Pengaturan Kontrol Akses bagi Pendataan Data

This page is intentionally left blank

Asrianda

Pengaturan Kontrol Akses bagi Pendataan Data

UNIMAL PRESS

Judul: **Pengaturan Kontrol Akses bagi Pendataan Data**
X + 126 hal., 15 cm x 23 cm
Penulis: **Asrianda**
Editor**: Eva Darnila**

Cetakan Pertama: 2016

Perancang Sampul:
Penata Letak: **Eriyanto**
Pracetak dan Produksi: **Unimal Press**

Penerbit:

UNIMAL PRESS

Unimal Press
Jl. Sulawesi No.1-2
Kampus Bukit Indah Lhokseumawe 24351
PO.Box. 141. Telp. 0645-41373. Fax. 0645-44450
Laman: www.unimal.ac.id/unimalpress.
Email: unimalpress@gmail.com

ISBN: **978-602-1373-57-6**

KATA PENGANTAR

Dengan mengucapkan puji dan syukur kehadirat Allah SWT, dengan rahmat dan karunia-Nya yang telah memberi hidayah kepada Penulis untuk menyelesaikan buku yang berjudul "Pengaturan Kontrol Akses bagi Pendataan Data" tidak lupa selawat dan salam kepada Rasullullah SAW.

Rasa terima kasih Penulis ucapkan kepada Almarhum dan Almarhummah Ayahanda Muhammad dan Ibunda Hasanah, Kakanda Fauziah, Abanda Fakhrurrazie, Abanda Muzakkir, Al Chaidar dan keponakan penulis Dara Nurfika Sari yang telah banyak membantu Penulis dalam menyelesaikan Buku ini. Tidak lupa terima kasih kepada Bapak Prof. DR. Apridar, SE, M.Si, selaku Rektor Universitas Malikussaleh yang banyak mendorong penulis untuk lebih giat lagi bekerja dan berkarya dalam mengembangkan ilmu pengetahuan dan hasil penelitian berserta pengabdian demi kemajuan kampus tercinta ini. Dan juga pula saya ucapkan banyak terima kasih kepada teman-teman di program Studi Teknik Informatika Universitas Malikussaleh yang telah memberi semangat dan inspirasi kepada Penulis untuk menyelesaikan buku ini.

Dengan buku ini Penulis ingin membagi pengalaman yang sering Penulis alami dalam menyusun buku hasil penelitian yang telah penulis lakukan beserta referensi yang muktahir sekarang ini, dan Penulis juga merasa ilmu yang Penulis dapatkan selama ini belum ada apa-apanya di dunia kontrol akses, minimal dengan buku ini dapat membuka wawasan bagi para mahasiswa, peneliti juga masyarakat umum dan para teman-teman sejawat Dosen yang ingin mempelajari Kontrol Akses dan dapat mengembangkan lebih lanjut lagi.

Kritik dan saran dari pembaca sekalian sangatlah membantu bagaimana buku ini harus disusun dan disajikan lebih baik lagi.

Lhokseumawe, April 2016

Penulis

Daftar Isi

Kata Pengantar .....**Error! Bookmark not defined.**
Daftar Isi vi

BAB 1. PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 6
1.3 Batasan Masalah 7
1.4 Tujuan Penelitian 8
1.5 Manfaat Penelitian 8

BAB 2. TINJAUAN PUSTAKA 9
2.1 Kontrol Akses 9
2.1.1 *Read Access* 13
2.1.2 *Write Access* 13
2.1.3 *Execute Privilege* 13
2.1.4 *Delete Access* 14
2.2 Komponen Utama Kontrol Akses 14
2.3 Aspek Kebijakan dalam Keamanan 15
2.4 *Access Control Lists* (ACL) 16
2.5 Model–model Kontrol Akses 17
2.5.1 *Mandatory Access Control* (MAC) 18
2.5.2 *Discretionary Access Control* 19
2.5.3 *Role Based Access Control* (RBAC) 21
2.5.4 Entitas dalam RBAC 22
2.6 *Prinsip Least Privilege* 25
2.7 Pemisahan Tugas 25
2.8 Penyelenggaraan keamanan dalam RBAC 26
2.9 Struktur Role Based Access Control 27
2.10 Spesifikasi RBAC 30
2.10.1 *Core RBAC* 32

2.10.1.2 Fungsi Pendukung dalam Sistem *Core RBAC*.....35
2.10.1.3 *Review Functions*.....36
2.10.1.4 Hierarchical RBAC.....37
2.10.1.5 *Static Separation of Duty Relations* (SSD).....39
2.10.1.6 *Dynamic Separation of Duty Relations* (DSD).....44
2.11 Hambatan di RBAC.....46
2.11.1 *Separation of Duty* (SOD).....46
2.11.2 *Mutual Exclusion* (ME).....49
2.12 Penurunan Ke wenangan Pengguna RBAC.....50
2.13 Pelanggaran SOD oleh pengguna akhir di RBAC.....53
2.14 Pelanggaran SOD oleh pengguna akhir di RBAC.....56
2.15 Pelanggaran terhadap SOD oleh Administrator Keamanan.....57
2.16 Pekerjaan Besar Administrator Keamanan.....59
2.17 Kajian Riset Terkait.....60

BAB 3. METODE PENELITIAN.....63
3.1 Tujuan Penelitian.....63
3.2 Bahan-bahan.....63
3.3 Analisa Sistem.....64
3.4 Analisa Permasalahan.....64
3.5 Perancangan.....67
3.5.1 Penetapan *Role* ke Pengguna.....68
3.5.2 Pendaftaran bagi para pengguna.....70
3.5.3 Pengguna kehilangan password.....72
3.5.4 Pemberian Hak Akses ke *Role*.....74
3.5.5 Pemberian Hak Akses Ke Pengguna.....75
3.5.6 Pelimpahan wewenang pengguna ke Pengguna lain.....76
3.6 Pemberian Hak Akses dalam DSD ANSI 2004.....77
3.7 Hambatan Hak Akses dalam DSD.....79
3.8 Hambatan Hak Akses dalam Objek.....82
3.9 Hak Akses Negatif dalam Kegiatan Baca.....83
3.10 Hak Akses Negatif dalam Kegiatan Tulis.....84
3.11 Hak Akses Negatif dalam Kegiatan Cetak.....85

3.12 Kamus Data.....85

BAB 4. HASIL DAN PEMBAHASAN 87
4.1 Hasil 87
4.1.1 Pengaturan Hak Akses 87
4.1.2 Penugasan Hak Akses 89
4.1.3 Implementasi dengan menggunakan *Negative* MEP 93
4.2 Pembahasan.....98
4.3 Implementasi dalam Pendataan Masyarakat Miskin 98
4.3.1 *Role* dalam Sistem Pendataan Masyarakat Miskin 98
4.3.2 Pelimpahan Wewenang.....103
4.4 *Authentication - Access Control Mechanism*.....107
4.4.1 *Static Separation of Duty* (SSD).....108
4.4.2 *Dinamic Separation of Duty* (DSD).....110

BAB 6. KESIMPULAN DAN SARAN 115
6.1 Kesimpulan 115
6.2 Saran 116

DAFTAR PUSTAKA 117

This page is intentionally left blank

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Informasi merupakan aset yang penting dan sangat berharga bagi setiap organisasi. Sekarang ini informasi disimpan secara terdistribusi dan diakses oleh pengguna, baik itu melalui jaringan intranet maupun internet. Resiko terjadinya informasi jatuh ke pihak yang tidak diinginkan dan dapat digunakan untuk merugikan organisasi. Menghindari hal tersebut, dalam menjaga keamanan informasi secara efektif, aman dan kuat di organisasi merupakan suatu kebutuhan sangat mendesak.

Pengaksesan informasi sumber daya dalam organisasi menawarkan peningkatan fungsionalitas informasi yang dijalankan oleh organisasi tersebut, baik itu pada jaringan internal maupun eksternal. Organisasi harus mengembangkan dan menegakkan kebijakan akses berguna melindungi informasi sensitif dan rahasia sehingga mencegah terjadinya pengaksesan informasi yang tidak diinginkan oleh pihak tertentu, sehingga seluruh isi data terhindar dari pencurian, baik itu disengaja maupun tidak disengaja (Ferraiolo *et al*, 1992). Pengungkapan informasi sensitif mengenai pelanggan perusahaan, rencana strategis produk yang diluncurkan atau siapa menjadi kompetitor bukan saja dapat menyebabkan kerugian finansial yang besar, juga menghilangkan keunggulan kompetitif, kehilangan reputasi serta pertanggungjawaban secara hukum tetapi juga memberikan kompetitor kesempatan mengeluarkan produk lebih awal (Schweitzer, 1996).

Membatasi akses informasi dapat dilakukan pengguna dengan memanfaatkan mekanisme hambatan hak akses pengguna, berguna mencegah pencurian informasi. Pengguna dapat mendeteksi terjadinya kesalahan dalam melakukan akses informasi. Pencegahan lebih bermanfaat dari pada melakukan pendeteksian, jika terjadi pencurian informasi. Kontrol akses menyediakan sarana untuk mengontrol sistem informasi yang memiliki akses ke sumber daya. Pembatasan akses dapat dilakukan bagi pengguna yang berwenang, mekanisme dibuat untuk memastikan bahwa informasi tersedia bagi pengguna yang diizinkan untuk mengaksesnya.

Keamanan termasuk salah satu aspek yang sangat penting dalam sebuah sistem. Seringkali masalah keamanan kurang mendapat perhatian dari pemilik dan para pengelola sistem. Keamanan dalam sebuah sistem masih berada dalam urutan kedua maupun dalam urutan terakhir dari daftar hal-hal yang di anggap penting. Apabila hal tersebut mengganggu performansi dari sistem seringkali keamanan akan dikurangi ataupun ditiadakan. Kehilangan data yang sangat penting atau terganggunya sistem sebagai jalur transaksi yang mengatur aktivitas bisnis akan menjadikan masalah tersebut sebagai bencana bagi para pemilik data maupun pemanfaat sistem tersebut. Oleh sebab itu sebuah sistem sangat penting dijaga keamanannya, dan pihak-pihak tertentu saja yang boleh mengakses sistem tersebut.

Keamanan sumber daya komputasi membutuhkan biaya yang mahal juga memerlukan pengawasan yang rumit. Administrator harus melakukan pengontrolan yang sangat ketat dan dapat menentukan setiap pengguna yang mana saja berhak mendapatkan hak akses untuk mengakses suatu sumber daya. Pengguna akan diberikan password dan adminstrator dapat memperbaharui hak perizinan pengguna untuk mengakses ke sumber daya. Dalam menerapkan suatu kebijakan hak akses yang dilakukan oleh pengguna untuk menjaga keamanan dalam sebuah sumber daya kadangkala membutuhkan biaya mahal. Pelaksanaan untuk

menyederhanakan kebijakan akses ke sumber daya akan berdampak positif dalam penghematan biaya yang akan dikeluarkan.

Pengontrolan terhadap hak akses berguna untuk membatasi pengguna yang akan mengakses informasi dan menjaga keamanan atas informasi yang dianggap rahasia, sehingga informasi tersebut tidak dapat diakses oleh pengguna yang tidak diinginkan. Informasi hanya dapat diberikan kepada pengguna yang telah diberikan otoritas akses terhadap sistem tersebut. Oleh sebab itu sangat diperlukan kontrol akses guna membatasi hak-hak apa saja dapat diakses oleh pengguna. Sehingga keamanan atas informasi dapat terjaga dari pihak-pihak yang tidak diinginkan. Kontrol akses akan mendukung kerahasian dan integritas keamanan dalam sebuah sistem sehingga pengguna akan dibatasi hak apa yang boleh dilakukan dalam mengakses sistem tersebut.

Sebagian besar pihak melakukan manipulasi data dalam sebuah sistem dilakukan pihak internal di organisasi tersebut, mencegah terjadinya manipulasi data dapat dilakukan dengan melakukan pengaturan hak akses bagi pengguna. Kontrol akses merupakan cara mencegah ancaman keamanan internal. *Role Based Access Control* (RBAC) merupakan mekanisme pengelolaan sejumlah besar hak akses pada basis data berukuran besar yang fleksibel. Dibandingkan model kontrol akses tradisional yaitu *Mandatory Access Control* (MAC) dan *Discretionary Access Control* (DAC) (Habib, 2011).

Kabupaten Aceh Utara mempunyai jumlah penduduk terbanyak dari 13 Kabupaten dan Kota yang ada di Provinsi Aceh. Permasalahan terjadi di Kabupaten tersebut sewaktu melakukan pendataan penduduk miskin, bantuan diberikan kepada suatu gampong untuk penduduk miskin kadangkala data didapatkan tidak sesuai dengan yang diharapkan. Untuk mencegah terjadinya manipulasi data yang telah di data sebelumnya harus dibuat suatu pengaturan data yang ketat sehingga data tersebut dapat dijaga keamanannya dari pihak yang tidak diinginkan. Membuat sistem

menjadi lebih aman menerapakan *static separation of duty* (SSD) dalam RBAC menetapkan bahwa *mutual exclusive roles* atau hak akses tidak harus ditugaskan kepada subjek yang sama di waktu yang bersamaan (Strembeck, 2004). *Dynamic separation of duty* (DSD) dapat menjamin pengguna tidak dapat mengaktifkan *role* secara bersamaan yang dinyatakan berdasarkan *mutually exclusive*. DSD sangat mirip dengan SSD kecuali konflik dalam DSD dibuat berdasarkan *role* apakah dapat diaktifkan di seluruh *sessions*. DSD secara efektif dapat menghentikan terjadinya konflik di antara *role* yang diaktifkan secara bersamaan, tetapi tidak dapat menghalangi *role* yang diaktifkan secara berurutan (Jansen, 1998).

Dalam menjaga keamanan sumber daya komputasi akan membutuhkan biaya yang sangat mahal juga memerlukan pengawasan yang sangat rumit. Administrator harus melakukan pengontrolan yang sangat ketat dan dapat menentukan setiap pengguna yang mana saja berhak mendapatkan hak akses untuk mengakses suatu sumber daya. Pengguna akan diberikan password dan adminstrator dapat memperbaharui hak perizinan pengguna untuk mengakses ke sumber daya. Dalam menerapkan suatu kebijakan hak akses yang dilakukan oleh pengguna untuk menjaga keamanan dalam sebuah sumber daya kadangkala membutuhkan biaya mahal. Pelaksanaan untuk menyederhanakan kebijakan akses ke sumber daya akan berdampak positif dalam penghematan biaya yang akan dikeluarkan.

Kontrol akses akan membatasi pengguna dan pihak berwenang yang telah diberikan akses oleh sistem saja yang dapat melihat sebuah informasi, dan informasi akan terlindungi dari pihak-pihak yang tidak sah guna memodifikasi informasi tersebut. Pengguna yang telah dapat memasuki sistem, informasi yang boleh di akses oleh pengguna tersebut harus dibatasi, dan sistem harus membatasi hak apa saja yang boleh dilakukan oleh pengguna tersebut. Kontrol akses dapat memberikan kemampuan kepada pengguna dalam mengakses informasi tersebut juga berkemampuan untuk membatasi

dengan memberikan hak apa saja yang boleh diakses oleh pengguna tersebut, sehingga informasi tersebut terjamin keamanannya.

Kontrol akses memegang peranan yang sangat penting dalam menjaga keamanan sumber daya maupun sistem, supaya tidak dipakai maupun digunakan oleh pihak yang tidak diinginkan. kontrol akses dapat melakukan suatu kebijakan dalam komponen perangkat lunak, maupun komponen perangkat keras yang berguna untuk membatasi hak akses ke sumber daya. Dalam menjaga keamanan sistem tidak cukup hanya dengan menggunakan password, sidik jari atau yang lainnya tetapi dengan cara membatasi hak apa saja yang dapat diakses oleh pengguna tersebut sehingga sistem dapat memberikan izin kepada resource, dan apabila diterapkan dapat dilakukan dengan cara melakukan beberapa tingkat keamanan yang harus dilalui.

Role dinyatakan sebagai *mutually exclusive* satu dengan yang lainnya, maka pengguna mempunyai wewenang melakukan semua *mutually exclusive roles* dan hanya dapat menjalankan salah satu *mutually exclusive roles* disebabkan oleh *mutually exclusive* pada *role* tersebut. *Dynamic separation of duty* (DSD) adalah satu pengguna tidak dapat mengaktifkan lebih dari satu *mutually exclusive roles* pada waktu yang sama. Jika pengguna ingin mengaktifkan *mutually exclusive roles* lain, maka pengguna harus keluar sebelum mengaktifkan *mutually exclusive roles*, hal ini berarti pengguna dapat memiliki *mutually exclusive roles* ganda, tetapi tidak dalam waktu yang sama (ANSI, 2004).

Penerapkan SSD dalam pendataan masyarakat miskin di Kabupaten Aceh Utara mengalami kendala, sewaktu menetapkan *role* bagi pengguna yang melakukan pendataan atau pengguna tersebut seorang RT di *gampong* dan pengguna itu termasuk dalam kategori miskin, oleh sistem pengguna tersebut ditolak karena SSD hanya memperbolehkan satu *role* dimiliki oleh satu pengguna. Dengan menggunakan DSD dapat menyelesaikan permasalahan di atas, tetapi hal tersebut akan mengalami kendala yang diakibatkan

rawan terjadinya manipulasi, misalnya pegawai negeri sipil (PNS) tidak termasuk dalam kategori miskin tetapi jika ada bantuan PNS tersebut akan dimasukkan sebagai penerima bantuan. Hal ini disebabkan DSD dapat mengaktifkan banyak *role* walaupun dalam waktu yang berbeda sehingga rawan terjadinya manipulasi data yang disimpan ke dalam database. Dalam penelitian ini akan mengimplementasikan hambatan RBAC yaitu dengan DSD sesuai fungsi dan mekanismenya, sehingga pengguna RBAC tidak akan kehilangan wewenang yang dimilikinya.

Penelitian difokuskan dengan mengimplementasikan RBAC dengan salah satu hambatan yaitu DSD dari segi *Mutually Exclusive* (ME) terhadap tingkat hak akses bukan dengan *role*, dan dapat diterapkan dalam menjaga keamanan data maupun sumber daya baik itu database maupun perangkat lunak lainnya. Dapat diterapkan dalam menjaga keamanan data maupun sumber daya baik itu database maupun perangkat lunak lainnya. Penelitian ini menekankan kemampuan untuk menspesifikasi dan menerapkan lebih luas kebijakan kontrol akses dan mengurangi jumlah pengguna yang melakukan akses terlarang serta meningkatkan produktivitas administrasi pada sumber daya. Penelitian ini menekankan kemampuan untuk menspesifikasi dan menerapkan lebih luas kebijakan kontrol akses dan mengurangi jumlah pengguna yang melakukan akses terlarang serta meningkatkan produktivitas administrasi pada sumber daya.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas pengaturan kontrol akses dengan menerapkan hambatan DSD di RBAC pada tingkat role, sistem mungkin aman dari ancaman keamanan internal namun pengguna RBAC akan kehilangan otoritasnya sampai batas tertentu. Peneliti mengusulkan model berdasarkan pembagian role di tingkat batasan pada hak akses.

Berdasarkan latar belakang di atas, maka dapat dirumuskan masalah adalah:

1. Bagaimana mengalokasikan layanan yang boleh diakses oleh pengguna yang diatur oleh administrator.
2. Bagaimana mengembangkan Role-Based Access Control (RBAC), yang menyediakan akses ke sumber daya berdasarkan tugas dan hak pengguna yang telah diberikan oleh sistem.
3. Bagaimana cara pelimpahan wewenang dari satu pengguna ke pengguna lain dengan cara memberikan pembatasan masa aktif kemudian wewenang tersebut akan dikembalikan kepada pengguna awal.
4. Bagaimana cara melakukan pemberian penugasan penanggung jawab kepada pihak yang berwenang untuk meminimalkan pembatasan masa aktif yang telah diberikan kepada pihak yang berwenang dalam menjaga keamanan sumber daya.

1.3 Batasan Masalah

Dalam penelitian ini perlu adanya batasan masalah agar persoalan yang dibahas tidak menyimpang dari hal-hal yang telah ditentukan sebelumnya. Selain itu, batasan masalah dapat membuat penelitian lebih terarah dan memudahkan pembahasan, sehingga tujuan penelitian dapat tercapai. Adapun batasan masalah yang harus diperhatikan dalam penelitian ini adalah sebagai berikut:

1. Hanya membahas hambatan di tingkat hak akses pada DSD bukan hambatan role di metode RBAC
2. Pengujian menggunakan RBAC yang dilakukan dalam database Microsoft
3. SQL Server 2000 dan Pemograman Microsof Visual Basic 6.0.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian tesis ini adalah:

1. Mengimplementasikan RBAC menggunakan DSD dengan *Mutually Exclusive* pada hambatan hak akses bukan hambatan di *role*.
2. Mempelajari tentang pembagian hak akses bukan berdasarkan role yang telah ditentukan dalam standarisasi RBAC dalam pengaturan kontrol akses pendataan masyarakat miskin
3. Untuk menerapkan otentikasi dan otorisasi dalam pengaturan kontrol akses pendataan masyarakat miskin di Kabupaten Aceh Utara.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut:

1. Untuk Memudahkan pengaturan kontrol akses para pengguna dalam pendataan masyarakat miskin di Kabupaten Aceh Utara
2. Menghindari hambatan yang akan terjadi antar pengguna, pemisahan tugas secara dinamis dan statis dengan menggunakan RBAC.
3. Memudahkan administrator dalam mengontrol para pengguna dalam melakukan pengaturan kontrol akses pendataan masyarakat miskin.

-⌂-

BAB 2

TINJAUAN PUSTAKA

2.1 Kontrol Akses

Kontrol akses adalah pusat keamanan dalam sebuah komputer dengan cara membatasi pengguna untuk mengakses sumber daya, dan dimotivasi oleh kebutuhan untuk membocorkan akses dalam memperoleh informasi yang tersedia di sumber daya. Mekanisme keamanan sistem komputer yang paling penting berusaha untuk menjamin arus informasi tetap aman. Berarti tidak ada aliran yang tidak sah bekerja di dalam sistem komputer sehingga informasi yang berjalan di dalam sistem tersebut harus benar-benar aman. Seperti dalam pemerintahan ataupun sistem kemiliteran, keamanan akan mensyaratkan bahwa proses dalam melakukan pentranferan data pada sistem keamanan yang mempunyai proteksi tingkat tinggi ke para pengguna sampai ke tingkat yang lebih rendah, dapat melakukan pentranferan serta melakukan pencegahan untuk membaca data secara langsung sehingga *file* yang mempunyai klasifikasi rahasia akan tetap terjaga dan juga dihambat secara tidak langsung dalam mengakses informasi tersebut. Pencegahan juga dilakukan untuk mencegah para pengguna yang cerdik dengan cara berkoloborasi dengan pengguna yang memiliki wewenang untuk mengakses informasi tersebut (Rotenberg, 1974).

Proses ini akan diidentifikasi siapa yang sedang melakukan request untuk mengakses suatu resource tertentu dan apakah pengguna tersebut memiliki hak akses (authorized) untuk mengakses *resource* tersebut. Kontrol akses akan memproteksi data terhadap

akses yang dilakukan oleh pengguna yang tidak memiliki hak akses terhadap resource tersebut. Kontrol akses mendukung dengan terwujudnya *confidentiality* dengan memastikan data hanya bisa dilihat oleh orang yang memiliki hak akses untuk melihat data tersebut, serta mendukung *integrity* dengan memastikan data hanya bisa ditulisi dan diubah oleh pengguna yang memiliki hak akses untuk melakukan penulisan ataupun pengubahan terhadap data tersebut.

Kontrol akses adalah proses dalam melakukan mediasi setiap permintaan ke sumber daya dan data akan dikelola oleh sistem dengan cara menentukan apakah permintaan tersebut harus diberikan atau ditolak. Pada model kontrol akses secara tradisional akan membatasi skenario yang akan muncul, dengan memerlukan pengembangan secara terbuka di mana keputusan yang akan diberikan untuk melakukan akses tergantung pada sifat (atribut) dari pemohon ketimbang identitas. Pembatasan terhadap kontrol akses akan ditegakkan, walaupun itu berasal dari otoritas yang berbeda (Linares, 2008)

Dalam melakukan proses mediasi pada setiap permintaan ke sumber daya, dan data akan dikelola oleh sistem dengan cara menentukan apakah permintaan tersebut harus diberikan atau ditolak. Pada model kontrol akses secara tradisional akan membatasi skenario yang akan muncul, dengan memerlukan pengembangan secara terbuka di mana keputusan yang diberikan untuk melakukan akses tergantung pada sifat (atribut) dari pemohon ketimbang identitas.

Kontrol akses adalah sekumpulan metode yang dipergunakan untuk melindungi aset informasi, meskipun informasi tersebut dapat diakses oleh setiap orang tetapi akan tetap memerlukan perlindungan terhadap informasi yang lainnya. Kontrol akses akan mendukung kerahasian dan integritas dari sebuah sistem supaya sistem itu aman, serta akan melindungi kerahasian informasi dari orang yang tidak berhak untuk mendapatkan informasi tersebut. Penggunaan kontrol

akses dilakukan untuk memastikan bahwa orang yang berhak saja yang dapat mengakses informasi tersebut, dan akan memberikan kemampuan untuk mendikte informasi yang mana saja boleh dilihat ataupun dimodifikasi oleh pengguna.

Dalam bidang komputasi akan dimotivasi oleh kebutuhan untuk membocorkan akses ke informasi yang tersedia di dalam sumber daya komputasi dan entitas sehingga mempunyai wewenang dalam mengakses informasi di sistem tersebut. Suatu entitas adalah istilah umum yang mengacu pada agen aktif yang mampu untuk memulai atau melakukan perhitungan baik itu pengguna pada waktu memanggil program atau mengakhirinya. Agen dalam pemograman tersebut bertidak atas nama pengguna, dan *a running daemon process, a thread of execution, a hosting system, or a networking device* (Benantar, 2006).

Kontrol akses berfungsi untuk mengatur segala proses yang berhubungan dengan pengontrolan akses. Sebuah entitas yang meminta akses ke sebuah sumber daya disebut sebagai akses dari subyek. Sebuah subyek merupakan entitas yang aktif karena dia menginisiasi sebuah permintaan akses. Sebuah sumber daya yang akan diakses oleh subyek disebut sebagai obyek dari akses. Obyek dari akses merupakan bagian yang pasif dari akses karena subyek melakukan aksi terhadap obyek tersebut. Tujuan dari kebijakan kontrol akses adalah mengijinkan hanya subyek yang mempunyai otorisasi yang bisa mengakses obyek yang sudah diijinkan untuk diakses. Hal ini mungkin juga ada subyeks yang sudah mempunyai otorisasi tapi tidak melalukan akses terhadap spesifik obyek tertentu.

Untuk mengatur segala proses yang berhubungan dengan pengontrolan akses. Sebuah entitas yang meminta akses ke sebuah sumber daya disebut sebagai akses dari subyek. Sebuah subyek merupakan entitas yang aktif karena dia me nginisiasi sebuah permintaan akses. Sebuah sumber daya yang akan diakses oleh subyek disebut sebagai obyek dari akses. Obyek dari akses merupakan bagian yang pasif dari akses karena subyek melakukan

aksi terhadap obyek tersebut. Tujuan dari kebijakan kontrol akses adalah mengizinkan hanya subyek yang mempunyai otorisasi yang bisa mengakses obyek yang sudah diizinkan untuk diakses. Hal ini mungkin juga ada subyek yang sudah mempunyai otorisasi tapi tidak melak ukan akses terhadap spesifik obyek tertentu.

Kontrol akses adalah sekumpulan metode yang dipergunakan untuk melindungi aset informasi, meskipun informasi tersebut dapat diakses oleh setiap orang tetapi akan tetap memerlukan perlindungan terhadap informasi yang lainnya. Kontrol akses akan mendukung kerahasian dan integritas dari sebuah sistem supaya sistem itu aman, serta melindungi kerahasian informasi dari orang yang tidak berhak untuk mendapatkan informasi tersebut. Penggunaan kontrol akses dilakukan untuk memastikan bahwa orang yang berhak saja yang dapat mengakses informasi tersebut, dan memberikan kemampuan untuk mendikte informasi yang mana saja boleh dilihat ataupun dimodifikasi oleh pengguna.

Model akses dapat dikategorikan ke dalam kemampuan untuk membaca atau menulis informasi ke dalam ruang penyimpanan yang akan melakukan proses, ataupun pada penyimpanan sekunder pada jaringan dan perangkat lainnya yang menjalankan sistem tersebut. Kemampuan ini didapat secara implisit dengan mempunyai hak istimewa yang langsung dimiliki oleh entitas, dan akan bertindak baik secara langsung maupun secara tidak langsung melalui layanan yang diberikan oleh komputasi untuk mengakses sistem tersebut (Abadi et al, 1993).

Pendekatan secara tradisional dalam melakukan otorisasi untuk mendistribusikan ke beberapa pengguna baik itu secara *single user* (peminta akses ke sumber daya ataupun pemiliknya tergantung pada model yang dilakukan) bertanggung jawab serta membuktikan akses yang diperbolehkan untuk dilakukan serta membuktikan bahwa pengguna dapat mengambil mandat dari pengguna yang lainnya (Bauer *et al*, 2005).

Pada *level* yang lebih tinggi, kebijakan akses dalam pengendalian mekanisme dalam menterjemahkan hak akses bagi pengguna untuk mengakses informasi dalam sebuah sistem, hal tersebut akan dilakukan dalam beberapa kebijakan yang berguna untuk menjaga keamanan misalnya lookup tabel sederhana. Hal tersebut akan berguna untuk memberikan ataupun menolak pengguna yang mengakses sistem tersebut. Meskipun cara tersebut tidak memiliki standar yang dapat diterima dengan baik dalam menentukan kebijakan yang akan diterapkan (Ferraiolo *et al*, 2003). Tabel di bawah ini adalah contoh dari hak akses pengguna.

2.1.1 *Read Access*

Kemampuan yang diberikan kepada pengguna untuk melihat informasi pada sumber daya sistem seperti *file*, data dan lain sebagainya. Kemampuan tersebut yang telah diberikan adalah tidak dapat mengubah, menghapus, ataupun menambah tetapi hanya dapat melihat serta menyalin informasi tersebut.

2.1.2 *Write Access*

Kemampuan yang diberikan kepada pengguna untuk dapat menambah, mengubah, atau menghapus informasi yang ada dalam sumber daya. Kemampuan untuk melakukan pembacaan informasi (*read access*) di dalam sumber daya juga telah dapatkan oleh pengguna yang mendapatkan hak akses ini.

2.1.3 *Execute Privilege*

Hak akses yang diberikan kepada pengguna untuk menjalankan program yang ada di dalam sumber daya.

2.1.4 *Delete Access*

Memungkinkan pengguna untuk menghapus sumber daya sistem yang ada (misalnya *file*, *record*, direktori dan program). Namun, jika pengguna memiliki akses tulis namun tidak menghapus akses, mereka bisa menimpa *field* atau *file* tersebut.

2.2 Komponen Utama Kontrol Akses

Langkah pertama untuk setiap sistem kontrol akses diawali dengan melakukan analisis integritas sistem dengan menggunakan teknologi yang akan membantu hardware untuk menjalankan sistem tersebut. Hal tersebut akan membuat perangkat lunak dapat dijalankan dengan aman sehingga akan mencegah penyerangan untuk mengakses sumber daya. Kerentanan yang terjadi meliputi otentikasi yang lemah sehingga perlindungan terhadap aliran komunikasi akan menjadi sangat rawan dan mengakibatkan penyimpanan data dalam keadaan berbahaya. Tahap kedua identitas yang terpercaya merupakan proses penting untuk melakukan identifikasi dan otentikasi pengguna. Komputer dan aplikasi dapat memastikan bahwa para pelaku yang akan memasuki sistem tersebut sebagai pelaku yang sah. Tahap ketiga alokasi komposisi kontrol akses dapat diandalkan berdasarkan role yang telah diberikan oleh sistem. Tahap keempat adalah perlindungan informasi yang terfokus pada perlindungan data sepanjang siklus hidupnya dimanapun data tersebut disimpan atau data yang akan dikirimkan. Hal tersebut juga dapat berfungsi untuk melindungi hak cipta, maupun melindungi informasi rahasia (Microsoft, 2005).

