

Tugas EL-695 Keamanan Sistem Informasi

TOPIK : KEAMANAN BASIS DATA

**JUDUL : ARSITEKTUR DAN PROTOTIPE KEAMANAN
DATABASE MULTILEVEL**

Nama : Dedi Wardhana

NIM : 232 01 065

Program Magister Teknik Sistem Komputer

Jurusan Teknik Elektro

Fakultas Teknologi Industri

Institut Teknologi Bandung

ARSITEKTUR DAN PROTOTYPE KEAMANAN

DATABASE MULTILEVEL

Pendahuluan

Database multilevel merupakan sistem yang kompleks. Dalam database multilevel terdapat relasi-relasi. Relasi-relasi ini mengikuti aturan-aturan tertentu. Multilevel yang melekat pada database disini menunjukkan bahwa database memiliki level-level yang membedakan satu obyek database dengan obyek database lainnya. Level-level ini diperlukan untuk menentukan subyek yang boleh mengaksesnya.

Untuk menjamin akses database multilevel oleh subyek-subyek yang berhak diperlukan mekanisme keamanan tertentu. Banyak penelitian telah dilakukan dan menghasilkan arsitektur-arsitektur dan prototipe-prototipe keamanan database multilevel yang unik.

Arsitektur Keamanan Database Multilevel

Arsitektur keamanan database multilevel dapat dibagi ke dalam dua jenis utama. Jenis pertama adalah arsitektur yang menggunakan *trusted computing base (TCB)* eksternal untuk mengendalikan akses obyek database. Jenis ini disebut juga sebagai arsitektur *kernelized, Hinke-Schaefer*, atau *TCB subset DBMS (Database Management System)*. Arsitektur ini berbeda dari arsitektur-arsitektur yang mendelegasikan *mandatory access control (MAC)* kepada sistem manajemen database internal. Jenis kedua ini disebut juga sebagai arsitektur *trusted subject DBMS*.

Setiap database memiliki sekumpulan aturan sensitivitas data yang mengatur relasi antar data. Dalam pendekatan *Hinke-Schaefer* relasi ini didekomposisikan ke dalam fragmen-fragmen *single-level* atau *system-high*. Keamanan sistem manajemen database multilevel (*Multilevel Secure Database Management System* atau *MLS DBMS*) menyimpan fragmen-fragmen ini secara fisik ke dalam obyek *single-level* (sebagai contohnya, file-file, segmen-segmen, atau perangkat-perangkat keras yang terpisah). *MLS DBMS* memaksakan *mandatory access control (MAC)* pada setiap permintaan untuk mengakses obyek *single-level* atau *system-high* ini.

Pendekatan yang kedua menggunakan *trusted network* untuk pemisahan perijinan selain mengandalkan pada sistem operasi multilevel. Variasi ini juga mendekomposisikan database multilevel ke dalam fragmen-fragmen *system-high*. Tetapi dalam kasus ini *DBMS* mereplikasi data tingkat rendah dibawah fragmen-fragmen yang lebih tinggi tingkatannya. Pada jaringan multilevel *MLS DBMS* memisahkan data secara fisik dengan mendistribusikannya ke host sistem *DMBS* yang lainnya. Prototipe *Unisys Secure Distributed DBMS (SD-DBMS)* menggunakan pendekatan ini dan digunakan dalam proyek riset *NRL Trusted DBMS (TDBMS)*.

Pendekatan *TCB subset DBMS*

Arsitektur ini pertama kali didokumentasikan oleh Thomas Hinke dan Marvin Schaefer di System Development Corporation. *DBMS* ini dirancang untuk sistem operasi *Multics* dengan tujuan agar sistem operasi tersebut menyediakan semua kendali akses. Rancangan ini mendekomposisikan database multilevel ke dalam beberapa atribut dan kolom *single-level* dengan atribut-atribut yang memiliki sensitivitas yang sama tersimpan

bersama pada segmen-segmen sistem operasi *single-level*. Sebagai contohnya, untuk memenuhi permintaan *request*, proses *DBMS* diselenggarakan pada level *user* yang mengoperasikannya. Karena adanya aturan *mandatory access control (MAC)* dari sistem operasi, *DBMS* hanya memiliki akses yang sama levelnya atau dibawahnya. Kemudian *DBMS* menggabungkan elemen-elemen dari relasi yang sama untuk merekonstruksi *tuple* yang dikembalikan ke *user*.

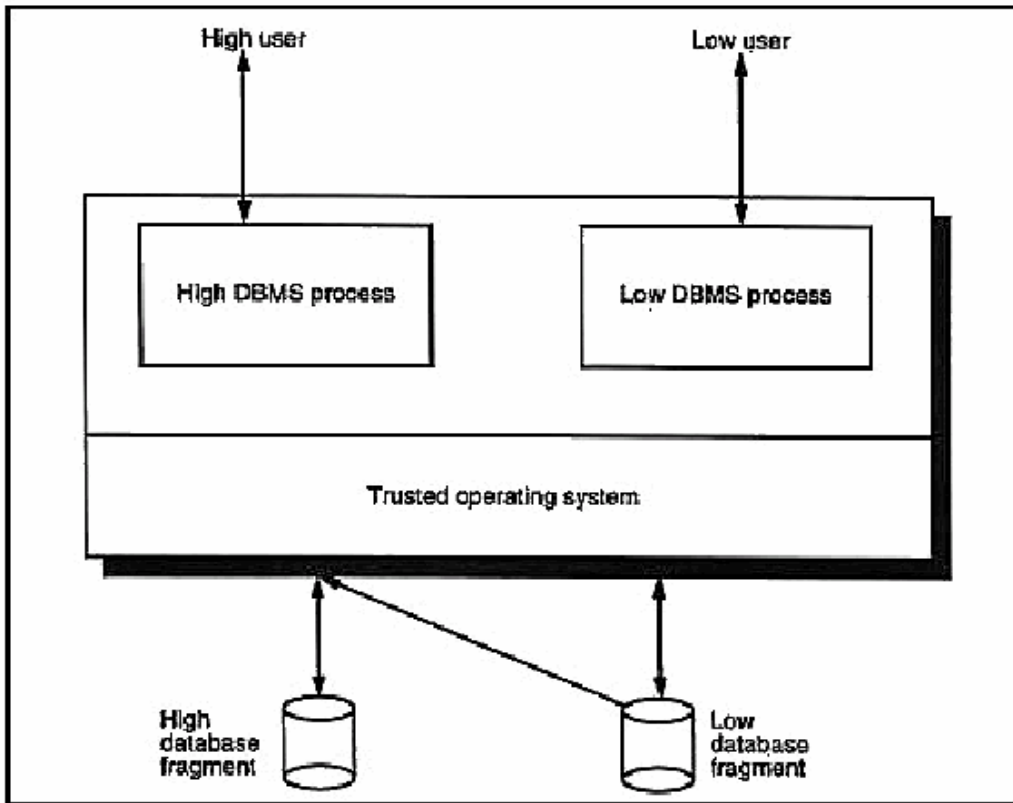
Pendekatan *Hinke-Schaefer* memiliki dua karakteristik utama, yaitu:

