

**Mekanisme *Discretionary Access Control*
untuk *Database Security***

PAPER

Oleh :

**NAMA : DIAN ERLIASARI
NIM : 232 000 72**



**MAGISTER TEKNOLOGI INFORMASI
FAKULTAS ELEKTROTEKNIK
INSTITUT TEKNOLOGI BANDUNG
2001**

DAFTAR ISI

	Hal
I. Pendahuluan	1
II. Dua pendekatan terhadap keamanan data	2
III. <i>Discretionary Access Control</i>	2
3.1 Prinsip Dasar <i>Discretionary Access Control</i>	3
3.2 Prinsip <i>Need To Know</i>	4
3.3 Penerapan mekanisme DAC pada sistem <i>database</i>	5
IV. Kesimpulan	15

Mekanisme *Discretionary Access Control* untuk *Database Security*

I. Pendahuluan

Hal mengenai keamanan data (*data security*) sering dibahas bersamaan dengan hal integritas data (*data integrity*). Keamanan (*security*) lebih condong memiliki pengertian untuk keamanan data terhadap *unauthorized disclosure*, perubahan data atau penghapusan data, sedangkan integritas (*integrity*) lebih condong pada ketepatan atau validitas data. Secara sederhana, penjelasan kedua konsep itu adalah keamanan berarti mengamankan data terhadap *user* yang tidak berwenang dan integritas berarti mengamankan data terhadap *user* yang berwenang. Kedua konsep juga memiliki persamaan yaitu sistem harus memahami *constraints* tertentu dimana *user* harus mematuhi. *Constraints* tersebut biasanya dispesifikasikan dalam bahasa-bahasa tertentu dan harus terus dicek oleh DBA dalam katalog¹ sistem tersebut.

Pada aspek keamanan, terdapat hal-hal yang harus kita perhatikan yaitu :

1. Aspek legal, sosial, dan etika, misalnya apakah seseorang yang membuat permohonan untuk *credit card* memiliki hak hukum untuk informasi yang diinginkannya.
2. Pengontrolan secara fisik, misalnya apakah ruangan komputer penyimpanan informasi dijaga dengan baik
3. Pertanyaan-pertanyaan kebijakan, misalnya bagaimana organisasi yang memiliki sistem menentukan siapa yang berhak mengakses dan apa saja yang berhak diakses.
4. Masalah-masalah operasional, misalnya jika skema *password* digunakan, bagaimana *password-password* tersebut diamankan? Seberapa sering *password-password* tersebut diubah untuk menjaga keamanan.

¹ Katalog adalah tempat dimana semua skema (eksternal, internal, konseptual) dan semua pemetaan yang saling berhubungan (eksternal/konseptual, konseptual/internal) disimpan.

5. Pengontrolan *hardware*, misalnya apakah unit pemrosesan menyediakan *feature* keamanan seperti pengamanan penyimpanan data atau mode operasi keamanan.
6. Dukungan sistem operasi, misalnya apakah sistem operasi yang digunakan akan menghapus isi memori dan *disk files* bilamana telah selesai melakukan proses apapun.
7. Hal mengenai sistem *database* itu sendiri, misalnya apakah sistem *database* memiliki konsep kepemilikan data (*data ownership*).

Dalam paper ini, aspek keamanan yang akan dibahas adalah aspek keamanan terhadap sistem *database*.

II. Dua pendekatan terhadap keamanan data

DBMS modern biasanya mendukung salah satu atau kedua pendekatan keamanan data yaitu *Discretionary Access Control* dan *Mandatory Access Control*. Penjelasan singkat dari kedua pendekatan ini adalah²:

Discretionary Access Control: setiap user memiliki hak akses berbeda-beda (dikenal dengan nama *privileges*) pada objek yang berbeda-beda pula.

Mandatory Access Control: setiap objek diberi label dengan tingkat klasifikasi tertentu, dan setiap user diberikan tingkat clearance tertentu. Kemudian setiap objek data tertentu dapat diakses oleh *user* dengan level *clearance* yang sesuai.

Aspek keamanan sistem *database* yang akan dibahas pada paper ini adalah pendekatan *discretionary access control*.