Pada sebuah contoh keamanan pada perancangan sistem yang memerlukan Mekanisme perlindungan yang penting dalam komputer dan berkembang sesuai dengan perkembangan aplikasi dalam menjaga keamanan sistem komputasi ada tiga hal penting yang harus diterapkan (Saltzer *et al*, 1975) antara lain:

1. Komputasi dasar yang terpercaya adalah perangkat keras dan perangkat lunak sistem harus mampu menjaga kerahasiaan dan integritas data.
2. Otentikasi adalah harus diketahui siapakah yang dapat mengawasi sistem tersebut. Misalnya, pengguna jika ingin melakukan penghapusan berkasnya harus membuktikan apakah perintah tersebut itu miliknya, dan bukan pengguna lain yang tidak di ketahui identitasnya.
3. Otorisasi atau kontrol akses adalah apakah pengguna yang melakukan kegiatan tersebut benar-benar dapat dipercaya, apakah hal itu benar perintah yang ia lakukan. Misalnya, benar bahwa pengguna tersebut yang memberikan perintah untuk melakukan penghapusan berkas itu.

Penentuan untuk melakukan pendekatan yang akan mengatasi kompleksitas dalam pengembangkan sistem. Hal tersebut sangat diperlukan untuk mengidentifikasi pendekatan yang *holistic* sehingga dapat digunakan untuk mendefinisikan persyaratan akses yang akan diperlukan dalam meningkatkan kemudahan yang berguna untuk melakukan pemprosesan identifikasi. Hal ini akan mengakibatkan kontrol akses dapat digunakan pada segala aplikasi, dan berguna dalam melindungi sistem tersebut dari pihak yang tidak diinginkan.

2.3 Aspek Kebijakan dalam Keamanan

Sekarang ini perhatian kebutuhan suatu penanganan keamanan dalam sebuah organisasi baik itu pemerintahan maupun perusahaan bisnis sangat tergantung pada informasi pengolahan data dalam melakukan sebuah operasional bisnis yang dijalankannya, baik itu bidang keuangan maupun informasi teknologi lainnya (Ruthberg *et al*, 1989).

Integritas dan ketersedian atas kerahasian kunci keamanan da lam sistem perangkat lunak, database dan jaringan keamanan data telah menjadi kekhawatiran yang sangat besar di dalam segala sektor. Masalah korupsi atau pengungkapan data yang dilakukan dengan cara tidak sah atau pencurian sumber daya perusahaan dapat mengganggu sebuah organisasi yang dimiliki oleh perusahaan tersebut. Akibat hal tersebut akan berdampak secara hukum sehingga kepercayaan publik akan berkurang (Roskos *et al*, 1989).

Mekanisme perlindungan awal terdiri dari perangkat keras dan sistem operasi yang berguna untuk menjalankan perangkat tersebut. Pengontrolan akses dalam sistem adalah awal dari pertahanan utama untuk melindungi pembacaan informasi yang dilakukan oleh pengguna yang tidak sah. Sistem serta pengaksesan dalam sumber daya akan mengidentifikasi para pengguna sehingga pengguna yang terpercaya saja yang dapat mengakses sistem tersebut.

Identitas dalam manajemen tersebut telah berkembang sebagai suatu disiplin ilmu tersendiri, yang bertujuan untuk mengurangi biaya pemeliharaan repositories. Identitas yang mungkin ada dan berpotensi untuk digunakan dalam sistem tersebut, sehingga penegakan konsistensi dalam pencapaian pemetaan identitas menjadi ambigu dalam mewakili entitas yang sama atau beberapa kelompok entitas yang berkoloborasi secara bersama (Bishop, 1988).

2.4 *Access Control Lists* (ACL)

Salah satu model kontrol akses yang umum digunakan adalah dengan menggunakan *access control list* (ACL). ACL berisi daftar untuk mengidentifikasikan pengguna yang memiliki akses terhadap objek tertentu, termasuk tipe dan ruang lingkup dari akses tersebut. Bila menggunakan ACL pada setiap bagian data, database maupun aplikasi yang memiliki daftar pengguna, sehingga pengguna yang berkaitan dengan akses tersebut yang diperbolehkan untuk mengakses informasi.

Administrator dalam menjaga keamanan kontrol akses sangat mudah untuk melihat pengguna yang memiliki hak akses dalam mengakses informasi. Perubahan hak akses yang diberikan kepada pengguna sangat mudah dilakukan oleh administrator. Administrator hanya menambahkan atau menghapus pengguna dari ACL. Setiap rangkaian data atau aplikasi yang dijalankan memiliki daftar ACL tersendiri, bisa jadi daftar informasi tersebut ada ataupun tidak, hal tersebut sesuai dengan pemberian hak akses yang diperoleh oleh pengguna dari administrator. Jika terjadi pelanggaran keamanan yang dilakukan oleh pengguna, administrator akan menemukan potensi pelanggaran keamanan tersebut. Sehingga hak akses yang diberikan kepada pengguna dapat ditarik kembali oleh administrator, hal ini menjadi masalah dalam ACL. Administrator akan memeriksa satu per satu daftar dalam ACL kemudian akan menghapus pengguna dari daftar yang ada dalam ACL. Ketika pengguna dibutuhkan pada tanggung jawab yang berbeda pada organisasi, hal tersebut akan menjadi permasalahan yang rumit.

Administrator bukan hanya menghilangkan pengguna dari ACL, administrator harus menentukan izin yang mana perlu untuk dihilangkan, atau dibiarkan saja, juga izin mana harus dimodifikasikan. Atas permasalahan tersebut administrator akan berupaya untuk memperbaiki daftar dalam ACL. Dalam beberapa kasus lain, aturan yang lebih terperinci dapat diterapkan untuk ACL dalam membatasi akses ke bagian tertentu dalam mengakses sumber daya. Hal ini menunjukkan bahwa ACL membutuhkan biaya yang sangat tinggi dan pelaksanaan administrasi juga mengalami kesulitan (Sandhu *et al*, 2004).

2.5 Model–model Kontrol Akses

Model dalam kontrol akses sangat berfungsi untuk menentukan jenis kontrol akses yang diperlukan dalam mendukung kebijakan keamanan. Model kontrol akses merupakan sebuah

framework yang menjelaskan bagaimana subjek mengakses objek. Ada tiga tipe utama model kontrol akses yaitu *mandatory, discretionary*, dan *role-based*. Tiap tipe model memakai metode berbeda untuk mengkontrol bagaimana subjek mengakses objek dan mempunyai kelebihan serta keterbatasan masing- masing. Beberapa model dipakai secara eksklusif dan kadang-kadang model tersebut dikombinasikan sehingga mampu mencapai tingkat keperluan keamanan yang dibutuhkan.

2.5.1 *Mandatory Access Control* (MAC)

MAC adalah sebuah mekanisme untuk mengontrol pengguna atau suatu proses yang memiliki akses ke sumber daya dalam sebuah sistem. Kebijakan yang dilakukan oleh MAC ditentukan untuk memfasilitasi pengelolaan dalam memelihara pengontrolan akses ke sumber daya. Perencanaan dalam sebuah sistem MAC mempunyai tiga bentuk pengontrolan yang harus dipertimbangkan yaitu *policies*, *models* dan *mechanisms*. Kebijakan MAC adalah suatu persyaratan yang sangat menentukan dalam mengelola akses, siapa yang mengelola akses tersebut, dan dalam keadaan apa serta informasi apa yang dapat diakses. Kebijakan tersebut dapat menjangkau beberapa platform komputasi dan aplikasi. Dalam mengevaluasi dan menganalisis MAC secara eksklusif dapat dilakukan dengan banyak mekanisme, model keamanan biasanya mengambarkan sifat dari sistem yang menggunakan MAC tersebut. Model dalam MAC adalah suatu kebijakan keamanan yang diterapkan dalam sistem dan berguna untuk membuktikan keterbatasan teoritis dari suatu sistem (Hu *et al*, 2011).

Kebijakan akses yang dilakukan oleh MAC berdasarkan ketentuan yang ditentukan oleh otoritas pusat (Samarati *et al*, 2011). Keputusan kebijakan kontrol akses di MAC ditentukan oleh central *authority*, bukan dilakukan oleh para pemilik individu dari objek tersebut, dan pemilik tidak dapat mengubah hak akses yang telah

dibuat (Pfleeger, 1997). *Mandatory access control* didasarkan pada kebijakan pengembangan sistem yang tidak dapat diubah oleh pengguna perseorangan.

MAC akan memperlakukan kontrol akses berdasarkan keamanan informasi yang melekat pada pengguna dan objek. Hal ini menunjukkan bahwa hubungan tersebut tidak dapat dirubah oleh pemilik objek. MAC dapat mengetahui akses yang akan dilakukan antara subjek dan objek secara konsisten. Jika terjadi duplikasi pada objek, kontrol akses akan berhubungan dengan objek semula dan akan menerapkan hal tersebut pada objek yang terduplikasi (Na *et al*, 2000).

2.5.2 *Discretionary Access Control*

Discretionary Access Control (DAC) adalah model akses sumber daya berdasarkan identitas pengguna. Seorang pengguna akan diberikan hak akses ke sumber daya dengan ditempatkan pada sebuah ACL (*Access Control List*) yang berhubungan dengan sumber daya tersebut. Pengentrian pada sumber daya dengan ACL dikenal sebagai ACE (Access *Control Entry*). Ketika seorang pemakai atau sekelompok pemakai sebagai pemilik dari suatu obyek dalam model DAC, pengguna dapat memberikan izin ke pengguna lain atau kelompok lain (Sandhu *et al*, 1999).

Subjek memiliki otoritas dengan batasan tertentu, dalam menentukan objek-objek apa yang dapat diakses. Contohnya adalah penggunaan daftar kontrol akses (*access control list*). Daftar kontrol akses merupakan sebuah daftar petunjuk bagi pengguna yang memiliki hak akses ke sumber daya tertentu. Misalnya daftar yang menunjukkan subjek atau pengguna yang memiliki akses ke objek dan hak apa yang mereka miliki sehingga mempunyai kaitan dengan objek.

Akses dibatasi berdasarkan otorisasi yang diberikan pada pengguna. Hal ini berarti bahwa subjek diizinkan untuk menentukan tipe akses apa yang dapat terjadi pada objek yang mereka miliki. Jika organisasi menggunakan model *discretionary access control*, administrator jaringan dapat mengizinkan pemilik sumber daya mengontrol siapa yang dapat mengakses *file*/sumber daya tersebut.

Kontrol akses pada DAC akan membat asi akses terhadap objek berdasarkan identitas para pengguna ataupun grup. Hubungan antara kontrol akses dengan pemilik dari objek tersebut akan sangat menentukan sehingga akan sangat sulit mempertahankan konsistensinya. Kontrol akses dapat dibentuk dalam satu unit subjek-objek, dan pengguna yang telah memiliki izin dapat mengizinkan pengguna lainnya untuk mengakses sumber daya tersebut. Kebijakan dari kontrol akses dapat diubah oleh pemiliknya sendiri, dan pemilik tersebut secara optional dapat melimpahkan kewenangan yang mereka miliki sehingga akan mengalami kesulitan untuk mengontrol informasi secara efisien (Na *et al*, 2000).

Implemetasi umum dari *discretionary access control* adalah melalui *access control list* (ACL) yang dibuat oleh pemilik, diatur oleh administrator, dan dijalankan oleh sistem. Dengan demikian kontrol ini tidak termasuk dalam lingkungan kontrol yang terpusat sehingga dapat memberi kemampuan pada pengguna untuk mengakses informasi secara dinamis, kebalikan dari aturan yang lebih statis pada mandatory access control. Mekanisme *discretionary access control* (DAC) secara umum efektif namun mempunyai kelemahan tertentu. Secara khusus, pengguna tidak sah dapat menipu pengguna sah untuk memperlihatkan data sensitif.

Dalam sebuah kebijakan keamanan dan yang dibutuhkan baik oleh pemerintah sipil maupun organisasi bisnis yang intinya keamanan tidak akan diberi persyaratan tapi setiap organisasi akan memiliki persyarakatan keamanan yang unik, karena keamanan yang dilakukan banyak yang mengalami kesulitan untuk memenuhi MAC

dan kontrol DAC. DAC adalah kontrol akses yang mekanismenya memungkinkan pengguna sistem tersebut untuk mengizinkan atau melarang pengguna lain dalam melakukan akses ke objek yang di bawah kendali mereka (Rochaeli *et al*, 2005).

2.5.3 *Role Based Access Control* (RBAC)

Role Based Access Control (RBAC) telah banyak diterapkan pada aplikasi yang berhubungan dengan pengontrolan akses sumber daya. Berbeda dengan kontrol akses tradisional, RBAC tidak menerapkan hak akses para pelaku atau subjek, sebaliknya memberikan hak akses untuk peran (*roles*). Perancang kebijakan atau administrator sangat berperan dalam memberikan hak kepada para pelaku, sehingga subjek akan mendapatkan akses ke objek melalui *role* yang telah diberikan oleh administrator (Khayat *et al*, 2005).

Fitur yang disediakan oleh RBAC menjadi daya tarik bagi para perusahaan yang menerapkan teknologi informasi, pertama RBAC yang mengorganisir subjek dan role secara alamiah sesuai dengan struktur yang diterapkan pada perusahaan tersebut (Ferraiolo *et al*, 1997). Hubungan antara hak akses dengan para pelaku yang diterapkan pada perusahaan tersebut, pertama RBAC memberikan tugas keamanan pada kontrol akses sebagai prioritas tertinggi untuk mengontrol akses ke sumber daya. Hal tersebut mengakibatkan RBAC akan me nerapkan keamanan yang sangat ketat dalam melakukan kontrol akses ke sumber daya. Kedua RBAC dalam menerapkan hak akses kepada pengguna membutuhkan waktu yang singkat, dengan cara menghubungkan subjek dengan *role*, sehingga memerlukan penunggasan hak akses untuk *role* pada setiap subjek (Khayat *et al*, 2005).

Kontrol akses dalam mengambil keputusan ditentukan oleh role, sehingga pengguna sebagai bagian dari sebuah organisasi akan mendapatkan hak akses sesuai dengan role yang didapatkannya.

Kebijakan yang dilakukan oleh RBAC akan membuat kontrol akses yang didapatkan oleh pengguna berdasarkan keputusan yang diperoleh dalam sebuah organisasi. Pengguna tidak bisa mengambil hak akses pengguna lain, hal inilah dasar perbedaan antara RBAC dan DAC (Ferraiolo *et al*, 1992).

Dalam menangani kebijakan keamanan, RBAC adalah fondasi utama yang harus didefinisikan dengan baik sebelum *Security Constraint Specifiers* (SCS) atau disebut juga sebagai petugas yang menentukan kendala dalam menjaga keamanan. SCS ini akan menentukan keterbatasan yang terjadi seperti kendala dalam penerapan sistem. RBAC memiliki banyak variasi yang akan dilakukan dengan menggambarkan bagaimana menentukan keterbatasan yang terjadi tanpa kehilangan ciri khasnya. Struktur RBAC meliputi *roles, permission, user, dan session* (Chen *et al*, 1996).

Dalam RBAC, *role* didefinisikan sebagai suatu gagasan yang merupakan dasar dari kebijakan kontrol akses (Sandhu et al, 1999). Pendefinisian *role* masih menjadi perdebatan sehingga diperlukan menjelasan konsep *role* itu tersendiri, sehingga untuk melakukan koloborasi dengan kontrol akses masih disesuaikan dengan kebutuhan. Dalam ilmu prilaku, *role* didefinisikan sebagai pola yang ditentukan sesuai prilaku seseorang dalam situasi tertentu berdasarkan posisi orang tersebut (Hawkins *et al*,1983). Dengan kata lain pendefinian *role* adalah tugas yang didapatkan oleh seseorang sesuai dengan tanggung jawabnya. Namun demikian, sangat sulit untuk menggambarkan pengertian *role* karena bahasa yang diterapkan masih bersifat ambigu (Zhu, 2003).

2.5.4 Entitas dalam RBAC

Role adalah entitas yang sangat mendasar dalam RBAC dan mempunyai istilah sangat luas sehingga tidak ada definisi yang jelas di dalam kepustakaan. Sebenarnya *role* dianggap sebuah abstrak di

dalam sekumpulan hak akses, sehingga jika ingin mendefinisikan *role* serta mengimplementasikan *role* dalam suatu kasus maka *role* merupakan suatu jembatan yang digunakan untuk menghubungkan sekumpulan hak akses pada sekumpulan pengguna. Hak akses yang ditugaskan pada *role* tergantung pada jenis *role* yang akan digunakan, kemudian pengguna akan ditugaskan pada *role* (Klarl *et al*, 2009).

Pengguna merupakan manusia yang berinteraksi dengan sistem komputer (Ferraiolo *et al*, 2003). Pengguna akan mendapatkan akses ke sistem setelah di otentikasi. Setelah mendapatkan akses ke sistem informasi manajemen seorang pengguna melakukan proses untuk mengaktifkan beberapa *role* untuk menjalankan beberapa tugas (Habib, 2011).

Subjek merupakan suatu proses yang dilakukan atas nama pengguna. Di dalam praktek sebenarnya proses atau program komputer yang menjalankan tugas atau aktivitas atas nama pengguna, kemungkinan besar tugas yang dilakukan akan memerlukan keterlibatan beberapa subjek yang aktik berguna untuk menjalankan tugas tertentu. Jalur akses yang akan dilakukan oleh subjek pada setiap proses untuk menemukan apakah proses tersebut dipanggil oleh pengguna yang resmi atau tidak. Hal tersebut dilakukan pada pengguna yang memasukkan identitasnya untuk melakukan login ke *website* bank pada sistem perbankan *online*. Program utama yang beroperasi atas nama pengguna untuk *login* didefinisikan sebagai subjek (Ferraiolo *et al*, 2003).

Permission merupakan wewenang yang diberikan kepada pengguna atau subjek yang melakukan beberapa operasi atau tindakan yang dilakukan kepada objek (Ferraiolo et al, 2003). *Permission* adalah suatu keistimewaan yang diperoleh oleh pengguna, jika pengguna tersebut masuk ke dalam sistem maka operasi yang dilakukan terhadap objek tertentu dapat dilakukan karena pengguna tersebut telah mendapatkan *permission* (Habib, 2011).

Objek di dalam kontek kontrol akses dianggap sebagai sumber daya di mana pengguna dapat melakukan beberapa kegiatan (Ferraiolo *et al*, 2003). Objek dapat berupa *file*, *folder, directory, record*, atau *table* dan lain-lain. Tipe objek tergantung dengan sifat pada sistem yang dibuat. Suatu objek bisa menjadi *entity* pasif yang hanya berisi informasi ataupun bisa menjadi sebuah *entity* aktif berupa printer atau program komputer yang pernah dibahas oleh *Department of Defense in an orange book* (DOD, 1985).

Operation merupakan proses komputer yang dijalankan oleh subjek dan subjek itu sendiri dijalankan oleh pengguna (Ferraiolo *et al*, 2003). Operation bisa berupa *write, read, execute, delete, update* dan sebagainya. Jenis *Operation* tergantung pada jenis objek yang dijalankan. Misalnya pengguna pada *online banking* memasukkan informasi yang dibutuhkan untuk dapat *login* ke sistem. Hal tersebut menyatakan bahwa program utama akan bertindak sebagai subjek mewakili pengguna untuk menjalankan operasi yang ada pada sistem seperti pengecekan saldo, mentransfer uang, melihat informasi pribadi ataupun mengubah sandi dan lain- lain (Habib, 2011).

Session merupakan sebuah contoh komunikasi yang dilakukan oleh dua *entity* (Ferraiolo *et al*, 2003). Di RBAC *session* adalah pemetaan di antara *role* aktif dengan pengguna. *Session* hanya boleh memiliki satu pengguna dan dapat mengaktifkan banyak *role* (ANSI, 2004). Dalam praktek sebenarnya pengguna akan terikat untuk memiliki satu *session* pada waktu yang bersamaan atau waktu yang berbeda. Hal ini pengguna akan ditetapkan untuk dapat memiliki salah satu *role* aktif per *session* atau pengguna dapat mengaktifkan beberapa *role* per *session*. Hal tersebut tergantung implementasi yang akan diterapkan di RBAC (Habib, 2011).

2.6 *Prinsip Least Privilege*

Dalam prinsip yang bertujuan untuk memenuhi integritas, prinsip paling utama adalah mengharuskan pengguna diberikan hak keistimewaan yang diperlukan untuk melakukan suatu pekerjaan. Meskipun hal tersebut penting untuk dilakukan tetapi masih memerlukan identifikasi pekerjaan apa yang digunakan oleh pengguna. Dalam melakukan pekerjaan tersebut akan dilakukan dengan cara menentukan set minimum sehingga dibutuhkan hak untuk melakukannya. Membatasi pengguna dengan cara memberikan keistimewaan hak kepada mereka. Pelaksanaan suatu pekerjaan bukan hak yang telah diberikan kepada mereka sehingga akan ditolak karena itu hal tersebut bukan tugas yang diberikan kepada pengguna. Pekerjaan yang bukan hak mereka harus ditolak itu sesuai dengan kebijakan dalam menjaga. Persyaratan yang dilakukan untuk membatasi hak bagi pengguna akan dilakukan oleh administrator. Dengan melalui RBAC pembatasan dan pemberian hak bagi pengguna dapat dilakukan dengan sangat mudah (Ferraiolo *et al*, 1992).

2.7 Pemisahan Tugas

Mekanisme RBAC dapat digunakan oleh sistem administrator dalam menegakkan kebijakan untuk melak ukan pemisahan tugas bagi para pengguna. Pemisahan tugas yang dianggap sangat penting dalam menghalangi penipuan, penipuan dapat terjadi jika ada kesempatan untuk melakukan dengan cara kolaborasi di antara berbagai pekerjaan yang terkait sesuai dengan kemampuan yang dimiliki. Pemisahan tugas mensyaratkan bahwa bagian tertentu dalam melak ukan transaksi, tidak boleh satupun individu yang diizinkan untuk mengeksekusi transaksi di bagian data tersebut. Umumnya digunakan dalam transaksi yang terpisah sehingga diperlukan cara untuk melakukan tugas untuk mengotorisasi tugas yang telah dilakukan. Sehingga hal itu akan mengakibatkan tidak

ada nya individu tunggal yang mampu melaksanakan kedua transaksi tersebut (Clark *et al*, 1987).

Pemisahan tugas adalah suatu pertimbangan penting dalam sistem nyata, di mana tugas yang dilakukan bervariasi tergantung pada aplikasi yang akan digunakan. Dalam situasi nyata, transaksi tertentu saja yang harus dibatasi. Misalnya melakukan transaksi dengan cara mengotorisasi dalam membatasi untuk melakukan pembayaran, tetapi transaksi tersebut akan mengirimkan saran bagi administrator supaya tidak melakukan hal tersebut. Pemisahan tugas dapat berupa statis atau dinamis. Kepatuhan dengan pemisahan statis persyaratan yang akan ditentukan dengan cara melakukan penugasan secara individu untuk *role* dan alokasi transaksi tersebut akan dilakukan oleh *role*. Kasus ini menjadi lebih sulit karena pemisahan tugas yang dinamis diharuskan sesuai dengan persyaratan dan hanya dapat dilakukan selama sistem tersebut berjalan (Ferraiolo *et al*, 1992).

Tujuan di balik pemisahan tugas yang dinamis adalah untuk memungkinkan fleksibilitas yang lebih dalam sebuah operasi. Pertimbangan kasus tersebut dimulai dari dalam otorisasi dalam menjalankan tugas. Sebuah kebijakan statis bisa mengharuskan bahwa tidak ada individu yang dapat berfungsi sebagai inisiator sehingga pelaksanaan tugas akan berfungsi sebagai pemberi kuasa. Hal ini dapat diimplementasikan dengan memastikan bahwa tidak ada orang yang dapat melakukan *role* inisiator juga dapat melakukan *role* kuasa (Sandhu *et al*, 2004).

2.8 Penyelenggaraan keamanan dalam RBAC

RBAC sangat fleksibel digunakan karena dapat mengambil karakteristik dari sebuah organisasi dalam melakukan sebuah kebijakan yang terstruktur. Salah satu kebijakan RBAC yang terbesar adalah kemampuannya dalam mendukung sebuah administrasi. Setelah transaksi *role* yang ditetapkan dalam sebuah

sistem, transaksi ini cenderung tetap *relative* konstan atau berubah perlahan-lahan dari waktu ke waktu. Tugas administrasi adalah untuk melakukan pemberian dan pencabutan membership dari bagian role yang berinisial nama tertentu dalam sebuah sistem. Ketika anggota baru akan memasuki organisasi tersebut, administrator akan memberikan keanggotaan untuk *role* yang ada. Bila fungsi orang tersebut berubah dalam sebuah organisasi, keanggotaan pengguna untuk *role* yang sudah ada dapat dengan mudah dihapus dan yang baru dapat diberikan. Akhirnya, ketika seseorang meninggalkan organisasi, semua keanggotaan *roles* tersebut akan dihapus. Untuk organisasi yang mengalami omset besar dalam jumlah personil, kebijakan keamanan berbasis *role* adalah satu- satunya pilihan yang logis (Elliott *et al*, 2010).

Keuntungan utama dari RBAC adalah fleksibilitas dan manajemen yang relatif mudah. Fleksibilitas memungkinkan administrator dapat membuat *privilege* yang seminimal mungkin untuk setiap *user*, menghindari konflik dari tugas antar *user*, pemisahan tugas secara dinamis maupun statis. Administrator dengan mudah untuk mendaftarkan atau menolak seorang *user* kedalam sebuah *role* berdasarkan tanggung jawab dan tugasnya (Ferraiolo *et al*, 1992).

2.9 Struktur Role Based Access Control

RBAC telah mendapatkan perhatian yang sangat luas di aplikasi komersial. Model RBAC didefinisikan dalam empat model komponen: *Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations (SSD), Dynamic Separation of Duty Relations (DSD)*. Core RBAC didefinisikan sebagai koleksi minimum dalam elemen RBAC di mana rangkaian elemen tersebut berelasi sepenuhnya untuk memperoleh sistem dalam RBAC (Ferraiolo *et al*, 2001).

Dalam struktur RBAC hak akses akan ditugaskan ke *roles* bukan langsung kepada pengguna, dan *roles* akan digunakan oleh pengguna, kemudian pengguna membuat sebuah *session*, dan *session* akan mendapatkan izin (*permission*) dari *role* yang didapatkan oleh pengguna kemudian akan mengaktifkan perannya (*roles*). *Roles* yang didapatkan oleh pengguna disesuaikan dengan pengelolaannya. *Roles* yang didapatkan pengguna disusun sesuai dengan hirarkinya di mana *role* yang diperoleh oleh senior lebih berkuasa dan memiliki hak akses yang lebih dari pada *role* yang didapatkan oleh junior (Mohammed *et al*, 1994).

Dalam suatu organisasi, *role* diciptakan untuk merepresentasikan berbagai fungsi pekerjaan. Hak akses melakukan operasi/proses tertentu pada sistem yang diberikan kepada *role* tertentu. Anggota staf (pengguna sistem lainnya) mendapatkan *role* tersebut, dan hanya melalui *role* mereka mendapatkan hak akses untuk melakukan fungsi-fungsi sistem. Pengguna tidak diberikan hak akses secara langsung untuk melakukan operasi tertentu, tetapi hanya mendapatkan melalui *role* mereka, pengelolaan hak akses bagi pengguna individu di permudah dengan menentukan *role* yang sesuai untuk pemakai, hal ini menyederhanakan operasi umum, seperti menambah pengguna, atau mengubah unit organisasisi pengguna.

Ada tiga aturan utama dalam RBAC, yaitu:

1. Pemberian *role*. Seorang pengguna atau disebut juga subyek dapat menjalankan transaksi atau proses pada sistem jika pengguna tersebut diberikan *role* yang diizinkan untuk menjalankan transaksi. *Role* bisa dianggap sebagai kelompok pengguna, seseorang diizinkan melaksanakan operasi yang menjadi hak kelompoknya. Pemberian *role* dilakukan dengan menyatakan bahwa seorang pengguna adalah anggota dari kelompok tersebut.
2. Otorisasi *role*. Pemberian otorisasi kepada pengguna untuk menjalankan transaksi dilakukan dengan mengaktifkan *role*

yang hendak dipakai oleh pengguna tersebut. Dalam kondisi pengguna tidak melakukan apapun ke sistem, semua *role* sifatnya pasif. Pengaktifan *role* bersifat sementara selama sesi berlangsung. Apabila sesi berakhir, *role* kembali di nonaktifkan. Dengan aturan nomor 1 di atas, dapat dipastikan bahwa *role* yang diaktifkan hanya *role* yang memang dimiliki pengguna tersebut.

3. Otorisasi Transaksi. Pengguna dapat menjalankan transaksi hanya jika transaksi telah diotorisasi untuk *role* pengguna yang diaktifkan. Dengan aturan nomor 1 dan nomor 2, dipastikan bahwa pengguna dapat melakukan transaksi hanya untuk yang mereka yang telah diotorisasi.

Untuk mendefinisikan model RBAC, beberapa konvensi berikut akan berguna:

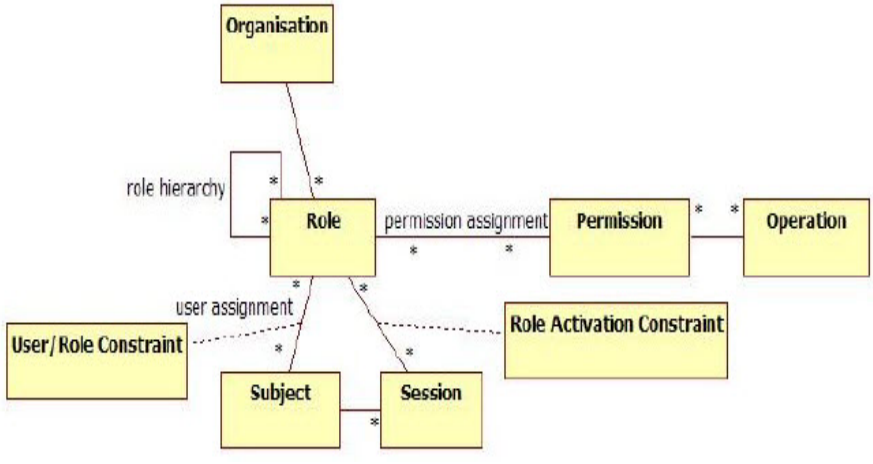
a. S = Subyek = Seorang pengguna atau aplikasi lain (*automated agent*)
b. R = *Role* = Fungsi pekerjaan atau jabatan yang menentukan tingkatan otoritas
c. P = *Permissions* = Izin dari mode akses terhadap sumber daya
d. SE = Sesi = Pemetaan yang melibatkan S, R dan/atau P
e. SA = Pemberian role ke subyek (*subject assignment*)
f. PA = Pemberian izin akses ke proses/transaksi
g. RH = Hirarki peran yang urut parsial (*partially ordered role hierarchy*). RH juga dapat ditulis: ≥ (notasi ini: x ≥ y berarti x mewarisi hak akses y.)
h. Subyek dapat memiliki peran lebih dar i satu.
i. *Roles* dapat mempunyai anggota beberapa subyek.
j. *Roles* yang dapat memiliki lebih dari satu hak akses.
k. Sebuah izin akses dapat diberikan ke lebih dari satu peran.

