ISSN: 1978 - 8282

PENERAPAN ARSITEKTUR THREE-TIER TERHADAP OPTIMALISASI KEAMANAN DISTRIBUTED DATABASE

Muhammad Yusup[1]
Untung Rahardja[2]
Eko Prasetiyani[3]

muhamad.yusup@faculty.raharja.ac.id,
untung.rahardja@faculty.raharja.ac.id,
prasetiyani@si.raharja.ac.id,

Diterima: 8 April 2010/ Disetujui: 17 Maret 2010

ABSTRACT

Along with the advancement in information technology and the increasing rate of rapid development encourages the development of the security levels on the server database system.Security level is viewed from a variety of tier, either on the Application Tier, Database Tier, and Network Tier. However, the security application server database system has been only limited to the Application Tier, for example, the application of user access rights with Token IP, IP Address, Mac Address and the Global Password Management (GPM) in the RME server. For Tier Network security is set up through the router, but the security of the Database Tier well untouched. Currently RME Server is still using the concept of Two-Tier, which is between the Database Tier and Application Tier is still in one server, use Windows Authentication as user access rights on the database side. Thus the application of the course is very vulnerable to security of the server database system, in which SQL commands are executed by the user difficult to control because it has not applied Mapping User and Server Roles in the database.These problems then the RME server with Two-Tier concept should be optimized as it can provide a significant impact on the continuity of 4 (four) pillar IT E-learning in Higher Education environment Raharja. To overcome these two needed a Three-Tier concept is applied to iRME server. By using the Three-Tier, allows the physical separation between the Database Tier and Application Tier on different servers so that expected can improve safety on the database server.This architecture has the potential to be applied backups of the Application Tier server problems, allowing also applied iRME server mirroring as advanced development. Can be concluded that the Three-Tier concept

1. **Dosen Jurusan Teknik Informasi, AMIK Raharja Informatika**
   Jl. Jend Sudirman No. 40 Modern Cikokol-Tangerang Telp. 5529692
2. **Dosen Jurusan Sistem Informasi, STMIK Raharja**
   Jl. Jend Sudirman No. 40 Modern Cikokol-Tangerang Telp. 5529692
3. **Mahasiswa Jurusan Sistem Informasi, STMIK Raharja**
   Jl. Jend Sudirman No. 40 Modern Cikokol-Tangerang Telp. 5529692

can be a current solution to improve security at the database server environment Raharja College.

ABSTRAK

Seiring dengan kemajuan teknologi informasi dan laju perkembangannya yang semakin pesat mendorong adanya perkembangan tingkat keamanan pada sistem database server. Tingkat keamanan tersebut dapat dijaga dari berbagai tier, baik pada Application Tier, Database Tier, maupun Network Tier. Namun penerapan keamanan sistem database server saat ini baru sebatas pada Application Tier saja, misalnya penerapan hak akses user dengan IP Token, IP Address, Mac Address serta Global Password Management (GPM) pada server RME. Untuk keamanan Network Tier sudah diatur melalui router, tetapi keamanan dari Database Tier belum disentuh dengan baik. Saat ini Server RME masih menggunakan konsep Two-Tier, yaitu antara Database Tier dan Application Tier masih berada dalam satu server, menggunakan Windows Authentication sebagai hak akses user pada sisi database. Sehingga penerapan tersebut tentunya sangat rentan terhadap keamanan sistem database server, dimana perintah-perintah SQL yang dijalankan oleh user sulit untuk dikendalikan karena belum diterapkannya User Mapping dan Server Roles pada database. Dari permasalahan tersebut maka server RME dengan konsep Two-Tier perlu dioptimalkan karena dapat memberikan dampak yang cukup signifikan terhadap kelangsungan 4 (empat) pilar IT E-learning di lingkungan Perguruan Tinggi Raharja. Untuk mengatasi hal tersebut maka dibutuhkan suatu konsep Three-Tier yang diterapkan pada server iRME. Dengan menggunakan Three-Tier, memungkinkan adanya pemisahan secara fisik antara Database Tier dan Application Tier pada server yang berbeda sehingga diharapkan dapat meningkatkan keamanan pada sisi database server. Arsitektur ini memiliki potensi untuk diterapkan backup server jika Application Tier mengalami masalah, memungkinkan juga diterapkan mirroring pada server iRME sebagai pengembangan lanjutan. Dapat disimpulkan bahwa dengan konsep Three-Tier ini dapat menjadi sebuah solusi terkini dalam meningkatkan keamanan database server di lingkungan Perguruan Tinggi Raharja.

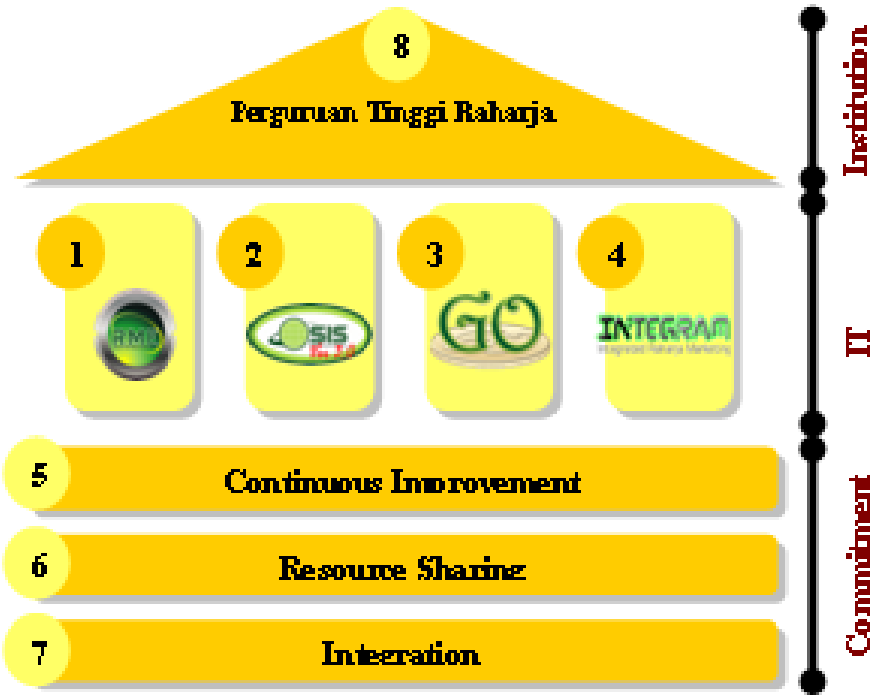
Kata kunci : Three-Tier, client/server, iRME

PENDAHULUAN

Perguruan Tinggi Raharja merupakan perguruan tinggi yang bergerak di bidang ilmu komputer yang berada di Propinsi Banten dan terletak hanya 10 (sepuluh) menit dari Bandara Internasional Soekarno-Hatta. Banyak penghargaan yang telah di raih, salah satunya adalah memenangkan *WSA* 2009 - Indonesia *E-Learning and Education Category of Intranet Product Raharja Multimedia Edutainment (RME)*. Pada saat ini Perguruan Tinggi Raharja pun telah meningkatkan mutu dan kualitasnya melalui sertifikat akreditasi dari Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT) diantaranya menyatakan bahwa program studi Diploma Tiga Komputerisasi Akuntansi di AMIK Raharja Informatika terakreditasi A. Selain itu, Perguruan Tinggi Raharja telah masuk peringkat 100 universitas teratas dan perguruan tinggi terbaik di Republik

Indonesia, serta diperolehnya sertifikasi ISO 9001:2008 dengan nomor sertifikat: JKT 6007007 sebagai pengakuan mutu pengelolaan manajemen Perguruan Tinggi.