III. *Discretionary Access Control (DAC)*

Discretionary Access Control adalah suatu alat dalam membatasi akses kepada objek-objek berdasarkan identitas subjek-subjek dan atau kelompok subjek dimana objek tersebut dimiliki³. Pengontrolannya adalah *discretionary* dalam artian suatu subjek dengan hak akses tertentu dapat memberikan hak

² C.J.Date, *An Introduction To Database Systems*, Seventh Edition, Addison-Wesley, 2000, Hal 506-512.

³ Definisi diambil dari *National Computer Security Center, A guide to understanding Discretionary Access Control in trusted systems*, Version 1, National Computer Security Center, 30 September 1987.

tersebut ke subjek yang lainnya. Pengontrolan dengan DAC digunakan untuk membatasi akses user kepada objek-objek yang dilindungi pada sistem. Tipe-tipe akses merupakan operasi yang *user* lakukan pada objek-objek, biasanya untuk setiap objek, *user* atau kelompok *user* memiliki wewenang untuk mendistribusikan dan me-revoke akses pada objek tersebut. Tipe-tipe akses yang biasa dilakukan adalah dengan standard SQL yaitu operasi *select*, *insert*, *delete*, *update* dan *grant*. *User* dapat memberikan *hak* akses kepada objek yang mereka kontrol berdasarkan peraturan “perlu tahu” (*need to know*) ataupun “siapa yang saya suka” ataupun peraturan lainnya, sehingga dapat disimpulkan bahwa mekanisme DAC mengontrol akses berdasarkan identitas keseluruhan subjek dan objek serta *access privileges* dimana :

1. *Security Objects* adalah entitas pasif yang berisi atau menerima informasi (*database*, relasi, view, fakta, suatu *tuple*, juga segmen memori, printer, dan sebagainya)
2. *Security Subjects* adalah entitas aktif, yaitu dalam bentuk USER (manusia) atau proses operasi yang mewakili seorang *user*. Keamanan subjek bertanggung jawab pada perubahan bentuk *database* dan menyebabkan informasi mengalir dengan objek dan subjek yang berbeda.
3. *Access privileges* adalah hak-hak operasi yang diberikan pada subjek sesuai kebutuhan dan kepentingan subjek terhadap objek.

Sebelum menjelaskan bagaimana mekanisme DAC pada model *database*, berikut ini akan dijelaskan terlebih dahulu kebijakan DAC secara teknis.

3.1 Prinsip Dasar *Discretionary Access Control*

Untuk dapat memahami penerapan mekanisme DAC, ada baiknya kita mengetahui terlebih dahulu prinsip dasar DAC⁴:

Jika *O* merupakan himpunan dari *security objects*, *S* merupakan himpunan dari *security subjects*, *T* merupakan himpunan *access privileges* dan untuk merepresentasikan peraturan akses adalah *P* yang merupakan himpunan predikat⁵.

⁴ Prinsip ini didapat dari paper *Gunther Pernul, Database Security*, Department of Information Systems University of Essen

Tuple⁶ $\langle o,s,t,p \rangle$ ($o \in O, s \in V, t \in T, p \in P$) dinamakan peraturan akses (*access rule*) dan fungsi f didefinisikan untuk menentukan jika suatu wewenang $f(o,s,t,p)$ itu valid atau tidak yaitu:

$$F: O \times S \times T \times P \rightarrow \{True, False\}$$

Untuk setiap $\langle o,s,t,p \rangle$, jika $f(o,s,t,p)$ mempunyai nilai True maka subjek s memiliki **wewenang** t untuk mengakses objek o dengan range yang didefinisikan oleh predikat p .

Pada mekanisme DAC ini terdapat prinsip yang dinamakan **prinsip pendelegasian hak** (*principle of delegation of rights*) yaitu: suatu hak adalah bagian (o,t,p) dari peraturan akses. Seorang subjek s ke i (s_i) yang memiliki hak (o,t,p) diperbolehkan untuk mendelegasikan/memberikan haknya tersebut pada subjek yang lain misalnya s ke j (s_j) dimana $i \neq j$.