Pembatasan lain dapat dilakukan untuk mewujudkan aturan yang mengikat, misalnya untuk mencegah dua peran yang berlawanan dimiliki oleh seseorang. Contohnya peran untuk

membuat *account* pengguna harus dipisah dengan peran untuk menentukan otorisasi aksesnya. Dalam notasi himpunan, berlaku ketentuan sebagai berikut:

a) PA $\subseteq P \times R$, adalah relasi many-to-many antara izin akses dan peran
b) SA $\subseteq S \times R$, adalah relasi many-to-many antara subyek dan peran
c) RH $\subseteq R \times R$

Kesemua konvensi di atas beserta dengan hubungannya dapat dilihat pada Gambar 2.1.



Gambar 2.1 Model *Role-Based Access Control* (RBAC).

2.10 Spesifikasi RBAC

Role Based Access Control (RBAC) merupakan kerangka kerja yang berfungsi untuk mengontrol akses pengguna ke sumber daya berdasarkan *role*. Secara signifikan hal tersebut dapat mengurangi biaya administrasi dalam melakukan kebijakan kontrol akses yang banyak digunakan pada organisasi besar (Liu *et al*, 2004). Standar ANSI untuk RBAC dikembangkan sebagai standarisasi atas kebutuhan kalangan pembeli baik dari

pemerintahan maupun pengembang teknologi informasi yang mendefinisikan secara konsisten beragam fitur yang ada dalam RBAC (Sandhu *et al*, 2000).

Standarisasi RBAC memiliki empat komponen yaitu pada *core* RBAC yang mendefinisikan fungsi intinya pada hak akses (*permissions*), pengguna (*user*), sesi (*sessions*), dan *role*, sementara itu tiga komponen lainnya sebagai komponen pendukung yang menambahkan dukungan untuk *hierarchy of roles*, kendala pada *role* pengguna (*constraints on the roles of a user*), dan kendala pada pengaktifan *role* pada sesi (*constraints on the active roles in a session*), di mana fungsi ini diterapkan secara formal dalam *set based specification language* (Spivey, 1992).

Penemuan kesalahan dan komplikasi standarisasi RBAC sebagian besar disebabkan oleh interaksi dalam melakukan efisiensi di konsep RBAC. Misalnya dalam perintah *AddInheritance* yang berfungsi untuk membatasi *hierarchical* RBAC yang tidak pernah dapat menambahkan pengwarisan dalam pra-kondisi, karena salah melakukan pendefinisian yang diakibatkan dalam melakukan pengambilan pengwarisan secara langsung dari *transitive inheritance* yang selalu salah. Di contoh yang lainnya ada lima dari tujuh fungsi pemetaan yang tidak diperlukan untuk mendefinisikan referensi pemodelan *core RBAC*, begitu juga di dalam melakukan pengupdatean pengguna dan penugasan perizinan dalam perintah *assigned users dan assigned permissions* di dalam administrasi di semua perintah *core RBAC* (Spivey, 1992) .

Dalam standarisasi ANSI RBAC memiliki empat komponen yaitu SSD (*static separation of duties*) dan DSD (*dynamic separation of duties*) yang disebut dengan *constrained* RBAC (Liu *et al*, 2004) adalah sebagai berikut:

1. *Core RBAC* adalah pendukungan core yang berfungsi untuk melakukan perizinan (*permissions*), pengguna (*user*), sesi (*session*) dan *Role*

2. *Hierarchical RBAC* adalah dukungan tambahan untuk *hierarchy of roles*
3. SSD adalah untuk melakukan penambahan dukungan yang terjadi pada *constraints on the roles of a user*
4. DSD adalah penambahan dukungan pada *constraints on the active roles in a session*

2.10.1 *Core RBAC*

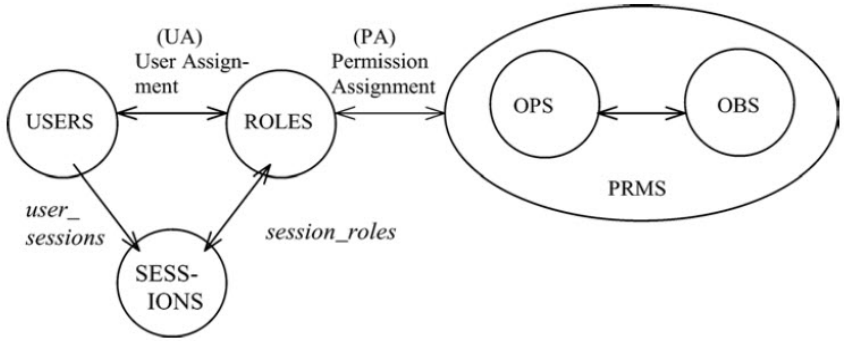
Core RBAC mewujudkan aspek yang penting dari RBAC. Konsep dasar RBAC adalah pengguna yang ditugaskan untuk menjadi *role*, hak akses yang ditugaskan pada *role*, dan pengguna memperoleh izin dengan menjadi anggota *role*. *Core RBAC* mencakup persyaratan bahwa tugas *user-role* dan *permission-role* dapat menjadi relasi *many to many*. Dengan demikian pengguna yang sama dapat ditugaskan ke banyak *role*. *Role* yang tunggal dapat memiliki banyak pengguna. Untuk perizinan, izin tunggal dapat ditugaskan kepada banyak *role* dan *role* yang tunggal dapat memperoleh banyak perizinan. *Core RBAC* mencakup persyaratan *user-role* ditinjau dari *role* yang akan ditugaskan kepada pengguna, serta dapat ditentukan oleh pengguna yang akan ditugaskan sebagai *role*. Suatu *role* yang sama untuk *permission-role* dapat dilakukan penambahan fungsi. *Core RBAC* juga mencakup konsep *session* pengguna yang memungkinkan untuk melakukan aktivasi yang selektif dan deaktivasi *role*. *Core RBAC* secara bersamaan akan menggunakan izin pengguna untuk menjalankan berbagai *role*. Hal ini menghindarkan produk yang akan membatasi pengguna untuk mengaktifkan salah satu *role* pada waktu bersamaan (Ferraiolo *et al*, 2001).

Core RBAC adalah konsep dari relasi *role*, di mana *role* merupakan semantik yang membentuk dan merumuskan kebijakan. Gambar di bawah ini yang menggambarkan *User Assignment* (UA) dan *Permission Assigment* (PA). Tanda panah yang menunjukkan

hubungan *many to many* (misalnya pengguna dapat ditugaskan untuk satu atau lebih dari *role* yang dijalankan, dan *role* dapat ditugaskan ke satu pengguna atau lebih). Pengaturan ini akan menyediakan fleksibilitas yang besar dengan rincian dari tugas perizinan yang diberikan untuk *role* dan dari pengguna untuk *role*. Tanpa melakukan hal tersebut akan menjadi permasalahan yang rumit karena pengguna sering melakukan akses ke sumber daya, yang diakibatkan oleh pengontrolan yang terbatas atas akses ke sumber daya yang diberikan ke pengguna. Pengguna mungkin perlu membuat *direktory* atau memodifikasi *file*, misalnya tanpa membuat *file* yang baru mungkin mereka perlu menambahkan *record* ke file tersebut (ANSI, 2004).

Pada Gambar 2.4 tanda panah *users* akan menuju ke *sessions* yang menunjukkan bahwa satu pengguna boleh memiliki satu atau banyak session tetapi satu *session* hanya boleh memiliki satu pengguna dan hal ini disebut dengan *user_sessions*. Dengan pola yang sama panah ganda yang menuju di antara *sessions* dengan *roles* menunjukkan bahwa satu *session* dapat memiliki banyak *role* yang aktif dan satu *role* dapat memiliki lebih dari satu *session*, atau disebut dengan *session_roles*. Ada beberapa definisi yang dapat dijelaskan tentang *session* bahwa pengguna dapat memiliki satu atau lebih *session* pada waktu yang bersamaan (C hadwick *et al*, 2007).

Istilah PRMS, OPS, OBS terdiri dari hak akses, operasi dan objek, hak akses yang dilakukan bisa *write, insert, delete, update* dan lain- lain (ANSI, 2004).



Gambar 2.2 Core RBAC (ANSI)

Secara dasarnya gambar yang diatas dapat didefinisikan sebagai berikut:

"*User*": adalah sebuah objek yang akan mengeluarkan permintaan tersebut ke dalam sebuah sistem yang dapat juga disebut sebagai pengguna, peralatan, jaringan, atau perangkat sistem agen cerdas.

"*Role*": adalah sebuah entitas yang akan menghubungkan pengguna dengan "*permission*" yang akan disesuaikan dengan fungsi kerja dalam lingkungan sebuah organisasi.

"*Object*": adalah sebuah entitas yang telah dimodifikasi atau dirubah dan akan menjadi peran utama untuk menerima dan menyimpan sebuah informasi.

"*Operation*": adalah sebuah jenis dari operasi obyek yang pelaksanaan programnya akan tergantung pada sistem yang akan diimplementasikan, seperti operasi dalam sistem *file* termasuk *read, write dan execute*, dan akan melakukan operasional dalam sistem database termasuk *insert, delete*, *append* dan *update*.

"*Permission*": berarti kemampuan yang dapat dioperasikan baik oleh satu atau lebih objek yang akan dilindungi oleh RBAC, hal ini akan dibentuk oleh seperangkat operasi dan objek. Pengguna akan ditugaskan sebagai *roles* yang akan disesuaikan dengan perizinan yang didapatkan untuk mengoperasikan objek. Para

pengguna akan diasosiasikan dengan *roles,* dan *roles* akan diasosiasikan untuk memperoleh perizinan.

2.10.1.1 Administrative *commands dalam Core RBAC*

Membuat dan melakukan pemeliharaan element *sets* pada *core RBAC* adalah *users*, *roles*, OPS (*operations*), dan OBS (*objects*). OPS dan OBS telah ditetapkan pada sistem ini, dan administrator hanya mebuat dan menghapus *user* dan *role* dengan cara membangun hubungan antara *role* dan *operations* yang ada pada object. Administrator memerlukan fungsi bagi pengguna yaitu *AddUser* dan *DeleteUser* dan fungsi untuk *role* adalah *AddRole* dan *DeleteRole* (Ferraiolo *et al*, 2001).

Membuat dan melakukan pemeliharaan hubungan antara dua relasi dalam *core RBAC* adalah *user-to-role* atau juga hubungan tugas UA (*assignment relation*), dengan perintah *AssignUser* dan *DeleteUser*. *Permission-to-role* (PA) adalah tugas *user* dalam *role* di mana fungsi yang diperlukan adalah *GrantPermission* dan *RevokePermission* (Ferraiolo *et al*, 2001).

2.10.1.2 Fungsi Pendukung dalam Sistem *Core RBAC*

Fungsi pendukung dalam sistem yang diperlukan adalah manajemen *session* dalam membuat keputusan pada akses kontrol. Fungsi ini akan menciptakan *session* dengan cara menetapkan satu *default set role* yang aktif pada pengguna di *session* awal. Komposisi *default set* ini dapat diubah oleh pengguna selama *session* ini berlangsung dengan menambahkan *role* atau menghapus *role*. Fungsi ini berkaitan dengan menambah dan *dropping* pada *role* yang aktif, penambahan pada fungsi (Ferraiolo *et al*, 2001) ini adalah sebagai berikut:

a. *CreateSession* berfungsi untuk menciptakan session pengguna serta memberikan pengguna dengan *default set role* aktif
b. *DropActiveRole* berfungsi untuk menambahkan *role* sebagai *role* aktif untuk *session* pada saat ini
c. *CheckAccess* yang berfungsi untuk menentukan apakah *session* subjek memiliki *permission* untuk melakukan permintaan operasi pada objek

2.10.1.3 *Review Functions*

Saat *user-to-role* memberikan izin wewenang kepada *role* yang lain, dimungkinkan untuk melihat relasi dari kedua pengguna tersebut dari sudut pandang *role*. Misalnya, dari relasi UA, administrator harus memiliki fasilitas untuk melihat aktifitas pengguna yang telah ditugaskan ke *role* tertentu, juga dapat melihat semua *role* yang telah diberikan kepada pengguna. Selain itu, administrator dimungkinkan untuk melihat hasil dukungan dari fungsi sistem untuk menentukan session atribute tertentu seperti *role* yang aktif dari *session* tertentu atau total izin domain untuk *session* tertentu. Oleh sebab itu tidak semua implementasi dalam RBAC menyediakan fasilitas untuk me lihat *role*, pengguna, *session permissions*, atau *role* yang aktif pada suatu session. Fungsi ini telah ditetapkan sebagai fungsi optional/*advance* dalam persyaratan ini. Ulasan dari fungsi *Mandatorym* (M) dan *Optional* (O) (Ferraiolo *et al*, 2001) adalah sebagai berikut:

a. *AssignedUsers* (M) berfungsi untuk mengembalikan *set* pengguna yang ditugaskan (*Assigned*) ke *role* tertentu
b. *AssignedRoles* (M) berfungsi untuk melakukan pengembalian *set role* yang ditugaskan ke pengguna tertentu
c. *RolePermissions* (O) berfungsi untuk mengembalikan *set of permissions* yang relah diberikan kepada *role* tertentu

d. *UserPermissions (O*) berfungsi untuk mengembalikan *set of permissions*

e. Pengguna atau *assigned roles*

f. *SessionRoles (O)* berfungsi untuk mengembalikan *set of active roles* yang berkaitan dengan *session*

g. *SessionPermissions (O)* berfungsi untuk mengembalikan *set of permissions* yang tersedia di *session (*gabungan dari semua *permissions assigned ke sesssion 's active roles)*

h. *RoleOperationsOnObject (O)* berfungsi untuk mengembalikan *set of operations role* yang akan ditampilkan pada objek

i. *UserOperationsOnObject (O)* berfungsi untuk mengembalikan *set of operations* pengguna yang mungkin tampil pada suatu objek (diperoleh baik secara langsung atau melalui *assigned roles)*

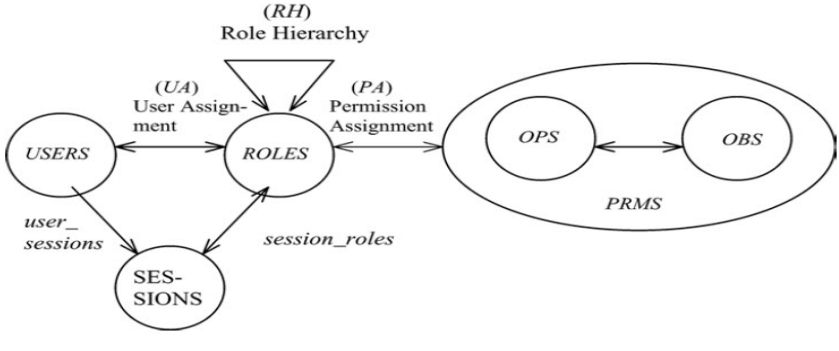
2.10.1.4 Hierarchical RBAC

Hierarchical RBAC akan menambahkan *role hierarchies* sesuai kebutuhan yang diperlukan. Sebuah *hierarchies* secara matematis disesuaikan dengan urutan parsial yang akan mendefinisikan hubungan secara senioritas di antara *role*, di mana *role* senoir memperoleh izin dari *role* yunior, dan *role* junior akan memperoleh keanggotaannya sebagai pengguna dari *role* senior. Pernyataan di bawah akan mengakui dua jenis *role hierarchies* (Ferraiolo, 2001) yaitu:

1. *Hierarchical RBAC* dalam kasus ini akan mendapatkan dukungan secara partial dan dapat bertidak secara acak sebagai *role hierarchies*. Dengan menyertakan konsep pengwarisan perizinan berganda serta menjadi keanggotaan *role*.
2. Dengan terbatasnya *hierarchical RBAC*, sebagai sistem mungkin akan menerapkan pembatasan pada *role hierarchies*.

Paling umum hirarki hanya terbatas pada struktur yang sederhana yaitu seperti *trees or inverted trees*.

Model komponen yang memperkenalkan *role hierarchies* (RH) seperti yang ditunjukkan pada gambar di bawah ini. *role hierarchies* umumnya disertakan sebagai aspek kunci dari model RBAC dan disertakan sebagai bagian dari produk yang ditawarkan oleh RBAC. *Hierarchies* secara alami akan menyusun *role* dan mempunyai wewenang serta tanggung jawab dalam sebuah organisasi. *Role hierarchies* akan mendefinisikan hubungan pengwarisan di antara *role*. Warisan yang telah digambarkan dalam bentuk *permissions*, yaitu r1 akan mengwariskan *role* r2. Jika seluruh hak akses dari r2 juga *privileges* dari r1. Dalam beberapa implementasi yang telah diterapkan dalam RBAC, *role permissions* tidak dikelola secara sentral, sedangkan *role hierarchies* berbeda. Dalam sistem ini, *role hierarchies* yang dikelola akan mempunyai hubungan dengan pengguna. *Role* r1 akan mengandung *role* dari r2 jika semua pengguna yang sah untuk r1 yang dizinkan oleh r2. Bagaimanapun juga bahwa pengguna itu menunjukkan ia memiliki r1 setidaknya seluruh hak akses yang dimiliki oleh r2, sedangkan pengwarisan izin untuk r1 dan r2 tidak menunjukkan tugas yang akan diberikan kepada pengguna (ANSI, 2004).



Gambar 2.3 *Hierarchical* RBAC (ANSI)

2.10.1.5 *Static Separation of Duty Relations* (SSD)

Konflik kepentingan di dalam sebuah sistem RBAC disebabkan oleh pengguna yang memperoleh otorisasi perizinan yang berhubungan melalui konflik *role*. Salah satu sarana berupa konflik kepentingan yang melalui pemisahan tugas yaitu batasan dilakukan pada tugas pengguna ke *role*. Keterbatasan RBAC terjadi dalam berbagai bentuk, contoh yang paling umum bahwa pemisahan tugas ditetapkan bersama tugas pengguna yang berkaitan dengan *role*. Keterbatasannya RBAC dapat ditunjukkan dengan cara mengimplementasikan beberapa bentuk pemisahan tugas yang lainnya (Gligor *et al*, 1998).

Dalam RBAC, SSD digunakan untuk menghindari konflik kepentingan di antara *role* yang saling bertentangan. Konflik kepentingan terjadi di antara *role* yang saling bertentangan sehingga kepentingan di antara *role* mungkin muncul saat pengguna ingin menjalankan hak akses dari kedua *role* yang saling bertentangan. *Role* yang saling bertentangan itu terjadi karena pertentangan kepentingan di antara *role* satu dengan lainnya. SSD merupakan keterbatasan RBAC yang diterapkan pada saat *user-role* melakukan tugasnya, hal tersebut ditunjukkan pada Gambar 2.6 (ANSI, 2004).

Pada model RBAC akan menetapkan relasi SSD berkaitan dengan keterbatasan RBAC di antara *user-role* yang mendapat tugas melebihi dari pasangan *role* (misalnya pengguna tidak dapat ditugaskan secara bersama di kedua *role* dalam SSD). Meskipun di dunia nyata kebijakan SSD itu ada, hal ini didefinisinya dalam membatasi dua aspek yang sangat penting. Aspek pertama adalah ukuran dari rangkaian *role* dalam SSD. Aspek kedua adalah kombinasi di antara *role* dari sekumpulan penunggasan dengan penguna yang akan dibatasi. Dalam model SSD dapat didefinisikan dengan dua argumen di mana serangkaian *role* termasuk dua atau beberapa *role* lebih besar dengan mengindikasikan kombinasi di

antara *role*, sehingga akan terjadi suatu pelanggaran dari kebijakan SSD (ANSI, 2004).

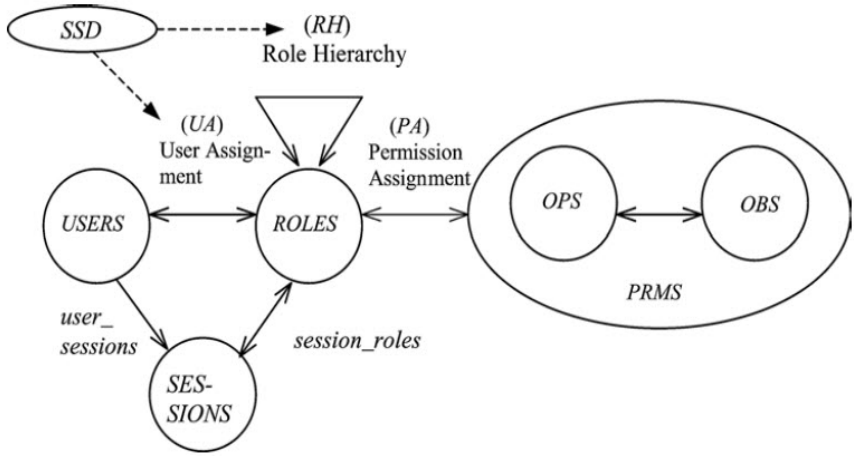
Komponen Hierarchical RBAC akan menambahkan relasi untuk mendukung hirarki relasi untuk *role*. Hirarki adalah cara alamiah untuk menyusun tugas *role* dalam mencerminkan jalur sebuah organisasi yang berwenang dan pertanggungjawabannya. *Static Separation of Duty Relations* (SSD) akan menambahkan relasi antara *role* secara eksklusif sehubungan dengan tugas yang didapatkan oleh pengguna. Sehubungan dengan potensi *inkonsistensi Static Separation of Duty Relations* (SSD) akan memperoleh warisan relasi dari hirarki *role*. Model Static *Separation of Duty Relations* (SSD) mendefinisikan relasi baik itu ada pada hirarki *role* atau bukan hirarki dalam *role* (Chen *et al*, 2007).

SSD didefinisikan dalam (ANSI, 2004) sebagai "relasi SSD yang diterapkan sebagai pembatasan penunggasan di antara *user-role*. Keanggotaan pengguna di salah satu *role* dapat menghentikan pengguna menjadi anggota dari salah satu *role* atau beberapa *role* yang lain tergantung pada peraturan yang akan diberlakukan pada *Separation of Duty* (SOD) statis". Ketika terjadi konflik kepentingan di antara dua buah *role*, *role* tersebut dinyatakan sebagai *mutually exclusive roles* sehingga dalam pelaksanaan SSD pengguna yang ditugaskan ke salah satu dari dua *role mutual exclusive* (ANSI, 2004). Pengguna tidak dapat ditugaskan kepada dua *mutually exclusive roles*. Keterbatasan pada RBAC ini diterapkan pada saat *user-role* bertugas bukan pada saat pengaktifan *role*, hal ini dinamakan dengan *Static Separation of Duty* (SSD) (Perelson *et al*, 2000).

Para peneliti mengajukan model SSD dengan menitikberatkan pada *entity* yang saling bertentangan. Kebijakan pelaksanaan SSD secara langsung diterapkan akan mengalami kesulitan, sehingga pelaksanaannya sulit diverifikasi apakah RBAC telah memenuhi keterbatasan *Static Mutually Exclusive Role* (SMER) atau tidak. Tetapi untuk memastikan verifikasi terhadap keterbatasan SMER,

kebijakan SSD diimplementasikan pada tugas yang dilakukan tidak efesien dan pada tipe satunya lagi digunakan di RBAC standard ANSI 2004 adalah *Static Mutually Exclusive Role* (SMER) dan *Dynamic Mutually Exclusive Role* (DME R) (Li *et al*, 2007).

Seperti yang diilustrasikan pada gambar di bawah ini, relasi SSD mungkin ada di dalam Hierarchical RBAC. Ketika akan diterapkan relasi SSD dihadapan *role hierarchy*, hal ini menjadi perhatian khusus untuk dapat diaplikasikan dan akan memastikan bahwa warisan antara pengguna tidak akan mengurangi kebijakan SSD. Oleh sebab itu *role hierarchy* telah dipilih untuk ditetapkan dan akan dimasukkan sebagai keterbatasan pengwarisan dalam SSD (Gavrila *et al*, 1998).



Gambar 2.4 *Static Separation of Duty Relations* (ANSI)

Pada contoh kasus untuk menerapkan SSD pada dua *mutually exclusive roles* "R1" dan "R2", kedua *role* tidak pernah memiliki pengguna yang ditugaskan secara bersama-sama, hal ini berarti bahwa satu pengguna tidak dapat ditugaskan pada dua *mutually exclusive roles*. Definisi SSD yang diterapkan pada (ANSI, 2004) sebagai berikut:

SSD ⊆ ℕ) adalah sekumpulan pasangan (rs, n) di mana setiap rs merupakan serangkaian *role* dan n adalah angka ≥ 2, dengan property ini bahwa pengguna tidak dapat ditugaskan pada n

atau *role* lebih banyak dari set rs (rs, n) ϵ SSD. Berbagai kebijakan SSD dapat diterapkan secara berbeda, misalnya pengguna hanya dapat menjalankan satu *role* dari sekumpulan *role* atau pengguna tidak dapat menjalankan semua *role* dari *role* yang telah ditetapkan n (min) = card | rs | -1.

Spesifikasi resmi SSD yang diterapkan (ANSI, 2004) adalah sebagai berikut:

$\forall$ (rs, n) ϵ SSD, $\forall t \subseteq$ rs : | t | $\geq$ n $\Rightarrow$ *assignet_users* (r) =

SSD yang didefinisi ulang dengan *role hierarchy* (RH) di (ANSI, 2004) adalah:

$\forall$ (rs,n) ϵ SSD, $\forall$t $\subseteq$ rs : | t | $\geq$ n $\Rightarrow$ *authorized users* (r) =

Dalam sebuah kasus yang terjadi di departement store role pengawas berhubungan dengan bagian keuangan dan pegawai mempunyai hubungan dengan bagian gudang akan memverifikasi barang yang diterima, hal ini dinyatakan sebagai *mutually exclusive roles*. Jadi, alasan inilah menyatakan bahwa role ini merupakan petugas dalam orga nisasi dan mempunyai wewenang untuk memverifikasi barang yang diterima dan tidak mempunyai wewenang untuk menolak pembayaran terhadap barang yang telah diterima. Oleh sebab itu, kedua *role* akan berada dalam ME sehingga mengakibatkan dalam SSD satu orang pengguna tidak dapat ditugaskan dalam dua buah *role*. Di dalam SSD seorang pengguna tidak akan pernah memiliki dua buah *mutually exclusive roles*. Ketentuan ini kadangkala tidak fleksibel sebab dalam situasi khusus bisa jadi seorang pengguna perlu untuk mengaktifkan *mutually exclusive roles* yang lainnya dengan bantuan SSD (Habib, 2011).

Contoh lainnya pada sebuah bank yang mempergunakan aplikasi berbasis SSD mempunyai dua *role* yaitu teller dan nasabah dinyatakan sebagai *mutually exclusive roles*. Tujuan diperlakukan

hal ini supaya tidak ada pengguna mempunyai dua *role* baik itu sebagai nasabah maupun teller bank sehingga dapat menghindari kecurangan yang akan terjadi di bank tersebut. Namun, dengan cara ini teller bank tidak dapat menjadi nasabah dan tidak bisa membuka rekening di bank di mana dia bekerja. Kenyataannya tidak ada bank membatasi pegawainya untuk membuka rekening bank di mana pengawai bank tersebut bekerja, hal ini merupakan keterbatasan yang tidak dapat dihindari dalam SSD. Sehingga untuk menghindari dari permasalahan ini DSD adalah sebuah pilihan untuk diterapkan. Tujuan dari penerapan *Dynamic Separation of Duty Relations* (DSD) membuat operasional yang dilakukan oleh pengguna menjadi fleksibel, namun sebaliknya SSD akan menjadi sebuah hambatan yang berguna karena bagaimanapun menggunakan SSD pengguna tidak dapat mengaktifkan dua *mutually exclusive roles*, jika kasus tersebut SSD akan menjadi pilihan dalam memenuhi kebutuhan organisasi. Dengan diterapkannya DSD hal tersebut terjadi oleh kebutuhan organisasi untuk mengaktifkan dua *mutually exclusive roles* karena alasan tertentu (Nash *et al*, 1990).

Ada dua aspek penting yang digunakan oleh SSD yaitu pada aspek pertama adalah besarnya *role* yang digunakan dalam SSD dan kedua adalah kombinasi diantara sekumpulan pengguna ditetapkan pembatasannya dalam SSD. SSD dapat diaplikasikan dengan *Hierarchical RBAC*. Sehingga perhatian dalam merancang pengwarisan bagi pengguna mengakibatkan SSD akan mempengaruhinya sesuai dengan tujuan yang telah ditetapkan. Dalam hal ini SSD sebagai hambatan bagi pengguna yang sah untuk mengaktifkan *role* yang mempunyai kaitan dengan SSD (ANSI, 2004).

2.10.1.6 *Dynamic Separation of Duty Relations* (DSD)

Dynamic Separation of Duty Relations memungkinkan pengguna diberi wewenang dua atau lebih untuk mendapatkan *role*. Wewenang diberikan jika tidak terciptanya konflik kepentingan sehingga dapat bertindak secara mandiri, namun hal tersebut menjadi permasalahan sewaktu *role* diaktifkan secara bersama-sama (Ferraiolo *et al*, 2001).

DSD merupakan tipe kedua dari SOD yang diimplementasikan di RBAC (ANSI, 2004). Perbedaan di antara SSD dan DSD adalah waktu di mana hambatan tersebut diterapkan, SSD digunakan sewaktu *user-role* ditugaskan, sementara DSD digunakan pada saat *role* diaktifkan. DSD diimplementasikan pada saat *role* diaktifkan hal ini merupakan proses dinamis yang dinamakan dengan *Dynamic Separation of Duty* (ANSI, 2004). Ketika dua buah *role* menimbulkan konflik kepentingan di antara *role* itu dikatakan sebagai *mutually exclusive* yang terjadi pada satu sama lainnya, kemudian DSD akan menugaskan pengguna kepada kedua *mutually exclusive roles* yang bertentangan dengan SSD. Sehingga pengguna mendapatkan kewenangan untuk menjalankan atau mengaktifkan kedua *role* namun pada saat bersamaan atau pada *session* pengguna yang sama (Ferraiolo *et al*, 2001). Hal ini lebih fleksibel namun batasan yang didapatkan lebih lemah dibandingkan dengan SSD (ANSI, 2004).

Untuk mengimplementasikan banyak *role* di DSD disebut sebagai *mutually exclusive roles* sehingga seluruh atau sebahagian dari *mutually exclusive roles* dapat ditugaskan ke pengguna. Kebijakan di DSD ditentukan seberapa banyak *role* yang dapat diaktifkan oleh pengguna dalam waktu yang bersamaan. Definisi DSD dapat dibuat (ANSI, 2004) sebagai berikut:

DSD $\subseteq$ (2 ROLES x N) merupakan sekumpulan pasangan (rs, n) di DS D, di mana setiap rs adalah serangkaian *role* dan n

merupakan angka ≥ 2, dengan *property* bahwa pengguna tidak dapat mengaktifkan n atau banyaknya *role* yang telah ditetapkan dari *role* rs pada setiap dsd ϵ DSD. Definisi secara formal di (ANSI, 2004) adalah sebagai berikut:

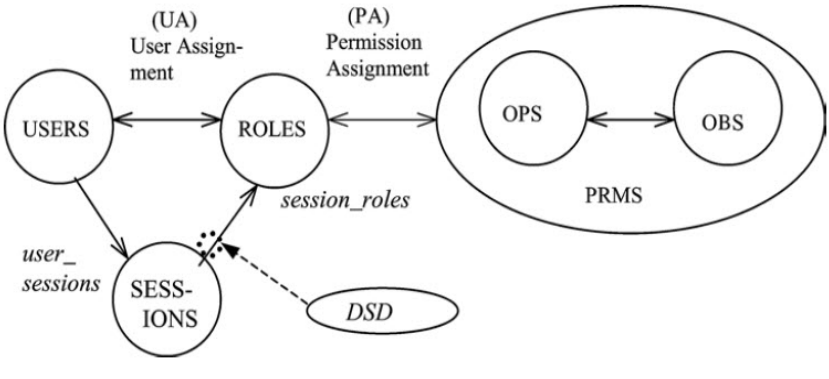
$\forall rs \epsilon 2^{ROLES}, n \epsilon N, (rs,n) \epsilon DSD \Rightarrow n \geq 2. | rs | \geq n$, and

$\forall s \epsilon SESSIONS, \forall rs \epsilon 2^{ROLES}, \forall role\_subset \epsilon 2^{ROLES},$

$\forall n \epsilon N, (rs,n) \epsilon DSD, role\_subset \subseteq rs,$

$role\_subset \subseteq session\_role(s) \Rightarrow | role\_subset | < n.$

Pada saat *role* akan dideklarasikan sebagai *mutually exclusive roles* berguna untuk mengimplementasikan kebijakan DSD dengan menggunakan dua cara. Cara pertama jika salah satu ada pasangan dari *mutually exclusive roles* dan pengguna akan dibatasi hanya mengaktifkan salah satu *role* pada setiap pasangan tersebut pada waktu yang sama atau pada saat *session* pengguna, kedua akan ada seperangkat yang akan didefinisikan lebih dari dua *role* dan jumlah pengguna akan dibatasi pada saat mengaktifkan *role* dalam waktu yang sama atau bersamaan dengan session pengguna. Relasi DSD memberikan pada organisasi fleksibilitas operasional yang besar dibandingkan dengan relasi SSD (Li *et al*, 2005).