1. *DBMS* multilevel sebenarnya merupakan sekumpulan *DBMS single-level* yang bekerja secara bersamaan
2. Database multilevel dapat didekomposisikan ke dalam sekumpulan database *single-level* atau *system-high*, dan masing-masing merupakan bagian dari database multilevel secara konseptual

Ada dua variasi dari arsitektur ini: tersentralisasi, dan terdistribusi. Pada pendekatan tersentralisasi tiap-tiap *DMBS single-level* adalah proses-proses terpisah yang berjalan pada suatu *trusted operating system*, dan database multilevel didekomposisikan ke dalam fragmen-fragmen *single-level* yang masing-masing disimpan di dalam obyek sistem operasi *single-level* (sebagai contohnya, file-file atau segmen-segmen). Sementara *DBMS* memungkinkan untuk dipercaya melakukan beberapa fungsi kendali akses, *trusted operating system* dapat memaksakan aturan kendali akses secara penuh kepada semua akses yang dilakukan *DBMS* terhadap obyek-obyek *DBMS*. Gambar 1 mengilustrasikan pendekatan ini.

Pada arsitektur ini *user* tidak beroperasi dalam mode multilevel tetapi pada level sesi yang terselenggara dengan *trusted operating system*. Setiap *user* berinteraksi dengan

DBMS pada tingkat sesi *user*, dan banyak *DBMS* yang berlainan berjalan pada tingkat-tingkat sensitivitas yang berlainan pula boleh beroperasi pada saat yang bersamaan.



Gambar 1. Arsitektur *TCB subset*

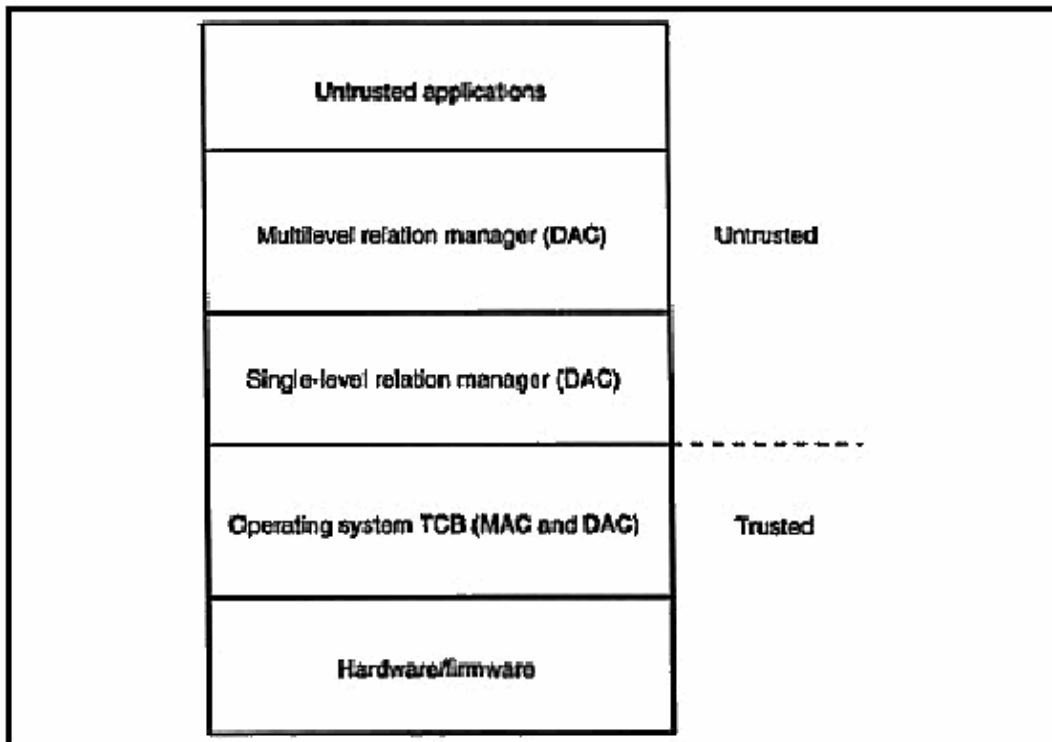
Ada dua prototipe yang dikembangkan menggunakan konsep *Hinke-Schaefer*, yaitu *SeaView DBMS* dan *LDV DBMS*.

Secure Distributed Data Views (SeaView) DBMS

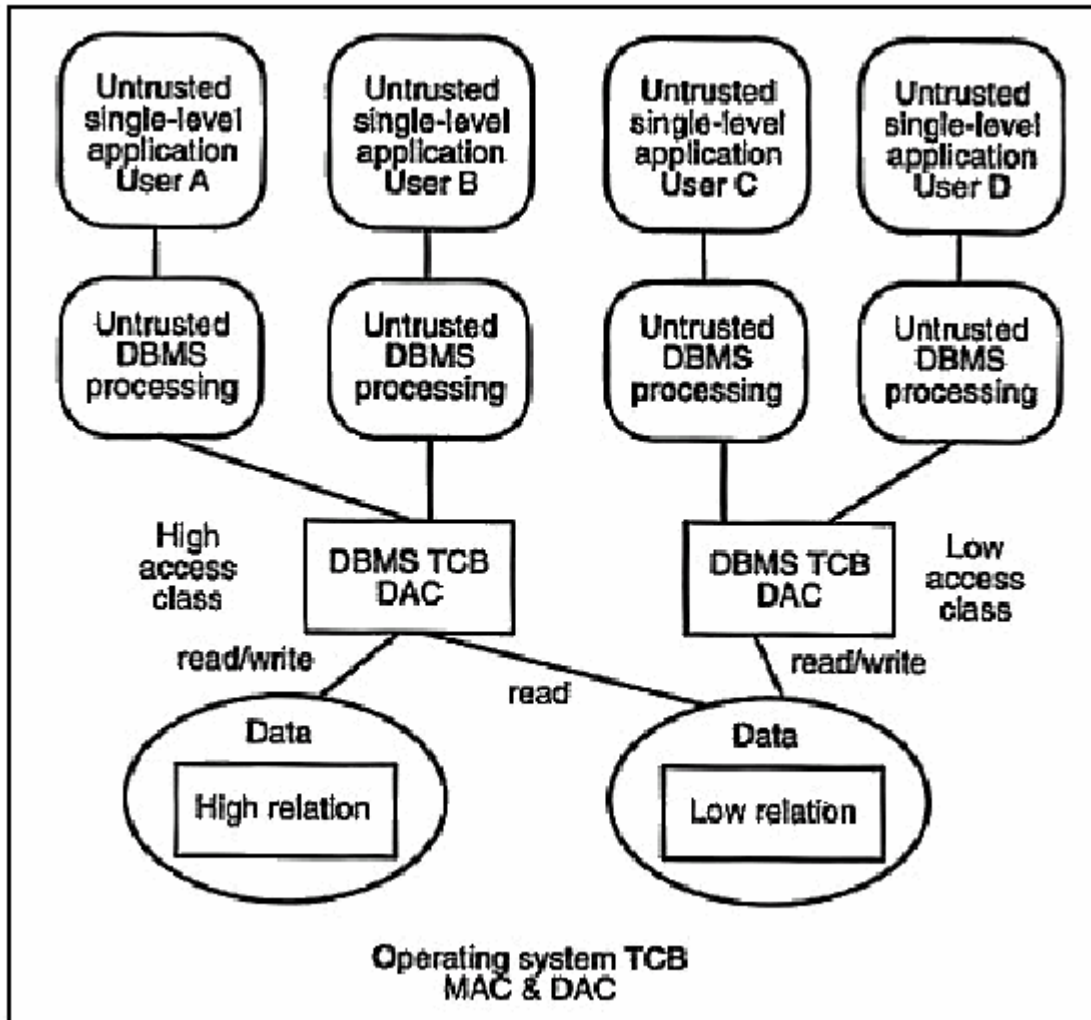
Dalam pendekatan *SeaView* sebuah relasi multilevel didekomposisikan ke dalam relasi-relasi *single-level* yang didasarkan pada penamaan tingkat elemen (*elemen-level labelling*). Setiap *tuple* (catatan) didekomposisikan dan disimpan ke dalam fragmen-fragmen *single-level* tertentu. Fragmen-fragmen dengan jenis relasi dan level yang sama