3.2 Prinsip Need To Know

Prinsip *Need to Know* merupakan salah satu peraturan yang dipakai dalam mekanisme DAC. Berikut ini adalah penjelasannya.

Setiap *security object* berhubungan dengan satu atau lebih *project*⁷ yang dinamakan *compartments*. Suatu *security subject* diperbolehkan untuk mengakses suatu objek bilamana suatu subjek memiliki pengetahuan (need to know) mengenai isi dari objek tersebut. Contoh dari prinsip ini adalah :

Rumah sakit memiliki berbagai data yang berbeda-beda baik itu data medis, data keuangan, data pribadi. *Compartments* dari hal ini adalah :

$$\text{Compartments:} \{ \text{data medis, data keuangan, data pribadi} \}$$

Lalu kita memiliki satu *compartment* untuk objek dengan data seperti berikut :

$$\text{Comp(O)} = \{ \text{data medis, data keuangan} \}$$

Maka : Hak subjek S diberikan bilamana $\text{Comp(O)} \subseteq \text{NTK(S)}$ artinya, suatu subjek dapat melihat data dari Comp(O) bila dia telah diberikan hak

⁵ Predikat digunakan untuk membatasi akses pada objek yang tergantung pada beberapa constraints.

⁶ Tuple adalah terminologi relasional formal untuk baris atau *record*. Satu *tuple* = satu baris = satu *record*.

⁷ *project* dalam hal ini menurut saya merupakan kumpulan data baik itu baik itu berupa database, tabel, ataupun view.

mengenai keseluruhan data (dalam hal ini *compartments*-nya) dan data pada $\text{comp}(O)$ adalah himpunan bagian dari *compartments*.

Prinsip ***Need to Know*** ini dibahas lebih lanjut lagi dengan peraturan pada *privileges* yaitu:

Read : Subjek S dibolehkan untuk membaca Objek O jika $\text{Comp}(O) \subseteq \text{NTK}(S)$

Write : Subjek S boleh me-*write* objek O jika $\text{Comp}(O) \supseteq \text{NTK}(S)$

Contoh dari prinsip ini adalah:

Compartments: { data medis(M), data keuangan (K), data pribadi(P) }

$\text{Comp}(O) = \{M, F\}$

$\text{NTK}(S1) = \{F\}$

$\text{NTK}(S2) = \{P\}$

$\text{NTK}(S3) = \{P, M\}$

$\text{NTK}(S4) = \{F, M, P\}$

Artinya:

$S1$ hanya dapat *write* pada O .

$S4$ hanya dapat membaca (*read*) O .

$S2$ dan $S3$ tidak dapat mengakses O .

3.3 Penerapan mekanisme DAC pada sistem *Database*

Penerapan DAC pada sistem *database* diaplikasikan dengan menggunakan bahasa *Structured Query Language* (SQL) dan salah satunya diterapkan dengan menggunakan ***view*** (dapat juga diterapkan pada tabel). *View* merupakan hal yang ideal untuk *security*, misalnya dalam:

1. Pengontrolan data-data baik yang *dependent* maupun yang *independent*
2. Pengontrolan secara statistik

Masalah yang timbul bilamana kita mengaplikasikan mekanisme DAC dengan *view* adalah:

1. Dengan hanya menggunakan satu *view*, tidak mungkin memisahkan akses ***read*** dan ***write***. Hal ini dapat kita atasi dengan membuat dua buah *view*

dengan struktur yang sama namun peraturan akses *read* dan akses *view* kita pisahkan. *View* tersebut dapat diakses oleh *user* (subjek) berdasarkan peraturan akses yang subjek miliki.

2. Terdapat sistem-sistem yang tidak siap bilamana terdapat *tuple* baru atau *tuple* yang di-*update* dan *tuple* tersebut harus memenuhi syarat predikat P pada definisi *view* yang dibuat. Untuk itu, predikat pada *view* harus mengikutsertakan aturan bilamana terdapat *tuple* baru ataupun *tuple* yang di-*update*.
3. Penambahan *tuple* baru melalui *view* akan menghasilkan nilai *null* dalam *database*-nya. Hal ini terjadi karena *view* merupakan hasil pemilihan *tuple-tuple* dari suatu atau beberapa tabel dan sifatnya adalah *read only*, sehingga subjek memang tidak dapat melakukan operasi *write*.