Gambar 2.5 ***Dynamic Separation of Duty Relations (ANSI, 2004)***

2.11 Hambatan di RBAC

Hambatan di RBAC memiliki batasan tertentu dan implementasi yang dilakukan menggunakan SOD (*Separation of Duty*). Penelitian yang telah dilakukan dengan menerapkan berbagai jenis batasan serta berbagai tipe lainnya dalam hambatan di RBAC sesuai dengan ketentuan SOD. Hal ini mencakup tentang waktu, tempat dan hambatan berdasarkan cara mengakses sistem yang dilakukan oleh pengguna pada waktu dan lokasi tertentu. Kendala ini dilakuan sesuai dengan ketentuan nama *attributes* yang diberikan (Kuhn *et al*, 2010). Hambatan di RBAC merupakan model komponen ketiga dalam RBAC. Komponen tersebut berhubungan dengan berbagai jenis hambatan dalam RBAC. Ini merupakan komponen yang penting dalam keamanan sistem. Hal tersebut memungkinkan RBAC untuk mengimplementasikan keamanan informasi dalam melindungi sistem dari ancaman internal maupun ancaman eksternal, keadaan tersebut dapat terverifikasi dengan menggunan model kontrol akses di RBAC (Jaeger *et al*, 2001). Untuk mengekspresikan hambatan di RBAC telah didefinisikan menggunakan metode grafis dengan cara melakukan analisis menggunakan UML (Ray *et al*, 2004).

Penelitian yang pernah dilakukan mengizinkan akses ke sistem tergantung pada lokasi subjek, hal ini dapat memperluas model yang menggabungkan hierarchies, inheritance dan *separation of duty constraints*. RBAC dapat menyediakan keamanan yang ketat disebabkan oleh kendala yang ada. Penambahan fitur baru di RBAC berdampak luas pada konflik kepentingan yang berguna untuk menjaga kekuatan keamanan pada suatu sistem (Jaeger, 1999).

2.11.1 *Separation of Duty* (SOD)

Hambatan di RBAC di sebut *Separation of Duty* (SOD), SOD digunakan untuk membatasi pengguna dalam melakukan tugas-tugas tertentu sesuai dengan ketentuan serta batasan yang diberikan oleh

organisasi. SOD ini diimplementasikan supaya pengguna melakukan tugas sesuai wewenang yang diberikan oleh organisasi tersebut. Konsep SOD memiliki sejarah yang panjang dan digunakan untuk memperluas tanggung jawab dan kewajiban lebih dari satu pengguna dalam meminimalkan penipuan yang dilakukan oleh pihak internal maupun pihak eksternal (Simon *et al*, 1997).

SOD merupakan parameter keamanan yang digunakan untuk memperkuat pengamanan sistem informasi, yang berguna untuk meminimalkan kemungkinan terjadinya penipuan yang dilakukan dengan berbagai cara serta membagi beberapa sub tugas yang digunakan lebih dari satu pengguna. Tugas bisnis dijalankan oleh satu *role*, ada kemungkinan terjadinya banyak penipuan dibandingkan dengan situasi di mana lebih dari satu *role* yang terlibat dalam menjalankan tugas bisnis tersebut (Gligor *et al*, 1998), (Crampton, 2003), (Simon *et al*, 1997), (Sandhu, 1988). Misalnya pada kasus bank locker, bank tersebut membutuhkan dua buah kunci untuk membuka locker tempat penyimpanan uang para nasabahnya. Dengan menggunakan dua buah kunci, keamanan penyimpanan uang nasabah akan lebih terjamin sewaktu membuka kunci locker dibandingkan dengan menggunakan satu kunci. Kedua kunci tersebut harus dijaga dan dioperasikan dua pengguna yang berbeda. Umumnya satu kunci dipegang oleh nasabah bank yang memiliki locker dan satunya lagi dipegang oleh pejabat bank. Jadi pemilik maupun pejabat bank saja yang dapat membuka locker tersebut (Habib, 2011).

Pada kasus lainnya sewaktu melakukan pembayaran yang dilakukan pelanggan dengan menggunakan cek yang diberikan kepada bank. Dalam kasus ini untuk mencairkan cek tersebut harus ada dua buah tanda tangan dari petugas bank untuk menyetujui prosedur pembayaran yang akan dilakukan. Jadi, pada contoh tersebut menunjukkan bahwa terdapat kebutuhan untuk menerapkan konsep SOD pada sistem yang dilakukan secara manual maupun sistem otomatis (Schaad *et al*, 2005).

Penerapan SOD dapat diimplementasikan dengan membagi pengontrolan lebih dari satu pengguna untuk menjalankan tugas penting dan sensitif berguna meminimalkan terjadinya peluang untuk melakukan kecurangan. Tugas tersebut dilakukan dengan cara melibatkan lebih dari satu pengguna dibandingkan hanya menggunakan satu pengguna, sehingga keamanan untuk mengakses sistem dapat terjamin. Jika *role* tunggal mendapatkan hak akses untuk membuat, menandatangani serta menyetujui pemesanan pembelian di sebuah departement store, maka memungkinkan peluang terjadinya kecurangan dalam proses pembelian. Hal itu terjadidiakibatkan satu orang yang dilibatkan dalam proses bisnis tersebut.

Tidak ada yang bisa untuk menghentikannya karena pengguna itu memiliki *role* yang mempunyai wewenang untuk melakukan semua hak akses dalam menjalankan seluruh proses bisnis yang bertentangan dengan konsep SOD. Tapi kalau *role* ini dipecah menjadi dua *role* yang berbeda dan memiliki tugas bisnis berbeda sehingga tugas bisnis akan dijalankan dengan berbagai *role*, sehingga tugas pengguna akan dibatasi sesuai dengan hak akses yang telah diberikan. Dengan menjalankan konsep SOD sedikit kemungkinan terjadinya kecurangan bila dibandingkan dengan menjalankan *role* tunggal (Habib *et al*, 2009).

Konsep SOD diterapkan menimbulkan masalah, pengguna sistem dapat memilih untuk tidak menggunakannya atau menerapkan konsep tersebut. Berarti konsep tersebut direkomendasikan untuk diterapkan pada sistem. SOD diimplementasikan di RBAC harus secara efisien, karena SOD telah diselidiki secara rinci namun belum pernah diimplementasikan dalam segala model kontrol akses sehingga informasi yang didapatkan mengenai SOD tidak sangat mendetail. Generasi selanjutkan dari RBAC akan menjadi sangat dinamis dalam pembatalan *role* (Sandhu et al, 2008). Terdapat banyak jenis SOD

yang telah dibahas dalam berbagai literatur tapi hanya dua buah yang diimplementasikan dalam RBAC standar ANSI.

2.11.2 *Mutual Exclusion* (ME)

Penerapan kebijakan SOD memunculkan gagasan untuk mengimplementasikan *Mutual Exclusion* (ME). *Mutually exclusive* adalah situasi diterimanya salah satu alternatif secara otomatis dengan cara tidak memasukkan alternatif lain. ME merupakan konsep dasar dari SOD. Dalam praktek yang sebenarnya SOD diterapkan dengan mengimplementasikan bantuan dari ME. Pada dasarnya ME yang menentukan situasi hambatan antara dua atau beberapa *role* yang memiliki hambatan di antara mereka, maka *role* tersebut akan dideklarasikan sebagai *mutually exclusive roles* (MER).

Seorang pengguna akan dibatasi untuk menggunakan, menjalankan atau mengaktifkan kedua MER tersebut, dengan batasan tipenya bisa menjadi statis maupun dinamis. Ada juga yang menyatakan hak akses tersebut sebagai *Mutually Exclusive Permissions* (MEP) bukan *mutually exclusive roles* (MER). Dirinci secara mendetail diketahui bahwa terdapat hak akses tertentu yang membuat *role* menjadi *mutually exclusive*. Cara pelaksanaan ME akan menjadikan perbedaan tersebut akan menunjukkan kemudahan bagi pengguna (Habib, 2010).

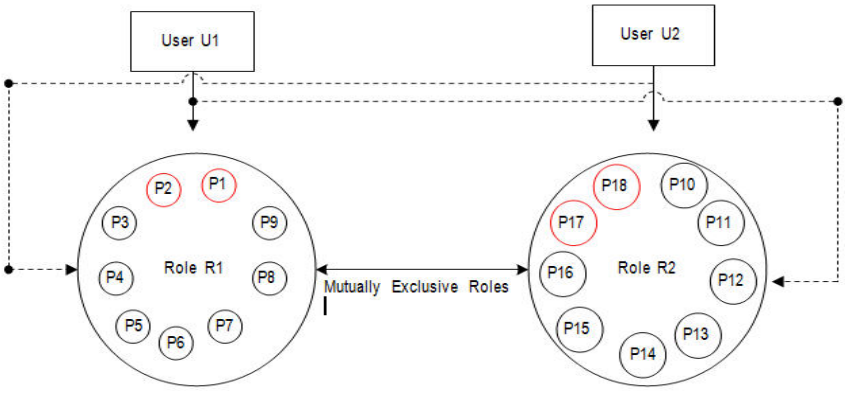
Tipe hak akses terbagi dalam dua jenis yang pertama hak akses positif berguna untuk memberikan wewenang ke pengguna untuk melakukan beberapa tindakan pada objek, sedangkan hak akses negatif membatasi pengguna untuk melakukan tindakan ke objek (Habib, 2010). Hak akses negatif jenis hak akses dapat diberikan *role* dan *role* dapat memberikan ke pengguna berguna untuk membatasi pengguna supaya tidak dapat mengaktifkan atau menjalankan hak akses tertentu. Akhirnya pengguna tidak terikat dalam melakukan tindakan tersebut.

2.12 Penurunan Ke wenangan Pengguna RBAC

Dalam RBAC *role* menjadi perhatian utama dan memiliki posisi strategis, sehingga administrator keamanan membuat *role* sesuai dengan kebutuhan struktur organisasi. Umumnya *role* terdiri dari dua bagian yang salah satunya hak akses yang saling bertentangan dan menciptakan konflik kepentingan dengan hak akses *role* lainnya. Bagian lainnya terdiri dari hak akses yang tidak menciptakan hambatan dengan hak akses dari *role* lainnya. Hak akses yang saling bertentangan dikatakan sebagai *mutually exclusive permissions* dalam mengimplementasikan DSD. Ini berarti bahwa *role* hanya dapat memiliki dua jenis hak akses (Habib, 2011).

Ada kemungkinan bahwa *role* akan mendapatkan seluruh hak akses yang saling bertentangan atau sebaliknya bahwa *role* hanya terdiri dari hak akses yang tidak menimbulkan konflik kepentingan dengan hak akses lainnya. Tetapi jarang terjadi, biasanya *role* merupakan kombinasi hak akses yang saling menimbulkan hambatan dan hak akses yang tidak terjadinya hambatan. Pada kasus untuk menerapkan DSD, pengguna diberi otoritas untuk memiliki dua *mutually exclusive roles* dan dapat mengaktifkan hanya satu *role* dalam satu sessions akibat *mutually exclusive roles* (ANSI, 2004).

Pada RBAC standar, saat *role* dinyatakan sebagai MER untuk menerapkan DSD, semua hak akses akan menimbulkan pengaruh terhadap ME dan menjadi *mutually exclusive permissions* (ANSI 2004).



Gambar 2.6 Hambatan Hak Akses di MER

Pada Gambar 2.6 menjelaskan ranah kewenangan pengguna di RBAC. Terjadinya hambatan di antara dua *role* yaitu R1 dan R2 yang dideklerasikan sebagai *mutually exclusive roles* untuk menerapkan DSD. Alasan terjadinya hambatan pada kedua *role* tersebut dikarenakan bahwa hak akses P1 dan P2 terhadap *role* R1 berada dalam konflik kepentingan dengan hak akses P17 dan P18 terhadap *role* R2. Terjadinya hambatan dari kedua *role* ditampilkan dalam lingkaran merah putus-putus. Ada dua pengguna yang U1 dan U2 mempunyai otoritas pada kedua *role* tersebut, disebabkan *mutually exclusive roles* salah satu pengguna hanya dapat mengaktifkan satu *role* dalam satu session untuk melaksanakan DSD.

Di gambar 2.6 tanda panah padat yang berasal dari pengguna untuk *role* memperlihatkan *role* tersebut diaktifkan dan tanda panah putus-putus menunjukkan bahwa pengguna tidak dapat mengaktifkan *role* yang menjadi sasarannya. Hal ini dianggap pengguna U1 hanya dapat mengaktifkan hak akses P8 dari *role* R2 dalam *session* yang sama, tetapi keduanya berada dalam *role* ME sehingga pengguna U1 tidak mendapatkan akses untuk mengaktifkan MER R2. Saat ini dapat dilihat bahwa pengguna U1 diaktifkan hak aksesnya dari *role* R1 dan seluruh hak akses dari *role* R2 yang telah dibuat tidak tersedia secara otomatis, hal ini sebagai

hasil penerapan dari DSD. Hal ini menjelaskan ranah otoritas pengguna U1 akan berkurang dengan mengabaikan fakta bahwa pengguna yang ingin mengaktifkan supaya tidak terjadinya hambatan hak akses P12 dari MER lainnya. Pada pola yang sama ranah otoritas pengguna U2 akan dikurangi sebagai hasil dari penerapan DSD (ANSI, 2004).

Jika pengguna dituntut untuk diberikan otoritas hak akses apapun dan hak akses tersebut tidak menciptakan hambatan diantara mereka maka pengguna harus mampu untuk mengaktifkan hak akses setiap saat bahkan pada saat mengimplementasikan DSD. Hal ini akan menjadi kelemahan besar dalam mengimplementasikan DSD pada tingkat *role* jika pengguna dibatasi untuk tidak mengaktifkan hambatan hak akses yang berasal dari hambatan *rol*e. Hal ini menjadi masuk akal dan tidak perlu ada pengurangan dalam ranah otoritas pengguna di RBAC. Jika di satu sisi penerapan DSD untuk menghindari pengontrolan dilakukan oleh satu orang maka untuk meminimalkan kecurangan sebagai hasil dari penerapan DSD pada tingkat *role* harus dikurangi ranah otoritas pengguna RBAC, sehingga hal ini tidak akan menjadi solusi optimal.

Misalkan, ada dua *role* masing-masing memiliki seribu hak akses dan kedua *role* dibuat *mutually exclusive roles* yang disebabkan oleh lima hak akses dari satu *role* mengakibatkan terjadinya hambatan diantara mereka dengan tiga hak akses dari *role* lainnya. Kedua *role* memiliki beberapa hambatan di hak akses, karena itulah dibuat sebagai *mutually exclusive roles* untuk menjalankan DSD. Saat pengguna dapat mengaktifkan hanya satu *role* dalam *session* yang sama untuk mengimplementasikan DSD.

Hal ini menunjukkan bahwa P5 bukan hambatan hak akses dari satu *role* atau P7 bukan hambatan hak akses dari *role* lainnya tidak akan tersedia untuk pengguna sebagai akibat dari mengaktifkan satu dari dua *mutually exclusive roles*. Hal ini merupakan kekurangan dalam mengimplementasikan DSD pada tingkat *role*, juga mengurangi ranah otoritas pengguna RBAC yang

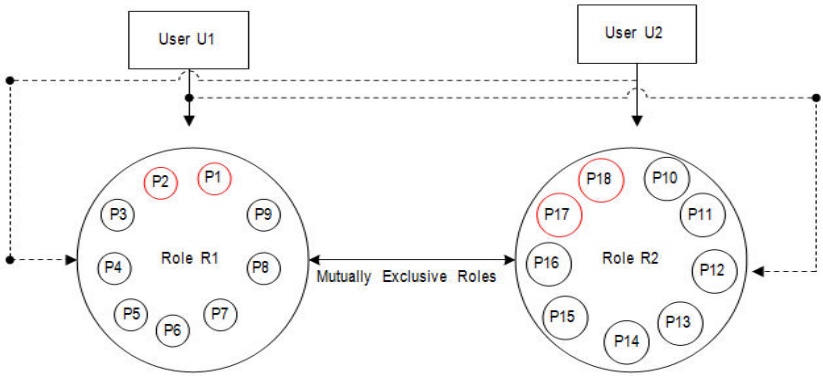
bertentangan dengan *Completeness Property*. Sebagai hasil penerapan DSD pada tingkat *mutually exclusive roles*, setelah mengaktifkan salah satu MER kewenangan pengguna tidak dibolehkan untuk mengaktifkan bukan hambatan hak akses dari MER lainnya. Pengguna RBAC sebagian harus berkorban dari otoritas mereka jika DSD akan diimplementasikan pada tingkat *role* (Habib, 2011).

2.13 Pelanggaran SOD oleh pengguna akhir di RBAC

Dalam RBAC *role* menjadi perhatian utama dan memiliki posisi strategis, sehingga administrator keamanan membuat *role* sesuai dengan kebutuhan struktur organisasi. Umumnya *role* terdiri dari dua bagian yang salah satunya hak akses yang saling bertentangan dan menciptakan konflik kepentingan dengan hak akses *role* lainnya. Bagian lainnya terdiri dari hak akses yang tidak menciptakan hambatan dengan hak akses dari *role* lainnya. Hak akses yang saling bertentangan dikatakan sebagai *mutually exclusive permissions* dalam mengimplementasikan DSD. Ini berarti bahwa *role* hanya dapat memiliki dua jenis hak akses (Habib, 2011).

Ada kemungkinan bahwa *role* akan mendapatkan seluruh hak akses yang saling bertentangan atau sebaliknya bahwa *role* hanya terdiri dari hak akses yang tidak menimbulkan konflik kepentingan dengan hak akses lainnya. Tetapi hal tersebut jarang terjadi, biasanya *role* merupakan kombinasi hak akses yang saling menimbulkan hambatan dan hak akses yang tidak terjadinya hambatan. Pada kasus untuk menerapkan DSD, pengguna diberi otoritas untuk memiliki dua *mutually exclusive roles* dan dapat mengaktifkan hanya satu *role* dalam satu *sessions* akibat *mutually exclusive roles* (ANSI, 2004).

Pada RBAC standar, saat *role* dinyatakan sebagai MER untuk menerapkan DSD, semua hak akses akan menimbulkan pengaruh terhadap ME dan menjadi *mutually exclusive permissions* (ANSI 2004).



2.7 Hambatan Hak Akses di MER

Pada Gambar 2.7 menjelaskan ranah kewenangan pengguna di RBAC. Terjadinya hambatan di antara dua *role* yaitu R1 dan R2 yang dideklerasikan sebagai *mutually exclusive roles* untuk menerapkan DSD. Alasan terjadinya hambatan pada kedua *role* tersebut dikarenakan bahwa hak akses P1 dan P2 terhadap *role* R1 berada dalam konflik kepentingan dengan hak akses P17 dan P18 terhadap *role* R2.

Terjadinya hambatan dari kedua *role* ditampilkan dalam lingkaran merah putus-putus. Ada dua pengguna yang U1 dan U2 mempunyai otoritas pada kedua *role* tersebut, disebabkan *mutually exclusive roles* salah satu pengguna hanya dapat mengaktifkan satu *role* dalam satu *session* untuk melaksanakan DSD. Di gambar 2.7 tanda panah padat yang berasal dari pengguna untuk *role* memperlihatkan *role* tersebut diaktifkan dan tanda panah putus-putus menunjukkan bahwa pengguna tidak dapat mengaktifkan *role* yang menjadi sasarannya.

Hal ini dianggap pengguna U1 hanya dapat mengaktifkan hak akses P8 dari *role* R2 dalam *session* yang sama, tetapi keduanya berada dalam *role* ME sehingga pengguna U1 tidak mendapatkan akses untuk mengaktifkan MER R2. Saat ini dapat dilihat bahwa pengguna U1 diaktifkan hak aksesnya dari *role* R1 dan seluruh hak akses dari *role* R2 yang telah dibuat tidak tersedia secara otomatis, hal ini sebagai hasil penerapan dari DSD. Hal ini menjelaskan ranah otoritas pengguna U1 akan berkurang dengan mengabaikan fakta bahwa pengguna yang ingin mengaktifkan supaya tidak terjadinya hambatan hak akses P12 dari MER lainnya. Pada pola yang sama ranah otoritas pengguna U2 akan dikurangi sebagai hasil dari penerapan DSD (ANSI, 2004).

Jika pengguna dituntut untuk diberikan otoritas hak akses apapun dan hak akses tersebut tidak menciptakan hambatan diantara mereka maka pengguna harus mampu untuk mengaktifkan hak akses setiap saat bahkan pada saat mengimplementasikan DSD. Hal ini akan menjadi kelemahan besar dalam mengimplementasikan DSD pada tingkat *role* jika pengguna dibatasi untuk tidak mengaktifkan hambatan hak akses yang berasal dari hambatan *role*. Hal ini menjadi masuk akal dan tidak perlu ada pengurangan dalam ranah otoritas pengguna di RBAC.

Jika di satu sisi penerapan DSD untuk menghindari pengontrolan dilakukan oleh satu orang maka untuk meminimalkan kecurangan sebagai hasil dari penerapan DSD pada tingkat *role* harus dikurangi ranah otoritas pengguna RBAC, sehingga tidak akan menjadi solusi optimal. Misalkan, ada dua *role* masing-masing memiliki seribu hak akses dan kedua *role* dibuat *mutually exclusive roles* yang disebabkan oleh lima hak akses dari satu *role* mengakibatkan terjadinya hambatan diantara mereka dengan tiga hak akses dari *role* lainnya. Kedua *role* memiliki beberapa hambatan di hak akses, karena itulah dibuat sebagai *mutually exclusive roles* untuk menjalankan DSD.

Saat pengguna dapat mengaktifkan hanya satu *role* dalam *session* yang sama untuk mengimplementasikan DSD. Hal ini menunjukkan bahwa P5 bukan hambatan hak akses dari satu *role* atau P7 bukan hambatan hak akses dari *role* lainnya tidak akan tersedia untuk pengguna sebagai akibat dari mengaktifkan satu dari dua *mutually exclusive roles*. Hal ini merupakan kekurangan dalam mengimplementasikan DSD pada tingkat *role*, juga mengurangi ranah otoritas pengguna RBAC yang bertentangan dengan *Completeness Property*. Sebagai hasil penerapan DSD pada tingkat *mutually exclusive roles*, setelah mengaktifkan salah satu MER kewenangan pengguna tidak dibolehkan untuk mengaktifkan bukan hambatan hak akses dari MER lainnya. Pengguna RBAC sebagian harus berkorban dari otoritas mereka jika DSD akan diimplementasikan pada tingkat *role* (Habib, 2011).

2.14 Pelanggaran SOD oleh pengguna akhir di RBAC

Sesuai standar RBAC, DSD didefinisikan sedemikian rupa sehingga pengguna dapat mengaktifkan salah satu dari dua *mutually exclusive roles* dalam waktu yang sama atau di *session* yang sama. Setiap pengguna untuk mengaktifkan MER lainnya maka pengguna tersebut harus keluar dari *role* yang diaktifkannya, lalu mengakhiri *session* kemudian *login* lagi dengan *session* baru maka saat ini pengguna sudah dapat mengaktifkan MER lainnya. Sebagaimana yang telah dibahas mengenai SOD, definisi dari SOD diterapkan untuk mencegah terjadinya konflik kepentingan.

Hal ini berarti pengguna harus mampu melaksanakan hanya pada tingkat tertentu, itupun ditentukan oleh otoritas. Dengan kata lain, jika terjadi dua *role* yang saling bertentangan dan dapat menimbulkan konflik kepentingan ketika diaktifkan secara bersamaan maka pengguna tidak diperbolehkan untuk mengaktifkan kedua *role* dalam *session* yang sama atau di *session* lainnya.

Sebagaimana yang telah dijelaskan bahwa terdapat dua jenis SOD yaitu SSD dan DSD. SSD merupakan tipe terbatas dan DSD sedang istirahat dalam pengertian pengguna dapat mengaktifkan hampir seluruh *mutually exclusive roles* di *session* berbeda yang berlawanan dengan SSD (ANSI, 2004).

Kedua cara penerapan SOD tidak sesuai yang diharapkan, seperti kasus penerapan SSD pada satu pengguna yang tidak pernah ditugaskan kepada dua *role* yang saling bertentangan, hal tersebut tidaklah realistis. Dengan tipe yang sama tentang SOD terjadi pada kasus di mana kewajiban petugas dibutuhkan untuk ditetapkan sebagai supervisor terletak pada tingkat yang lebih tinggi dari hirarki yang berwenang dalam organisasi (Sandhu, 1990).

Pada penerapan DSD sesuai standar RBAC, pengguna mendapat kebebasan untuk mengaktifkan hampir seluruh atau sebanyak mungkin *mutually exclusive roles* yang ditugaskan di waktu berbeda atau dalam *session* berbeda. Setiap pengguna mengaktifkan dua *mutually exclusive roles* bahkan dalam *session* yang berbeda konsep SOD akan di langgar. Sebab itu perlu dukungan penerapan DSD dengan konsep yang berbeda untuk mencegah pelanggaran SOD (ANSI, 2004).

2.15 Pelanggaran terhadap SOD oleh Administrator Keamanan

Pada pembahasan terdahulu SOD dapat dilanggar oleh pengguna akhir tetapi juga bisa dilanggar oleh administrator keamanan secara tidak disengaja. Dalam standar RBAC merekomendasikan penerapan DSD pada tingkat *mutually exclusive roles* mengakibatkan hak akses tertentu akan menimbulkan konflik kepentingan satu sama lainnya. Sehingga, administrator keamanan menangani *mutually exclusive roles* bukan mengawasi hambatan yang terjadi di hak akses karena tidak ada mekanisme untuk menjaganya agar dapat mengetahui hambatan di hak akses. Dengan

cara seperti ini pengguna akan diberikan *role* tersebut dan bisa melanggar DSD saat mengaktifkan kedua hak akses yang saling bertentangan (ANSI, 2004).

Mendeklarasikan *mutually exclusive roles* dasarnya merupakan cara pintas untuk mendeklasikan *mutually exclusive permissions*. Misalkan, ada sebuah penawaran dari bank kecil hanya mempunyai tiga tipe karyawan seperti teller, manajer akuntansi, dan manajer cabang. Teller akan berhubungan dengan transaksi harian, transaksi manajer akuntansi dengan cara membuka rekening baru dan tawaran dari manajer cabang untuk mengembangkan kebijakan pemasaran yang baru untuk menghasilkan pendapatan dengan mendapatkan nasabah yang lebih banyak untuk bank. Untuk mencapai tujuan tersebut manajer cabang mengumumkan sebuah kebijakan bahwa jika seorang nasabah baru akan membuka rekening di bank tersebut, maka nasabah akan mendapatkan hadiah sebuah pemanggang roti yang baru pada kunjungan berikutnya ke bank. Manajer akuntansi saat membuka rekening baru maka rekening nasabah tersebut secara otomatis akan memenuhi syarat untuk mendapatkan pemanggang roti pada kunjungan berikutnya. Saat nasabah berkunjung ke bank pada kunjungan berikutnya, teller tersebut akan mendapatkan peringatan bahwa nasabah tersebut mendapatkan hadiah. Teller akan menekan tombol hadiah dan pemanggang roti secara otomatis akan dikirim ke alamat nasabah.

Dalam mengimplementasikan SOD kedua *role* manager akuntansi dan teller dibuat *mutually exclusive* sehingga pengguna tidak dapat memiliki kedua *role* untuk mencegah mendapatkan pemanggang roti bagi dirinya sendiri dengan cara membuat rekening baru tapi palsu. Setelah diluncurkan program ini, berhasil menarik banyak nasabah dan pengelolaannya menjadi rumit. Sehingga, kantor pusat memperhatikan permasalahan tersebut dan mengarahkan departemen TI untuk memberikan hak akses membuka rekening baru ke teller untuk membantu manajer akuntansi. Departemen TI melakukan hal tersebut disebabkan oleh tidak ada

kekhawatiran tentang SOD karena kedua *role* telah dinyatakan sebagai *mutually exclusive roles*. Saat teller mulai mengaktifkan kedua hak akses yang bertentangan tiba-tiba ada pengiriman pemanggang roti dalam jumlah besar terhadap rekening palsu baru. Departemen TI menyatakan tidak terjadi permasalahan pada sistem tersebut tetapi akhirnya kedua *role* manager akuntasi dan teller adalah *mutually exclusive* dan tidak dapat diaktifkan oleh pengguna yang sama.

Persoalannya bahwa peraturan SOD telah ditetapkan pada tingkat *role* di mana perlu diberlakukan pada tingkat hambatan di hak akses. Dari contoh ini bisa dirasakan bahwa SOD tidak akan diimplementasikan pada tingkat hambatan hak akses maka sistem akan selalu beresiko sebab hambatan hak akses dapat ditugaskan ke *role* yang sama setiap saat secara tidak sengaja disebabkan SOD di diimplementasikan di *role* sehingga tidak mengetahui hambatan di hak akses. Hal ini menunjukkan memberlakukan aturan SOD pada tingkat hambatan hak akses merupakan pendekatan yang lebih realitas dan handal dibandingkan dengan menerapkan SOD pada tingkat hambatan *role* (Habib, 2011).

2.16 Pekerjaan Besar Administrator Keamanan

Administrator keamanan secara manual membuat hak akses satu persatu dan menyatakan hambatan *role* sebagai *mutually exclusive roles* hal ini sesuai standar RBAC. Hal ini membutuhkan waktu dan pekerjaannya cenderung rawan terhadap kesalahan. Seharusnya ada beberapa mekanisme yang otomatis untuk membuat hak akses dan *mutual exclusion* harus dibuat secara otomatis pada tingkat hak akses. Tidak adanya membuat hak akses dalam RBAC akan membuat sulitnya pengelolaan otoritas pengguna. Disebabkan pembuatan hak akses tidak mudah untuk mencabut serta menetapkan hak akses ke *role* secara manual. Sehingga sistem harus mampu

mengurangi sebagian besar beban kerja administrator keamanan (Habib, 2011).

2.17 Kajian Riset Terkait

Belokoszto lszki. A, David M. Eyers, Wei Wang dan Ken Moody dalam penelitiannya menjelaskan manfaat kontrol akses di bidang kesehatan. Kontrol akses akan bekerja sehingga pengguna yang telah diberikan hak akses untuk mengakses data tersebut dan akan menyelesaikan pembaharuan hak akses yang telah diberikan sesuai dengan kebijakan yang didapatkannya. Hal tersebut akan mengalamai kesulitan untuk diterapkan terutama dalam organisasi yang berskala besar.