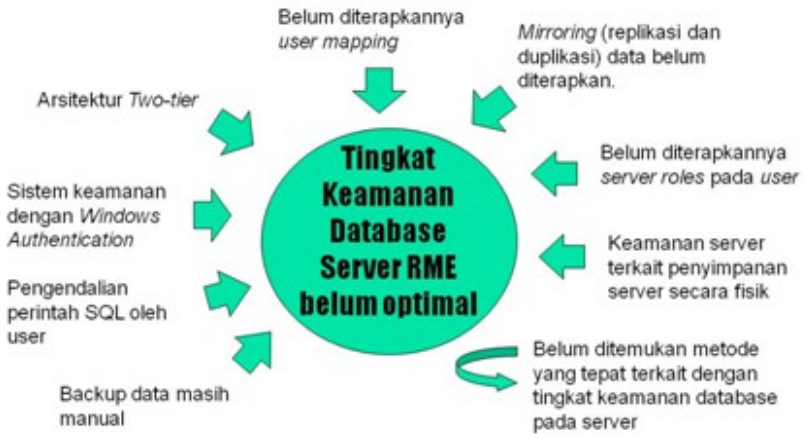
Perguruan Tinggi Raharja mempunyai 4 (empat) pilar *IT E-learning* yang terdiri dari *SIS (Student Information Services)*[1], *RME (Raharja Multimedia Edutainment)*[2], *INTEGRAM (Integrated Raharja Marketing)*[3], dan *GO (Green Orchestra)*[4] adalah instrumen Perguruan Tinggi Raharja sebagai kampus unggulan sesuai dengan visinya yaitu menuju perguruan tinggi unggulan yang menghasilkan lulusan yang berkompeten di bidang sistem informasi, teknik informatika dan sistem komputer serta memiliki daya saing yang tinggi dalam era globalisasi.



Gambar 1. 4 (empat) Pilar *IT E-Learning* Perguruan Tinggi Raharja

Gambar satu di atas menjelaskan bahwa 4 (empat) Pilar *IT E-learning* tersebut berjalan harmonis membentuk bangunan yang kokoh pada institusi Perguruan Tinggi Raharja dengan ditopang oleh keberadaan Server RME yang secara berkesinambungan melakukan perbaikan *(countinues improvment),* melakukan *resource sharing* dari dan oleh Pribadi Raharja serta saling terintegrasi (*integration*) satu sama lain. Namun, apakah keberadaan Server RME yang saat ini berjalan dengan konsep Arsitektur *Two-tier* tersebut memiliki tingkat keamanan yang optimal pada sisi *database tier* dalam mendukung kelangsungan 4 (empat) pilar *IT E-learning* di lingkungan Perguruan Tinggi Raharja?

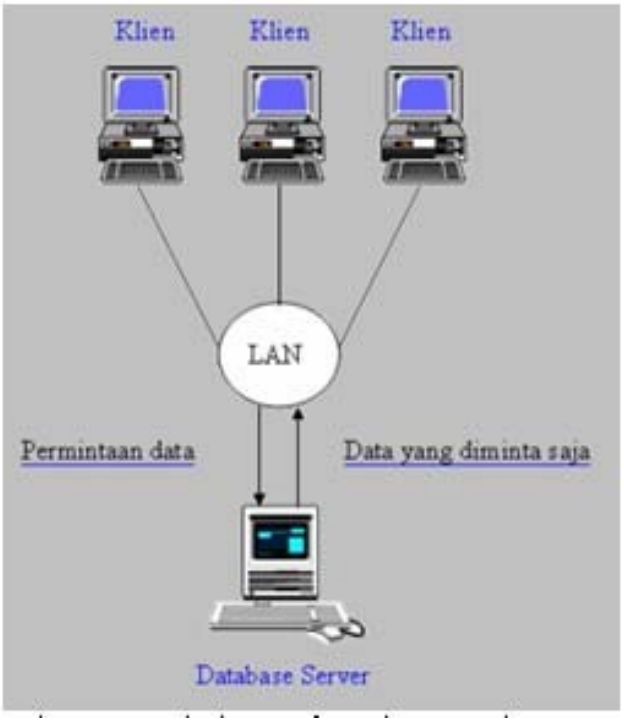
Sementara perkembangan teknologi kini telah banyak membuat perubahan pada cara berpikir manusia. Dengan laju pertumbuhan teknologi yang makin cepat, kebutuhan akan informasi dari hari ke hari terus meningkat sehingga menuntut kelancaran, dan kecepatan proses distribusi informasi. Arsitektur jaringan di lingkungan Perguruan Tinggi Raharja merupakan model konektivitas pada jaringan yang membedakan fungsi komputer sebagai *client/server*. Arsitektur ini menempatkan sebuah komputer sebagai *server* dalam hal ini adalah Server RME. Server ini yang bertugas memberikan pelayanan *IT E-learning* kepada terminal-terminal lainnya yang terhubung dalam sistem jaringan yang berhubungan dengan sistem keamanan database pada server RME. Misalnya tingkat keamanan tersebut ditinjau dari berbagai *tier*, baik pada *Application Tier*, *Database Tier*, maupun *Network Tier*. Namun dalam perjalanannya, ada beberapa permasalahan terkait dengan tingkat keamanan database server RME yang belum optimal. Keamanan sistem database server saat ini baru sebatas pada *Application Tier* dan *Network Tier* saja, misalnya penerapan hak akses *user* dengan *IP Token*, *IP Address*, *Mac Address* serta menerapkan aplikasi dengan *Global Password Management (GPM)* pada server RME. Untuk keamanan *Network Tier* sudah diatur melalui *router*, tetapi keamanan dari *Database Tier* belum disentuh dengan baik.



Gambar 2. Identifikasi masalah pada server RME

Untuk itu dilakukan beberapa identifikasi masalah terkait dengan tingkat keamanan database Server RME yang belum optimal, diantaranya adalah Server RME masih menggunakan konsep *Two-Tier*, yaitu antara *Database Tier* dan *Application Tier* masih berada dalam satu server. Dalam model *client/server*, pemrosesan pada sebuah aplikasi terjadi pada *client* dan *server*. *Client/server* adalah tipikal sebuah aplikasi *Two-Tier* dengan banyak *client* dan sebuah *server* yang

dihubungkan melalui sebuah jaringan. *Web Application* ditempatkan pada komputer *client* dan mesin database dijalankan pada server jarak-jauh. Aplikasi client mengeluarkan permintaan ke database yang mengirimkan kembali data ke *client*-nya.



Gambar 3. Arsitektur *Client/Server* (*Two-Tier*)

Model *Two-Tier*[5] terdiri dari tiga komponen yang disusun menjadi dua lapisan: *client* (*requesting service*) dan *server* (*providing service*). Tiga komponen tersebut yaitu: 1) User Interface: adalah antar muka program aplikasi yang berhadapan dan digunakan langsung oleh user, 2). Manajemen Proses, dan 3). Database. Model ini memisahkan peranan *user interface* dan database dengan jelas, sehingga terbentuk dua lapisan. Server database berisi mesin database, termasuk tabel, prosedur tersimpan, dan *trigger* (yang juga berisi aturan bisnis). Dalam sistem *client/server*, sebagian besar logika bisnis biasanya diterapkan dalam database. Server database menangani: 1). Manajemen data, 2). Keamanan, query, trigger, prosedur tersimpan, dan 3). Penanganan kesalahan. Namun dari penjelasan diatas, masih terdapat kelemahan pada arsitektur ini. Selain menjalankan tugas-tugas tertentu, kinerja dan skalabilitas merupakan tujuan nyata dari sebagian besar aplikasi. Adapun kekurangan dari model *client/server* adalah: 1). Kurangnya skalabilitas, 2). Tidak ada tingkat menengah untuk menangani keamanan dan transaksi. Selain itu, tingkat keamanan database server RME yang belum optimal terkait dengan penggunaan *Windows Authentication* sebagai

hak akses *user* pada sisi database. Setiap *user* yang akan mengakses database pada server RME, bisa saja *user* menjalankan perintah *drop* atau *delete* pada database, baik itu disengaja atau pun tidak. Yang menyebabkan database atau *table* menjadi hilang karena terhapus atau dihapus. Sehingga pengunaan *Windows Authentication* oleh *user* tersebut tentunya sangat rentan terhadap keamanan sistem database server, dimana perintah-perintah SQL yang dijalankan oleh *user* sulit untuk dikendalikan oleh *administrator*.