dimasukkan ke dalam segmen sistem operasi yang sama. Jika ada *request user*, *DBMS* menggabungkan fragmen-fragmen *single-level* pada level yang sama atau yang dibawah level sesi *user* dan mengembalikan *tuple* sesuai dengan kriteria yang diinginkan *user*. Karena setiap *user* berinteraksi dengan proses *DBMS single-level*, *DBMS* tidak mengetahui data-data yang berada di atas levelnya.

Arsitektur *SeaView* didasarkan pada satu pendekatan yang disebut sebagai *TCB subsets*. Pendekatan ini secara hirarkis membuat lapisan-lapisan komponen software. Oracle adalah salah satu contoh database yang menggunakan pendekatan ini. Gambar 2 dan 3 memperlihatkan lapisan-lapisan pada arsitektur *SeaView*.



Gambar 2. Arsitektur *SeaView*



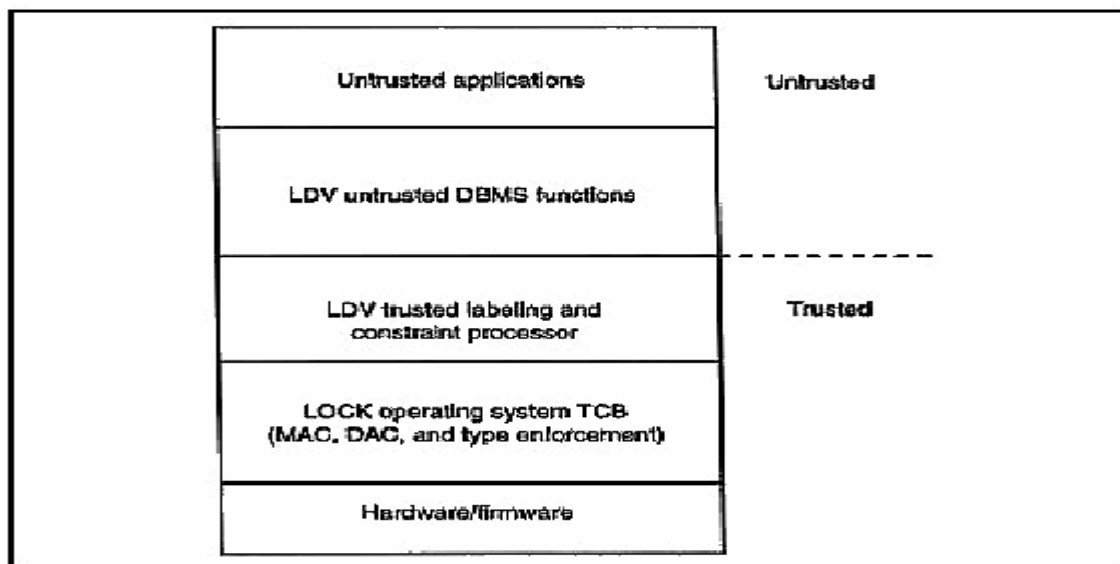
Gambar 3. Arsitektur *SeaView*

Lock Data Views (LDV) DBMS

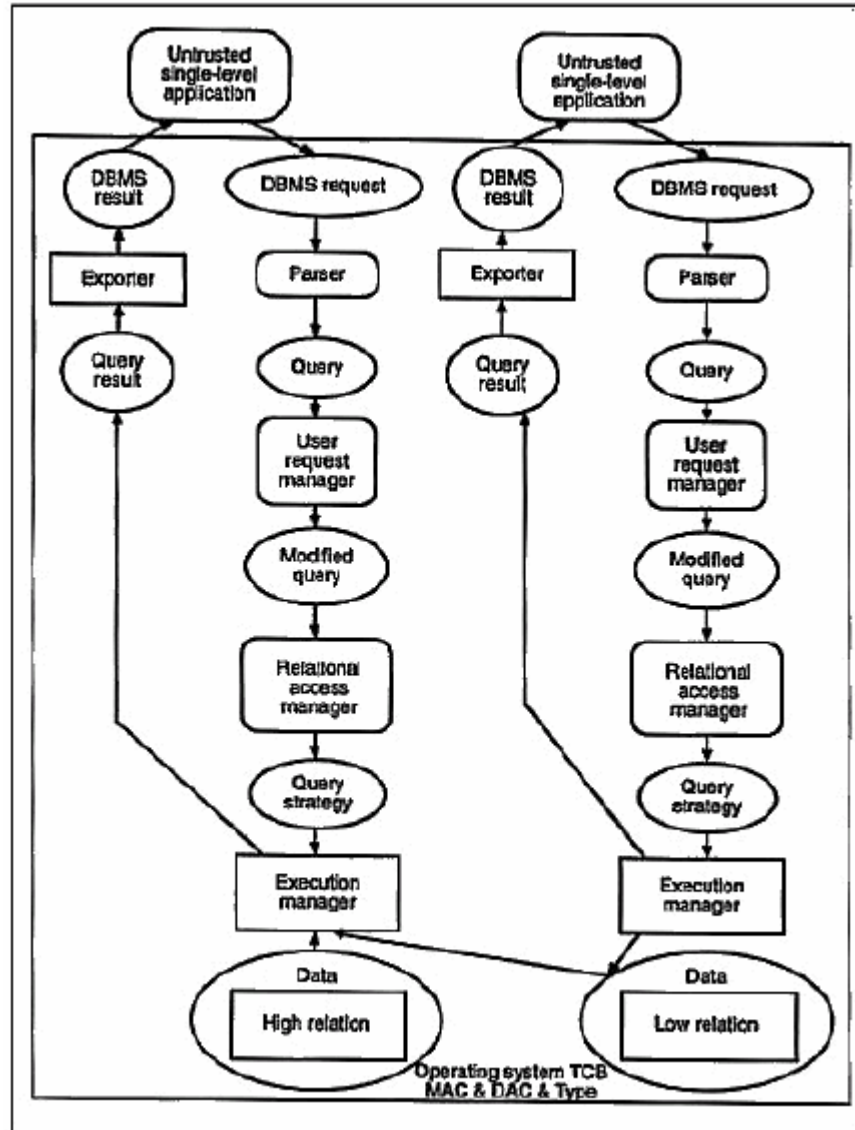
Rancangan *LDV* didasarkan pada kontrol akses dan *type enforcement* khusus dari sistem operasi *LOCK (Logical Coprocessing Kernel)*. Sebagai tambahan terhadap *mandatory access control (MAC)* yang didasarkan pada level sensitivitas dan *discretionary access control (DAC)* yang didasarkan pada daftar kendali akses (*access control lists*), *LOCK* juga melakukan kendali akses didasarkan pada domain (atau tugas). *LOCK* menyelenggarakan sebuah domain dan tabel jenis yang disebut sebagai *Domain*