Penelitian ini termodifikasi oleh *Electronic Health Record* (EHR) di UK *National Health Service* (NHS). Dalam NHS kontrol akses pada *record* pasien harus ditegakkan dalam melakukan distribusi data yang berskala besar di mana lokasi domain (misalnya rumah sakit) akan memiliki berbagai tingkat otonomi, sehingga untuk menjadi lebih terukur maka ia harus mendukung kebijakan distribusi di seluruh domain. Persyaratan lebih lanjut dari *two extremes of policy granularity*, di satu sisi mungkin NHS *wide standard* akan membuat suatu kebijakan yang diperlukan, dan di sisi lain UK *data protection* dapat mengakibatkan persyaratan bahwa pasien dapat memilih untuk membatasi akses ke *record* mereka sendiri, jadi dalam cara ini akan membuat administrator melakukan kebijakan tertentu.

Setiap administrasi yang akan dilakukan sangat terdistribusi, jika perlu dengan menetapkan kontrol akses atas kebijakan representasi kontrol akses itu sendiri, maka kebijakan dalam penyimpanan data akan melakukan pengontrolan sesuai dengan kepercayaan serta hak istimewa dan akan berkompetensi secara beragam.

Kajian riset terkait lainnya adalah sebagai berikut:

Tabel 2.1 Kajian Riset Terkait

No	Judul	Masalah	Kontribusi	Peneliti
1	*ROLE-BASED ACCESS CONTROL FOR INFORMATION FEDERATIONS IN THE INDUSTRIAL SERVICE SECTOR*	Penyediaan tool yang berbasis *role* pada kebijakan organisasi jika perubahan kebijakan itu sering berubah-ubah	Merancang solusi kontrol akses pada perusahaan penyedian produk listrik dan otomasi teknologi bagi pelanggan pada sektor industri	Kunz.S, Evdokimo v. S, Fabian.B, Stieger.B, Strembeck . M Mark
2	*INCREASING PERFORMANCE AND GRANULARITY IN ROLE-BASED ACCESS CONTROL SYSTEMS A CASE STUDY IN GRSECURITY*	Penggunaan pendelegasian pada suatu organisasi akan meningkatkan fleksibilitas dan skalabilitas yang akan mengurangi keunggulan utama pada RBAC	Mengajukan sebuah teknik analisis keamanan dalam melakukan pendelegasian hak akses dalam memberikan definisi yang tepat dalam RBAC dibandingkan analisis yang dipelajari dalam literatur.	Li N, Tripunitar a. M. V
3	*Secure RBAC with Dynamic, Efficient, & Usable DSD*	Pelaksanaan *Dynamic Separation of Duty* (DSD) dapat membuka peluang terjadinya pelanggaran SOD pada standarisasi RBAC baik itu dilakukan oleh pengguna akhir maupun administrator keamanan	Mengajukan suatu konsep dalam memperluas standar RBAC dengan cara meningkatkan keamanan, fleksibelitas operasional serta efisiensi dalam penggunaan RBAC	Habib. M.A

This page is intentionally left blank

BAB 3

METODE PENELITIAN

3.1 Tujuan Penelitian

Sesuai dengan tujuan yang ingin dicapai dalam melakukan penelitian DSD (*Dynamic Separation of Duty*) dengan menggunakan hambatan pada hak akses bukan hambatan pada *role*, serta mengimplementasikan studi kasus pendataan masyarakat miskin di Kabupaten Aceh Utara bukan suatu hal yang sangat mudah, pendataan membutuhkan banyak pengguna yang akan menanggani permasalahan tersebut. Penerapan DSD pada tingkat *role* di satu sisi menjadikan sistem lebih aman dari ancaman keamanan internal tetapi di sisi lain menciptakan berbagai permasalahan bagi pengguna RBAC termasuk pengguna akhir maupun administrator keamanan.

3.2 Bahan-bahan

Penelitian ini akan menggunakan bahasa pemograman Microsoft Visual Basic 6.0 dan memakai database Microsoft SQL Server 2000 untuk merancang kontrol akses RBAC dan juga mempergunakan beberapa software aplikasi lainnya seperti Microsoft Office 2007 yang akan digunakan sebagai alat pendukung dalam perancangan sistem ini.

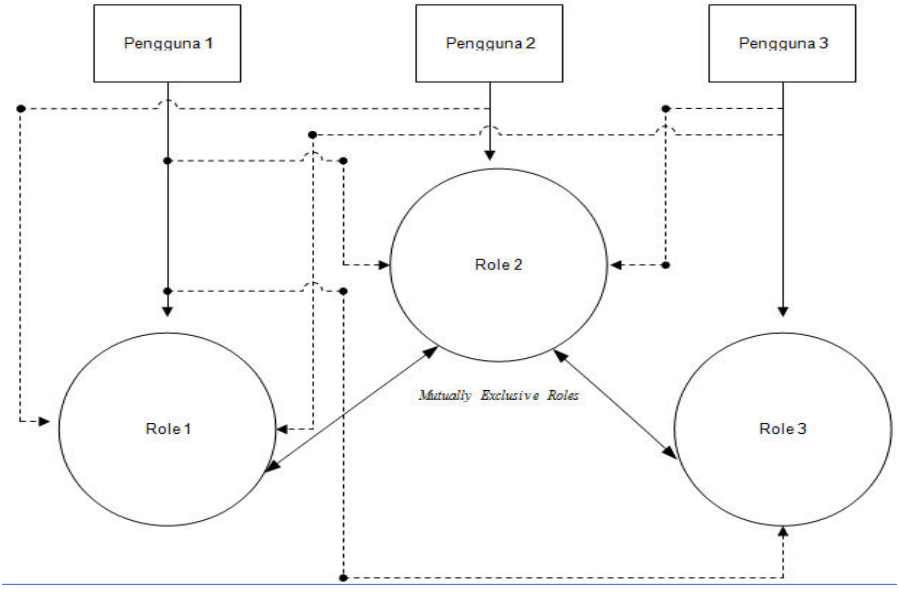
3.3 Analisa Sistem

Setelah mengetahui analisis dari algoritma yang akan dirancang, selanjutnya dilakukan analisis sistem. Dalam hal ini analisis sistem yang akan dilakukan terdiri dari analisis permasalahan dan analisis kebutuhan dari sistem yang akan dirancang.

3.4 Analisa Permasalahan

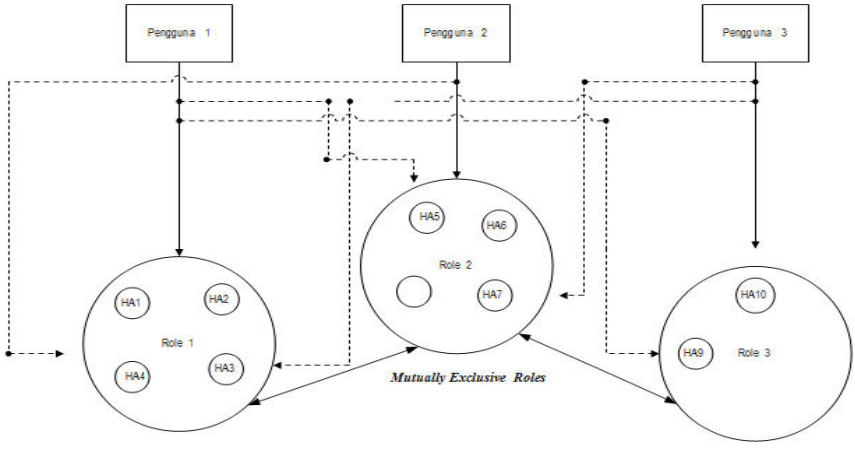
Berikut adalah model yang diajukan dan dijelaskan secara terperinci dengan mengikuti skenario yang digambarkan di bawah ini. Dalam ilustrasi gambar di bawah ini diproses secara lengkap sesuai yang dilakukan oleh *role*. Seperti dalam contoh melibatkan tiga pengguna dan tiga *role* dengan hak akses yang ganda. Misalkan kita memiliki tiga pengguna yaitu pengguna1, pengguna2 dan pengguna3 serta mempunyai otoritas untuk mengaktifkan tiga *mutually exclusive roles* yaitu Role1, Role2 dan Role3. Setiap pengguna akan mengaktifkan salah satu dari tiga *mutually exclusive roles*, maka pengguna tersebut tidak dapat mengaktifkannya baik itu sebagian maupun keseluruhan *role* yang ada dalam *session* yang sama sesuai dengan definisi DSD standar.

Pada gambar 3.1 pengguna1 ingin mengaktifkan Role1 dan pengguna2 mengaktifkan Role2 begitu juga pengguna3 mengaktifkan Role3. Tanda panah putus-putus memperlihatkan pengguna tidak dapat mengaktifkan *role* diakibatkan pelaksanaan DSD dari segi *mutually exclusive roles*. Akibat penerapan DSD pada tingkat *role*, pengguna RBAC ke hilangan ranah otoritas mereka.



Gambar 3.1 Tugas Pengguna dalam *Mutually Exclusive Roles*

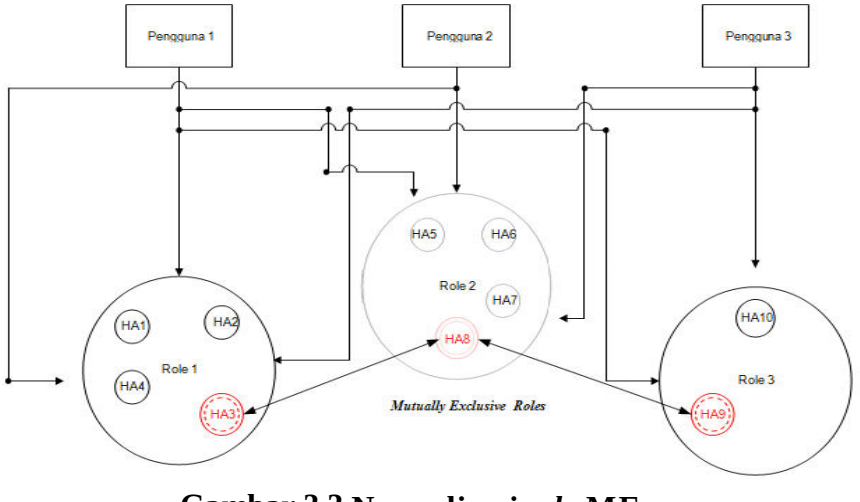
Pada gambar 3.2 menampilkan hak akses pengguna dan telah ditentukan secara bersama hasil dari penerapan DSD pada tingkat *mutually exclusive roles*. Role1 terdiri dari hak akses HA1 sampai HA4, Role2 hak aksesnya HA5, HA6, HA7, HA8 dan Role3 hak aksesnya adalah HA9 dan HA10.



Gambar 3.2 Hak Akses dalam MER

Penerapan ranah otoritas pengguna mengalami kekurangan besar dalam mengimplementasikan DSD dari segi *mutually exclusive*, diakibatkan oleh pengguna RBAC kehilangan bagian dari ranah otoritas mereka. Sekarang kita akan menganalisa skenaria model yang peneliti usulkan dengan mengimplementasikan DSD pada tingkat *mutually exclusive permissions*. Dalam gambar 3.3 menunjukkan *normalisasi role* di lingkaran merah yang memuat *mutually exclusive permissions* dalam lingkaran merah putus-putus dan sisa hak akses terletak di luar lingkaran merah.

Sekarang pengguna1 bisa mengaktifkan semua hak akses dari ketiga *role* tersebut setiap saat kecuali hak akses yang saling bertentangan dan DSD akan diimplementasikan di tingkat konflik hak akses bukan tingkat *role*. Misalkan pengguna1 ingin mengaktifkan hak akses HA3 di Role1, dia tidak dapat mengaktifkan hak akses HA8 di Role2 dan dia bisa mengaktifkan semua sisa hak akses di Role2, begitu juga dengan pengguna3.

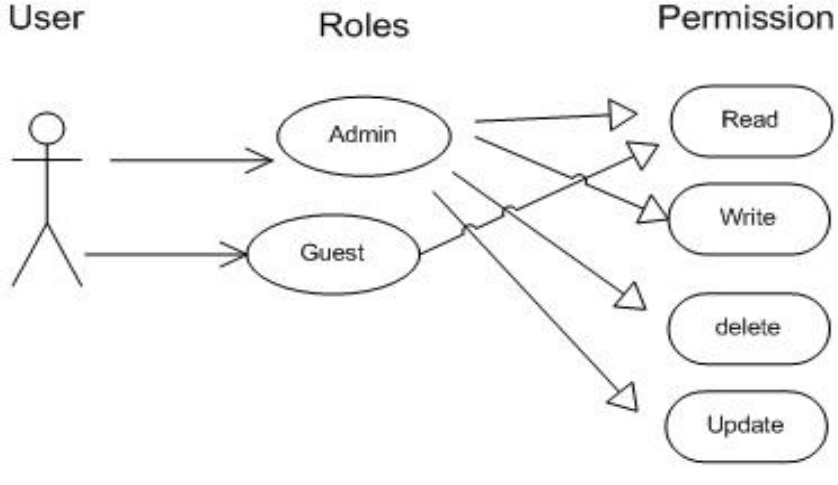


Gambar 3.3 Normalisasi *role* ME

3.5 Perancangan

Berikut adalah diagram analisis permasalahan dari sistem yang akan dibuat dengan menggunakan use case diagram yang merupakan model untuk menunjukkan bagaimana sistem itu bekerja dalam RBAC. Kontrol akses dalam sistem informasi pendataan penduduk miskin di Kabupaten Aceh Utara dapat meningkat secara signifikan dengan mengembangkan dan menerapkan model kontrol akses yang dapat memperluas prinsip yang terdapat dalam RBAC. Kontrol akses dengan menggunakan RBAC akan menambahkan penggunaan informasi kontekstual yang bersifat dinamis untuk keputusan dalam melakukan kontrol akses, sehingga akan didapatkan kontrol akses yang fleksibel dalam menjaga keamanan sumber daya.

Dalam melakukan klasifikasi akan diberikan *roles* dari masing-masing pengguna yang akan melakukan akses di dalam database tersebut, contoh gambar dibawah ini.



Gambar 3.4 *Role Based Acess Control Model*

Dibawah ini keterangan gambar 3.4 adalah sebagai berikut:

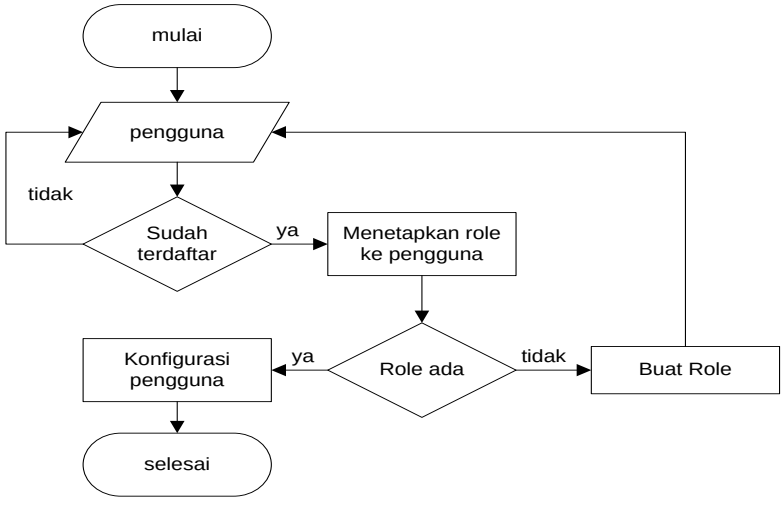
User : Seorang pengguna adalah manusia atau agen independen

Roles : Sebuah peran adalah jenis fungsi pekerjaan dalam suatu organisasi yang memiliki beberapa izin terkait ke sumber daya aman dan data.

Permissions (operasi): Perizinan adalah hak artau wewenang yang diberikan untuk mengakses atau melakukan operasi pada sumber daya yang telah dilindungi. Obyek adalah sumber daya yang dilindungi atau tugas dalam sistem. Operasi adalah suatu tindakan yang dijalankan dilakukan oleh pengguna.

3.5.1 Penetapan *Role* ke Pengguna

Model kontrol akses dengan menggunakan RBAC dapat digambarkan sebagai berikut:



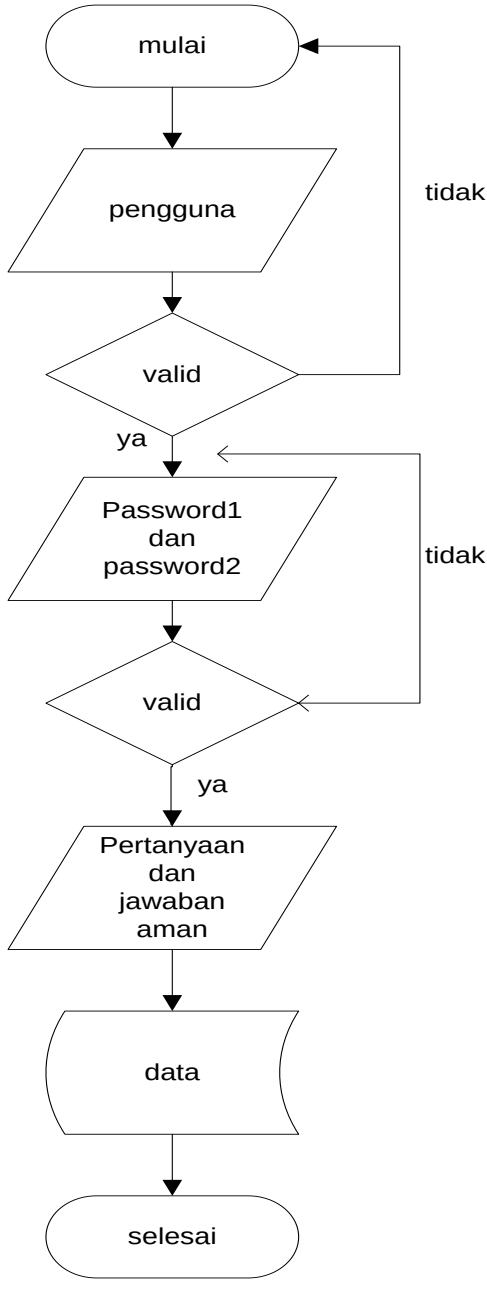
Gambar 3.5 Diagram Metode Kontrol Akses dalam Database

Dari gambar di atas dapat dijabarkan bahwa akses data dalam sebuah database harus dikontrol dengan ketat dan akses akan diberikan kepada yang berhak, dengan RBAC segala aktivitas atas akses ke data tersebut dapat diberikan dan dikontrol kepada siapa saja yang berhak untuk mengakses data tersebut.

Dalam penelitian ini sistem akan memeriksa *role* pengguna, *username* dan *password* ketika pengguna melakukan *login* ke sistem. Ada dua fungsi yang akan dilakukan oleh semua pengguna sebelum memasuki sistem yaitu mengubah *password* dan mengatur pertanyaan yang aman. Semua pengguna dapat mengubah *password* mereka dan mengisi *username* dan *password* dengan benar setelah mereka *log in*. Jika pengguna lupa *password*, pengguna akan mendapatkan *password* dengan cepat dengan menjawab pertanyaan aman yang telah ditetapkan sebelumnya.

Untuk meningkatkan keamanan *prototype*, sistem otomatis akan *log-off* jika tidak tindakan di sistem dalam jangka waktu tertentu, waktunya dapat diatur dan disesuaikan oleh administrator. Fungsi ini berguna untuk mencegah terjadinya kebocoran informasi yang tidak diharapkan. Sebagai contoh, salah satu pengguna lupa *log-off* di sistem tersebut dan meninggalkan komputer, orang lain dapat melihat informasi yang ada di sistem tersebut sehingga pihak yang tidak berkepentingan akan mengubah atau melakukan hal-hal yang tidak diinginkan. Oleh sebab itu secara otomatis *log-off* juga mengontrol skenario ini.

3.5.2 Pendaftaran bagi para pengguna

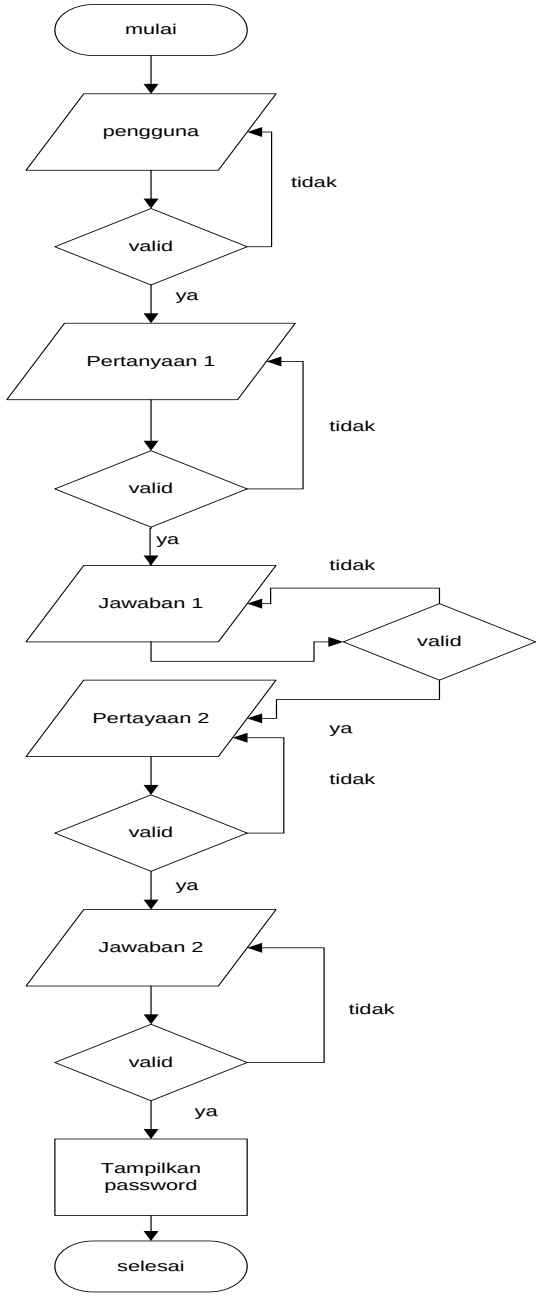


Gambar 3.6 Diagram Pendaftaran Pengguna

Pada gambar 3.6 skenario pendaftaran pengguna dilakukan oleh sistem, dengan cara pengguna dapat membuat *username* sendiri dan *password* dengan syarat *username* tersebut tidak boleh sama dengan *username* yang telah terdaftar di sistem. Sewaktu pembuatan password, sistem akan memberikan kotak pertanyaan ke pengguna beserta jawabannya yang diisi oleh sipengguna. Pertanyaan dan jawaban tersebut bebas diisi oleh pengguna serta jawabannya juga bebas yang diberikan otoritas penuh kepada sipengguna untuk membuat pertanyaan sendiri serta jawab sendiri.

Dalam ilustrasi gambar diatas sistem memberikan kotak masukan kepada pengguna untuk memasukkan *username*nya, oleh sistem dilakukan seleksi apakah username tersebut valid jika tidak valid silahkan pengguna memasukkan *username* lainnya. Jika valid pengguna akan diberikan kotak masukkan password1 dan password2 jika passwod1 dan password2 sama atau valid maka sistem mengajukan pertanyaan atau jawaban keamanan kepada pengguna jika pengguna lupa akan *password*nya. Jika pertanyaan dan jawaban telah dimasukkan *username* dan *password* akan disimpan ke sistem.

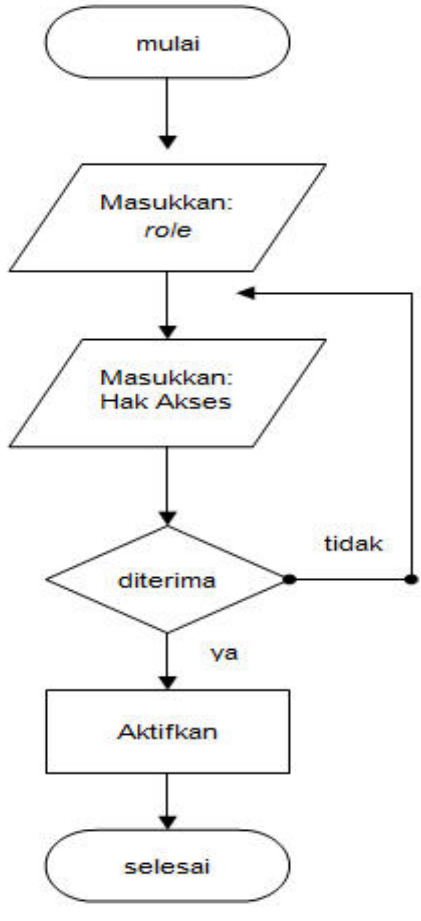
3.5.3 Pengguna kehilangan password



Gambar 3.7 Diagram Pengguna Kehilangan *Password*

Pengguna kehilangan *password* sistem dapat mengembalikan lagi, jika pengguna tersebut kehilangan *password*nya atau lupa *password* apa yang disimpan disistem. Dalam gambar 3.7 diilustrasikan sistem memberikan kotak masukkan kepada pengguna. Pengguna memasukkan *username* yang telah daftarkan ke sistem. Kemudian sistem mengecek apakah pengguna tersebut telah ada dalam sistem jika tidak ada sistem akan memberikan kotak isian pengguna lagi. Jika pengguna ada atau valid *username* di sistem, sistem akan kotak isian pertanyaan pertama. Pengguna mengisi pertanyaan yang dibuat pada sistem jika tidak valid maka sistem akan memberikan kotak isian pertanyaan pertama. Jika pertanyaan pertama valid sistem akan memberikan jawaban pertama dan pengguna harus menjawabnya sesuai jawaban yang diisi sewaktu pengguna melakukan pendaftaran pertama sekali disistem. Jika jawaban pertama tersebut salah sistem akan mengeluarkan kotak isian jawaban pertama. Jika jawaban pertama benar sistem akan menlanjutkan ke pertanyaan kedua jika pertanyaan kedua salah sistem akan mengeluarkan kotak isian pertanyaan kedua. Jika pertanyaan kedua benar atau valid sistem akan melanjutkan ke jawaban kedua. Jika jawaban kedua tidak valid sistem akan kembali ke kotak isian jawaban kedua. Jika jawaban kedua tersebut valid sistem akan menampilkan *password* pengguna tersebut.

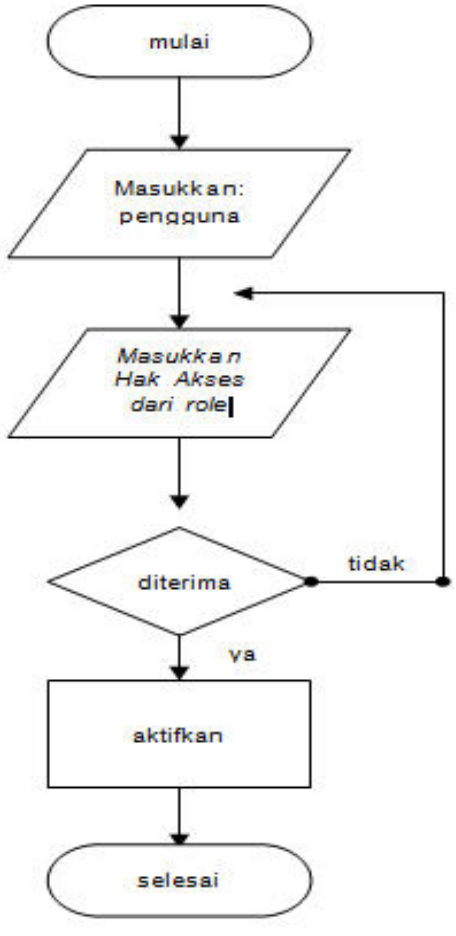
3.5.4 Pemberian Hak Akses ke *Role*



Gambar 3.8 Pemberian Hak Akses ke *Role*

Pada gambar 3.8 administrator memasukkan *role* kemudian memasukkan hak akses, sistem menanyakan apakah hak akses tersebut diterima atau ditolak, jika diterima hak akses dalam *role* diaktifkan kalau hak akses tersebut ditolak maka administrator memasukkan hak akses yang baru.

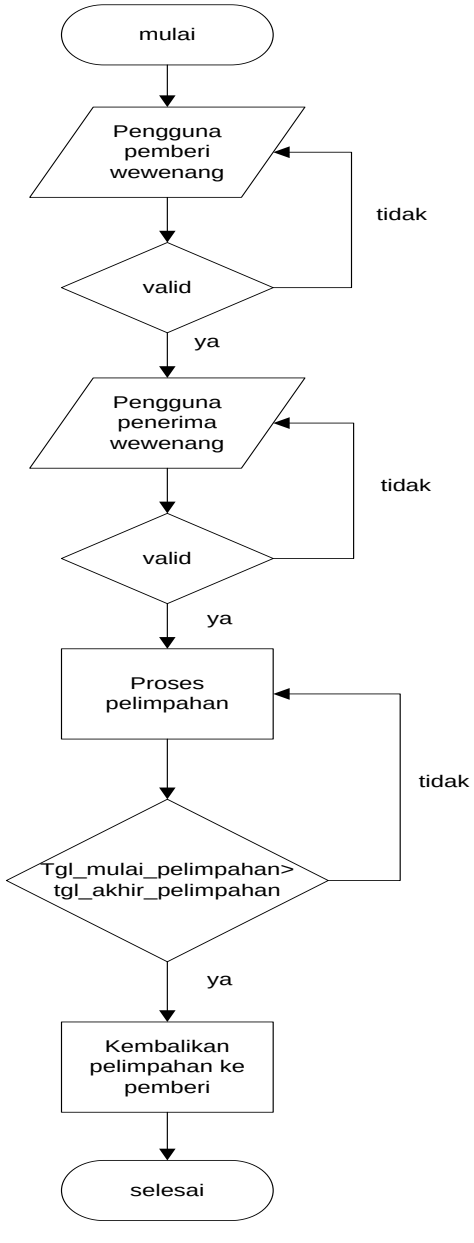
3.5.5 Pemberian Hak Akses Ke Pengguna



Gambar 3.9 Pemberian Hak Akses ke Pengguna

Pengguna perlu diberikan hak akses untuk masuk ke dalam sistem, administrator memasukkan pengguna yang akan diberikan hak akses dari *role*, sistem menanyakan apakah pengguna tersebut diterima hak aksesnya dalam *role* tersebut, jika diterima maka pengguna akan mendapatkan hak aksesnya, jika ditolak oleh sistem maka administrator memberikan hak akses baru ke pengguna.