Dari permasalahan diatas dapat dirumuskan bahwa belum ditemukan metode yang tepat terkait dengan tingkat keamanan pada database server yang saat ini sedang berjalan dilingkungan Perguruan Tinggi Raharja.

PEMBAHASAN

1. Literature Review

Banyak penelitian yang sebelumnya dilakukan berkenaan dengan *Literature Review* mengenai konsep *Three-Tier* dan *Distributed Database*. Dalam upaya pengembangan keamanan sistem database server ini perlu dilakukan studi pustaka sebagai salah satu dari penerapan metode penelitian yang akan dilakukan. Diantaranya adalah mengidentifikasikan kesenjangan (*identify gaps*), menghindari pembuatan ulang (*reinventing the wheel*), mengidentifikasikan metode yang pernah dilakukan, meneruskan penelitian sebelumnya, serta mengetahui orang lain yang spesialisasi dan area penelitiannya sama dibidang ini. Beberapa *literature review* tersebut adalah sebagai berikut:

1. Penelitian ini dilakukan oleh Bob Bretl Allen, Allen Otis, Marc San Soucie, Bruce Schuchardt, R. Venkatesh, 1998, dari Gemstone Systems Inc, berjudul *" Persistent Java Objects in Three-Tier Architectures "*. Penelitian ini membahas tentang arsitektur *Three-Tier* yang secara logis yang memisahkan fungsi aplikasi ke dalam komponen antarmuka pengguna, server komponen logika bisnis, dan database komponen. Banyak produk-produk server aplikasi dan produk-produk *middleware* menyediakan dukungan untuk membangun dan menggunakan aplikasi yang menggunakan arsitektur *Three-Tier*. Lapisan aplikasi *Three-Tier* menawarkan keuntungan signifikan atas pendekatan-pendekatan lain. Komponen dari arsitektur standar *Three-Tier* terdiri dari presentasi dan logika aplikasi pada *client*, aplikasi dan logika bisnis dalam server aplikasi tingkat menengah, dan data yang dikelola oleh database[6].

2. Penelitian ini dilakukan oleh Lieven Desmet, Bart Jacobs, Frank Piessens, Wouter Joosen, 2004, dari Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS), berjudul *" A Generic Architecture for Web Applications to Support Threat Analysis of Infrastructural Components "*. Penelitian ini menjelaskan bahwa tentang analisis ancaman yang berguna dari platform aplikasi web, beberapa arsitektur asumsi tentang aplikasi tersebut harus dibuat. Dokumen ini menjelaskan generik khas arsitektur *Three-Tier* aplikasi web. Ini berfungsi sebagai dasar untuk menganalisis ancaman pada infrastruktur yang paling penting komponen dalam arsitektur[7].
3. Penelitian yang dilakukan oleh Roberto Baldoni, Carlo Marchetti, Alessandro Termini, 2002, dari IEEE International Symposium on Reliable Distributed Systems (SRDS02), berjudul *" Active Software Replication through a Three-tier Approach "*. Penelitian ini membahas tentang sebuah logika replikasi adalah seperangkat protokol dan mekanisme pelaksanaan teknik replikasi perangkat lunak. Three-Tier terdiri replikasi pendekatan dalam memisahkan logika replikasi dari kedua klien dan server ditiru oleh embedding logika seperti itu di *middle-tier*[8].
4. Penelitian ini dilakukan oleh Jun Lin Lin dan Margaret H. Dunham dari Southerm Methodist University dan Mario A. Nascimento berjudul " *A Survey of Distributed Database Checkpointing*". Penelitian ini membahas mengenai *checkpointing* pada *database* terdistribusi dan pendekatan-pendekatan yang digunakan. Penelitian ini bermula dari adanya banyak survey yang dilakukan berkenaan dengan proses *recovery database*, dan banyak teknik yang diusulkan untuk mengatasinya. Dengan *distributed database checkpointing*, dapat mengurangi waktu proses *recovery* suatu kegagalan didalam *database* terdistribusi. *Checkpointing* dapat digambarkan sebagai suatu aktivitas menulis informasi ke penyimpanan yang stabil selama operasi normal dalam rangka mengurangi jumlah pekerjaan pada saat *restart*. Penelitian ini membantah bahwa sedikit batasan dan sedikit sumber daya menjadi masalah dalam pendekatan database terdistribusi, serta Membantah bahwa *checkpointing* hanya dapat digunakan untuk sistem distribusi yang *multidatabase*. Meskipun penelitian ini telah banyak dilakukan namun cukup rumit dalam implementasinya. Dengan penelitian ini kita dapat mengembangkan *database* terdistribusi dengan *checkpointing* untuk mempercepat proses *recovery database*[9].
5. Penelitian ini dilakukan oleh David J. DeWitt dari Universitas Wisconsin dan Jim Gray tahun 1992 berjudul "*Parallel Database Systems:The Future of High Performance Database Processing*". Penelitian ini dilakukan dengan konsep *database* terdistribusi yang merupakan database yang disimpan pada beberapa komputer yang terdistribusi satu sama lain. Pada penelitian ini, dijelaskan Sistem

database paralel mulai menggantikan *Mainframe* komputer besar untuk pengolahan data dan transaksi tugas. Paralel database komputer memiliki arsitektur yang berkembang dari penggunaan perangkat lunak yang eksotik untuk perangkat keras yang paralel. Seperti kebanyakan aplikasi, user menginginkan *hardware* sistem *database* yang murah, cepat. Ini menyangkut tentang prosesor, memori dan disk. Akibatnya, konsep *hardware database* yang eksotis tidak sesuai untuk teknologi saat ini. Di lain sisi, ketersediaan *microprocessors* cepat, murah dan kecil menjadi paket standar murah tapi cepat sehingga menjadi *platform* yang ideal untuk sistem *database* paralel. Stonebraker mengusulkan rancangan sederhana untuk spektrum disain yaitu *shared memory, shared disk* dan *shared nothing*. Dan bahasa yang digunakan dalam *database* adalah SQL sesuai dengan standar ANSI dan ISO. Dengan penelitian ini, kita dapat mengembangkan sistem *database* agar dapat digunakan diberbagai ruang lingkup[10].

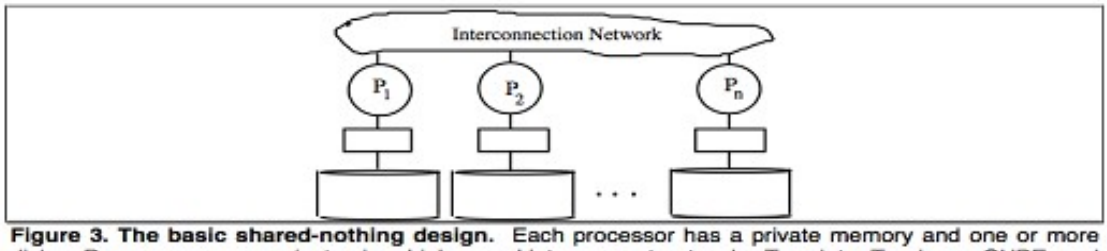


Figure 3. The basic shared-nothing design. Each processor has a private memory and one or more disks. Processors communicate via a high-speed interconnect network. Teradata, Tandem, nCUBE, and the newer VAXclusters typify this design.

