubahan atau pencurian program
- **Jaringan Komunikasi :**
  - Penyadapan
  - Pemutusan kabel
  - Interfensi Gelombang dan radiasi

# KEAMANAN DATA (5)

- **Data / Database Administrator :**
  - Kebijakan pengamanan dan prosedur yang lemah
  
- **Hardware :**
  - Bencana alam
  - Kehilangan data karena listrik mati
  - Pencurian peralatan

# KEAMANAN DATA (6)

- **Database :**
  - Pengubahan yang tidak sah atau penduplikasian data
  - Pencurian data
- **User :**
  - Menggunakan hak akses orang lain
  - Viewing dan penutupan data yang tidak sah
  - Kekurangan staf yang terlatih
  - Memasukkan virus/worm/trojan.
- **Programmer :**
  - Membuat jebakan (trapdoor)
  - Mengubah program.

# KEAMANAN DATA (6)

Langkah – langkah pengamanan data :

- **Level DBMS :**

- Menerapkan mekanisme otorisasi dan autentifikasi untuk mengatur hanya user tertentu saja yang dapat mengakses data sesuai dengan yang dibutuhkan

- **Level Sistem Operasi :**

- Super user pada sistem operasi dapat melakukan apapun terhadap database. Oleh karena itu dibutuhkan mekanisme pengamanan yang baik bagi sistem operasi

# KEAMANAN DATA (6)

## ■ Level Jaringan :

- Setiap data yang dikirimkan melalui network harus dienkripsi untuk menghindari terjadinya pembacaan data oleh orang yang tidak berhak (Eavesdropping) dan penyalahgunaan hak akses/berpura pura sebagai authorized user (masquerading)

## ■ Level Fisik :

- Akses terhadap fisik komputer memungkinkan perusakan data oleh penyusup. Oleh karena itu diperlukan mekanisme konvensional untuk mengamankan fisik komputer (gembok / kunci).
- Komputer juga harus terlindungi dari kejadian bencana alam seperti kebakaran, banjir, gempa, dll.

# Otorisasi

- Otorisasi merupakan pemberian hak yang istimewa terhadap user sehingga dapat mempunyai akses yang sah terhadap sistem atau objek sistem.
- Bentuk otorisasi dalam basis data :
  - **Read authorization** – mengizinkan pembacaan data, tapi tidak diizinkan modifikasi data.
  - **Insert authorization** – mengizinkan penambahan data baru, tapi tidak diizinkan modifikasi data yang sudah ada
  - **Update authorization** – mengizinkan modifikasi, tapi tidak diizinkan menghapus data
  - **Delete authorization** – mengizinkan penghapusan data

# Otorisasi (2)

- Bentuk otorisasi untuk modifikasi skema basis data :
  - **Index authorization** – mengizinkan pembuatan dan penghapusan index.
  - **Resources authorization** – mengizinkan pembuatan relasi baru.
  - **Alteration authorization** – mengizinkan penambahan atau penghapusan suatu atribut di dalam suatu relasi.
  - **Drop authorization** – mengizinkan penghapusan suatu relasi.

# Otorisasi dan View

- View adalah objek basis data yang berisi perintah query ke basis data.
- Setiap kali sebuah view diaktifkan, pemakai akan selalu melihat hasil querynya.
- Berbeda dengan tabel, data yang ditampilkan di dalam view tidak bisa diubah.
- View menyediakan mekanisme pengamanan yang fleksibel namun kuat dengan cara menyembunyikan sebagian basis data dari user lain.



# Otorisasi dan View (2)

- User dapat diberikan otorisasi pada View, tanpa harus diberikan otorisasi terhadap relasi yang digunakan di dalam definisi view.
- View dapat meningkatkan keamanan data dengan mengizinkan user untuk hanya dapat mengakses data sesuai dengan pekerjaannya masing – masing.
- Kombinasi antara level keamanan di tingkat relasional dengan level keamanan di tingkat view dapat digunakan untuk membatasi hak akses user, sehingga mereka hanya mengakses data sesuai dengan kebutuhannya saja.

# Otorisasi dan View (3)

Contoh View :

- Misalkan seorang pegawai bank membutuhkan informasi nama – nama nasabah dari setiap cabang yang ada, tetapi tidak diizinkan untuk mengetahui secara spesifik jumlah pinjamannya.
  - Penyelesaian : tolak akses langsung ke relasi *pinjaman* , tapi berikan akses terhadap view *nasabah-pinjam* yang hanya terdiri dari nama nasabah beserta cabangnya dimana mereka melakukan pinjaman
  - View *nasabah-pinjam* didefinisikan di dalam SQL sebagai berikut :  
**create view** *nasabah-pinjam* **as**  
    **select** *nama-cabang, nama-pelanggan*  
    **from** *peminjam, pinjaman*  
    **where** *peminjam.no-pinjaman = pinjaman.no-pinjaman*

# Otorisasi dan View (4)

- Pegawai tersebut diizinkan untuk melihat view dengan SQL :  
**Select \* from *nasabah-pinjam***
- Ketika pemroses query menterjemahkan hasilnya ke dalam query yang terdapat di dalam relasi aktualnya di basis data, si pegawai memperoleh query yang ada di *peminjam* dan *pinjaman*.
- Pada saat pegawai bank tersebut melakukan query, harus dicek dulu otorisasinya sebelum pemrosesan query me-replace view oleh definisi viewnya.