3.5.6 Pelimpahan wewenang pengguna ke Pengguna lain



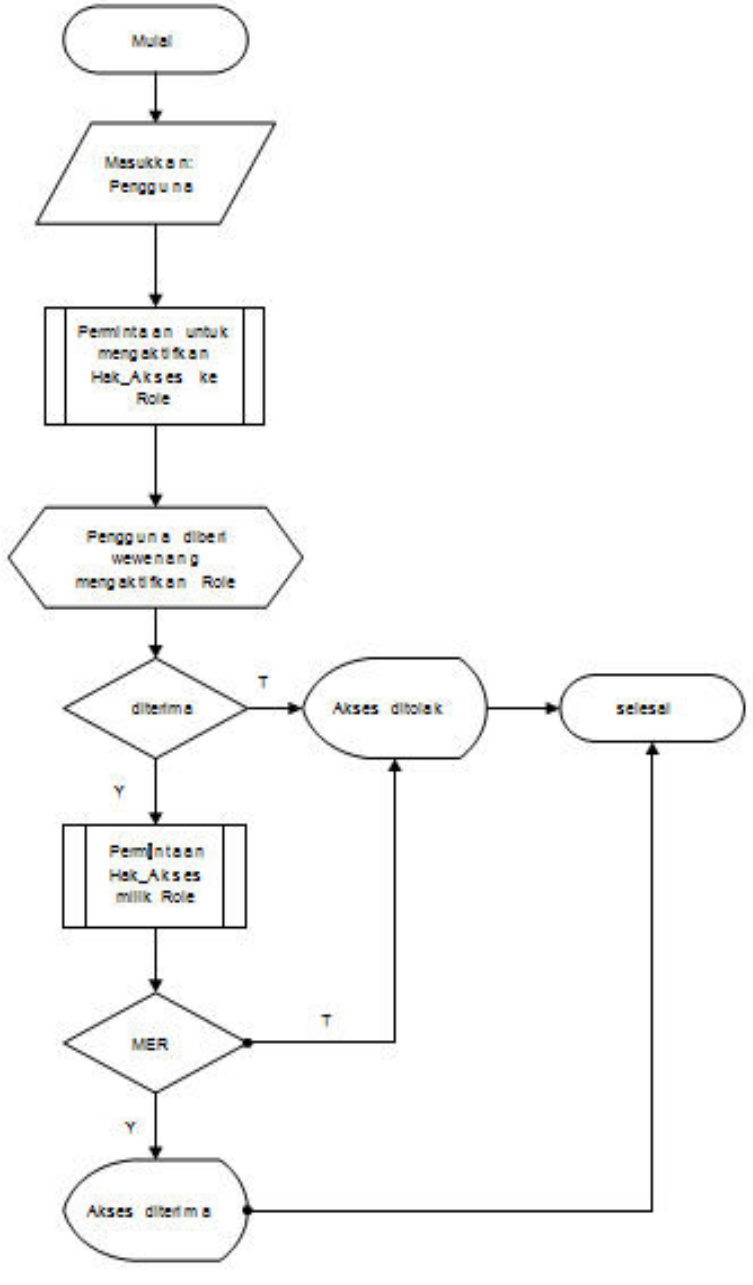
Gambar 3.10 Diagram Pelimpahan wewenang

Pelimpahan wewenang ke pengguna lain dilakukan oleh pengguna pertama dan diberikan wewenang oleh pengguna pertama ke pengguna kedua. Proses alur kerjanya pengguna memberikan wewenang kemudian sistem mengecek apakah wewenang tersebut valid atau tidak. Jika tidak valid kembali kekotak isian pengguna pemberi wewenang, jika valid keluar kotak isian pengguna penerima wewenang. Sistem akan mengecek kembali apakah penerima wewenang tersebut valid atau tidak. Jika tidak valid kembali ke kotak isian penerima wewenang, jika valid sistem langsung memproses pelimpahan berhasil dilaksanakan.

Setelah pelimpahan berhasil diberikan ke pengguna kedua dari pengguna pertama, sistem akan mengecek kembali apakah tgl_akhir_pelimpahan>tgl_sekarang. Jika kondisi tidak valid maka pelimpahan dilanjutkan, jika kondisi valid maka proses pelimpahan berakhir dan wewenang pengguna kedua dicabut dan akan kembali menjadi wewenang pengguna pertama.

3.6 Pemberian Hak Akses dalam DSD ANSI 2004

Pada DSD ANSI 2004 pemberian kontrol akses dilakukan oleh administrator kepada pengguna seperti pada gambar di bawah ini.



Gambar 3.11 Pemberian Hak Akses dalam *Role*

Administrator memberikan hak akses yang ada dalam *role* ke pengguna. Kemudian pengguna mengaktifkan hak akses yang diberikan oleh administrator. Jika aksesnya diterima maka pengguna berhak mengakses sistem sesuai dengan kewenangan yang diberikan ke pengguna tersebut, jika aksesnya ditolak maka pengguna tidak dapat mengakses sistem.

Proses kerja dalam ilustrasi gambar 3.6 adalah administrator memasukkan nama pengguna kemudian sistem melakukan proses permintaan pemprosesan pengaktifan hak akses dalam *role*. Kemudian sistem memproses kembali administrator diberi hak untuk mengaktifkan wewenang untuk mengaktifkan *role* bagi pengguna. Sistem memproses pemberian wewenang yang dilakukan oleh administrator, jika akses ditolak langsung keluar dari sistem. Jika akses diterima sistem melakukan permintaan hak akses dalam *role*. Jika MER ditolak oleh sistem permintaan tersebut tidak diterima oleh sistem. Jika permintaan diterima pemberian akses diproses oleh sistem dan pengguna tersebut mendapatkan wewenang hak akses dalam *role*.

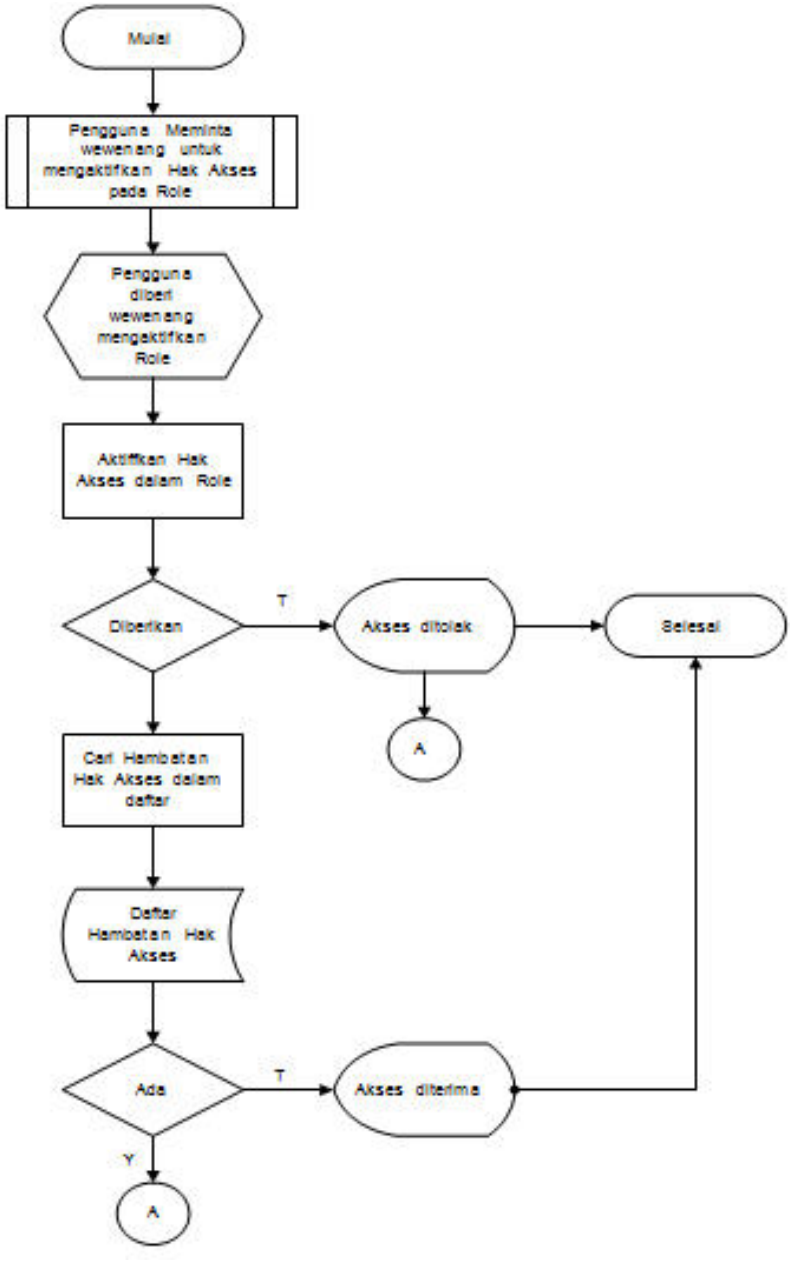
3.7 Hambatan Hak Akses dalam DSD

Pada gambar 3.12 hambatan hak akses dilakukan dengan langkah sebagai berikut:

1. Sistem menerima permintaan pengguna untuk mengaktifkan hak akses pada *role* yang diberikan oleh administrator.
2. Pertama sistem memverifikasi apakah pengguna yang telah diberikan wewenang boleh melakukan akses ke dalam sistem atau tidak. Jika pengguna tidak diberi wewenang untuk mengakses sistem maka pengguna akan menerima pesan bahwa aksesnya ditolak.
3. Dalam kasus ini pengguna berwenang untuk meminta hak akses, sistem memeriksa permintaan hak akses apakah

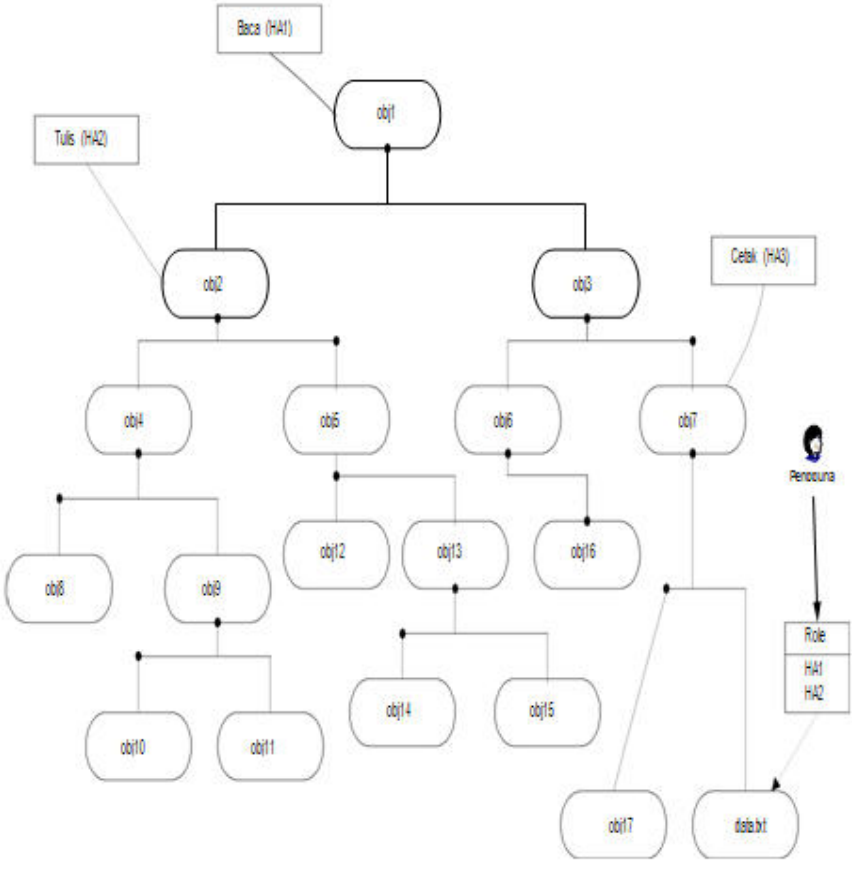
terjadi hambatan dengan hak akses lain atau tidak. Jika hak akses tidak terjadi hambatan dengan hak akses lain maka sistem memeriksa hak akses pengguna dalam daftar hambatan hak akses.

4. Jika dalam daftar hak akses yang ada, pengguna sudah pernah diberi akses untuk mengakses hak akses tersebut, maka permintaan hak aksesnya ditolak.
5. Jika permintaan hak aksesnya belum diaktifkan maka pengguna mendapatkan akses untuk mengaktifkan hak akses yang diminta.



Gambar 3.12 Hambatan Hak Akses dalam *Role*

3.8 Hambatan Hak Akses dalam Objek



Gambar 3.13 Hambatan Hak Akses dalam Objek

Pada gambar 3.13 menunjukkan objek berbeda di simpan di bawah pohon direktori yang berbeda. Tiga kegiatan berbeda ditugaskan ke direktori berbeda untuk membuat tiga hak akses yaitu HA1, HA2 dan HA3. Administrator membatasi akses ke direktori bagi pengguna ingin mengaksesnya kegiatan tersebut.

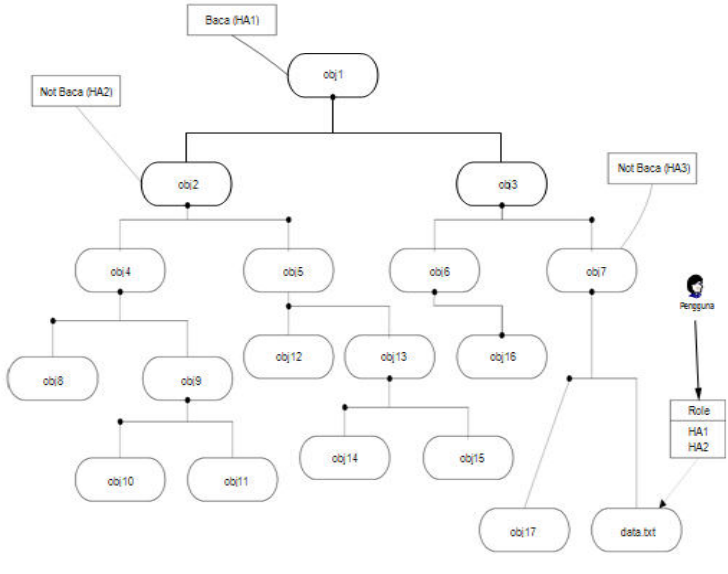
Kegiatan akses dibatasi ke pengguna yang mendapatkan akses HA1 yang mempunyai tugas untuk membaca, hak akses HA2 untuk tulis, dan HA3 untuk cetak. Hak akses HA2 dan HA3 mutually

exclusive permissions dibuat oleh administrator maka pengguna yang mempunyai hak akses HA1 tidak dapat mengakses HA2 dan HA3.

Setiap direktori berada dalam pohon direktori mewakili objek tertentu. Permintaan pengguna untuk mengaktifkan hak akses HA1, maka sistem menemukan hak akses sudah dinyatakan sebagai *mutually exclusive permissions* dengan hak akses HA2. Sistem memeriksa apakah pengguna tersebut sudah diaktifkan hak akses HA2. Jika pengguna tersebut sudah diaktifkan hak akses HA2 maka pengguna mendapatkan pesan aksesnya ditolak.

3.9 Hak Akses Negatif dalam Kegiatan Baca

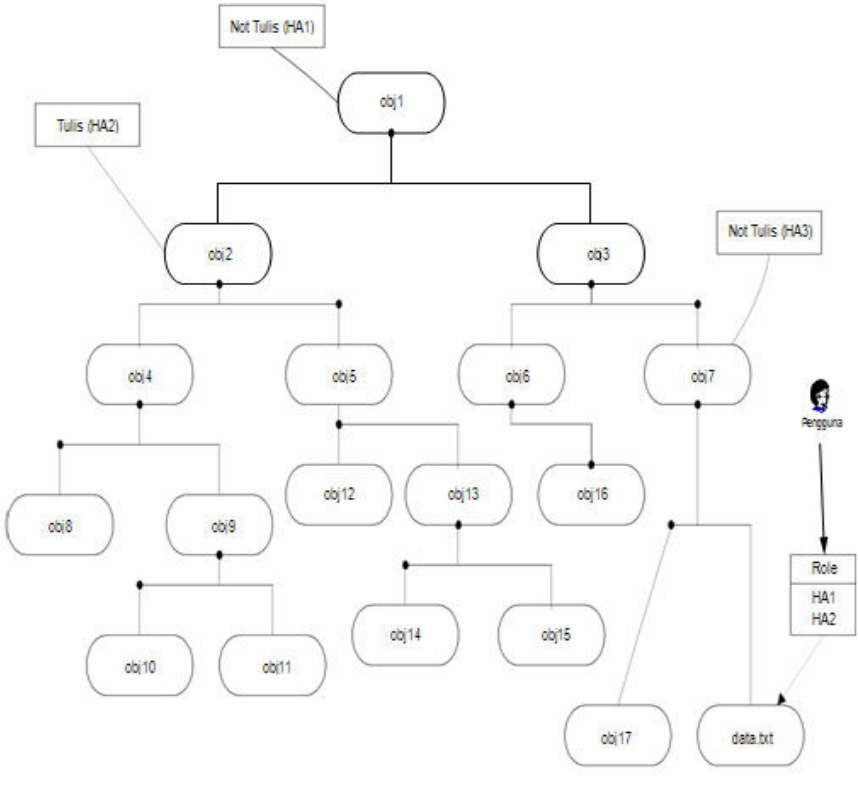
Di gambar 3.14 pengguna melakukan permintaan hak akses HA1 untuk mengakses berkas data.txt, akses HA1 adalah baca. Pengguna melakukan akses baca data.txt harus melewati objek obj7, tapi objek obj7 dimiliki oleh hak akses HA3 dengan akses Not Baca. Maka oleh sistem permintaan pengguna untuk akses berkas data.txt ditolak.



Gambar 3.14 Pohon Direktori dalam Kegiatan Baca

3.10 Hak Akses Negatif dalam Kegiatan Tulis

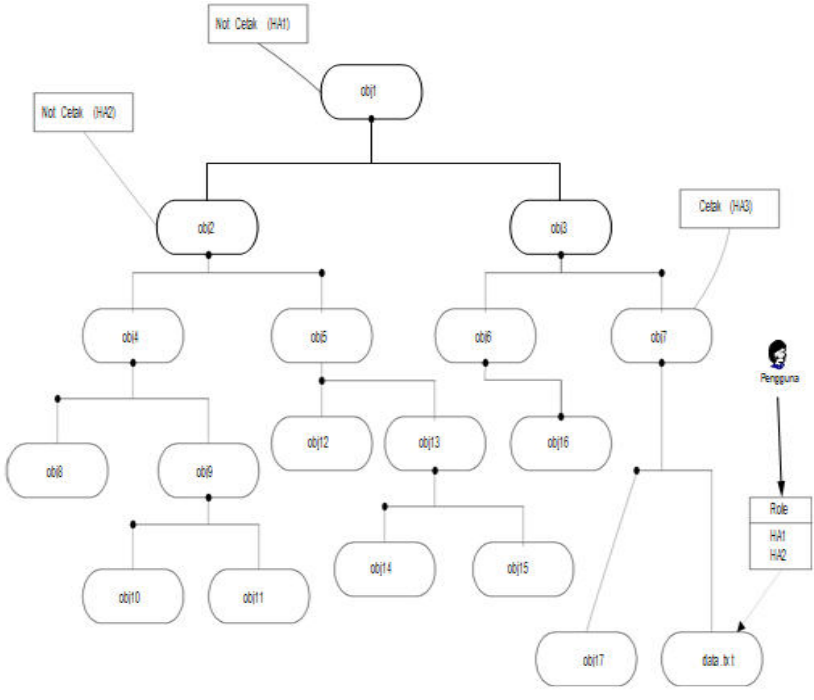
Gambar 3.15 pengguna melakukan permintaan hak akses HA2 untuk mengakses berkas data.txt, akses HA2 adalah tulis. Pengguna melakukan akses tulis data.txt harus melewati objek obj1 dan obj7, obj1 adalah hak akses HA1 dengan aksesnya Not Tulis dan objek obj7 dimiliki oleh hak akses HA3 dengan akses Not Tulis. Maka oleh sistem permintaan pengguna untuk mengakses berkas data.txt ditolak dikarenakan pengguna harus mengakses objek obj1 dan obj7 yang aksesnya tidak boleh melakukan akses tulis.



Gambar 3.16 Pohon Direktori dalam Kegiatan Tulis

3.11 Hak Akses Negatif dalam Kegiatan Cetak

Di gambar 3.17 pengguna melakukan permintaan hak akses HA3 untuk mengakses berkas data.txt, akses HA3 adalah cetak. Pengguna melakukan akses cetak data.txt, permintaan akses tersebut pengguna harus melewati objek obj1, objek obj1 dimiliki oleh hak akses HA1 dengan akses Not Cetak. Oleh sistem permintaan pengguna untuk akses tulis berkas data.txt ditolak.

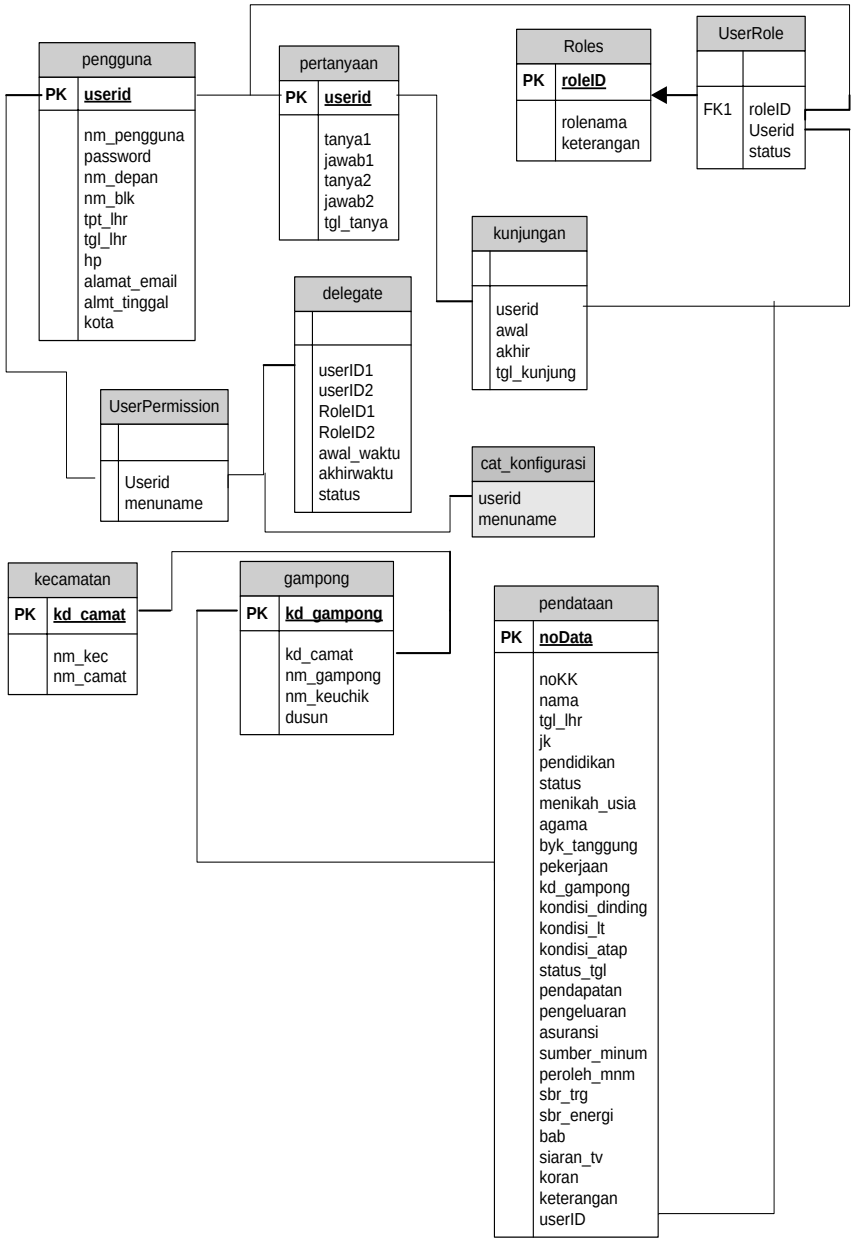


Gambar 3.17 Pohon Direktori dalam Kegiatan Cetak

3.12 Kamus Data

Dalam sistem yang akan peneliti rancang, semua informasi data kemiskinan akan disimpan dengan aman dan hanya dapat diakses oleh pihak yang berwenang ataupun ada data-data yang dapat dipublikasikan dan pihak-pihak yang membutuhkan data tersebut dapat mengaksesnya. Desain database yang baik redundasi

data harus seminimal mungkin, sehingga memberikan kemudahan dalam melakukan pemeliharaan data.



Gambar 3.18 Kamus Data

-⌂-

BAB 4

HASIL DAN PEMBAHASAN

4.1 Hasil

Dalam bab ini dibahas hasil uji coba, hal ini dilakukan untuk mengetahui cara kerja dan pengembangan sistem yang didapatkan dengan menggunakan *dynamic separation of duty* (DSD). Apakah dengan menggunakan *mutually exclusive roles* (MER) maupun dengan menggunakan *mutually exclusive permissions* (MEP) yang dilakukan secara *positive permissions* maupun *negative permissions*.

4.1.1 Pengaturan Hak Akses

Permintaan pengguna untuk mengaktifkan hak akses dari *role*, sistem akan memeriksa terlebih dahulu apakah hak akses yang diminta sudah ada dan telah dideklarasikan sebagai MEP. Jika sistem menemukan hak akses telah diaktifkan maka sistem akan menolak hak akses pengguna dan jika hak akses belum digunakan sebagai MEP, sistem mengaktifkan hak akses pengguna tersebut. Pada gambar 3.13 terlihat objek yang berbeda tersimpan dalam direktori berbeda, dengan mempunyai tiga kegiatan ditugaskan ke direktori berbeda dan menciptakan tiga buah hak akses yaitu HA1, HA2 dan HA3. Setelah membuat ketiga hak akses, administrator mendeklarasikan hak akses HA1, HA2 dan HA3 sebagai *Mutually Exclusive Permissions*.

Kegiatan hak akses dapat diperbaharui setiap saat setelah kegiatan membuat serta mengaktifkan perubahan dalam mengakses sumber daya yang diminta oleh pengguna. Sistem akan menyimpan riwayat hak akses yang telah diaktifkan oleh pengguna. Sistem melakukan verifikasi pengaktifan hak akses yang telah dilakukan oleh pengguna. Riwayat hak akses akan disimpan dalam jangka waktu yang telah ditentukan atau pada saat kegiatan tersebut dilakukan. Setelah kegiatan selesai dilakukan, riwayat terhapus dalam database sehingga sistem tidak menyimpan riwayat kegiatan tersebut selamanya.

Pada gambar 3.14 objek tersimpan dalam struktur pohon direktori, sehingga kegiatan tersebut akan mempengaruhi direktori lain yang tersebar beserta objeknya. Pengguna mempunyai *role* terdiri dari hak akses HA1, HA2 dan HA3 berkeinginan untuk melakukan kegiatan "baca" ke berkas "data.txt". Permintaan untuk melakukan kegiatan tersebut objek yang diinginkan di bawah direktori "obj7".

Sistem menerima hak akses yang diminta pengguna dan melakukan pemeriksaan apakah terjadinya hambatan hak akses yang ditentukan saat melakukan permintaan hak akses "baca". Sistem melakukan pemeriksaan pada daftar yang ada dan ditemukan bahwa "baca" memiliki dua hambatan yang saling bertentangan yaitu "tulis" dan "cetak" seperti dalam tabel 1.

Tabel 2 Daftar Kegiatan Hak Akses

No	Kegiatan	Hambatan
1	Cetak	Koreksi
2	Tulis	Baca
3	Tulis	Koreksi
4	Baca	Cetak

4.1.2 Penugasan Hak Akses

Administrator memberikan kegiatan “baca”, “tulis” dan “cetak” pada semua direktori dan membuat hak akses kemudian mendeklarasikan hak akses tersebut sebagai *Mutually Exclusive Permissions*. Dengan demikian hasil yang dibutuhkan dapat diperoleh tetapi hal ini membutuhkan waktu dan rawan terjadinya kesalahan.

Penggunaan cara yang lain untuk mendapatkan hasil yang sama adalah dengan cara membuat beberapa hak akses dan *role*, kemudian *role* tersebut ditetapkan ke pengguna. Administrator telah membuat tiga buah hak akses yaitu HA1, HA2 dan HA3 ke *role* yang baru yaitu “R1”. Kegiatan “baca” ditetapkan ke direktori “obj7 ” dan “obj2 ” yang mengakibatkan terciptanya hak akses baru yaitu HA4 dan HA5. Kemudian memecahkan *role* yang berkaitan denga hak akses, diakibatkan oleh *Mutually Exclusive* pengguna akan mengaktifkan hak akses HA4 dari *role* R2 dan hak akses HA5 dengan *role* baru R3.

Pengguna dapat mengaktifkan salah satu hak akses tersebut dari dua rangkaian hak akses yang digunakan oleh pengguna sebagai “S1” atau “S2”. Pada rangkaian hak akses S1 terdapat dua rangkaian hak akses yaitu “tulis (obj2)” dan “baca (obj2), pada rangkaian hak akses S2 juga mempunyai dua hak akses yaitu “cetak (obj7)” dan “baca (obj7)”.

Hambatan hak akses dengan menggunakan *mutually exclusive permissions* (MEP) hak akses HA4 dan HA5 berasal dari normalisasi *role* yaitu normalisasi R2 dan normalisasi R3. Pengguna berwenang untuk mengaktifkan semua *role* yaitu R1, R2, dan R3 dan juga berhak mengaktifkan *role* R1 atau nomalisasi R2 dan R3 yang disebabkan oleh *mutually exclusive roles* (MER) dalam mengimplementasikan DSD.

Perhitungan rangkaian ranah otoritas yang tersedia dalam hak akses dalam mengimplementasikan DSD. Kedua hak akses dipisahkan seperti yang terlihat di table 2 di bawah ini. Hak akses negatif memiliki tindakan negatif ditetapkan pada suatu objek yang menggantikan hak akses positif dan memiliki kegiatan positif sesuai dengan ketetapan pada objek yang sama sebagai hak akses negatif. Hak akses positif mempunyai hak akses negatif kemudian akan dibatalkan serta dihapus dari ranah pengguna RBAC.

Hak akses dalam kegiatan "baca"

HA1 = Baca (obj1)

HA2 = Tulis (obj2)

HA3 = Cetak (obj7)

HA4 = Baca (obj2)

HA5 = Baca (obj7)

Roles

R1 = {HA1, HA2, HA3}

R2 = {HA4}

R3 = {HA5}

Pengguna

P1 = {R1, R2, R3}

Hambatan dalam Hak akses

MEP = {HA4, HA5}

MEP MER

HA4 HA5 → Normalisasi R2 Normalisasi R3

Setelah mengimplementasikan DSD:

Menentukan ranah kewenangan untuk pengguna P1= {S1, S2}

S1 = {R1, R2}

S2 = {R1, R3}

Tabel 3 Pembuatan *role* baru dan hak akses baru dalam Kegiatan Baca

S1	S2
R1 + R2	R1 + R3
HA1 + HA2 + HA3 +HA4	HA1 + HA2 + HA3 + HA5
Baca(obj1) + Tulis(obj2) + Cetak(obj7) + Baca(obj2)	Baca(obj1) + Tulis(obj2) + Cetak(obj7) + Baca(obj7)
Baca(obj1) + Tulis(obj2) + Cetak(obj7)	Baca(obj1) + Tulis(obj2) + Cetak(obj7)
R1	R1

Dari hasil perhitungan komulatif kedua kelompok hak akses di tabel 2 tersebut sesuai dengan ranah otoritas yang tersedia didapatkan kombinasi hak akses yang sama dengan hak akses pada *role* 'R1". Kemudian dilihat dari kegiatan hak akses tulis di bawah ini.