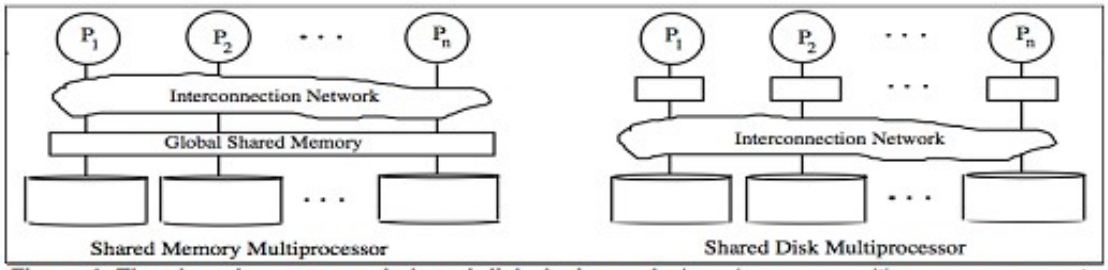
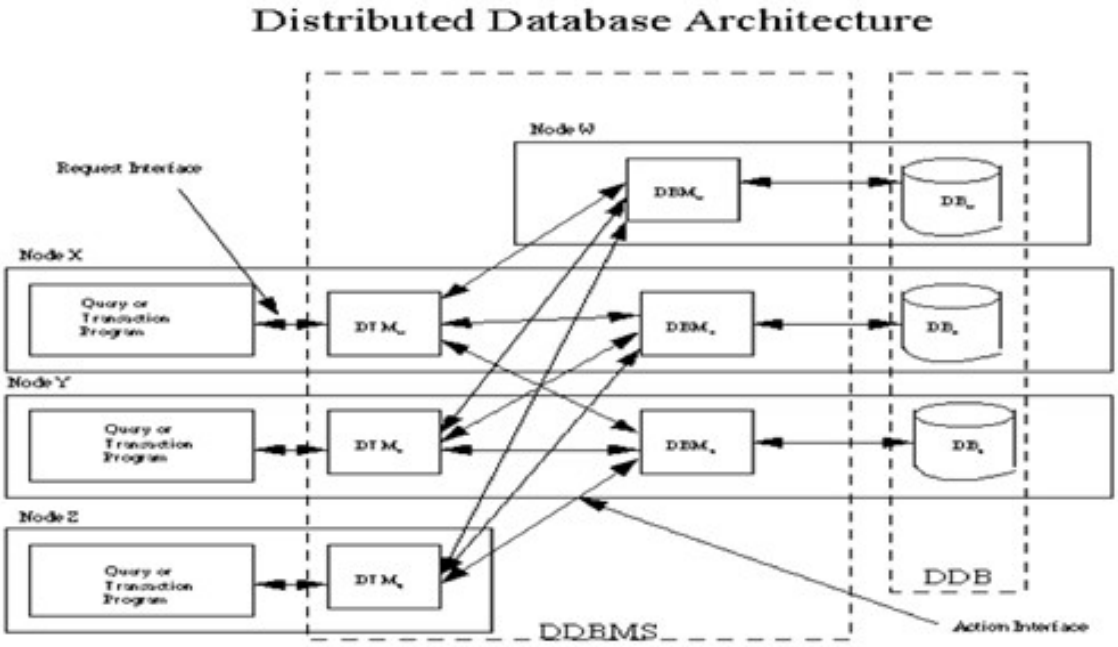


Figure 4. The shared-memory and shared-disk designs. A shared-memory multi-processor connects all processors to a globally shared memory. Multi-processor IBM/370, VAX, and Sequent computers are typical examples of shared-memory designs. Shared-disk systems give each processor a private memory, but all the processors can directly address all the disks. Digital's VAXcluster and IBM's Sysplex typify this design.

Gambar 3. *Desain Shared-Nothing, Shared-Memory* dan *Shared-Disk*

6. Penelitian ini dilakukan oleh Carolyn Mitchell dari Norfolk State University berjudul "*Components of a Distributed Database*" tahun 2004. Penelitian ini membahas tentang komponen-komponen didalam *database*. Salah satu komponen utama dalam DDBMS adalah *Database Manager*. "Sebuah *Database Manager* adalah perangkat lunak yang bertanggung jawab untuk memproses segmen data yang didistribusikan. Komponen utama lainnya adalah *Query User Interface*, yang merupakan sebuah program klien yang bertindak sebagai sebuah

antarmuka untuk Transaksi Manager yang terdistribusi. Sebuah Transaksi Manager terdistribusi adalah program yang menterjemahkan permintaan dari pengguna dan mengkonversi mereka ke *query database* manager, yang biasanya didistribusikan. Sebuah sistem *database* yang terdistribusi terbuat dari kedua manajer yaitu Database Manager dan Transaksi Manager Terdistribusi[11].



Gambar 4. Arsitektur *Distributed Database* dan Komponennya

7. Penelitian yang dilakukan oleh Hamidah Ibrahim, "*Deriving Global Integritas Dan Local Rules For Distributed Database*". Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Putra Malaysia, 43400 UPM Serdang. Ibrahim mengatakan bahwa tujuan terpenting didalam database sistem adalah menjamin konsistensi data, yang berarti bahwa data yang terdapat dalam database harus baik dan akurat. Didalam pelaksanaannya untuk menjaga konsistensi perubahan data sangat sulit, khususnya untuk didistribusikan dalam database. Dalam tulisan ini, menjelaskan sebuah algoritma penegakan aturan berdasarkan mekanisme untuk didistribusikan database yang bertujuan meminimalisir jumlah data yang harus ditransfer atau diakses diseluruh jaringan yang menjaga konsistensi dari database di satu situs, yaitu di situs mana pembaruan perlu dilakukan. Teknik ini disebut sebagai tes integritas generasi, yang berasal dari lokal dan global integritas, dan aturan yang telah efektif dapat mengurangi biaya kendala dalam memeriksa suatu data yang telah didistribusikan dalam lingkungan. Didalam penelitian ini telah berhasil menghasilkan sebuah sistem sentralistik yang besar dengan tingkat kehandalan yang tinggi untuk integritas data[12].