Definition Table (DDT). Dalam tabel ini domain-domain diiriskan dengan jenis-jenis data. Pada bagian irisan *access privileges* direkam (sebagai contoh, *read*, *write*, *execute*). *DDT* adalah mekanisme yang digunakan untuk mengeset rangkaian-rangkaian translasi obyek yang benar yang disebut juga sebagai *pipelines*. *Pipelines* ini digunakan untuk mengisolasi jalur-jalur eksekusi kepada sistem. Jadi, *LDV DBMS* merupakan sekumpulan *pipelines* yang mempunyai tanggung jawab terhadap manipulasi data multilevel. Tiga jenis *pipelines* utama dalam *LDV* adalah *response pipeline*, *update pipeline*, dan *metadata pipeline*. *Response pipeline* memproses permintaan untuk mengambil data. *Update pipeline* mengatur semua permintaan untuk mengubah database, termasuk operasi *insert*, *update*, dan *delete*. *Metadata pipeline* menangani semua perintah administrator untuk memanipulasi database metadata.

Mirip dengan pendekatan *TCB subset*, *LDV* berada diatas sistem operasi *LOCK*, dan database multilevel disimpan sebagai sekumpulan obyek-obyek *single-level* yang diproteksi oleh sistem operasi *LOCK*. Gambar 4 dan 5 mengilustrasikan arsitektur *LDV*.



Gambar 4. Arsitektur *LDV*



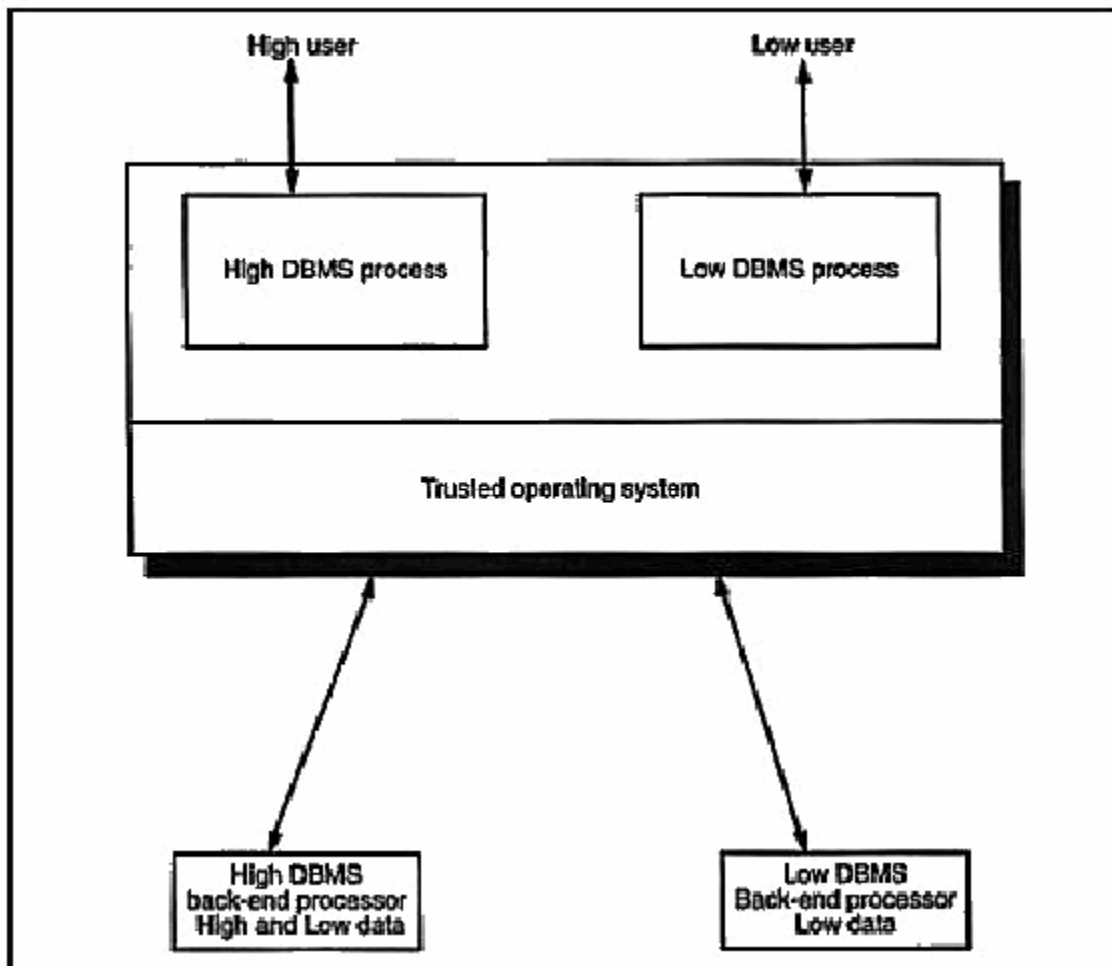
Gambar 5. Arsitektur LDV

Arsitektur Terdistribusi dengan Replikasi Data secara Penuh

Arsitektur ini menggunakan distribusi secara fisik dari database multilevel untuk mendapatkan *mandatory separation* dan kendali akses yang kuat. Arsitektur ini menggunakan banyak pengolah database back-end untuk memisahkan database ke dalam

fragmen-fragmen *sistem-high*. Pengolah *front-end* menjadi media semua akses *user* kepada database multilevel dan kepada pengolah database back-end single-level.