Terdapat lima operasi utama yang biasa dilakukan pada sistem *database* yaitu: *create*, *delete*, *write*, *read*, dan *execute*. Biasanya tipe-tipe akses yang ada identik dengan salah satu operasi tersebut ataupun kombinasi dari operasi-operasi yang ada, misalnya operasi *insert* pada tabel dilakukan dengan menggunakan operasi *create* dan *write* pada *tuple* di tabel tersebut.

Hampir semua sistem *database* yang mendukung DAC menyimpan peraturan akses dalam suatu **matriks kontrol**. Dalam bentuk yang sederhana, baris-baris pada matriks merepresentasikan *security subjects* dan kolom matriks merepresentasikan *security objects*. Irisan antara baris dan kolom matriks berisikan tipe akses dimana subjek memiliki wewenang pada objek.

Tabel 1 Tabel contoh Matriks Kontrol

Security	Objek 1	Objek 2
Subjek 1	read	Execute
Subjek 2	Create,read,write,delete	-

Dengan menggunakan DAC, informasi ini dapat direpresentasikan dengan menggunakan salah satu dari mekanisme:

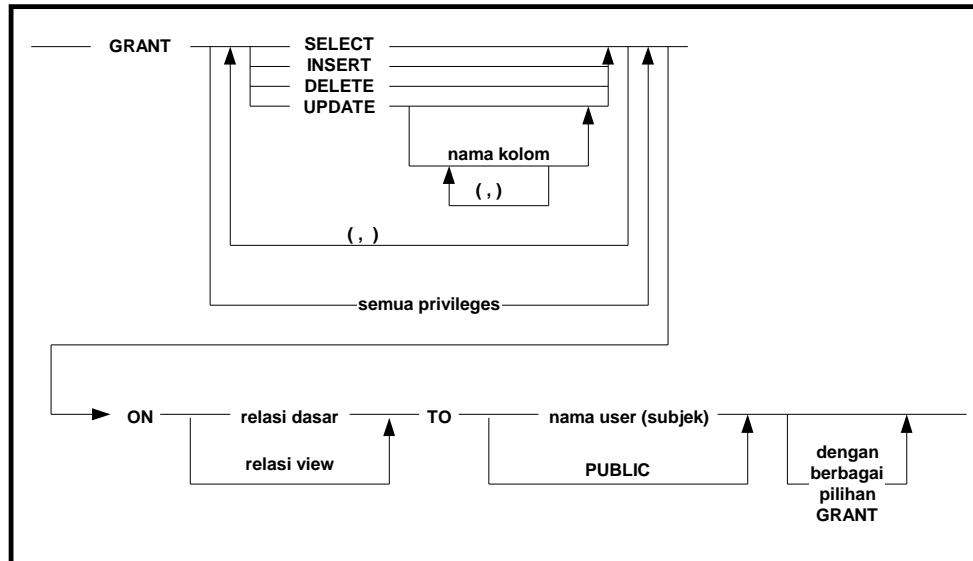
1. Kemampuan: subjek memiliki kemampuan untuk mengakses objek. Kemampuan tersebut dapat diberikan dari satu subjek ke subjek lainnya,

namu biasanya tidak dapat di-ubah tanpa adanya campur tangan *database administrator*.

2. Profil: profil adalah daftar dari objek-objek yang dimiliki oleh setiap subjek. Untuk membuat, menghapus dan merubah akses pada objek memerlukan banyak operasi-operasi karena profil dari *user* harus di-*update* pula.
3. *Access Control List* (ACL): penggunaan ACL biasanya dengan menggunakan matriks kontrol akses yaitu dengan mendeskripsikan daftar subjek yang berhubungan dengan objek-objek.
4. *Protection bits* (Bit Proteksi): selain ACL, terdapat pula bit yang berhubungan dengan objek-objek, menyatakan implementasi yang tidak lengkap dari model matriks kontrol akses karena akses pada subjek tidak dapat diterapkan pada *user* yang tunggal.
5. *Password*: setiap subjek memiliki *password* pada setiap objek yang dimilikinya, dimana *password* itu merupakan ‘tiket’ untuk mengakses objek tersebut.