8. Penelitian yang dilakukan oleh Steven P. Coy dari *University of Maryland* berjudul "*Security Implication of the Choice of Distributed Database Management System Model: Relational Vs Object Oriented"*. Penelitian ini menjelaskan bahwa keamanan data harus dibenahi ketika mengembangkan database dan diantaranya memilih antara *relational* dan *object oriented model*. Banyak faktor yang harus dipertimbangkan, terutama dari segi efektifitas dan efisiensi, juga apakah sekuritas dan integritas ini memakan sumber daya yang terlalu besar tidak semata-mata fitur keamanan. Kedua pilihan ini akan mempengaruhi kekuatan dan kelemahan dari database tersebut. Untuk *centralized database* kedua model ini bisa dikatakan sama baiknya. Namun untuk *distributed database*, *relational model* lebih unggul dibidang sekuritas. Ini lebih banyak disebabkan karena *object oriented model database* masih kurang maturitasnya. Sehingga didalam lingkungan heterogenous, proses integritasnya masih menimbulkan banyak masalah. OODBMS tetap saja masih perlu perkembangan teknologi lebih lanjut, namun di lingkungan homogenous, OODBMS dapat menjadi pilihan yang baik[13].
9. Penelitian yang dilakukan oleh Stephane Gançarski, Claudia León, Hubert Naacke, Marta Rukoz and Pablo Santini yang berjudul "*Integrity Constraint Checking in Distributed Nested Transactions over a Database Cluster*" adalah sebuah solusi untuk memeriksa integritas dan kendala global dalam berhubungan multi *database* sistem. Penelitian ini juga menyajikan hasil eksperimental yang diperoleh atas solusi PC *cluster* dengan Oracle9i DBMS. Tujuan adalah melakukan eksperimentasi untuk mengukur waktu yang dihabiskan dalam memeriksa kendala global dalam sistem yang terdistribusi. Alhasil menunjukkan bahwa *overhead* berkurang hingga 50% dibandingkan dengan pemeriksaan integritas yang terpusat. Studi menunjukkan bahwa sistem berkemungkinan besar melanggar *referential integrity* dan *global conjunctive constraints.* Namun dengan cara *distributed nested transactions*, dengan adanya eksekusi dan parallelism, integritas dapat lebih terjamin[14].
10. Penelitian ini dilakukan oleh Allison L. Powell James C.dkk, Perancis Departemen Ilmu Komputer Universitas Virginia, berjudul berjudul *"The Impact of Database Selection on Distributed Searching"*. Penelitian ini menjelaskan bahwa *distributed searching* terdiri dari 3 bagian yaitu *database selection, query processing*, dan *results merging*. Cukup beberapa *database* yang dijadikan *database* seleksi (tidak semuanya) dan performa akan meningkat cukup signifikan. Bila seleksi *database* dilakukan dengan baik, pencarian secara *distributed* akan berkinerja lebih baik dibandingkan pencarian secara sentralisasi. Pencarian *database* juga ditambahkan proses seleksi dan ranking sehingga secara potensial meningkatkan efektifitas pencarian data[15].

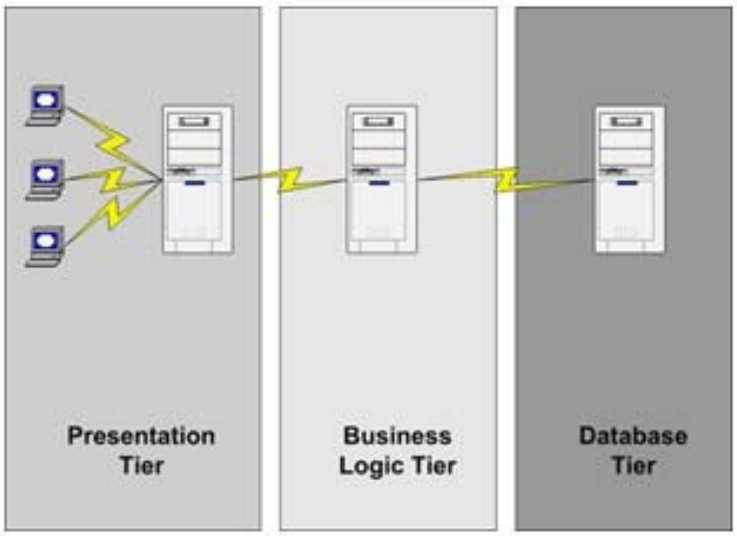
11. Penelitian ini dilakukan oleh Yin-Fu Huang dan HER JYH-CHEN (2001) dari Universitas Nasional Sains dan Teknologi Yunlin Taiwan, berjudul "*Fragment Allocation in Distributed Database Design*". Pada penelitian ini menjelaskan mengenai *Wild Area Network* (WAN), fragmen alokasi adalah isu utama dalam distribusi *database* desain karena kekhawatiran kinerja keseluruhan didistribusikan pada system *database*. Disini system yang diusulkan sederhana dan modelnya yang komprehensif mencerminkan aktivitas transaksi yang didistribusikan dalam *database*. Berdasarkan model dan informasi transaksi, dua bentuk algoritma dikembangkan untuk mendapatkan alokasi yang optimal seperti total biaya komunikasi yang sebisa mungkin diminimalkan. Hasilnya menunjukkan bahwa alokasi fragmentasi ditemukan dengan menggunakan algoritma yang tepat akan menjadi lebih optimal. Beberapa penelitian juga dilakukan untuk memastikan bahwa biaya rumus dapat benar-benar mencerminkan biaya komunikasi didunia nyata[16].

Dari sebelas *Literature Review* yang ada, telah banyak penelitian mengenai arsitektur *Three-tier* disamping itu juga telah ada pembahasan mengenai *Distributed Database*. Untuk penelitian terkait keamanan *Distributed Database* belum banyak dilakukan oleh para peneliti diatas. Ada juga peneliti yang khusus terkait keamanan database, seperti penelitian yang dilakukan Steven P. Coy dari *University of Maryland* menjelaskan bahwa *Distributed Database*, *Relational Model* lebih unggul dibidang sekuritas. Namun demikian dapat disimpulkan pula secara keseluruhan bahwa secara khusus belum ada pembahasan seputar masalah keamanan *Distributed Database*.

2. Pemecahan Masalah

Untuk mengatasi masalah diatas, maka dibutuhkan suatu konsep arsitektur *Three-tier*[17] yang diterapkan pada server iRME. Dengan konsep *Three-tier* tersebut memungkinkan adanya pemisahan secara fisik antara *database tier* dan *application tier* pada server yang berbeda sehingga diharapkan dapat meningkatkan keamanan pada sisi database. Arsitektur *Three-tier* merupakan inovasi dari arsitektur *client/ server*. Pada arsitektur *Three-tier* ini terdapat *application server* yang berdiri di antara *client* dan *database server*. Contoh dari *application server* yang adalah IIS (*Internet Information Services*). *Application Server* umumnya berupa business process layer, dimana yang dikembangkan saat ini menggunakan ASP (Active Server Pages). Sehingga dapat menempatkan beberapa *business logic* pada *tier* tersebut.

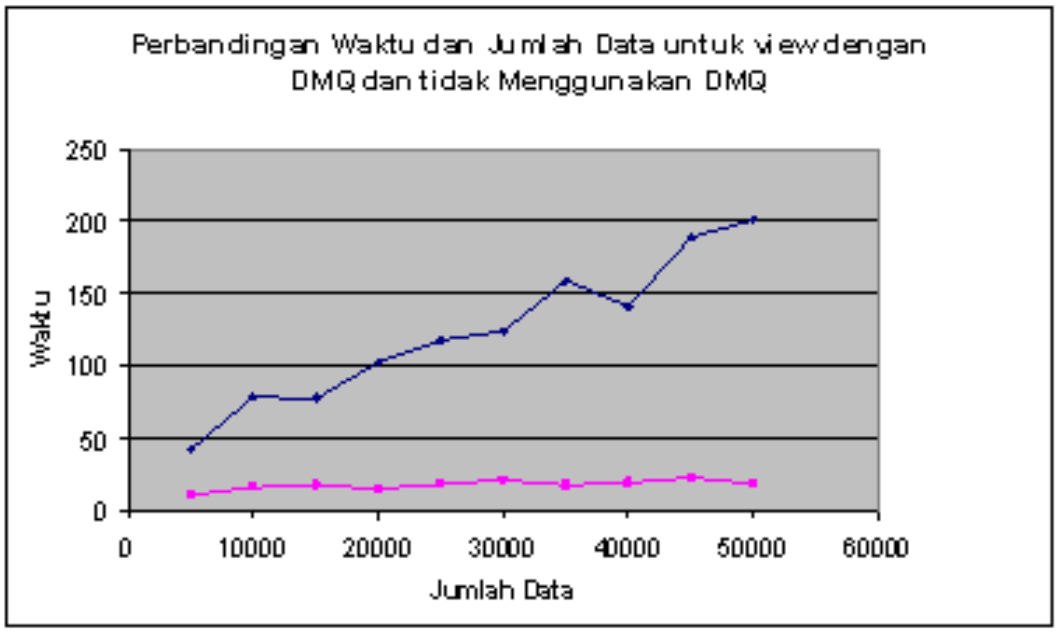
Arsitektur *Three-Tier* ini banyak sekali diimplementasikan dengan menggunakan *Web Application*. Karena dengan menggunakan *Web Application*, *Client Side* (Komputer Client) hanya akan melakukan instalasi Web Browser. Dan saat komputer *client* melakukan inputan data, maka data tersebut dikirimkan ke *application server* dan diolah berdasarkan *business process*-nya. Selanjutnya *application server* akan melakukan komunikasi dengan database server.Karena aplikasinya berbasiskan web, maka *application server* selalu mengirimkan Web *application*-nya ke komputer *client*. *Application server* ditempatkan pada sisi *client* dan hanya mengirimkan data ke dalam *database server*. Arsitektur *Three-Tier* adalah model yang membagi fungsionalitas ke dalam lapisan-lapisan, aplikasi-aplikasi mendapatkan skalabilitas, keterbaharuan, dan keamanan.