Hak akses dalam kegiatan "tulis"
HA1 = Baca (obj1)
HA2 = Tulis (obj2)
HA3 = Cetak (obj7)
HA4 = Tulis (obj1)
HA5 = Tulis (obj7)

Roles
R1 = {HA1, HA2, HA3}
R2 = {HA4}
R3 = {HA5}

Pengguna
P1 = {R1, R2, R3}

Hambatan dalam Hak akses
MEP = {HA4, HA5}
MEP MER
HA4 ⟷ HA5 ⟶ Normalisasi R2 ⟷ Normalisasi R3

Setelah mengimplementasikan DSD:

Menentukan ranah kewenangan untuk pengguna

P1 = {S1, S2}

S1 = {R1, R2}

S2 = {R1, R3}

Tabel 4 Pembuatan *role* baru dan hak akses baru dalam Kegiatan Tulis

S1	S2
R1 + R2	R1 + R3
HA1 + HA2 + HA3 +HA4	HA1 + HA2 + HA3 + HA5
Baca(obj1) + Tulis(obj2) + Cetak(obj7) + Tulis(obj1)	Baca(obj1) + Tulis(obj2) + Cetak(obj7) + Tulis(obj7)
Baca(obj1) + Tulis(obj2) + Cetak(obj7)	Baca(obj1) + Tulis(obj2) + Cetak(obj7)
R1	R1

Hak akses dalam kegiatan “cetak”

HA1 = Baca (obj1)

HA2 = Tulis (obj2)

HA3 = Cetak (obj7)

HA4 = Cetak (obj1)

HA5 = Cetak (obj2)

Roles

R1 = {HA1, HA2, HA3}

R2 = {HA4}

R3 = {HA5}

Pengguna

P1 = {R1, R2, R3}

Hambatan dalam Hak akses

MEP = {HA4, HA5}

Setelah mengimplementasikan DSD:

Menentukan ranah kewenangan untuk pengguna

P1 = {S1, S2}

S1 = {R1, R2}

S2 = {R1, R3}

Tabel 5 Pembuatan *role* baru dan hak akses baru dalam Kegiatan Cetak

S1	S2
R1 + R2	R1 + R3
HA1 + HA2 + HA3 +HA4	HA1 + HA2 + HA3 + HA5
Baca(obj1) + Tulis(obj2) + Cetak(obj7) + Cetak(obj1)	Baca(obj1) + Tulis(obj2) + Cetak(obj7) + Tulis(obj2)
Baca(obj1) + Tulis(obj2) + Cetak(obj7)	Baca(obj1) + Tulis(obj2) + Cetak(obj7)
R1	R1

4.1.3 Implementasi dengan menggunakan *Negative* MEP

Pada metode selanjutnya untuk mendapatkan hasil dengan menggunakan hak akses negatif yang diimplementasikan pada *mutually exclusive permissions* (MEP). Di sini digunakan tiga kegiatan berbeda yang ditetapkan pada objek berbeda dan digunakan untuk membuat hak akses yang berbeda. Setiap hak akses telah ditetapkan untuk *role* yang berbeda. Hak akses HA1 ditetapkan ke *role* "R1", hak akses HA2 ditetapkan ke *role* "R2", dan hak akses HA3 ditetapkan juga ke *role* "R3". Hak akses HA2 dan HA3 dideklarasikan sebagai *mutually exclusive permissions*.

Setelah menggunakan pendekatan ini, pengguna dapat mengaktifkan salah satu dari dua rangkaian hak akses yang tersedia

dan dapat diakulasikan dengan menggunakan S1 dan S2. Pengguna dapat membaca semua direktori terkecuali pengguna hanya dapat membaca direktori yang dimiliki oleh salah satu hak akses yang diinginkan sesuai dengan gambar 3.9.

Target dari administrator untuk mengimplementasikan kebijakan tersebut. Kebijakan akan diimplementasikan pada dua tahap yang dilakukan oleh administrator. Salah satu tahap terkait dengan penetapan hak akses pada *role* dan kemudian pengguna dapat menggunakan tahap lainnya untuk mendeklarasikan hak akses sebagai *mutually exclusive permissions*.

Hak akses dalam kegiatan "baca"
HA1 = Baca (obj1)
HA2 =Not Baca (obj2)
HA3 = Not Baca (obj7)

Roles
R1 = {HA1}
R2 = {HA2}
R3 = {HA3}

Pengguna
P1 = {R1, R2, R3}

Hambatan dalam Hak akses
MEP = {HA2, HA3}

Hambatan dalam hak akses akibat dalam menciptakan *mutually exclusive permissions*. Hak akses HA2 dan HA3 berasal dari normalisasi *role*, masing-masing berasal dari "normalisasi R2" dan "normalisasi R3". Hal penting yang harus diingat bahwa kedua hak akses merupakan hak akses negatif berguna diterapkan untuk membatasi ranah kewenangan pengguna dengan cara yang mudah. Pengguna berwenang untuk mengaktifkan semua *role* yaitu R1, R2 dan R3 dan dapat mengaktifkan *role* tersebut kapan saja tanpa

batasan apapun. *Mutually exclusive* pada hak akses negatif akan diimplementasikan dengan cara pandang hak akses positif. Pengguna dapat mengaktifkan *role* baik "normalisasi R2" atau "normalisasi R3", hal ini disebabkan *mutually exclusive roles* dalam mengimplementasikan DSD. Di bawah ini ada dua *mutually exclusive permissions* menggunakan dua normalisasi *roles* sebagai *mutually exclusive roles.*

Setelah mengimplementasikan DSD:

Tersedia serangkaian ranah otoritas pengguna

P1= {S1, S2}

S1 = {R1, R2}

S2 = {R1, R3}

Melakukan kalkulasi rangkaian di ranah otoritas dari sisi hak akses dalam mengimplementasikan DSD. Kedua rangkaian tersebut dipecahkan hak aksesnya untuk ranah otoritas yang dapat dilihat di bawah ini. *Negative mutually exclusive permissions* diimplementasikan dari sisi *positive mutually exclusive permissions.*

P1= {R1} + {(R2) MER (R3)}

P1= {HA1} + {(HA2) MER (HA3)}

P1= {HA1} + {Not (HA2) + (HA3) OR (HA2) + Not (HA3)}

P1= {HA1 + Not (HA2) + HA3} OR {HA1 + HA2 + Not (HA3)}

P1= {Baca(obj1) + Not(Not Baca(obj2)) + Not Baca(obj7)} OR {Baca(obj1) + Not Baca(obj2) + Not(Not Baca(obj7))}

P1= {Baca(obj1) + Baca(obj2) + Not Baca (obj7)} OR {Baca(obj1) + Not Baca(obj2) + Baca(obj7)}

Setelah melakukan perhitungan pada kedua pasang rangkaian hak akses kumulatif dari ranah otoritas, maka didapatkan dua kombinasi hak akses yang berbeda. Dari perhitungan rangkaian di atas didapatkan hak akses positif ketika kegiatan negatif dilakukan

dengan menggunakan hak akses negatif. Di bawah ini akan melakukan kalkulasi dua rangkaian ranah otoritas dengan kegiatan "tulis", lihat gambar 3.16.

Hak akses dalam kegiatan "tulis"
HA1 = Not Tulis (obj1)
HA2 = Tulis (obj2)
HA3 = Not Tulis (obj7)

Roles:
R1 = {HA1}
R2 = {HA2}
R3 = {HA3}

Pengguna:
P1 = {R1, R2, R3}

Hambatan dalam Hak akses :
MEP = {HA1, HA3}

Setelah mengimplementasikan DSD:
Tersedia serangkaian ranah otoritas pengguna
P1= {S1, S2}
S1= {R2,R1}
S2 = {R2, R3}
P1= {R2} + {(R1) MER (R3)}
P1= {HA2} + {(HA1) MER (HA3)}
P1= {HA2} + {Not (HA1) + (HA3) OR (HA1) + Not (HA3)}
P1= {HA2 + Not (HA1) + HA3} OR {HA2 + HA1 + Not (HA3)}
P1= {Tulis(obj2) + Not(Not Tulis(obj1)) + Not Tulis(obj7)} OR {Tulis(obj2) + Not Tulis(obj1) + Not(Not Tulis(obj7))}

P1= {Tulis(obj2) + Tulis(obj1) + Not Tulis(obj7)} OR {Tulis(obj2) + Not Tulis(obj1) + Not Tulis(obj7)}

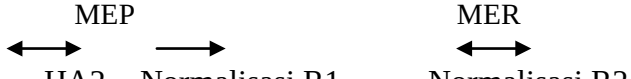
Dari perhitungan rangkaian di atas didapatkan hak akses positif ketika kegiatan negatif dilakukan dengan menggunakan hak akses negatif. Di bawah ini akan melakukan kalkulasi dua rangkaian ranah otoritas dengan kegiatan "cetak", lihat gambar 3.17.

Hak akses dalam kegiatan "cetak"
HA1 = Not Cetak (obj1)
HA2 = Not Cetak (obj2)
HA3 = Cetak (obj7)

Roles:
R1 = {HA1}
R2 = {HA2}
R3 = {HA3}

Pengguna:
P1 = {R1, R2, R3}

Hambatan dalam Hak akses :
MEP = {HA1, HA2}



Setelah mengimplementasikan DSD:
Tersedia serangkaian ranah otoritas pengguna
P1 = {S1, S2}
S1 = {R3, R1}
S2 = {R3,R2}
P1= {R3} + {(R1) MER (R2)}
P1= {HA3} + {(HA1) MER (HA2)}
P1= {HA3} + {Not (HA1) + (HA2) OR (HA1) + Not (HA2)}

P1= {HA3 + Not (HA1) + HA2} OR {HA3 + HA1 + Not (HA2)}
P1= {Cetak(obj7) + Not(Not Cetak(obj1)) + Not Cetak(obj2)} OR {Cetak(obj7) + Not Cetak(obj1) + Not(Not Cetak (obj2))}
P1= {Cetak(obj7) + Cetak(obj1) + Not Cetak(obj2)} OR {Cetak(obj7) + Not Cetak(obj1) + Cetak(obj2)}

4.2 Pembahasan

Dari hasil perhitungan kalkulasi yang telah dibuat, hak akses positif dan hak akses negatif sama-sama memiliki potensi dalam penyelesaikan suatu permasalahan dalam melakukan akses. Tetapi mempunyai dampak yang berbeda di mana yang telah dibahas sebelumnya bahwa hak akses positif memberikan wewenang ke pengguna dan hak akses negatif membatasi wewenang ke pengguna. Hak akses negatif tidak akan memberikan manfaat apapun tanpa menggunakan hak akses positif. Hak akses negatif sangat baik diterapkan jika saat hak akses positif diterapkan juga pada objek yang disimpan dalam struktur pohon direktori. Sehingga objek akan selalu terhubung satu sama lainnya melalui beberapa hirarki.

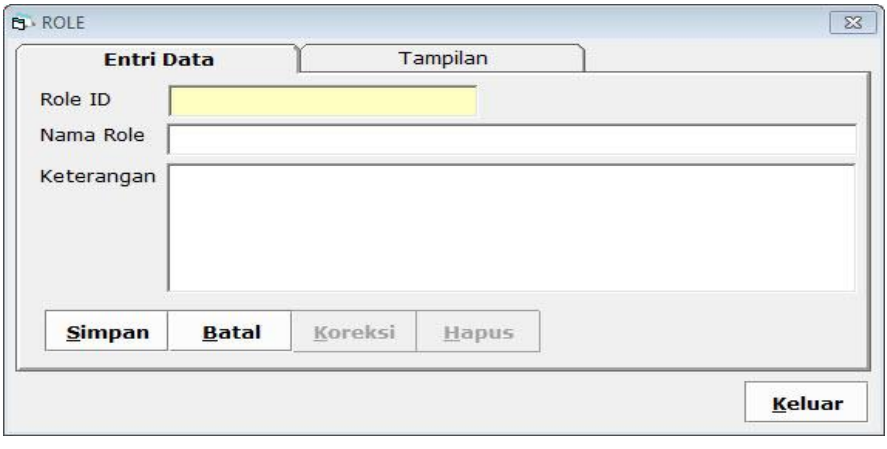
Penetapan hak akses negatif bagi role jika role tersebut belum ditetapkan hak akses positifnya pada struktur pohon direktori. Penetapan hak akses negatif dan positif bagi role akan sangat berguna bagi administrator, sehingga pengguna dapat dibatasi kewenangannya dengan cara sangat mudah.

4.3 Implementasi dalam Pendataan Masyarakat Miskin

4.3.1 *Role* dalam Sistem Pendataan Masyarakat Miskin

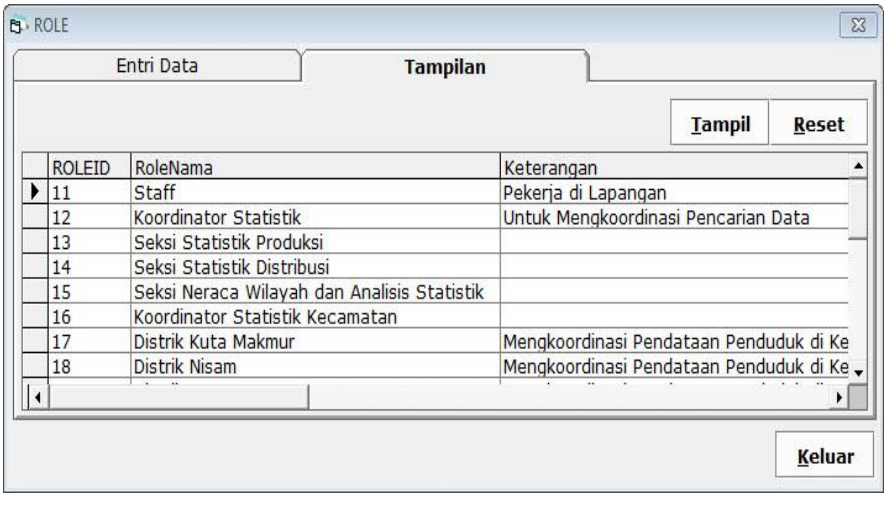
Role dalam sistem pendataan masyarakat miskin adalah pengelompokan tugas yang akan dilakukan oleh administrator. Pengguna mendapatkan role sesuai dengan *role* yang diberikan oleh

administrator. Administrator akan membuat *role* sesuai dengan kelompok manajemen yang telah ditetapkan oleh organisasi tersebut. Tugas *role* dapat diperoleh sesuai dengan wewenang yang diberikan oleh administrator kepada pengguna. Pada Gambar 4.1 adalah *form* untuk menambah maupun memodifikasi *role* baru.



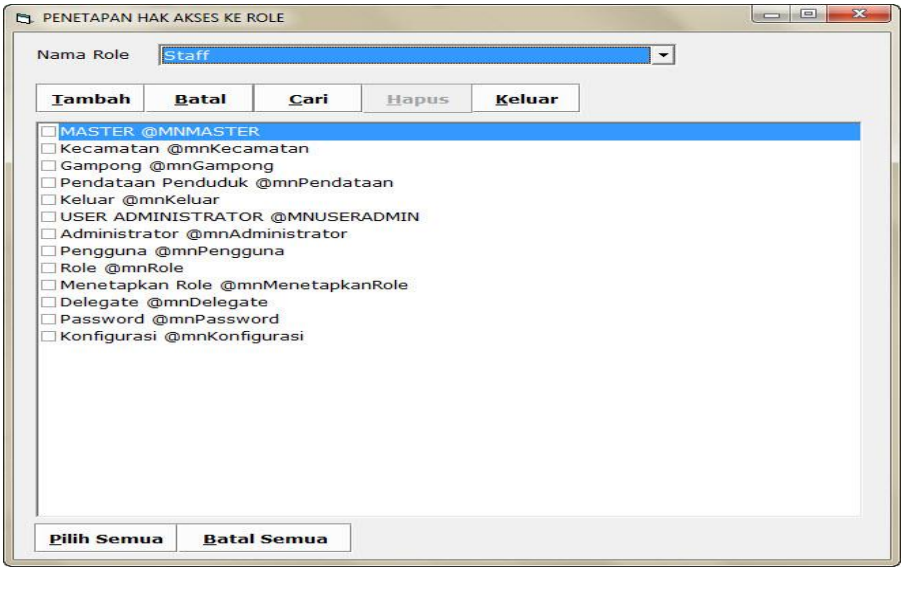
Gambar 4.1 GUI *Role*

Administrator memasukkan RoleID sesuai dengan kode yang diterapkan pada organisasi, kemudian administrator memasukkan nama *role*, dan keterangan *role* yang menerangkan *role* tersebut beserta wewenang yang boleh diterapkan pada *role* tersebut. *Role* bagi administrator merupakan inti dari sistem yang akan diterapkan dalam RBAC, *role* sangat penting pada sistem pendataan penduduk miskin di Kabupaten Aceh Utara. *Role* adalah peran atau wewenang yang didapatkan oleh pengguna. *Role* administrator mempunyai peran untuk melakukan pengontrolan sistem, menentukan hak akses yang didapatkan oleh *role*, kemudian memberikan *role* ke pengguna lain beserta hak apa yang dapat diakses oleh pengguna. Di bawah ini gambar *role* yang telah dimasukkan oleh administrator.



Gambar 4.2 *GUI* Tampilan *Role*

Administrator menentukan hak akses bagi *role*. Pemberian hak akses oleh administrator ke *role* dengan cara memilih menu apa saja yang boleh diakses oleh *role* tersebut.



Gambar 4.3 *GUI* Menetapkan Hak Akses bagi *Role*

Permission (hak akses) adalah persetujuan untuk melakukan suatu operasi pada satu objek. Objek dapat berupa sebuah program yang dieksekusi. Tipe operasi dan kontrol pada RBAC tergantung pada sistem dimana operasi itu dimplemtasikan. Pada sistem ini kontrol diberikan adalah berupa *form* yang dapat diakses oleh pengguna, dan pengguna yang mendapatkan akses tersebut telah mendapatkan *role* yang telah ditentukan oleh administrator. Manajemen *role* yang diberikan oleh administrator kepada pengguna sesuai dengan tanggung jawab dan wewenang yang didapatkan oleh pengguna. Hak akses ditentukan oleh administrator dan *role* akan mendapatkan hak akses sesuai dengan ketentuan yang diberikan oleh administrator, pengguna akan mendapatkan *role* tersebut sesuai dengan wewenang yang akan diberikan oleh administrator. Administrator dapat menghapus *role* maupun mengkoreksi *role* tersebut sesuai dengan struktur maupun tugas pada suatu organisasi tersebut.

Role berorientasi pada group, sekumpulan transaksi dibuat berbentuk objek berupa program dan berhubungan dengan data. Seorang administrator dapat menambah dan menghapus transaksi ke dalam sebuah *role* atau menolak pengguna pada suatu *role*. Di bawah ini adalah *source code* untuk melakukan menyimpan *role*.

```
Private Sub Simpan()
    roles1 = TampilKodeCombo(" select * from roles where
             rolenama='" & CboRole & " '" , Db)
    Db.ExecQuery (" insert into userrole(userID,roleID,status)
             values ('" & CboUserID & " ','" & roles1 & " ','"
             & CboStatus & " ')" )
    CmdBatal_Click
End Sub

Private Sub CmdSimpan_Click()
    Simpan
```

```
Db.ExecQuery (" insert into kunjungan(userid,awal,akhir,
    tgl_kunjung,keterangan) values ('" & UserID & " ','" &
    Waktu & " '," & _
    " '" & Time & " ','" & Format(Date, " yyyy/mm/dd" ) & "
    ','Menyimpan Data Role User'" & " )" )
End Sub
```

Di bawah ini adalah *source code* untuk melakukan penyimpanan hak akses di *role* dalam database.

```
Private Sub TambahData()
    On Error GoTo ErrTambah
    Dim u As Integer, Mname As String
    role1 = TampilKodeCombo(" select RoleID from Roles where
            roleNama='" & CboRole & " '" , Db)
    For u = 0 To List1.ListCount - 1
    If List1.Selected(u) = True Then
    Mname = List1.List(u) If Mname <> "" Then
    Db.ExecQuery (" insert into roleha(roleID,menuname) values
        ('" & role1 & " ','" & MenuNama(List1.List(u)) & " ')" )
    End If
    End If
    Next u
    MsgBox " KONFIGURASI HAK AKSES DI ROLE TELAH
        TERSIMPAN...!" , vbInformation, " INFORMASI"
    Exit Sub
    ErrTambah:
    MsgBox Err.Description & "" & CStr(Err.Number)
End Sub
```

Pada *source program* di atas untuk melakukan penambahan data hak akses ke *role*, administrator akan menentukan hak akses yang diberikan ke *role*, jika tidak terjadi kesalahan maka sistem akan mengeluarkan peringatan bahwa hak akses telah tersimpan jika tidak

sistem juga akan mengeluarkan peringatan bahwa kesalahan telah terjadi.

Hak dan izin diberikan pada *role* bukan pada pengguna. Pengguna memerlukan hak dan izin secara virtual dengan jalan memasukkan pengguna tersebut me njadi anggota dari r*ole* yang bersangkutan. *Role* berorientasi pada group, atau sekumpulan transaksi yang dibuat. Transakasi disini dapat merupakan obyek yang berupa program yang berhubungan dengan data. Seorang administrator dapat menambah dan menghapus transaksi ke dalam sebuah *role* atau bahkan menolak pengguna pada suatu *role*. Dengan mengelompokkan pengguna kedalam *role* maka ada memudahkan pada proses otorisasi dan kemampuan dalam melakukan pengamanan. Hal ini bertolak belakangan dengan *access list model* pada umumnya yang dilakukan dengan jalan mencari seluruh otorisasi yang ada kemudian mengalokasi hak dan izin untuk pengguna tersebut.

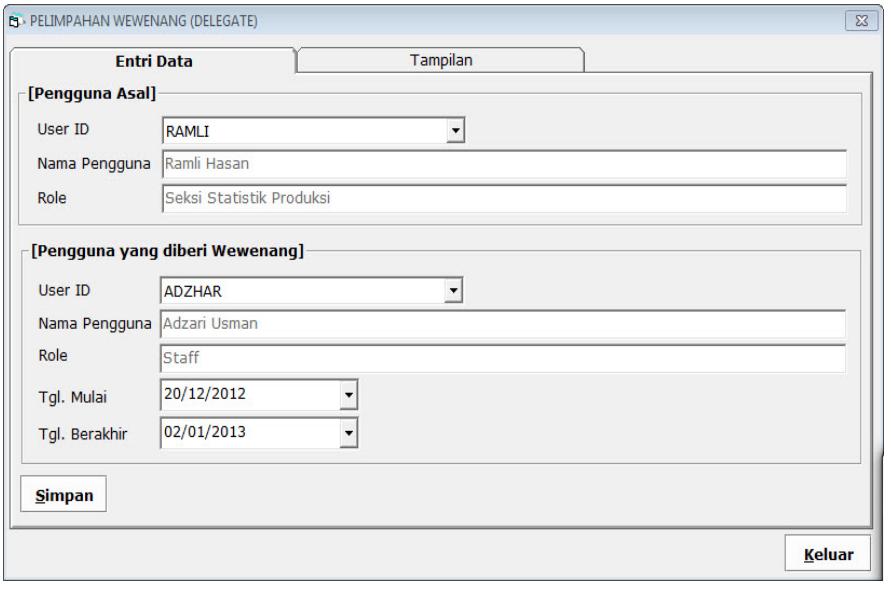
RBAC adalah sebuah metode yang cocok untuk digunakan dan dapat mengurangi komplesitas administrasi dan mengurangi biaya yang dibutuhkan dalam menjaga keamanan sebuah sistem. Pada dasarnya RBAC memberikan pengguna keanggotaan pada *role* berdasarkan kompetensi dan tanggung jawan masing-masing pengguna. Pada RBAC pengguna tidak dapat melakukan operasi atas inisiatif sendiri melainkan berdasarkan *role* yang telah diperoleh dari administrator.

4.3.2 Pelimpahan Wewenang

Pengguna akan melakukan pelimpahan wewenang ke pengguna yang lain, maka pengguna tersebut tidak dapat masuk ke dalam sistem sampai berakhirnya waktu pelimpahan wewenang yang diberikan tersebut berakhir. Jika waktu pelimpahan telah berakhir pengguna yang memberikan wewenang tersebut dapat

memasuki sistem dan pengguna yang menerima pelimpahan wewenang akan kembali mendapatkan wewenangnya kembali seperti semula.

Administrator tidak dapat membatalkan wewenang yang telah diberikan oleh pengguna ke pengguna yang lain, dan pengguna yang mendapatkan wewenang tersebut mempunyai hak akses ke dalam sistem seperti hak akses yang didapatkan oleh pengguna yang memberikan wewenang tersebut.



Gambar 4.4 G*UI* Pelimpahan Wewenang

Dibawah ini adalah gambar *source code* proses pelimpahan wewenang yang pertama adalah melakukan penyimpanan data ke *table delegate* sesuai dengan *field* userid1 adalah pengguna yang memberikan pelimpahan wewenang, dan *role* yang didapatkan oleh pengguna yang melakukan pelimpahan wewenang. Userid2 dan role2 yang berfungsi sebagai penerima pelimpahan wewenang, tanggal awal menerima pelimpahan wewenang dan tanggal berakhir pelimpahan yang diberikan. Kemudian menghapus *record* di *table* sementara yang berfungsi untuk menampung *record* hak akses yang diberikan oleh administrator kepada userid2. Kemudian melakukan

pencarian data apakah userid2 ada, jika ada lakukan penyimpanan sementara seluruh hak akses userid2 di *table* cat_konfigurasi. Setelah itu lakukan kembali penghapusan *record* userid2 di *table* userpermission, yang terakhir adalah koreksi *record* pengguna yang menerima pelimpahan wewenang dengan *record* pengguna yang memberikan pelimpahan wewenang di *table* userpermission.

```
Private Sub Cmdsimpan_Click()
  Simpan
End Sub
```

```
Private Sub Simpan()
  role1 = Mumang("select roleid from roles where rolenama='" & FTxtRole1 & "'", Db)
  role2 = Mumang("select roleid from roles where rolenama='" & FTxtRole2 & "'", Db)
  Db.ExecQuery ("insert into delegate(userid1,userid2,roleid1,roleid2,awal_waktu,akhirwaktu,status) values " & _
    "('" & CboUser1 & "','" & CboUser2 & "','" & role1 & "','" & role2 & "','" & Format(DTPicker1, "yyyy/mm/dd") & "'," & _
    "'" & Format(DTPicker2, "yyyy/mm/dd") & "','A'" & ")")
  Db.ExecQuery ("delete from cat_konfigurasi where userid='" & CboUser1 & "'")
  'Db.ExecQuery ("insert into cat_konfigurasi(userid) values ('" & CboUser1 & "')")
  CariMenu
  Db.ExecQuery ("delete from userpermission where userid='" & CboUser2 & "'")
  Db.ExecQuery ("update userpermission set userid='" & CboUser2 & "' where userid='" & CboUser1 & "'")
End Sub
```

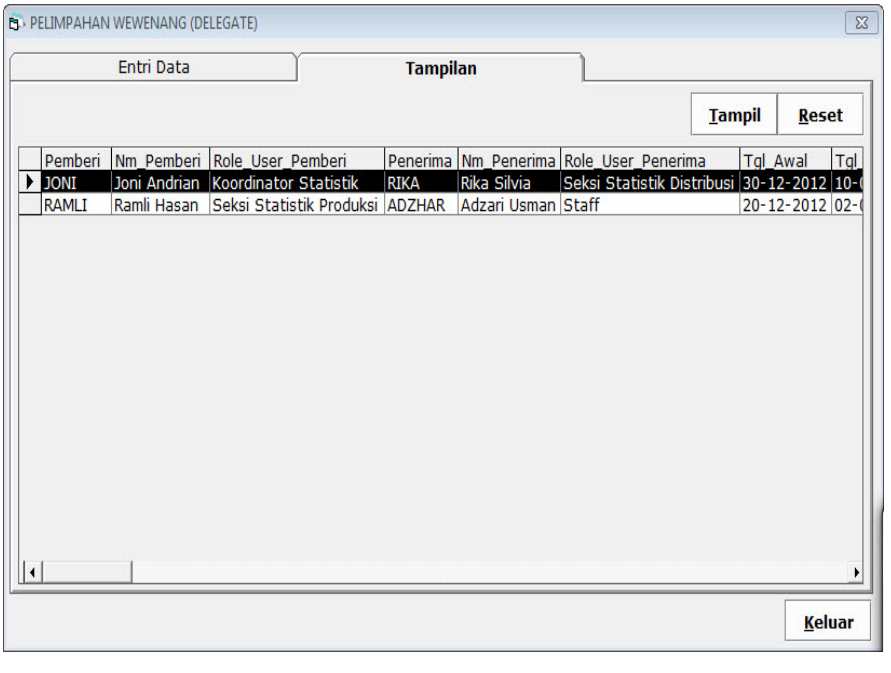
```
Private Sub CariMenu()
  Dim Sql$, Rs As ADODB.Recordset, u%
  Sql = "select * from userpermission where userid='" & CboUser2 & "'"
  Set Rs = Db.SelectRecord(Sql, 2, 2)
  If Not Rs.EOF Then
    For u = 0 To Rs.RecordCount - 1
      Db.ExecQuery ("insert into cat_konfigurasi(userid,menuname) values ('" & IIf(IsNull(Rs("userid")), "", Rs("userid")) & "'," & _
        "'" & IIf(IsNull(Rs("menuname")), "", Rs("menuname")) & "')")
    Rs.MoveNext
    Next u
  End If
  Rs.Close
End Sub
```

Gambar 4.5 Source code Proses Pelimpahan Wewenang

Gambar 4.6 dibawah ini adalah tampilan pengguna yang memberi wewenang, *role* pengguna yang memberi wewenang dan pengguna yang menerima wewenang, *role* pengguna yang menerima wewenang beserta tanggal awal penerimaan wewenang dan tanggal berakhirnya wewenang tersebut diberikan. Jika wewenang tersebut

telah berakhir maka secara otomatis sistem akan memberikan wewenang itu kembali ke pengguna yang telah melimpahkan wewenang dan pengguna yang menerima wewenang tersebut akan mendapatkan hak aksesnya kembali sesuai wewenang yang diberikan oleh administrator.

Tanggal berakhirnya wewenang akan disesuaikan dengan penanggalan sistem komputer, jika tanggal berakhirnya wewenang tersebut lebih besar dari tanggal sistem komputer maka lakukan proses pengembalian wewenang. Proses pengembalian wewenang tersebut dilakukan sewaktu sistem ini dijalankan.