Untuk menghasilkan suatu peraturan akses bagi subjek terhadap objek-objeknya, tipe akses yang dibuat menggunakan perintah GRANT dan REVOKE yang diterapkan pada pembuatan *view*. Dalam paper ini akan dibahas mengenai kedua akses tersebut.

Standar *grant* yang dilakukan untuk mekanisme DAC adalah sebagai berikut:



Gambar 1 Standar Perintah GRANT

Contoh dari standar ini adalah:

Misalkan kita memiliki tabel yang bernama supplier dengan atribut-atributnya:

Tabel 2 Struktur Data tabel supplier

Atribut	Tipe Data	Keterangan
kode_sup	char(6)	Primary key
nama_sup	char(30)	
alamat_sup	char(30)	
kota_sup	char(15)	
telpon_sup	char(21)	
fax_sup	char(16)	
hubungi_sup	char(21)	
kodepos_sup	char(5)	
saldo_awal	number(11)	

Peraturan akses/wewenang yang dibuat:

AUTHORITY AKS

GRANT SELECT(kode_sup, nama_sup, kota_sup), DELETE

ON supplier

TO dian,enrico;

Contoh ini memberikan penjelasan secara umum peraturan akses memiliki 4 komponen yaitu:

1. Nama, yaitu **AkS** (nama peraturan akses, AkS adalah akses supplier – **contoh**). Wewenang ini akan didaftarkan pada katalog dengan nama AkS ini.
2. Satu atau lebih *privilege*. Dalam contoh diatas, *privilege* yang digunakan adalah SELECT dan DELETE kemudian dispesifikasikan dengan pernyataan **GRANT**.
3. Nama tabel atau disini biasanya disebut *relation variables*⁸ dimana peraturan akses itu diaplikasikan (dalam contoh diatas diaplikasikan pada relasi supplier) dispesifikasi pada pernyataan **ON**.
4. Satu atau lebih *user* (lebih akuratnya lagi *user ID*), yang diberikan hak akses dengan jenis tertentu (*privilege*) pada *relation variables* tertentu, dispesifikasi dengan menggunakan pernyataan **TO**.

Perlu diingat, untuk *privilege* DELETE, kita tidak perlu menambahkan nama atribut-atribut, karena *privilege* DELETE merupakan *self-explanatory*.

Apa yang akan terjadi bilamana suatu subjek (atau seorang *user*) berusaha melakukan operasi tertentu pada beberapa objek dimana subjek tersebut tidak diberi hak?. Pilihan yang paling sederhana adalah menolak operasi yang subjek lakukan dengan menyediakan informasi diagnosa yang baik tentunya. Dalam situasi yang lebih sensitif, beberapa tindakan yang biasanya dilakukan adalah dengan mematikan program yang sedang dijalankan, mengunci *keyboard user*, dan sebagainya. Hal yang lebih baik dilakukan adalah dengan menyimpan data operasi-operasi yang *user* lakukan terhadap sistem *database* dalam file *log* supaya nantinya *database administrator* dapat membuat analisis terhadap usaha-usaha yang dilakukan oleh subjek dan langkah-langkah keamanan yang harus dilakukan. Salah satu usaha yang dapat dilakukan dalam menjaga keamanan objek-objek dari subjek yang tidak berwenang adalah dengan men-drop peraturan akses.

Standar untuk mem-drop peraturan akses adalah:

DROP AUTHORITY <nama peraturan akses>;