Gambar 5. *Architecture Three-Tier*

Arsitektur *Three-Tier* memiliki tiga tingkatan seperti berikut: 1) *Presentation Tier*, ini adalah tingkat paling atas aplikasi. *Presentation tier* menampilkan informasi yang berkaitan dengan layanan berupa hasil output ke *browser*/*Presentation Tier* dan semua tingkatan lain dalam jaringan. Namun ada kendala pada *Presentation Tier* jika pada *distributed database* memungkinkan suatu sistem menjadi lebih kompleks, karena banyaknya *database* yang tersebar dan jumlah data yang banyak dan terus meningkat didalam suatu organisasi maupun perusahaan. Jika suatu *database* memiliki sejumlah data yang tersimpan dengan banyak *query* dan tabel, suatu permintaan mengakibatkan proses pencarian data atau *source data* menjadi lambat. Selain itu banyaknya user yang mengakses hasil out menggunakan *web display* menyebabkan sistem menjadi lambat. Untuk permasalahan ini digunakan metode DMQ (*Data Mart Query*)[18]. DMQ merupakan metode yang menerapkan analogi "*Waste Space for Speed*". DMQ juga merupakan salah satu metode yang berbentuk

terhadap pemisahan antara "*Engine*" dan "*Display*". Dengan kata lain metode DMQ dapat langsung menampilkan *source code* pada *display* dan proses *query* yang dikerjakan pada *engine*. Secara umum DMQ menghasilkan sebuah *display data* yang jauh lebih cepat dibandingkan dengan menggunakan metode umum, karena DMQ tidak melakukan proses lagi dalam menampilkan data. Dan akhirnya DMQ merupakan suatu solusi yang dapat membantu kebutuhan *user* pada proses *display data* yang sebelumnya sangat lambat dan tidak efesien. Pada *Data Mart Query* sumber data berasal dari tabel. Jadi pada proses DMQ ini, mengalokasikan seluruh data yang dipilih ke dalam suatu tabel. Sehingga *user* tidak perlu memikirkan pembuatan struktur tabel tujuan, yang perlu dipikirkan hanyalah dimana data tersebut berada. DMQ ini digunakan untuk menghindari penggunaan *Query* majemuk.



Keterangan :

= Tidak menggunakan DMQ

= Menggunakan DMQ

Grafik 1. Perbandingan Waktu dan Jumlah data

DMQ akan mengorbankan besarnya kapasitas penyimpanan data (*space harddisc*) untuk meningkatkan kecepatan (*increase speed*). DMQ membutuhkan *triger update data* untuk menghasilkan data yang mutakhir. 2) *Application Tier*, Sebuah fungsi aplikasi dengan melakukan pemrosesan terperinci. 3) *Data Tier*, Tingkatan ini terdiri dari Database Server. Berikut informasi disimpan dan diambil. *Data Tier* tetap netral dan independen dari aplikasi server atau logika bisnis. Memberikan *Data Tier* sendiri juga meningkatkan skalabilitas dan kinerja. Arsitektur *Three-Tier* memiliki kelebihan diantaranya yaitu: 1) Segala sesuatu mengenai *database* terinstalasikan pada sisi *server*, begitu pula dengan pengkonfigurasiannya. Hal ini membuat harga yang harus dibayar lebih kecil, 2) Apabila terjadi kesalahan pada salah satu lapisan tidak akan menyebabkan lapisan lain ikut bermasalah misalnya terjadi *crash* pada *database tier* maka tidak berpengaruh pada *application tier*, 3) Perubahan pada salah satu lapisan tidak perlu menginstalasi ulang pada lapisan yang lainnya dalam hal ini sisi *server* ataupun sisi *client,* 4) Skala besar, 5) Keamanan dibelakang *firewall*, 6) Transfer informasi antara web server dan server database optimal, 7) Komunikasi antara sistem tidak harus didasarkan pada standar internet, tetapi dapat menggunakan protocol komunikasi yang lebih cepat dan berada pada tingkat yang lebih rendah, 8) Penggunaan *middleware* mendukung efisiensi query database dalam SQL di pakai untuk menangani pengambilan informasi dari database.

Arsitektur ini memiliki potensi untuk diterapkan backup server jika *Application Tier* mengalami masalah. Hal ini dapat mengurangi resiko hilangnya data pada database server yang diakibatkan oleh kerusakan hardware komputer, bencana alam, kesalahan user, virus, dan lain sebagainya. Ada beberapa jenis backup [19] yang tersedia pada SQL Server yang memiliki potensi untuk diterapkan pada iRME misalnya dengan Full Backup, Differential Backup, Incremental Backup, dan Transaction Log Backup. Memungkinkan diterapkan juga *scheduled task* untuk menjalankan proses backup ini secara otomatis oleh SQL Server. Selain itu dapat juga diterapkan mirroring (replikasi dan duplikasi) pada server iRME sebagai pengembangan lanjutan. Teknik replikasi [20] yang digunakan dengan *Sync*. *Sync* pada dasarnya mekanisme koordinasi proses-proses konkuren yang saling mempengaruhi satu sama lain agar pemakaian resource secara bersama dapat menjalin validitasnya. Ada 2 (dua) hal yang melatar belakangi penggunaan *sync* yaitu 1). pengaksesan yang dilakukan bersama-sama ke data yang sama sehingga data menjadi tidak konsisten, 2). *race condition* yaitu situasi dimana beberapa proses pengakses dan memanipulasi data secara bersamaan. Nilai akhir data tergantung dari proses mana yang selesai terakhir. Pada *sync* perangkat keras, terdapat proses *disabling interrupts* untuk menyelesaikan bounded-buffer harus menggunakan *round robin* memerlukan kode yang dibuat disekitar instruksi lock.

```
while (test_and_set (lock));
boolean bwaiting [N];
int j; /* Takes on values from 0 to N - 1 */
boolean key;
do{
      waiting [i] = TRUE;
      key = TRUE;
      While ((waiting[i] && key )
            Key = test_and_set (lock); /* Spin lock */
      waiting[i] = FALSE;

/****** CRITICAL SECTION ******/
j = (i + 1) mod n;
While ((j!=1) &&  (!waiting[j]))
j = (j + 1) % n;
If (j==1) //using hardware
lock = FALSE;  // Test_and_set
Else
Waiting [j] = FALSE;

/****** REMINDER SECTION *******/
} while (TRUE);
```