Pengolah front-end bertanggung jawab untuk mengarahkan *queries* ke pengolah database yang benar, memastikan tidak ada arus informasi yang salah, menjaga konsistensi data antara fragmen-fragmen database yang direplikasi, dan memberikan respon *query* pada *user* yang tepat. Sebagai tambahan pengolah front-end juga bertanggung jawab terhadap identifikasi dan otentifikasi *user*, dan proses audit.



Gambar 6. Arsitektur Terdistribusi dengan Replikasi Data secara Penuh

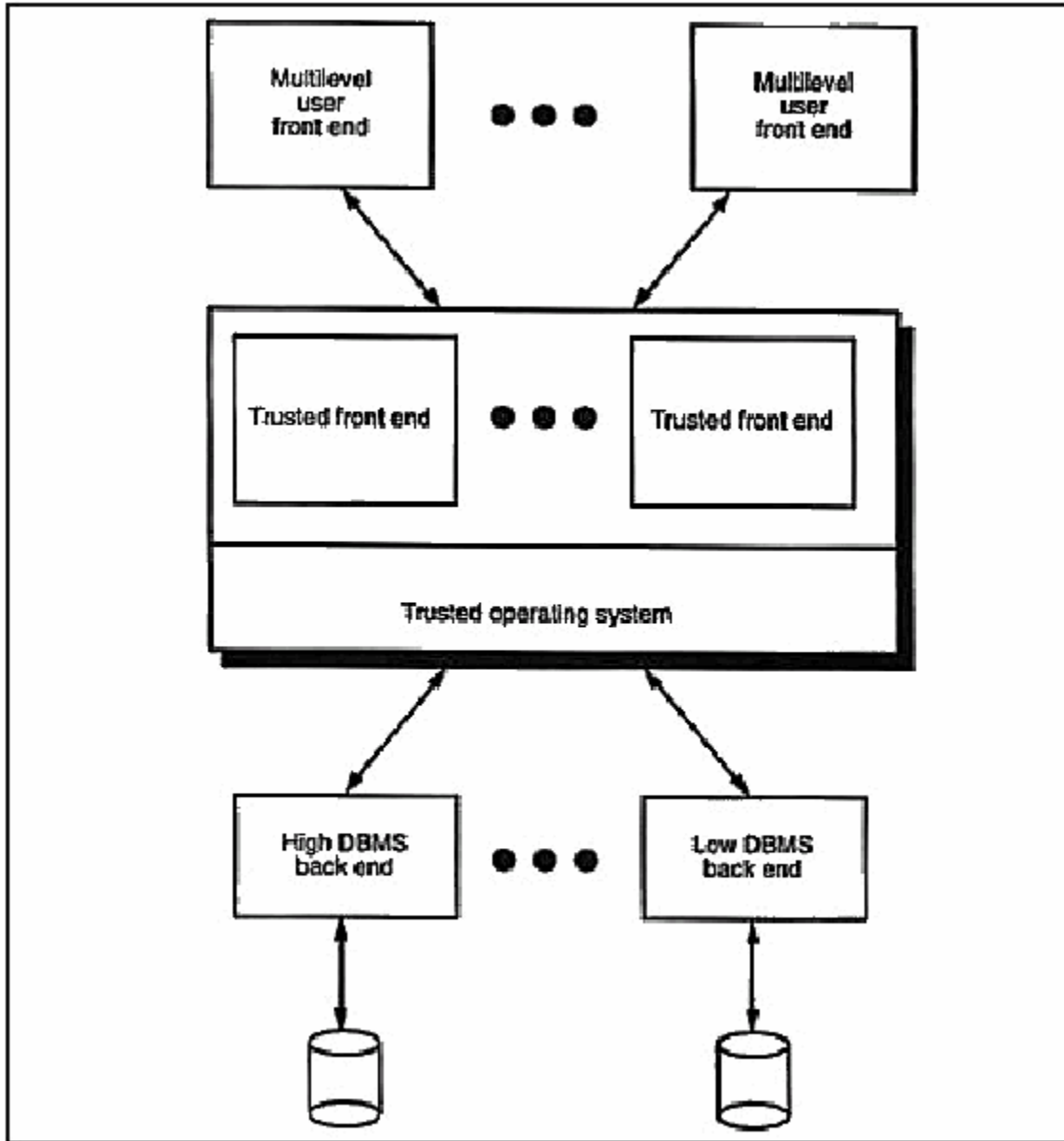
Arsitektur Terdistribusi dengan Replikasi Data secara Variabel

Berbeda dengan arsitektur sebelumnya, arsitektur ini membolehkan data untuk didistribusikan dan direplikasikan menurut kebutuhan penggunaan aktual. Pendekatan ini digunakan dalam proyek *Unisys Secure Distributed DBMS (SD-DBMS)*.

Pendekatan arsitektural yang diambil untuk mendapatkan *trusted operation* adalah dengan mendistribusikan relasi-relasi multilevel ke dalam fragmen-fragmen *single-level* dan memasukkan semua fragmen *single-level* ini banyak pengolah *DBMS back-end*. Gambar 7 mengilustrasikan arsitektur *SD-DBMS* yang disederhanakan menjadi dua level keamanan: *high* dan *low*. Arsitektur ini terdiri dari tiga jenis komponen: *user front end (UFE)*, *trusted front end (TFE)*, dan interkoneksi.

Perangkat *UFE* disini adalah dapat berupa *workstation* yang menjalankan mode *single-level* atau *trusted workstation* yang menjalankan mode multilevel di dalam suatu jangkauan tingkat-tingkat keamanan tertentu. *UFE* digunakan sebagai tempat aplikasi yang menyediakan antarmuka antara *end user* dan *TFE*.

Komponen *TFE* mengendalikan eksekusi semua perintah *DBMS* dan berlaku sebagai monitor referensi untuk akses database. *TFE* terdiri dari fungsi-fungsi *trusted* dan *untrusted* yang dibangun pada sistem operasi yang *trusted* dan *high-assurance*. Banyak host *DBMS back-end* berhubungan dengan *TFE*. Setiap host *DBMS back-end* beroperasi dalam mode *system high* pada kelas akses dalam jangkauan kelas akses *TFE*. Semua *DBMS back-end* ini memasukkan data pada kelas akses tertentu dan merespon *request* yang dibangkitkan oleh *TFE*.