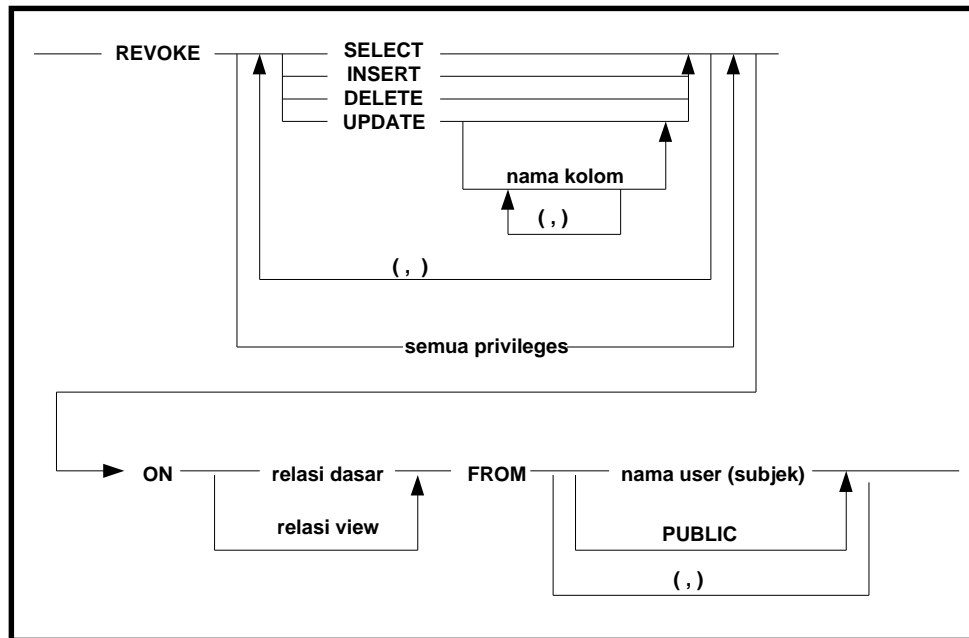
Contoh:

DROP AUTHORITY AkS;

⁸ Relation variables adalah variabel-variabel yang nilai-nilainya merupakan hasil relasi. Relation variables ini identik dengan tabel.

Perintah DROP juga dapat digantikan dengan REVOKE. Di beberapa paper mengenai keamanan database khususnya DAC, perintah REVOKE ini lebih sering digunakan dibandingkan dengan DROP.

Standar perintah *revoke* adalah sebagai berikut:



Gambar 2 Standar Perintah REVOKE untuk akses tertentu

Dari gambar diatas, kita dapat melihat bahwa perintah REVOKE diatas melakukan pembatalan akses kepada objek-objek sesuai dengan subjek-subjek yang ada. Contoh dari perintah REVOKE untuk standar gambar diatas adalah :

```

AUTHORITY RAKS
REVOKE SELECT(kode_sup, nama_sup, kota_sup), DELETE
ON supplier
FROM dian,enrico;

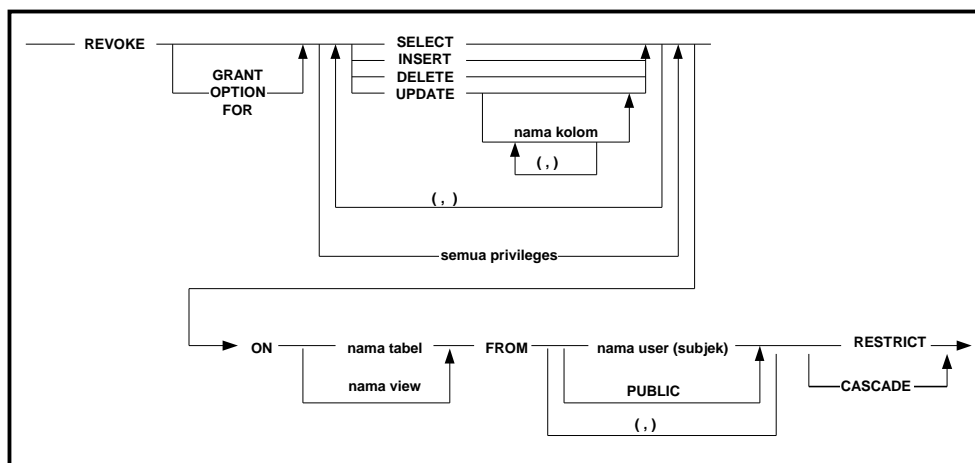
```

Contoh ini memberikan arti yaitu:

1. Nama peraturan akses yaitu RakS. Penamaan untuk peraturan akses ini bebas hanya saja penulis memberikan nama RakS yang bernama *revoke AkS*, dimana peraturan akses ini ditujukan untuk membatalkan akses untuk akses-akses pada relasi supplier (agar lebih mudah diingat).