Algoritma 1. *Sync* Perangkat Keras

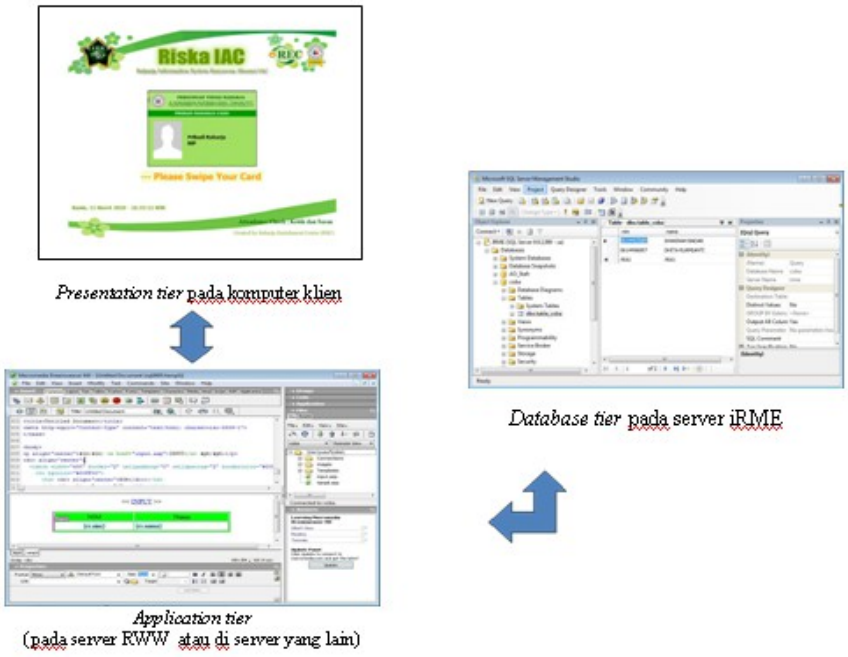
Selain *sync* pada perangkat keras, dapat pula *sync* diterapkan pada perangkat lunak, artinya ketika pembuatan perangkat lunak berjalan langsung diberikan kode *sync*, dengan pendekatan *low-level primitives:semaphore*[21] menggunakan operasi down.

```
type semaphore = integer,
procedure down (var: semaphore);
begin
s:=s-1;
if s <= 0 then
begin
tempatkan antrian pada antrian untuk semaphores
proses diblocked
End;
End;
```

Algoritma 2. *Sync* Perangkat Lunak

3. Implementasi

Untuk mengimplementasikan Arsitektur *Three-Tier*, maka disiapkan satu server yang secara khusus menangani *Database Tier* saja. Tahapan yang dilakukan adalah melakukan migrasi database dari server RME ke server iRME. Selanjutnya untuk migrasi program aplikasi, dapat lebih fleksibel ditempatkan di berbagai server misalnya pada server RWW (Raharja Wide Web), server REC, server REC2 atau ditempatkan di berbagai komputer *client* yang dapat juga dijadikan sebagai backup server.



Gambar 6. Penerapan Arsitektur *Three-Tier* pada Server iRME

Pada gambar 6 diatas dijelaskan bahwa ada pemisahan secara fisik antara *Application Tier* pada server RWW atau server lainnya dengan *Database Tier* pada server iRME. Jika terjadi masalah pada *Application Tier* tentu tidak terlalu khawatir akan kehilangan data karena *Application Tier* ini bisa ditempatkan di server mana saja asalnya terkoneksi dengan jaringan yang ada dilingkungan intranet, jika sewaktu-waktu server mengalami masalah. Masalah tersebut misalnya serangan virus/worm/trojan baik pada server atau jaringan, kerusakan pada harddisk, gagal *booting* akibat sistem *crash*, atau bahkan sampai pada kehilangan server secara fisik akibat bencana alam seperti gempa bumi, bencana banjir atau tindakan pencurian. Dengan pemisahan *tier* ini diharapkan dapat meminimalisasi kerugian yang ditimbulkan serta dapat meningkatkan keamanan sistem database.

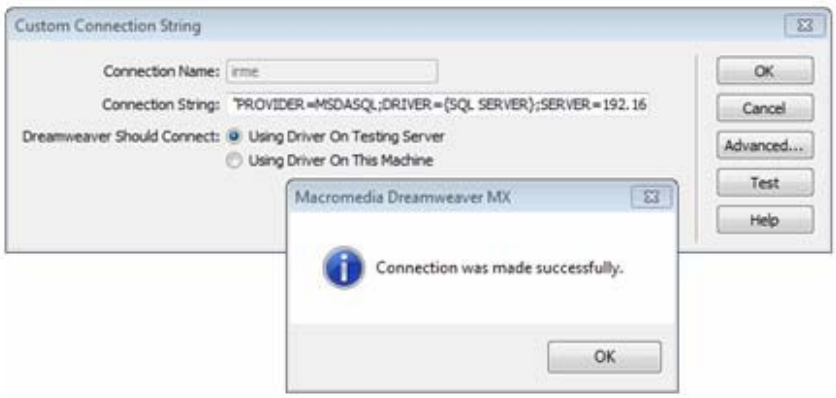
Selanjutnya untuk menghubungkan antara *Application Tier* ke *Database Tier* server iRME digunakan untuk koneksi provider. Koneksi provider yang digunakan sebagai drivernya adalah Microsoft SQL Server. Berikut ini adalah *connection string* provider sebagai berikut:

```
"PROVIDER=MSDASQL;DRIVER={SQL SERVER};SERVER=ServerName;
DATABASE=DatabaseName;USER ID=Username;PASSWORD=PasswordName"
```

Listing 1. Connection String Provider

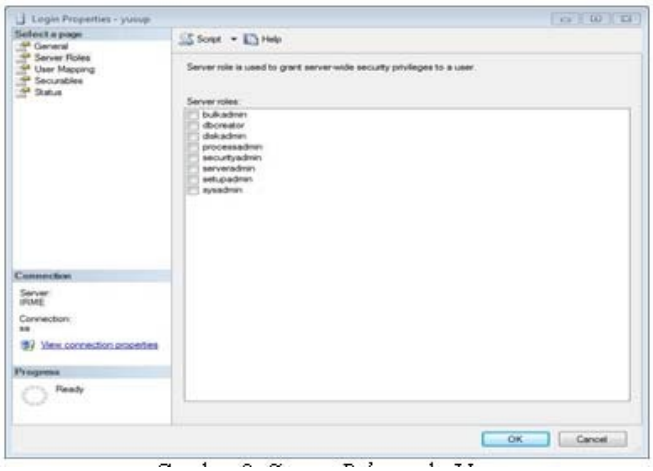
Setiap *user* yang terhubung ke *Database Tier* harus mempunyai hak akses *security login* berupa *login name* dan *password* menggunakan *SQL Server Authentication* bersamaan dengan default database yang diterapkan pada server

iRME. Untuk login ke dalam *Database Tier* server iRME, status login *enable/ disabled* dapat diatur dengan baik, serta dalam hal pengaturan akses *user* untuk koneksi ke database engine dengan status *grant* dan *deny* dapat juga dikelola dengan baik.