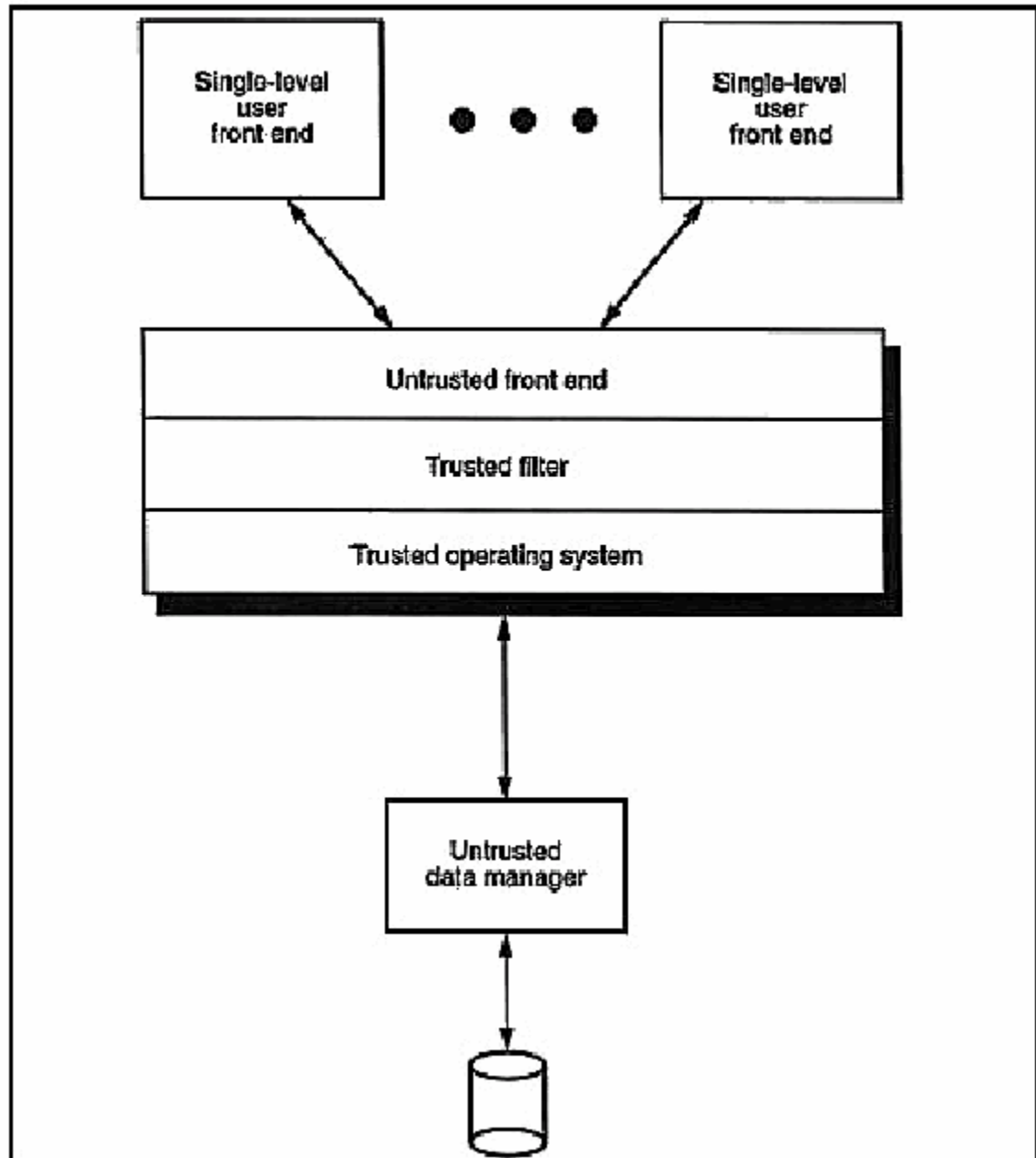
Gambar 7. Arsitektur Terdistribusi dengan Replikasi Data secara Variabel

Integrity-lock DBMS

Arsitektur *integrity-lock*, seperti yang diperlihatkan dalam gambar 6, terdiri dari tiga komponen: proses *front-end untrusted*, proses *trusted filter*, dan proses *data manager untrusted*. Proses *front-end untrusted* berinteraksi dengan *end user*. Proses ini

bertanggung jawab untuk melakukan *query parsing* dan memproses respon yang akan dikirimkan kepada *end user*. Proses *trusted filter* bertanggung jawab untuk melakukan enkripsi dan dekripsi obyek-obyek dan label-labelnya, melakukan identifikasi data-data yang dikembalikan oleh proses *data management*, dan melakukan *downgrading* obyek-obyek yang dikembalikan kepada *end user*. Misalkan disini obyek database merupakan sekumpulan *tuple*. Dalam kasus ini *trusted filter* akan membangkitkan *cryptographic checksum* dengan melakukan proses enkripsi kepada setiap *tuple* dan label sensitivitas dari tiap *tuple*, sehingga *tuple* terkunci. Residu proses enkripsi dikaitkan dengan *tuple* sebagai *checksumnya*. Database multilevel disimpan dibawah proses *data management*.

Ketika *end user* melakukan operasi seleksi terhadap database, *trusted filter* akan mengarahkan *data manager* untuk mengambil semua *tuple* sesuai dengan kriteria seleksi. *Tuple* ini dikembalikan ke *trusted filter*. *Trusted filter* memeriksa label sensitivitasnya dan membuang *tuple* yangt tidak lolos pengecekan *mandatory access policy*. Lalu proses ini memeriksa kembali apakah *checksumnya* benar. *Tuple* yang lolos dikembalikan ke *end user* yang melakukan operasi seleksi ini.