2. Untuk *privilege* SELECT pada kolom kode_sup, nama_sup dan kota_sup dan *privilege* DELETE di-revoke/dibatalkan aksesnya.
3. Pembatalan akses diaplikasikan pada relasi *supplier* (tabel supplier)
4. Subjek yang ditunjukkan dalam pembatalan tersebut adalah dian dan enrico.

Disini dapat dilihat bahwa perintah REVOKE ini hanya membatasi subjek-subjek untuk tidak mengakses objek-objek yang tadinya diperbolehkan namun sekarang tidak diperbolehkan, sedang subjek-subjek itu sendiri masih aktif. Yang harus kita pertanyakan sekarang ini adalah bagaimana kita men-non aktifkan subjek-subjek yang tadinya aktif menjadi tidak aktif. Tidak aktif dalam hal ini berarti bahwa subjek-subjek tersebut bukan saja tidak dapat mengakses segala sesuatu dalam *database* tersebut namun juga *database* harus menghiraukan mereka bilamana mereka mencoba mengaksesnya. Salah satu cara yang dilakukan adalah dengan menghapuskan peraturan akses yang subjek miliki bukan menghapuskan aksesnya saja. Jadi disini kita menghapuskan hak yang telah diberikan sebelumnya oleh DBA (biasanya). Standar REVOKE pada **SQL92** ini digunakan untuk menerapkan metode DAC, yaitu:



Gambar 3 Standar Perintah REVOKE pada SQL92

Apabila kita melihat gambar 2 dan gambar 3 dengan seksama, kita dapat melihat bahwa perbedaan kedua gambar tersebut adalah bahwa pada gambar 3 tertera perintah GRANT setelah perintah REVOKE. Hal ini berarti bahwa seorang DBA dapat:

1. Membatasi hak subjek pada objek-objeknya, dalam arti bahwa sebelumnya subjek-subjek telah mendapatkan hak akses yang lebih besar dibandingkan pada waktu DBA membatasinya sekarang. Jadi Subjek tersebut telah aktif sebelumnya, bukan dari subjek yang tidak ada menjadi subjek yang ada.
2. Menghapus peraturan akses subjek. Hal ini berarti bahwa suatu atau seorang subjek tidak diinginkan lagi untuk dapat mengakses *database* yang ada, sehingga subjek tersebut “dimatikan”. Apabila subjek tersebut memang diperbolehkan untuk mengakses *database* tersebut lagi, DBA atau orang yang berwenang dapat membuat peraturan akses untuk subjek tersebut ataupun mengaktifkan kembali peraturan akses untuk subjek tersebut (biasanya peraturan akses disimpan dalam file).

Kita asumsikan bahwa men-drop suatu *relation variables* akan secara otomatis men-drop peraturan akses apapun yang diaplikasikan pada *relation variables* tersebut, sehingga bilamana kita menghapus suatu *relation variables*, otomatis peraturan akses yang berada dalam *relation variables* tersebut akan hilang juga.

Contoh untuk membatasi subjek terhadap objek:

AUTHORITY contoh3

REVOKE GRANT DELETE

ON supplier

FROM dian,enrico;

Dari contoh diatas, terlihat bahwa subjek akan ditolak hak aksesnya untuk melakukan operasi DELETE. Dapat dilihat bahwa konstruksi perintah dengan menggunakan REVOKE GRANT (istilah non formalnya : “tolak haknya”) itu akan membatasi hak akses subjek untuk memasuki objek dengan menggunakan operasi tertentu.