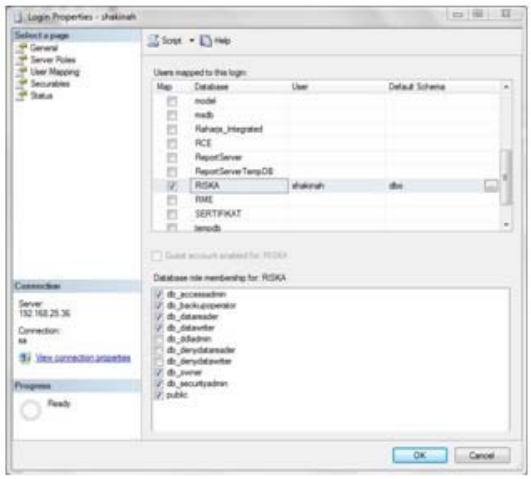
Gambar 7. Testing server dari *Application Tier* ke Server iRME

Setiap user yang akan mengakses database iRME, diberikan juga ketentuan *server roles* pada *user*. *Server roles* pada *user* merupakan aturan-aturan yang dapat diterapkan dan diberlakukan untuk *user* yang mengakses server iRME. Pada folder *server roles* akan tampil daftar-daftar *server roles*, yaitu: 1). Bulkadmin dapat menjalankan perintah *Bulk Insert,* 2). Dbcreator dapat menjalankan perintah *Create*, *Alter*, *Drop* dan *Restore* pada semua database. 3). Diskadmin dapat mengelola file disk. 4). Processadmin dapat menghentikan proses yang berjalan pada database. 5). Securityadmin dapat mengaturi login masing-masing propertisnya seperti *grant*, *deny*, dan *revoke* pada level server. 6). Serveradmin dapat mengubah opsi-opsi pengaturan pada server dan mematikan server database. 7). Setupadmin dapat menambah dan mengapus *link server* dan dapat menjalankan system *stored procedure.* 8). Sysadmin dapat melakukan seluruh aktifitas pada database.



Gambar 8. *Server Roles* pada *User*

Setiap *user* yang melakukan akses ke *Database Tier* server iRME juga diberikan aturan *User Mapping*. *User Mapping* berfungsi untuk menentukan database dan hak akses apa saja yang dapat digunakan oleh *user*. Administrator Server iRME dapat membatasi *user* untuk melakukan akses ke dalam database tertentu. *User Mapping* dengan aturan pembatasan *user* pada Server iRME terdiri dari: 1) db_accessadmin, *user* dapat melakukan perintah *alter* pada *user*, *create schema* dan dapat melihat database. 2) db_backupoperator *user* dapat melakukan *backup database*, *backup log*, *check point* dapat melihat database. 3) db_datareader *user* dapat membaca perintah *select* dan dapat melihat database. 4) db_datawriter *user* dapat menambah, menghapus dan mengupdate data dan melihat daftar database. 5) db_ddladmin *user* dapat menggunakan perintah-perintah untuk membuat *procedur*, *view schema* dan lain-lain. 6) db_denydatareader *user* tidak dapat membaca data dengan perintah *select*. 7) db_denydatawriter *user* tidak dapat menggunakan perintah *delete*, *insert*, dan *update*. 8) db_ owner *user* dapat menggunakan perintah *grant* dengan opsi *control*. 9) db_securityadmin *user* dapat menggunakan perintah *alter application, role*, dan *create schema*, dan *view definition*.



Gambar 9. *User Mapping* pada Server iRME

SIMPULAN

Berdasarkan uraian diatas bahwa keamanan *Database Tier* pada arsitektur *Two-Tier* memiliki beberapa kelemahan sehingga perlu diterapkannya arsitektur *Three-Tier*. Dengan diterapkannya konsep *Three-Tier* pada server iRME di lingkungan Perguruan Tinggi Raharja dapat mengoptimalkan keamanan pada sisi database sever. Konsep ini juga memiliki potensi untuk diterapkannya *Backup Server* jika *Application Tier* mengalami masalah, dan memungkinkan juga diterapkan *mirroring* pada server iRME sebagai pengembangan lanjutan.

PUSTAKA

1. Untung Rahardja, *Laporan Penelitian Student Information Services (SIS) Pada Perguruan Tingi Raharja*, Tangerang: Raharja Enrichment Centre, Perguruan Tinggi Raharja, 2007.

2. Untung Rahardja, *Laporan Penelitian Raharja Multimedia Edutainment (RME) Pada Perguruan Tingi Raharja*, Tangerang: Raharja Enrichment Centre, Perguruan Tinggi Raharja, 2007.

3. Lilik Agustin, *Desain dan Implementasi INTEGRAM pada Perguruan Tinggi Raharja*, Tangerang: STMIK Raharja, 2005.

4. Untung Rahardja, *Laporan Penelitian Green Orchestra (GO) Pada Perguruan Tingi Raharja*, Tangerang: Raharja Enrichment Centre, Perguruan Tinggi Raharja, 2008.

5. Brandon Salmon, Eno Thereska, Craig A.N. Soules, Gregory R. Ganger, *A Two Tiered Software Architecture for Automated Tuning of Disk Layouts*, Pennsylvania: Carnegie Mellon University, 2003.

6. Allen. B. B., Otis Allen., Soucie. S. M., Schuchardt. B., Venkatesh. R. *Persistent Java Objects in Three-Tier Architectures.* Gemstone Systems Inc. 1998.

7. Lieven Desmet,. Bart Jacobs., Frank Piessens., Wouter Joosen. *A Generic Architecture for Web Applications to Support Threat Analysis of Infrastructural Components*. Belgium : DistriNet Research Group. 2004.

8. Roberto Baldoni, Carlo Marchetti, Alessandro Termini. *Active Software Replication through a Three-tier Approach.* Italia : University La Sapienza Roma. 2002.

9. Lin. J. L., Dunham M. H. and Nascimento M. A. *A Survey of Distributed Database Checkpointing.* Texas: Department of computer science and engineering, Shoutern Methodist University. 1997.

10. DeWitt. D.J., Gray.J. *Parallel Database Systems: The Future of High Performance Database Processing.* San Francisco: Computer Sciences Department, University of Wisconsin. 1992.

11. Mitchell Carolyn. *Component of a distributed database.* Department of Computer science, Norfolk state University. 2004.

12. Hamidah Ibrahim. *Deriving Global And Local Integrity Rules For A Distributed Database.* Departement of Computer Science Faculty of Computer Science and Information Technology, University Putra Malaysia 43400 UPM Serdang. 2001

13. Steven P Coy. *Security Implications of the Choice of Distributed Database Management System Model: Relational Vs Object Oriented.* University of Maryland. 2008.

14. Stephane Gangarski, Claudia Leon, Hurbert Naacke, Marta Rukoz and Pablo Santini. *Integrity Constraint Checking In Distributed Nested Transactions Over A Database Clustur.* Laboratorie the Information Paris 6. University Pierre et Marie Curie 8 rue du Capitaine Scott, 75015, Paris. Centro de Computacion Paralela Y Distribuida, Universidad Central de Venezuela. Apdo. 47002, Los Chaguaramos, 1041 A, Caracas, Venezuela. 2006.

15. Allison L. Powell, James C. French, Jamie Callan, Margaret Connell and Charles L. Viles. *The Impact of Database Selection on Distributed Searching. 23rd ACM SIGIR Conference on Information Retrieval (SIGIR'00)*, pages 232-239, 2000.

16. Huang Yin-Fu dan JYH-CHEN HER. *Fragment Allocation in Distributed Database Design*. Nasional Yunlin Universitas Sains dan Teknologi Yunlin. Taiwan 640, R.O.C. 2001.

17. R. Baldoni , C. Marchetti. *Software Replication in Three-Tiers Architectures: is it a real challange?,* Italy: Universit'a di Roma "La Sapienza". 2003.

18. Rangga Praduwiratna. *Mengenal Jenis Backup pada SQL Server 2005*. www.Ilmukomputer.com. 2007.

19. Untung Rahardja, Retantyo Wardoyo, Shakinah Badar, *Application of Data Mart Query (DMQ) in Distributed Database System*, Proceeding International Conference on Creative Communication and Innovative Technology (ICCIT), Tangerang, pp. 109 - 118, Agustus 2009.

20. Cinny, Morrly. *Sinkronisasi Pada Database MySQL dan SQL Server Dengan Menggunakan Teknik Replikasi*, Jakarta: Pasca Sarjana Universitas Gunardarma. 2006.

21. Singhal, Mukes., R. Mahendra., *Distributed Semaphore*, Ohio: Department of Computer Science of The Ohio State University. 1994