# Otorisasi dan View (5)

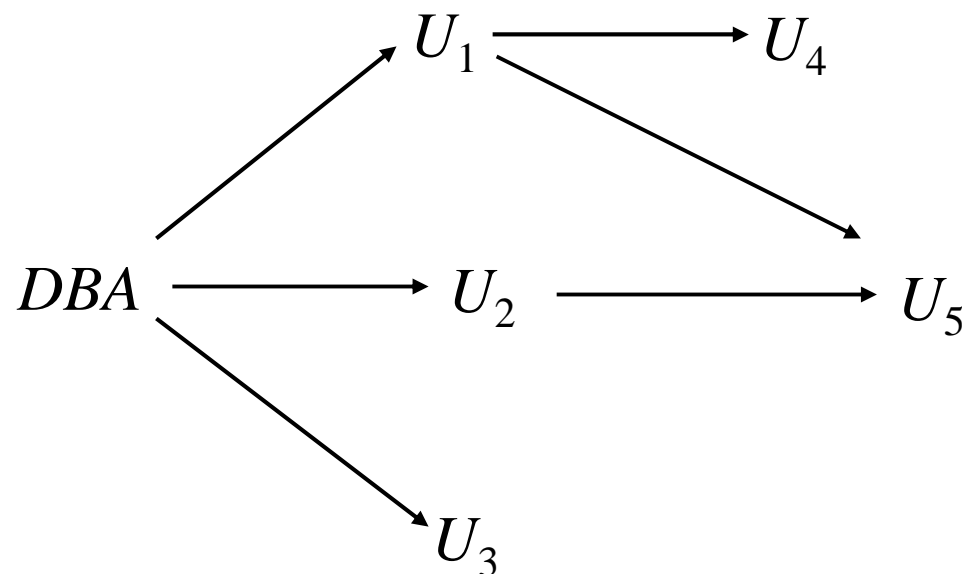
- Pembuatan view tidak membutuhkan **resource authorization** karena sesungguhnya tidak ada relasi yang dibuat.
- Pembuat view hanya memperoleh hak akses sesuai dengan yang diberikannya, tidak ada otorisasi lainnya diluar dari yang dia miliki.
- Misalkan, jika si pembuat view *nasabah-pinjam* hanya memiliki **read authorization** pada relasi *peminjam* dan *pinjaman*, maka dia hanya memperoleh otorisasi tersebut saja pada *nasabah –pinjam*.

# Pemberian Hak Akses (Privileges)

- Otorisasi dari satu user ke user lainnya dapat direpresentasikan ke dalam suatu grafik
- Node dari grafik ini adalah user.
- Root-nya adalah Database Administrator (DBA)
- Misalkan grafik untuk **update authorization** pada relasi *pinjaman*

# Pemberian Hak Akses (Privileges) - 2

- Bentuk  $U_i \rightarrow U_j$  mengindikasikan bahwa user  $U_i$  memiliki otorisasi untuk update pada relasi *pinjaman* terhadap  $U_j$ .



# Spesifikasi Keamanan di SQL

- Pernyataan **grant** digunakan untuk memberikan otorisasi :

**grant** *<daftar privilege>*  
**on** *<nama relasi / nama view>* **to** *<daftar user>*

- *<daftar user>* adalah :
  - Id-user
  - *Public*, yang memungkinkan semua daftar user yang valid untuk menggunakan hak akses yang diberikan
  - Role

# Spesifikasi Keamanan di SQL

- Pernyataan **grant** digunakan untuk memberikan otorisasi :

**grant** *<daftar privilege>*  
**on** *<nama relasi / nama view>* **to** *<daftar user>*

- *<daftar user>* adalah :
  - Id-user
  - *Public*, yang memungkinkan semua daftar user yang valid untuk menggunakan hak akses yang diberikan
  - Role



# Hak Akses di Dalam SQL

- **Select :**

Mengizinkan user untuk melihat data yang ada di relasi, atau kemampuan untuk melakukan query dengan menggunakan view

- Contoh : memberikan hak akses **select authorization** bagi user  $U_1$ ,  $U_2$ , dan  $U_3$  pada relasi *cabang*. “**grant select on cabang to  $U_1$ ,  $U_2$ ,  $U_3$** ”

- **Insert :**

Kemampuan untuk menambahkan record / tuple baru

# Hak Akses di Dalam SQL (2)

- **Update :**  
Kemampuan untuk update data dengan menggunakan pernyataan SQL
- **Delete :**  
Kemampuan untuk menghapus record / tuple
- **References :**  
Kemampuan untuk mendeklarasikan foreign key pada saat membentuk suatu relasi