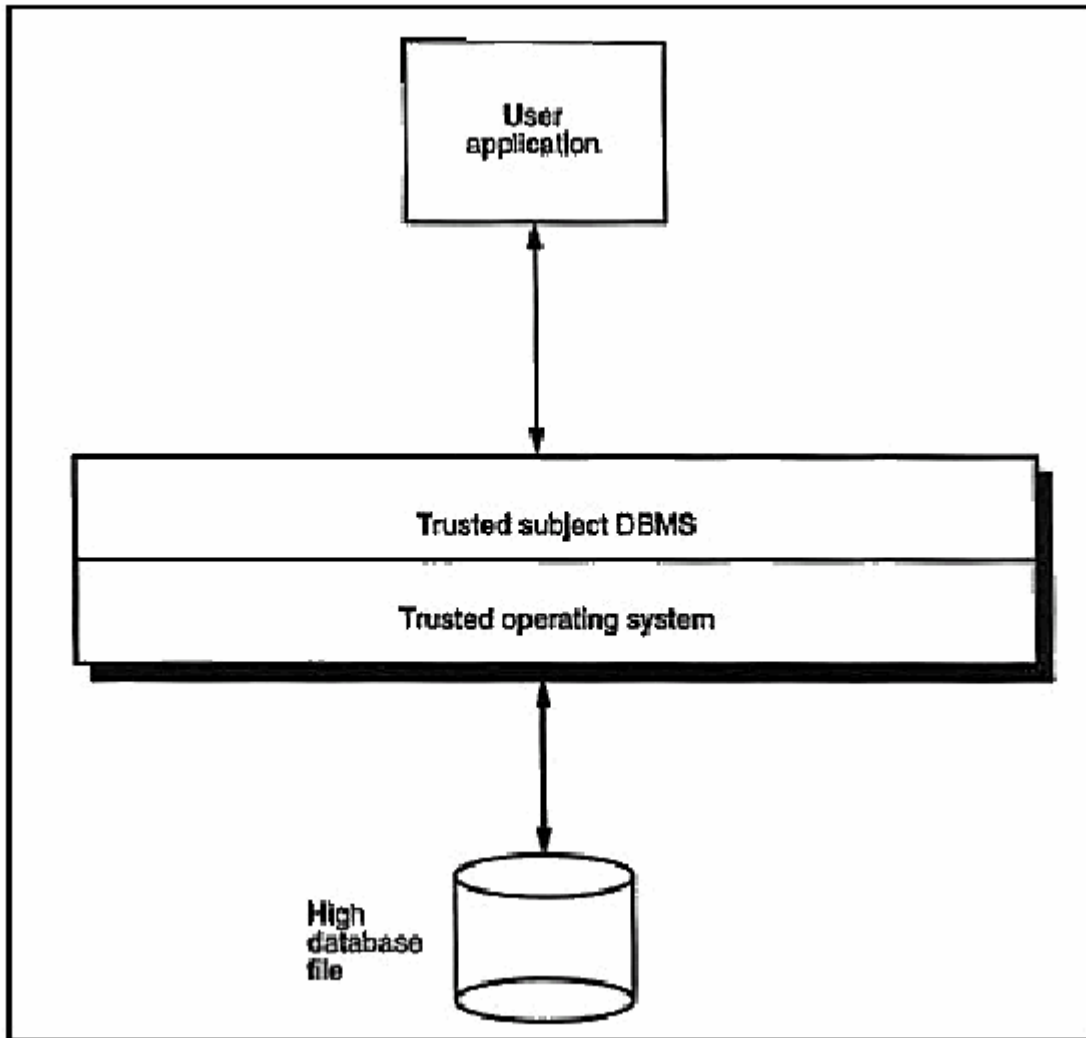
Gambar 8. Arsitektur *Integrity-Lock*

Trusted Subject-Monolithic DBMS

Pendekatan berdasarkan kernel-kernel *trusted operating system* untuk melakukan *access control enforcement* mengorbankan beberapa fungsionalitas *DBMS* untuk mendapatkan *mandatory assurance* yang lebih tinggi. Dengan pengecualian pada sistem

database *Oracle* yang menggunakan pendekatan *TCB subset*, semua produk *DBMS* yang dijual atau dibangun saat ini mengandalkan kepada sistem database itu sendiri untuk mengatur kendali akses terhadap obyek database.

Dengan pendekatan ini software *DBMS* berjalan di atas *trusted operating system*. Sistem operasi menyediakan isolasi kode *DBMS* dan mengendalikan akses terhadap database sehingga setiap akses terhadap database harus melewati *trusted DBMS*. *DBMS* menyimpan database multilevel dalam satu atau lebih file. *DBMS* mengkaitkan suatu label sensitivitas dengan setiap *tuple*. Label ini diperlakukan sebagai atribut relasi, meskipun dalam kenyataannya label itu merupakan atribut virtual yang tidak perlu dimasukkan.



Gambar 9. Arsitektur *Trusted Subject-Monolithic*

Prototipe Keamanan Database Multilevel

Ada tiga macam prototipe yang dibahas disini, yaitu: *SeaView*, *LVD*, dan *ASD* (*Advance Secure DBMS*). Tetapi yang akan dibahas berikut ini adalah prototipe *ASD* karena dua prototipe lainnya sudah dibahas sebelumnya.

Advance Secure DBMS (ASD)

Arsitektur *ASD* dapat dibagi menjadi dua bagian, yaitu: arsitektur internal dan arsitektur network. Arsitektur ini dapat dilihat pada gambar 10.

Arsitektur Internal *ASD*

Berbeda dengan *SeaView* dan *LDV*, *ASD* mengambil pendekatan yang memuat *mandatory access control (MAC)* dan *discretionary access control (DAC)* dalam aturan arsitektur internalnya. Aturan ini disebut juga sebagai *ASD TCB (Trusted Computing Base)*. Sementara arsitektur *SeaView* dan *LDV* secara umum mengandalkan sistem operasinya untuk memuat *MAC*, dan *DBMS* internal untuk memuat *DAC*.

Meskipun *ASD TCB* berjalan sebagai proses dibawah kendali *TCB* dari sistem operasi, tetapi ini tidak dapat dikatakan bahwa *ASD TCB* berada di dalam *TCB* dari sistem operasi. *ASD TCB* tidak berbagi domain proteksi dengan sistem operasi.

Seperti juga *DBMS* lainnya, *ASD* terdiri dari *trusted code* dan *untrusted code*. *Untrusted code* melakukan operasi-operasi *DBMS* yang tidak relevan dengan keamanan. Kode ini berjalan pada kelas akses dari proses yang dijalankan oleh *user*.

TCB dari sistem operasi memproteksi *ASD TCB* dari proses-proses lain yang berjalan dalam sistem operasi. *TCB* ini juga menjamin bahwa tidak ada proses yang dapat mengakses data *DBMS* kecuali melewati *ASD TCB*.