Gambar 4.6 *GUI* Tampilan Pelimpahan Wewenang

Ada dua buah *table* yang akan direlasikan yaitu *table* pengguna dan *table roles*. Gambar dibawah ini menunjukkan kode sumber dalam melakukan tampilan pelimpahan wewenang.

```
Private Sub CmdTampil_Click()
  Dim Sql$, Rs As ADODB.Recordset
  Sql = "select a.userID1 as Pemberi,b.nm_pengguna as Nm_Pemberi,d.rolenama as Role_User_Pemberi," & _
        "a.userID2 as Penerima,c.nm_pengguna as Nm_Penerima,e.rolenama as Role_User_Penerima," & _
        "convert(char(10),a.awal_waktu,105) as Tgl_Awal,convert(char(10),a.akhirwaktu,105) as Tgl_Berakhir," & _
        "case when a.status='A' then 'Aktif' else 'Tidak Aktif' end as Status " & _
        " from (((delegate a inner join pengguna b on a.userid1=b.userid) inner join pengguna c on a.userid2=c.userid) " & _
        " inner join roles d on a.roleid1=d.roleid) inner join roles e on a.roleid2=e.roleid where a.status='A'"
  Set Rs = Db.SelectRecord(Sql, 2, 2)
  With DataGrid1
    Set .DataSource = Rs
    .MarqueeStyle = dbgHighlightRowRaiseCell
    .Columns(0).Width = 1300
    .Columns(1).Width = 3000
    .Columns(2).Width = 3500
    .Columns(3).Width = 1300
    .Columns(4).Width = 3000
    .Columns(5).Width = 3500
    .Columns(6).Width = 1500
    .Columns(7).Width = 1500
    .Columns(8).Width = 2000
    .Refresh
  End With
End Sub
```

Gambar 4.7 *Source code* Tampilan Pelimpahan Wewenang

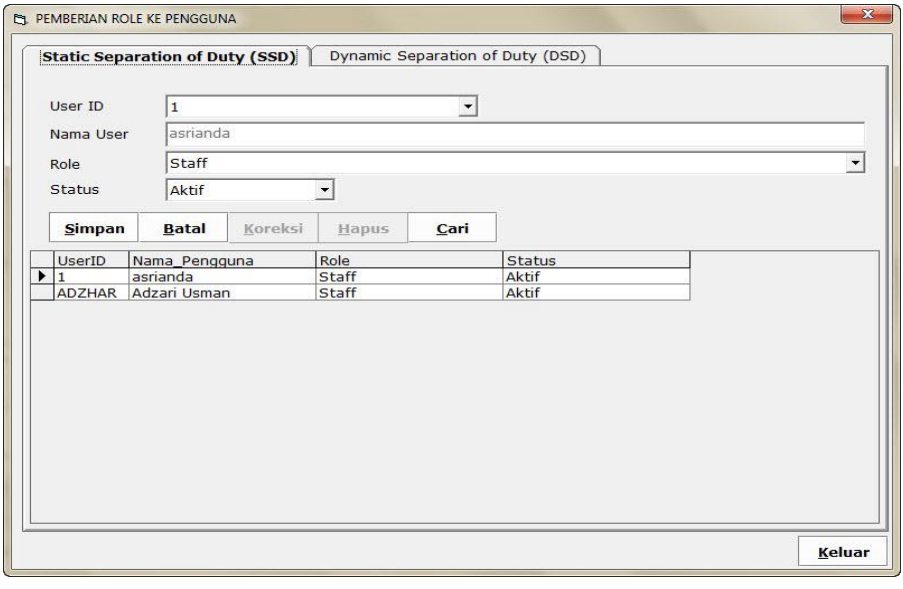
4.4 *Authentication - Access Control Mechanism*

Access Control Mechanism di mulai dengan identifikasi dan otentikasi. Identifikasi adalah pada tahap ini pengguna akan memberitahukan siapa dirinya. Otentikasi adalah tindakan untuk memverifikasi klaim identitas pengguna tersebut yaitu sesuatu yang mereka ketahui misalnya *password*, nomor induk karyawan atau sidik jari dan lain-lain. Proses otentikasi berfungsi sebagai kesempatan pengguna untuk menerima dan melakukan proses pengaksesan *resource*. Pihak pengguna harus mampu untuk memberikan informasi yang dibutuhkan oleh pemberi layanan dan berhak untuk mendapatkan *resource*nya. Sedangkan pihak yang memberi layanan harus mampu menjamin bahwa pihak yang tidak berhak tidak dapat mengakses *resource* ini.

Pengaksesan data yang dilindungi harus dibatasi untuk orang yang berwenang dalam mengakses data tersebut. Otorisasi akan menentukan sumber daya informasi yang diizinkan untuk diakses dan tindakan apa saja yang akan diizinkan dalam melakukan proses ke sistem tersebut yang dapat dilakukan oleh pengguna. Otorisasi untuk mengakses informasi di dalam sistem di mulai dengan kebijakan administrasi yang dibuat oleh organisasi tersebut. Kebijakan akan menentukan data apa yang dapat diakses, oleh siapa dan dalam kondisi apa. Dalam tesis ini otorisasi *role* didefinisikan oleh organisasi pendataan penduduk miskin. Administrator yang berada dalam *role* administrator dapat menetapkan r*ole* kepada pengguna yang akan menjalankan sistem tersebut.

4.4.1 *Static Separation of Duty* (SSD)

Static Separation of Duty (SSD) adalah pengguna yang sama tidak diperbolehkan untuk mendapatkan *role* yang berbeda, sehingga pengguna hanya memiliki satu *role* dalam tugas yang diberikan kepadanya. Dengan kata lain SSD digunakan untuk menjaga tidak adanya *role* ganda yang dapat dimiliki oleh seorang pengguna sehingga terjadinya manipulasi dapat dihindari. *Static separation* ditentukan dengan melakukan penugasan oleh pengguna dalam *role* dan mealokasikan transaksasi dengan *role*.



Gambar 4.8 *GUI Static Separation of Duty* (SSD)

Pada Gambar 4.8 administrator memberikan otorisasi *role* staff ke pengguna dan pengguna tidak mendapatkan *role* lainnya. Oleh sebab itu SSD dikatakan hanya satu pengguna dapat memiliki satu *role* saja, di mana administrator hanya dapat mengaktifkan satu pengguna untuk mendapat satu *role* dan pengguna tersebut akan ditolak oleh sistem jika administrator mengaktifkan pengguna tersebut di *role* yang lainnya. Pada SSD pengguna tidak dapat ditugaskan sebagai *mutually exclusive roles*, serta SSD berkemampuan untuk mengatasi potensi konflik kepentingan pada saat penggunakan ditugaskan sebagai *role*.

	UserID	nm_pengguna	Role	Hak_Akses
1	1	asrianda	Staff	mnGampong
2	1	asrianda	Staff	mnKecamatan
3	1	asrianda	Staff	mnKeluar
4	1	asrianda	Staff	MnMaster
5	1	asrianda	Staff	mnPendataan
6	ADZHAR	Adzari Usman	Staff	mnGampong
7	ADZHAR	Adzari Usman	Staff	mnKecamatan
8	ADZHAR	Adzari Usman	Staff	mnKeluar
9	ADZHAR	Adzari Usman	Staff	MnMaster
10	ADZHAR	Adzari Usman	Staff	mnPendataan

Gambar 4.9 Data Pengguna SSD

Pada Gambar 4.9 pengguna asrianda mendapatkan *role* sebagai staff dan mempunyai lima buah hak akses, dan pengguna tersebut tidak diperbolehkan mendapatkan *role* yang lain karena SSD memperbolehkan satu pengguna untuk mendapatkan satu *role*. Dalam penelitian ini *role* diperbolehkan untuk memiliki hak akses yang sama dengan *role* yang lainnya, hal ini disebabkan dalam standarisasi RBAC di SSD pengguna hanya boleh memiliki satu *role*. Jika pendataan serta keputusan untuk menetapkan siapa yang dianggap kotegori miskin dan berhak untuk mendapatkan bantuan, maka pekerjaan yang dilakukan akan sangat lambat dan membutuhkan waktu lama, hal ini disebabkan keputusan yang diberikan akan membutuhkan banyak pihak.

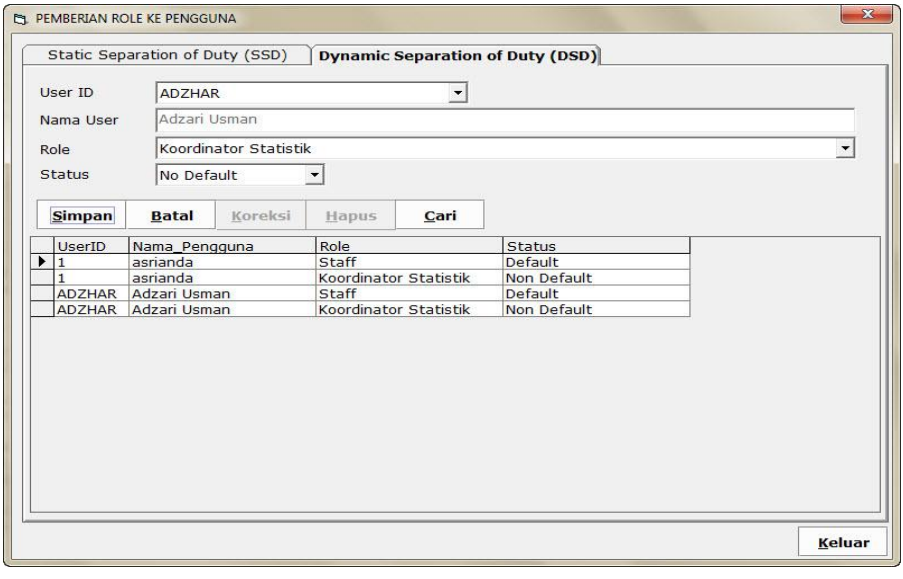
4.4.2 *Dinamic Separation of Duty* (DSD)

DSD disesuaikan dengan persyaratan yang ditentukan oleh sistem, dan bertujuan untuk lebih fleksibel dalam melakukan suatu operasi. DSD memungkinkan pengguna yang sama mendapatkan tugas dan role yang berbeda, sehingga memberikan fleksibilitas dalam bisnis dan membuat rentan terjadinya manipulasi.

DSD dimungkinkan pengguna dapat ditugaskan sebagai *mutually exclusive roles*, tetapi *role* tidak dapat diaktifkan dalam waktu yang sama. Di DSD memungkinkan pengguna diberi otorisasi dalam *role* sehingga memungkinkan tidak terjadinya konflik kepentingan ketika pengguna bertindak secara mandiri, tetapi terjadi permasalahan sewaktu kebijakan yang didapatkan diaktifkan secara bersamaan. Umumnya DSD memberikan efesiensi yang lebih tinggi dan fleksibilitas dalam operasionalnya.

Contoh pengguna A sebagai *role* koordinator statistik. Koordinator statistik diperbolehkan untuk melihat, maupun mengkoreksi data seluruh distrik yang ada, dan pengguna A tersebut sebagai *role* distrik Sawang, jika pengguna merubah data di *role* distrik Nisam sebagai data miliknya maka hal ini akan terjadi masalah yang besar, karena dapat terjadi konflik kepentingan di antara *role* tersebut. Tetapi standard DSD pengguna A harus keluar dari sistem dulu baru pengguna tersebut dapat mengaktifkan *role* yang lainnya.

Sifat DSD menyediakan dukungan istimewa karena setiap pengguna memiliki tingkat hak akses yang berbeda tergantung dari *role* yang dilakukan. Kemampuan DSD adalah untuk mengatasi isu-isu konflik yang terjadi sewaktu pengguna diberikan *role*, hal ini disebabkan pada SSD konflik akan terjadi pada *role* karena satu *role* hanya boleh dimiliki oleh satu pengguna. DSD memungkinkan pengguna diberi otorisasi dua *role* atau lebih yang tidak menciptakan konflik kepentingan ketika bertindak secara independen, namun mempunyai masalah dalam kebijakan jika *role* tersebut diaktifkan secara bersamaan.



Gambar 4.10 *GUI Dynamic Separation of Duty* (DSD)

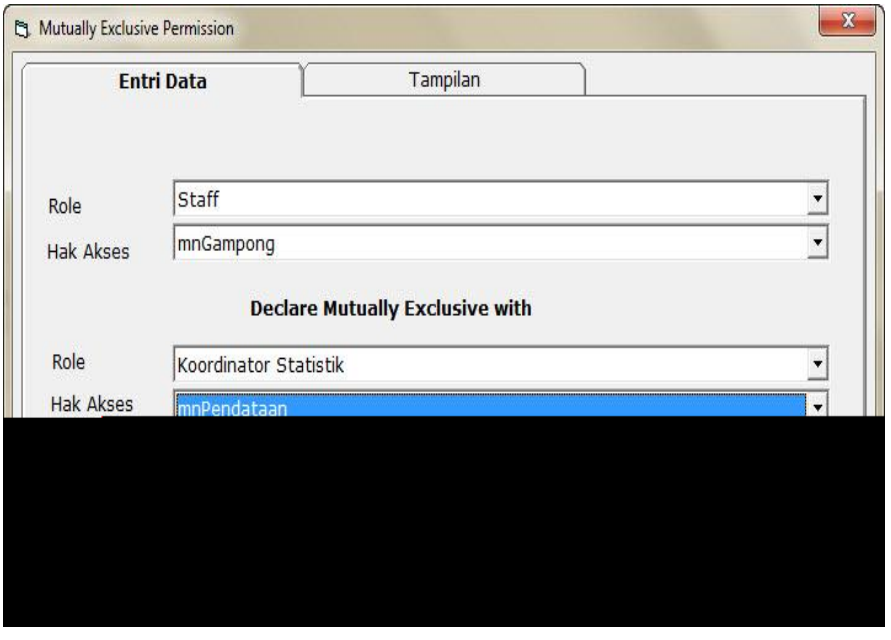
Pada Gambar 4.10 administrator memberikan otorisasi *role* staff ke pangguna dengan statusnya sebagai *default* kemudian pengguna tersebut akan diberikan *role* koordinator statistik dengan statusnya *non default* sampai seterusnya. Dalam permasalahan ini pengguna tersebut tidak boleh mendapatkan *role* yang sama atau *role* berbeda dengan status *default* yang ganda hal dilakukan sewaktu menerapkan *mutually exclusive permissions* akan mengalami masalah sewaktu memberikan otorisasi dan hak akses ke pengguna. Seorang pengguna akan memasukkan data penerima bantuan, dan akan diblokir dalam melakukan pembayaran bagi penerima bantuan, tetapi dia dapat melakukan pembayaran bantuan tersebut jika orang lain yang memasukkan data penerima bantuan.

	UserID	Role	Hak_Akses
1	ADZHAR	Koordinator Statistik	mnDelegate
2	ADZHAR	Koordinator Statistik	mnGampong
3	ADZHAR	Koordinator Statistik	mnKecamatan
4	ADZHAR	Koordinator Statistik	mnKeluar
5	ADZHAR	Koordinator Statistik	mnKonfigurasi
6	ADZHAR	Koordinator Statistik	MNMASTER
7	ADZHAR	Koordinator Statistik	mnMenetapkanRole
8	ADZHAR	Koordinator Statistik	mnPassword
9	ADZHAR	Koordinator Statistik	mnPendataan
10	ADZHAR	Koordinator Statistik	MNUSERADMIN
11	ADZHAR	Staff	mnGampong
12	ADZHAR	Staff	mnKecamatan
13	ADZHAR	Staff	mnKeluar
14	ADZHAR	Staff	MNMASTER
15	ADZHAR	Staff	mnPendataan
16	ADZHAR	Staff	mnPengguna
17	ADZHAR	Staff	mnRole
18	ADZHAR	Staff	MNUSERADMIN

Gambar 4.11 Data Pengguna DSD

Pada Gambar 4.11 pengguna adzhar mendapatkan *role* sebagai staff dan *role* koordinator statistik dengan hak akses sesuai ketentuan yang telah ditetapkan oleh administrator. Dengan didapatkannya dua *role* yang berbeda mudahnya terjadi manipulasi dalam sistem, pengguna jika menggunakan *role* yang lainnya maka pengguna tersebut harus keluar dahulu dan mengaktifkan *role* yang lainnya.

Administrator keamanan tidak perlu secara khusus menentukan kedua hambatan dalam hak akses yang memiliki *role* yang sama seperti *mutually exclusive permissions* karena akan dilakukan secara otomatis. Hal ini menjadi keberuntungan bagi administrator karena dengan cara ini administrator tidak lagi melakukannya secara manual.



Gambar 4.12 GUI *Mutually Exclusive Permission*

Dalam Gambar 4.12 menyatakan dua hak akses sebagai *mutually exclusive* dalam hak akses. Di antara *role* yang ada dapat ditentukan hak akses apa yang tidak boleh diakses oleh pengguna, jika pengguna telah mendapatkan ke dua *role* tersebut. Hak akses yang didapatkan menjadi menjadi *mutually exclusive permissions* secara otomatis.

Hal ini memberikan kemudahan bagi administrator dari pada mendeklarasikan hambatan dalam hak akses secara manual seperti *mutually exclusive permissions* yang dibuat secara satu-persatu.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dalam bab ini yang berisikan kesimpulan dan saran yang diambil dari perancangan, implementasi serta hasil pengujian yang dilakukan dengan menggunakan *dinamic separation of duty* (DSD) sesuai dengan standarisasi ANSI 2004 beserta dengan DSD yang menggunakan *mutually exclusive permission*s pada *prototype* yang telah peneliti rancang adalah sebagai berikut:

1. Penggunaan DSD dengan *mutually exclusive roles* rawan terjadinya manipulasi diakibatkan tidak membatasi hak akses pengguna berbeda dengan *mutually exclusive permissions* dapat membatasi hak akses pengguna.
2. Pada DSD standar dua buah *role* tidak dapat diaktifkan oleh pengguna yang sama dalam *session* yang sama tetapi dapat diaktifkan oleh pengguna yang sama pada *session* berbeda
3. Dengan menggunakan DSD *mutually exclusive permissions* dua *role* yang berbeda tidak dapat diaktifkan oleh pengguna yang sama walaupun *session*nya berbeda.
4. DSD digunakan jika suatu pekerjaan membutuhkan waktu serta keputusan yang cepat tapi rentan terjadinya manipulasi

5. Penerapan *mutually exclusive permissions* digunakan oleh organisasi jika pengguna menggunakan banyak *role* sehingga hak akses yang ada dalam *role* tersebut dapat dibatasi sesuai hak dan wewenangnya.
6. Dalam DSD untuk mengaktifkan *role* yang lainnya pengguna harus keluar dari sistem, hal ini mengakibatkan pengguna akan membutuhkan waktu yang lama dalam memasuki sistem tersebut jika sistem tersebut banyak mengaktifkan komponen yang lainnya.

6. 2 Saran

1. Dalam sistem ini dapat dikembangkan lebih baik lagi dengan melakukan laporan dalam bentuk web yang hasil pelaporannya dapat dilihat oleh masyarakat luas
2. Penerapan *mutually exclusive permissions* dapat dikembangk an dengan cara lebih luas lagi sehingga dapat diterapkan pada aplikasi apapun dalam membatasi hak akses pengguna
3. Penelitian ini dapat dikembangkan lebih lanjut lagi di wilayah dan Kabupaten lainnya yang ada di Aceh

-⌂-

DAFTAR PUSTAKA

Abadi. M. Burrows. M & Lampson. B., P lotkin, .1993. *A Calculus for Access Control in Distributed Systems.* TOPLAS. vol. 15, no. 4, pp. 706-734

ANSI. 2004. *Role Based Access Control*. American National Standards Institute, Inc.

Bauer. L, Garriss. S & Reiter. M. K. 2005. *Distributed proving in access-control systems*. in: Proceedings of 2005 IEEE Symposium on Security and Privacy, pp. 81–95.

Belokosztolszki. A, David. M. E, Wang. W & Moody. K. 2008. *Policy Storage for Role-Based Access Control Systems.* University of Cambridge Computer Laboratory. United Kingdom

Benantar. M. 2006. *Access Control Systems: Security, Identity Management and Trust Models.* Springer Science+Business Media, Inc. 233 Spring Street. New York, NY 10013. USA

Bishop, M. 1988. *Theft of Information in the Take-Grant Protection Model, Proceedings of the Workshop on Foundations of Computer Security*. MITRE TR M88-37. Franconia. NH, pp. 194-218

Chadwick. W. D, Xu. W, Otenko. S, Laborde. R & Nasser. B. 2007. *Multi-session Separation of Duties (MSoD) for RBAC.* In Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop (ICDEW '07). IEEE Computer Society. Washington. DC. USA

Chen. F & Sandhu. R. S. 1996. *Constraints for Role-Based Access Control*. ACM RBAC Workshop. MD. USA

Chen. F, Li. S & Yang. H, 2007, *Enforcing Role -Based Access controls in Software Systems with an Agent Based Service Oriented Approach, Software Systems with an Age nt Based Service Oriented Approach*. ICNSC : 483-488

Clark. D.D & Wilson. D.R, 1987, *IEEE Symposium on Computer Security and Privacy, A Comparison of Commercial and Military Computer Security Policies.* I

Crampton. J, 2003, *Specifying and enforcing constraints in role-based access control, In P roceedings of the Eighth ACM S ymposium on Access Control Models and Technologies* (Como, Italy, June 02 - 03, 2003). S ACMAT '03. ACM, New York, NY, 43-50. DOI= http://doi.acm.org/10.1145/775412.775419.

DoD 5200.28-STD Department of Defense, December 1985, *Trusted Computer System Evaluation Criteria (Orange Book),* National Computer Securit y Center

Elliott. A. A & Knight. G. S., 2010, *Role Explosion: Acknowledging the Problem*, Proceedings of the 2010 International Conference on Software Engineering Research & Practice.

Ferraiolo. D & Kuhn. R, 1992, "*Role-based Access Control*", Proceedings of 15th NIST-NCSC National Computer Security Conference, Baltimore, MD, 13-16 October 1992, pp. 554-563.

Ferraiolo. D & Barkley. J, June 1997, *Specifying and Managing Role Based Access Control Within a Corporate Intranet*, in Proc. of the 2nd ACM workshop on Role-Based Access Control, George Mason University, Fairfax, Virginia, USA, pp. 77–82.

Ferraiolo. D. F, Sandhu. R , Gavrila. S, Kuhn. D. R & Chandramouli. R, 2001, *Proposed NIST Standard for Role-based Access Control*, ACM Transaction o n Information and System Security, vol. 4, No. 3, pp. 224-274.

Ferraiolo. D., Kuhn. D. & C handramouli R., 2003, *Role-Based Access Control*, Artech House, Computer Security Series

Gavrila. S & Barkley. J, 1998, *Formal specification for RBAC user/role and role relationship management*, In Proceedings of the Third ACM Workshop o n Role Based Access Control, 81–90.

Gligor. V.D, Gavrila. S. I & Ferraiolo. D. F, 1998, *On the formal definition of separation-of duty policies and their composition*, In Proceedings o f the Symposium on Security and Privacy, IEEE Press, Los Alamitos, Calif.

Habib. M. A, 2010, *Mutual exclusion and role inheritance affecting least privilege in RBAC, Internet Technology and Secured Transactions* (ICITST), Internationa l Conference for , vol., no., pp.1-6, 8-11 November 2010, URL : http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5678530 &isnumbe r=5678008

Habib. M. A, 2011, *Role inheritance with object-based DSD*, Int. J. Internet Technology and Secured Transactions, Vol. 3, No. 2, pp.149–160.

Habib. A.M, Praher. C, *Object based dynamic separation of duty in RBAC*, Internet Technology and Secured Transactions, (ICITST), pp.512-516, 9-12 Nov. 2009.

Hawkins. D. I, Best. R.J & Coney. K. A, 1983, *Consumer Behavior*, Business Publications, P lano, Texas

Hu. V.C, K uh. D.R, Xie. T & Hwang. J, 2011, *Model Checking For Verification Of Mandatory Access Control Models And Properties*, International Journal of Software Engineering and K nowledge Engineering Vol. 21, No. 1 (2011) 103–127

Jaeger.T, 1999, *On the increasing importance of constraints. In Proceedings of the fourth ACM workshop on Role-based access control* (RBAC '99), ACM, New York, NY, US A, 33-42, DOI=10.1145/319171.319175 http://doi.acm.org/10.1145/319171.319175

Jaeger. T & Tidswell. J. E, 2001, *Practical safety in flexible access control models*, ACM Trans. Inf. Syst. Secur. 4, 2 (May 2001), 158-190. DOI=10.1145/501963.501966 http ://do i.ac m.o rg/10.1145 /501963.501966

Jansen. W. A, 1998, *Inheritance Properties of Role Hierarchies, In Proceedings of the 21st National Information Systems Security Conference*, Gaithersburg, MD, USA, 476-485

Khayat. E.J & Abdallah. A. E, 2005, *A Formal Model for Flat Role Based Access Control*, IFIP International Federation for Information Processing Volume 173, pp 233-246

Klarl. H, K linger. K, Molitorisz. K, Abeck. S & Emig. C, 2009, *Extending Role-Based Access Control for Business Usage*, *Emerging Security Information*, Systems and Technologies, SEC URWARE '09, Third International Conference on, vol., no., pp.136-141, 18-23 June 2009 doi: 10.1109/SEC URWARE.2009.28 URL:http ://ieee xp lore. ieee.org/sta mp /sta mp.jsp?tp= &arnumb er=5211011&isnumb er=5210974

Kuhn. D. R, Coyne. E. J & Weil. T. R, June 2010, *Adding Attributes to Role-Based Access Control*, IEEE Computer Society, pp 79-81

Kunz. S, Evdokimov. S, Fabian. B, Stieger. B & Strembeck. M, 2010, *Role-Based Access Control for Information Federations in the Industrial Service Sector*, ECIS 2010 Proceedings, Paper 15

Li. N, Byun. J & Bertino. E, 2005, *A Critique of the ANSI Standard on Role-Based Access Control*, tech. report 2005-29, Center for Education and Research in Information Security and Assurance, P urdue University

Li. N & Tripunitara. M. V, 2006, *Security Analysis in Role Based Access Control*, Journal ACM Transactions on Information and S ystem Security (TISSEC) , Volume 9 Issue 4, Pages 391 – 420, ACM New York, N Y, US A

Li. N, Tripunitara. M. V & Bizri. Z, 2007, *On mutually exclusive roles and*

separation- of-duty, ACM Trans. Inf. S yst. Sec ur. 10, 2, Article 5 (May), DOI=10.1145/1237500.1237501 http://doi.acm.org/10.1145/1237500.1237501

Linares. F. J. R, 2008, Advanced *Policy-based Access Control in RDBMS*, Universidad Polit_ecnica de Madrid & Fachhochschule Hannover

Liu. Y.A & Stoller. S.D, 2004, *Role-Based Access Control: A Corrected and Simplified Specification*, Computer Science Dept., State Univ. of New York at Stony Brook, S tony Brook, NY 11794

Microsoft. C, 2005, *Microsoft Windows Access Control: A path toward Compliance*, Risk Management, and Productivity, Retrieved in December 2006 diunduh pada http//download.microsoft.com/download/d/3/f/d3f4fe20-b522-40b8-8975-eb6f3076bdfd/, tanggal 10 September 2012

Mohammed. I & Dilts. D.M, 1994, Design 275-296. *for Dynamic User-Role-Based Security*, Computer and Security, 13(8), 661-671.

Na. S. Y & Cheon. S. H , 2000, *Role delegation in role-based access control*, In Proceedings of the Fifth ACM Workshop on Role-Based Access Contro l (RBAC'00), pages 39–44

Nash. M. J & Poland. K. R, 1990, *Some Conundrums Concerning Separation of Duty*, Proc. 1990 IEEE S ymposium on Security and Privacy, 201-207, May

Perelson. S & Botha. R. A, 2000, *Conflict analysis as a means of enforcing static separation of duty requirements in workflow environments*, *South Africa n Computer Journal*, (26):212 – 216, November

Pfleeger. C. P.,1997, *Security In Computing*, Second Edition, Prentice-Hall PTR

Ray. I, Li. N, France. R & K im. D, 2004, *Using uml to visualize role - based access control constraints*, In Proceedings of the ninth ACM symposium on Access control models and technologies (SACMAT

'04). ACM, New York, NY, USA, 115-124. DOI=10.1145/990036.990054 http://doi.acm.org/10.1145/990036.990054

Rochaeli. T & Eckert. C, 2005, *RBAC Policy Engineering with Patterns, Departement of Computer Science*, Darmstadt University of Technology, Proceedings of the Semantic Web and Policy Workshop, Galway, Ireland

Roskos. J.E, Boone. J.M & Mayfield. T, October 1989, *Integrity in Tactical and Embedded Systems*. Institute for Defense Analyses, HQ 89-034883/1

Rotenberg, L.J, 1974, *Making computers keep secrets*. Ph.D. Th., MIT, MAC TR-115

Ruthberg. Z.G & Polk. W.T, 1989, *Editors. Report of the Invitational Workshop on Data Integrity*, SP 500-168, Natl. Inst. of S tds. and Technology

Saltzer, J & Schroeder. M, 1975, *The protection of information in computer systems*. Proc. IEEE 63, 1278–1308.

Samarati. P & Vimercati. S. D. C. D, 2011, *Access Control: Policies, Models, and Mechanisms*, Springer Science Business Media, LLC, Springer US

Sandhu. R, 1988, *Transaction control expressions for separation of duties*, In Proceedings of 4th Aerospace Computer Security Conference (Orlando, Florida), pp. 282–286.

Sandhu. R, 1990, *Separation of Duties in Computerized Information Systems*, Proc. IFIP WG11.3 Workshop on Database Security, September.

Sandhu. R & Munawer. Q, 1999, *The ARBAC99 Model for Administration of Roles*. In Proceedings of the 15th Annual Computer Security Applications Conference, ACSAC. IEEE Computer Society, Washington, DC, 229.

Sandhu. R, Bhamidipati. V & Munawer. Q., *The ARBAC97 model for role-based administration of roles*, 1999, ACM Transactions on Information and Syste m Security (TISSEC), vol. 2, no. 1, pp. 105 – 135

Sandhu. R, Ferraiolo. D & Kuhn. R, July 2000, *The NIST model for role-based access control: Towards a unified standard*, In Proceedings of the 5th ACM Workshop on Role-Based Access Control, pages 47-63, Berlin, Germany

Sandhu, R., Ferraiolo. D & K uhn. R, 2004. "*American National Standard for Information Technology – Role Based Access Control*", ANSI INCITS 359-2004

Sandhu. R & Bhamidipati. V, 2008, *The ASCAA Principles for Next-Generation Role - Based Access Control*, Proc. 3rd International Conference on Availability, Reliability and Security (ARES), Barcelona, Spain, March 4-7, pp xxvii- xxxii. Presentation Keynote Lecture.

Schaad. A, Spadone. P & Weichsel. H, 2005, *A case study of separation of duty properties in the context of the Austrian " eLaw" process*, In Proceedings of the 2005 ACM Symposium on Applied Computing (Santa Fe, New Mexico, March 13 - 17, 2005). L. M. Liebrock, Ed. SAC '05. ACM, New York, NY,1328-1332. DOI= http://doi.acm.org/10.1145/1066677.1066976.

Schweitzer, J. A. 1996. *Protecting Business Information: a manager's guide*. Butterworth-Heinemann. Boston

Strembeck. M, 2004, *Conflict Checking of Separation of Duty Constraints in RBAC - Implementation Experiences*, In Proc. of the Confere nce on Software Engineering (SE),

Simon. R & Zurko. M,1997, *Separation of duty in role-based environments, In Proceedings of 10th IEEE Computer Security Foundations Workshop*, Rockport, Massachusetts, pp. 183–194.

Spivey. J. M, The Z Notation: *A Reference Manual*, Prentice-Hall, 2nd edition, 1992

Zhu. H, *Some Issues Of Role-Based Collaboration*, 2003, Electrical and Computer Engineering, IEEE CCECE 2003, Canadian Conference

-⌂-

RIWAYAT HIDUP

Identitas Pribadi

Nama Lengkap : Asrianda, S. Kom, M. Kom
Tempat/Tgl. Lahir : Lhokseumawe, 15 Mei 1973
Pendidikan : Magister Teknik Informatika
Pekerjaan : Dosen Fakultas Teknik Informatika Universitas Malikussaleh
Suku/ Bangsa : Aceh / Indonesia
Agama : Islam
Alamat : Jl. Kemuning No 8 Keude Aceh Lhokseumawe 24315

Jenjang Pendidikan

1. SD Negeri 10 Lhokseumawe tamat tahun 1985.
2. SMP Negeri 1 Lhokseumawe tamat tahun 1988.
3. SMA Negeri Lhokseumawe tamat tahun 1991.
4. Fakultas NGT Teknik Komputer Universitas Sisingamangaraja XII tamat tahun 1994.
5. STMIK-AMIK Riau Jurusan Teknik Informatika tamat tahun 2005
6. Magister Teknik Informatika Universitas Sumatera Utara tamat tahun 2013.

Karya Tulis Ilmiah 5 Tahun Terakhir

1. Kontrol Akses dalam Keamanan Data Pendataan Penduduk Miskin. Volume 6, Nomor 1, Mei 2013, ISSN 1979-0236, Jurnal Samudera
2. Spesifikasi dan Pengaturan Kontrol Akses dalam Pendataan Masyarakat Miskin di Kabupaten Aceh Utara. Volume 6,

Nomor 2, November 2013, ISSN 1979-0236, Jurnal Samudera

Buku Yang Pernah Ditulis 5 Tahun Terakhir

1. Buku Pemograman Visual Basic: Teori dan Implementasi, Unimal Press ISBN 978-602-1373-00-2. 2013. Unimal Press
2. Pemograman Database, Unimal Press ISBN 978-602-1373-01-9. 2012. Unimal Press

Lhokseumawe, 15 Juni 2016
Hormat Saya,

Asrianda, S. Kom, M. Kom.

-⌂-