# Hak Akses di Dalam SQL (3)

- **Usage :**

Pada SQL-92, memungkinkan user untuk menggunakan domain tertentu

- **All Privileges :**

Kemampuan untuk menggunakan semua hak akses yang ada

# Hak Akses di Dalam SQL (4)

- Di dalam SQL seorang DBA juga dapat memberikan hak akses kepada user tertentu untuk melakukan pemberian hak akses terhadap user lainnya dengan menggunakan “**with grant option**”.
- Contoh :  
“**grant select on cabang to  $U_1$  with grant option**”
- Contoh di atas memberikan hak akses kepada  $U_1$  untuk melakukan perintah **select** pada relasi *cabang* dan memperbolehkan  $U_1$  untuk memberikan hak akses tersebut kepada user lainnya.

# Role pada SQL

- Role memungkinkan hak yang sama diberikan kepada sekelompok pemakai sekali saja dengan membuat role yang sesuai.
- Hak akses dapat diberikan atau dicabut dari roles, sebagaimana halnya pada user
- Roles dapat diberikan kepada beberapa user dan bahkan kepada role lainnya.

# Role pada SQL (2)

- Roles yang didukung oleh standar SQL 99 :

```
create role teller  
create role manajer
```

```
grant select on cabang to teller  
grant update (jumlah) on account to teller  
grant all privileges on account to manajer
```

```
grant teller to manajer
```

```
grant teller to andri, desi  
grant manajer to alif
```

# Pencabutan Otorisasi di SQL

- Pernyataan **revoke** digunakan untuk pencabutan otorisasi hak akses

```
revoke <daftar privilege>  
on <nama relasi / nama view>  
from <daftar user> [restrict|cascade]
```

- Contoh :

```
revoke select  
on cabang  
from  $U_1, U_2, U_3$  cascade
```

- Pencabutan hak akses dari seorang user dapat menyebabkan user lainnya juga kehilangan hak akses itu (cascade)

# Pencabutan Otorisasi di SQL (3)

- *<daftar privilege>* bisa dibuat menjadi **all** untuk mencabut semua hak akses yang sedang dipegang oleh user yang bersangkutan.
- Jika *<daftar user>* meliputi **public** maka semua user akan kehilangan hak aksesnya.
- Jika hak akses yang sama diberikan dua kali untuk pengguna yang sama oleh pemberi otoritas yang berbeda, pengguna dapat mempertahankan hak aksesnya setelah pencabutan.
- Semua hak akses yang bergantung pada hak akses yang dicabut maka akan ikut tercabut pula hak aksesnya



# Audit Trails

- Audit trail adalah log dari semua perubahan (insert/delete/update) terhadap basis data bersamaan dengan informasi seperti user yang mana yang melakukan perubahan, dan kapan perubahan tersebut dilakukan
- Digunakan untuk melacak kalau perubahan data yang tidak sesuai (palsu/keliru)
- Dapat diimplementasikan dengan menggunakan *Trigger*, tapi banyak juga DBMS yang memberikan dukungan langsung untuk proses audit

# Enkripsi

- Data dapat dienkripsi pada saat ketentuan pada otorisasi database tidak memberikan perlindungan yang memadai .
- Teknik enkripsi yang baik :
  - Relatif mudah bagi user yang mempunyai hak akses untuk melakukan enkripsi dan deskripsi data
  - Skema enkripsi tidak tergantung pada kerahasiaan algoritmanya, tetapi pada kerahasiaan parameter dari algoritmanya yang disebut dengan kunci enkripsi (*encryption key*)
  - Menyulitkan penyusup untuk dapat mengetahui kunci enkripsi yang digunakan

# Enkripsi (2)

- *Data Encryption Standard (DES) :*
  - Menukar karakter dan mengaturnya kembali berdasarkan kunci enkripsi yang disediakan untuk pengguna yang memiliki hak akses melalui mekanisme yang aman. Skema ini tidak terlalu aman karena keynya yang harus dishare.
- *Advance Encryption Standard (AES) :*
  - Standar yang lebih baru untuk menggantikan DES, dan berdasarkan algoritma Rijndael, tapi juga tergantung dari key rahasianya yang harus dishare.

# Enkripsi (3)

- *Public Key Encryption :*
  - Masing – masing user mempunyai dua kunci :
    - Public Key : key yang dipublish untuk melakukan enkripsi data, tapi tidak untuk deskripsi data
    - Private Key : key yang hanya diketahui oleh user tertentu, dan digunakan untuk mendeskripsikan data
  - Skema enkripsi tersebut dibuat sedemikian rupa sehingga sangat sulit bagi orang lain untuk mendeskripsikan data yang hanya diberikan oleh public key
- Skema enkripsi public key RSA berdasarkan pada kesulitan untuk memfaktorkan sejumlah besar digit (100 an digit) ke dalam komponen – komponen utamanya