
# Sistem Basis Data

## OTORISASI DAN HAK AKSES

Alif Finandhita, S.Kom

# 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)

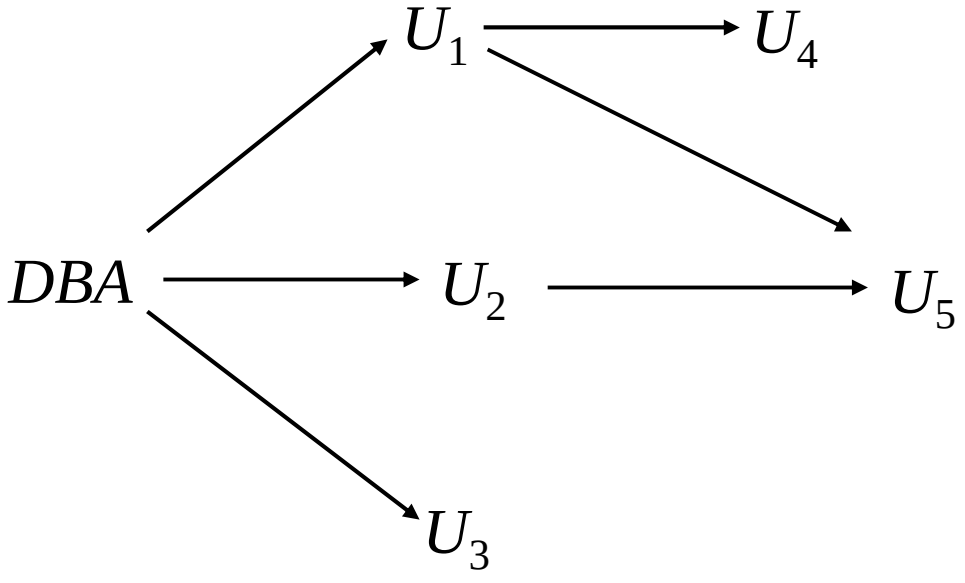
- Pembuatan view tidak membutuhkan **resource authorization** karena sesungguhnya tidak ada relasi yang dibuat.
- Pembuat view hanya memperoleh hak akses sesuai dengan yang diberikannya, tidak ada otorisasi lainnya diluar dari yang dia miliki.
- Misalkan, jika si pembuat view *nasabah-pinjam* hanya memiliki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

# Pemberian Hak Akses (Privileges) - 2

- Bentuk $U_i$ –> $U_j$ 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.

- *Andvance Encryption Standard (AES) :*
  - Standar yang lebih baru untuk menggantikan DES, dan berdasarkan algortima 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