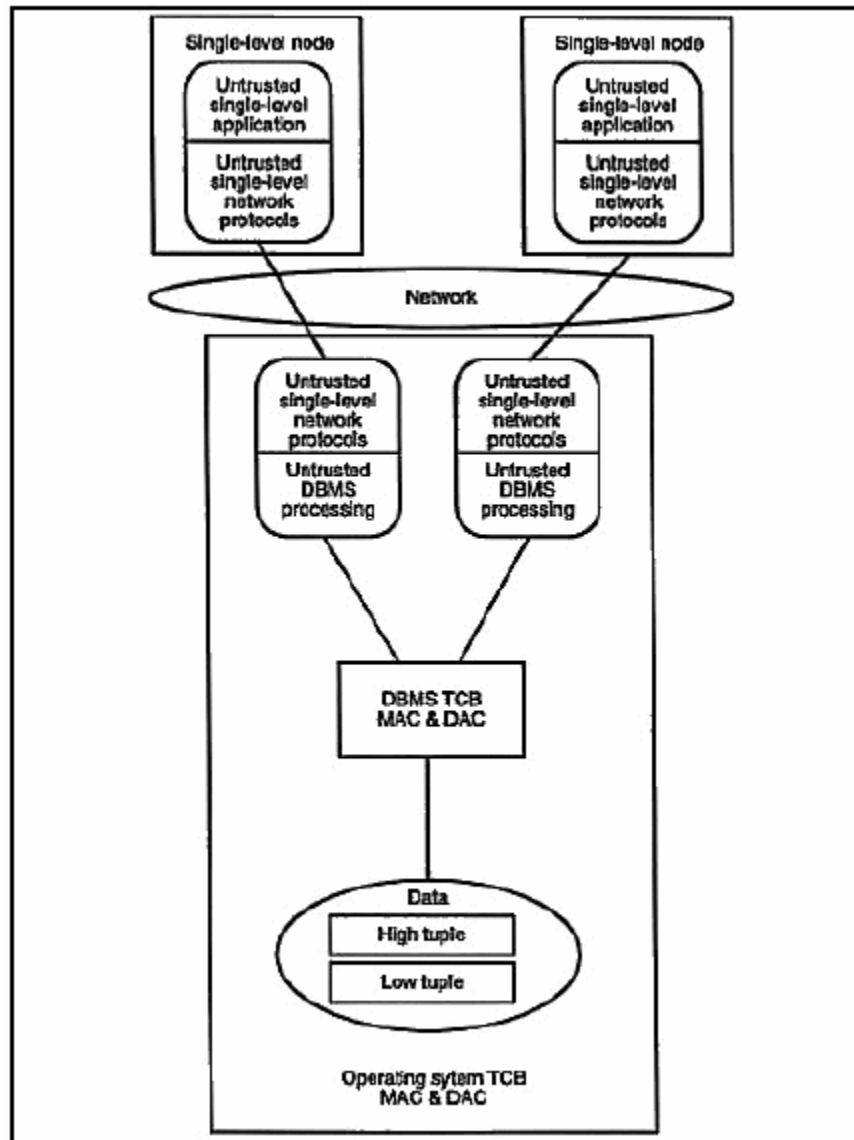
Secara konseptual semua *tuple* berada dalam satu file sistem operasi. File ini dikategorikan pada bagian terbawah dari *tuple* yang ada di dalam file. Setiap *tuple* yang dimasukkan ke file ini mengandung klasifikasi yang ditentukan oleh *ASD TCB* berdasarkan tingkat keamanannya.

Karena file data *ASD* berisi *tuple* yang diberi label dengan klasifikasi yang berlainan, file ini harus diproteksi dari modifikasi oleh proses-proses *untrusted* yang mungkin saja memiliki klasifikasi yang sama dengan file data *ASD*. Pendekatan yang digunakan *ASD* adalah dengan menggunakan sistem operasi yang mendukung aturan integritas *Biba*. Aturan ini menentukan label-label integritas kepada subyek dan obyek. Label-label ini adalah analogi label-label keamanan yang berkaitan dengan pengklasifikasian. Dengan berlakunya aturan integritas ini suatu subyek dapat menulis ke suatu obyek yang diproteksi hanya jika tingkat integritas dari subyek mendominasi tingkat integritas dari obyek. Untuk memproteksi data *ASD* obyek sistem operasi yang berisi data tersebut diberikan label *DBMS integrity compartment*. Label ini membatasi akses tulis ke obyek tersebut hanya kepada subyek yang memiliki *DBMS integrity compartment* yang lebih dominan. Karena hanya *ASD TCB* yang memiliki *integrity compartment* ini, tidak ada subyek lain yang dapat menulis secara langsung ke dalam obyek sistem operasi yang berisi data *ASD*. Tentunya, subyek-subyek lain dapat menggunakan fasilitas-fasilitas *ASD TCB* untuk memasukkan data *tuple* ke dalam obyek, tetapi hanya dibawah kendali *ASD TCB*.

Arsitektur Network *ASD*

ASD diimplementasikan sebagai *trusted server*, seperti yang ada pada gambar 7. Dibawah pendekatan ini, *ASD* beroperasi pada nodenya sendiri dalam network. *ASD* node melayani node-node lainnya yang mirip *single-level* tetapi beroperasi pada klasifikasi-klasifikasi keamanan yang berbeda. *ASD* menyediakan *multilevel engine* yang

memberikan fasilitas *sharing* antar level. Node *top secret* dapat mengambil data dari *ASD trusted server* yang memiliki data dikategorikan sebagai *node unclassified*.



Gambar 10. Arsitektur *ASD*

Penyediaan akses network perlu mempertimbangkan protokol network yang aman. Protokol network yang didukung oleh *ASD server* adalah *TCP*, *IP*, dan *SLIP*

(*Serial Line IP*). *SLIP* memperbolehkan protokol *TCP/IP* digunakan pada kabel serial, seperti kabel telepon. Ini membolehkan *ASD* untuk digunakan dalam lingkungan taktis.

Karena software protokol network secara aktual menangani data, software ini akan menjadi relevan terhadap keamanan jika secara kongkuren atau sekuensial software menangani data yang digolongkan pada banyak kelas akses. Sebagai contohnya, pesan *top secret* dikirimkan ke node *top secret*, kemudian pesan *unclassified* dikirimkan ke node *unclassified*. Kebocoran data dapat terjadi karena software ini. Tentu saja hal ini melanggar *mandatory security policy*.

Solusi yang diambil terhadap masalah ini adalah dengan menggunakan protokol network secara terpisah untuk tiap tingkat keamanan. Sesuai dengan konsep ini jika suatu host memiliki dua port serial, setiap port serial dikaitkan dengan software network protokol yang terpisah, dan port-port serial ini berhubungan dengan node yang beroperasi pada kelas akses yang berbeda, maka kebocoran data dapat dihilangkan.

Penutup

Berbagai arsitektur dan prototipe telah dikembangkan untuk memenuhi kriteria keamanan database multilvel yang memadai. Masing-masing memiliki konsep-konsep pendekatan yang unik. Dari berbagai arsitektur yang ada pendekatan *trusted subject* masih mendominasi produk-produk database saat ini.

DAFTAR PUSTAKA

- Notargiacomo, LouAnna. *Architectures for MLS Database Management Systems*.
- Hinke, Thomas H. *Multilevel Secure Database Management Prototypes*.
- Jajodia, Sushil. *Toward a Multilevel Secure Relational Data Model*.

DAFTAR ISTILAH

- *relations* : data yang tersimpan dalam tabel-tabel
- *attributes* : sejumlah kolom yang dimiliki *relation*
- *tuples* : sejumlah baris yang dimiliki *relation* pada suatu waktu
- *mandatory access control* : kendali akses yang mengatur kewajiban subyek agar subyek dapat mengakses obyek tertentu
- *discretionary access control* : kendali akses yang menentukan kebebasan yang dimiliki subyek terhadap obyek tertentu
- *integrity* : hal yang memaksakan kondisi-kondisi tertentu terhadap relasi-relasi
- *security* : hal yang memaksakan pemisahan antar data
- *multilevel engine* : penggerak multilevel