Beberapa contoh lainnya untuk mekanisme DAC dengan menggunakan beberapa *relation variables* dasar:

Tabel 3 Relation Variables S

S#	sname	status	kota
S1	Josa	20	Bandung
S2	Nita	10	Bandung
S3	Uway	30	Jakarta

S4	Marini	20	Jakarta
S5	Deri	30	Jakarta

Tabel 4 Relation Variables P

P#	Pnama	warna	berat	kota
P1	Nut	Merah	12.0	Bandung
P2	Bolt	Hijau	17.0	Jakarta
P3	Screw	Biru	17.0	Banjarmasin
P4	Screw	Merah	14.0	Bandung
P5	Cam	Biru	12.0	Jakarta
P6	Cog	merah	19.0	Bandung

Tabel 5 Relation Variables SP

S#	P#	Jumlah
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Tabel 6 Relation Variables (VIEW) LS

S#	snama	status	kota
S1	Josa	20	Bandung
S4	Marini	20	Jakarta

1. AUTHORITY CONTOH1

GRANT SELECT(P#, pnama, berat)

ON P

TO Luki, Anne, Indra;

Penjelasan:

User Luki, Anne, Indra dapat melihat *subset vertikal* pada *relation variables* dasar P. Contoh ini merupakan contoh untuk peraturan akses yang *value-independent*.

2. AUTHORITY CONTOH2

GRANT SELECT,UPDATE(snama,status), DELETE

ON LS

TO dian, lia;

Penjelasan:

Relation variable LS merupakan *view* dari *relation variable* S. *User* dian dan lia dapat melihat *horizontal subset* dari *relation variable* dasar S. Contoh ini merupakan contoh dari peraturan akses *value-dependent*. Walaupun *user* dian dan lia dapat melakukan operasi DELETE melalui *view* LS, namun mereka tidak dapat melakukan operasi INSERT dan UPDATE.

```
3. VAR SSPPR VIEW
  ( S JOIN SP SOIN ( P WHERE kota="bandung" ) {P#}
  {ALL BUT P#,jumlah};
```

```
AUTHORITY CONTOH3
GRANT SELECT
ON SSPPR
TO indra;
```

Penjelasan:

Ini merupakan contoh lain dari *value-dependent*; *User* indra dapat melakukan operasi SELECT terhadap *relation variables* *supplier* namun hanya untuk *supplier* (S) yang men-supply part yang ada di bandung.

```
4. AUTHORITY CONTOH4
GRANT SELECT,UPDATE(STATUS)
ON S
WHEN DAY() IN ('Mon', 'Tue', 'Wed', 'Thu', 'Fri')
AND NOW() >= TIME '09:00:00'
AND NOW() < TIME '17:00:00'
TO purchasing;
```

Penjelasan:

Disini kita mengembangkan perintah kita dengan menambahkan perintah WHEN untuk mendefinisikan pengontrolan yang kita inginkan. Peraturan akses contoh4 menjelaskan bahwa nilai status *supplier* dapat dirubah oleh *user* "purchasing" pada hari-hari tertentu. Ini merupakan contoh dari peraturan akses yang sifatnya *context-dependent* karena permintaan akses dapat diberikan berdasarkan dari isi peraturan akses yang telah dibuat oleh DBA atau *user* yang berwenang.

IV Kesimpulan

Metode *Discretionary Access Control* hanya merupakan salah satu dari banyaknya metode yang dapat digunakan untuk mengamankan *database*. Apabila dibandingkan dengan metode lainnya, dari aspek kemudahan penggunaan, tentunya metode DAC ini memang mudah digunakan dan mudah dimengerti, namun hal ini juga merupakan kelemahan dari metode ini. Hal ini dikarenakan apabila terdapat seorang *user* biasa yang mengetahui peletakan peraturan akses yang ditempatkan oleh DBA atau *user* berwenang maka *user* biasa tersebut dapat saja mengubah isi dari peraturan akses yang ada atau bahkan menghapusnya (masalah keamanan). Selain itu, DBA atau *user* yang berwenang juga harus dapat memperhatikan alur informasi karena mungkin saja peraturan akses yang dibuat menjadi menghambat alur informasi *database*.

DAFTAR PUSTAKA

1. C.J.Date, *An Introduction To Database Systems*, Seventh Edition, Addison-Wesley, 2000, Hal 506-512.
2. National Computer Security Center, *A guide to understanding Discretionary Access Control in trusted systems*, Version 1, National Computer Security Center, 30 September 1987.
3. Gunther Pernul, *Database Security*, Department of Information Systems University of Essen.