24548/H/06

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

PEMBANGUNAN PERANGKAT LUNAK ANTAR MUKA GRAFIS SCHEMA DAN CUBE EDITOR MS ANALYSIS SERVICE BERBASIS WEB

TUGAS AKHIR

RSIF
005.1
Nin
P-1
2005

PERPUSTAKAAN
ITS
Tgl. Terima | 6-8-2005
Terima Dari | H
No. Agenda Prp. | 222985

Disusun Oleh :

WIDYANTI KARTIKA NINGSIH

NRP. 5101.100.004

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2005

PEMBANGUNAN PERANGKAT LUNAK ANTAR MUKA GRAFIS SCHEMA DAN CUBE EDITOR MS ANALYSIS SERVICE BERBASIS WEB

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Siti Rochimah, MT.	Faizal Johan Atletiko, S.Kom.
NIP. 132.103.631	NIP. 132.300.414

Surabaya, Juli 2005

ABSTRAK

Aplikasi Web telah menjadi bagian yang tidak terpisahkan lagi dalam kehidupan sehari-hari. Seiring dengan meningkatnya teknologi internet membuat seorang administrator tidak perlu berada didepan komputer server untuk dapat melakukan pekerjaan administratif seperti membuat analisa proses secara online dari sebuah schema database untuk dibuat schema cube (data warehousing) dan melakukan perubahan pada schema yang sudah dibuat.

Untuk dapat melakukan hal tersebut SQL Server 7.0 maupun SQL Server 2000 telah menyediakan fasilitas pengaksesan tabel dimensi, cube, relasi dan koneksi database dalam Analysis Manager menggunakan DSO (Decision Support Object) library component MS OLAP. Dengan adanya DSO connection, seorang administrator dapat memanipulasi schema, tabel, cube, dimension, relasi suatu schema cube dari database SQL Server secara langsung dengan menggunakan bahasa pemrograman yang mendukung, seperti MS Visual Basic, VBScript, MS Visual C++, MS.Net.

Dalam Tugas Akhir ini dibuat aplikasi berbasis web yang merupakan tahap awal dari OLAP (Online Analytical Processing) suatu schema database dengan menggunakan SQL Analysis Service, Analysis Manager yang memungkinkan seorang administrator database untuk dapat melakukan pekerjaan administratif khususnya membuat schema dan cube (fact table) dari database yang dianalisa, membuat dan mengedit dimensi, dimana saja tanpa harus berada di komputer server.

KATA PENGANTAR

Segala puji dan syukur semata ditujukan ke hadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir berjudul "**Pembangunan Perangkat Lunak Antar Muka Grafis Schema dan Cube Editor Ms Analysis Service Berbasis Web**".

Mata kuliah Tugas Akhir dengan beban sebesar 4 satuan kredit disusun dan diajukan sebagai salah satu syarat untuk menyelesaikan program strata satu (S-1) pada jurusan Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan Tugas Akhir ini, Penulis berusaha untuk menerapkan ilmu yang telah didapat selama menjalani perkuliahan dengan tidak terlepas dari petunjuk, bimbingan, bantuan, dan dukungan berbagai pihak.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan dan mengandung kekurangan di sana-sini sehingga dengan segala kerendahan hati Penulis masih dan insya Allah akan terus mengharap saran serta kritik yang membangun dari rekan-rekan pembaca.

Surabaya, Juli 2005

UCAPAN TERIMA KASIH

Dengan mengucapkan syukur Alhamdulillah kepada Allah SWT, pada kesempatan ini Penulis hendak menyampaikan penghormatan dan terima kasih kepada pihak-pihak yang telah memberi bantuan baik itu berupa moril maupun material secara langsung maupun tidak langsung kepada :

1. Ibunda dan Ayahanda tercinta, yang telah dengan sabar merawat, membesarkan, mendidik dan menyayangi penulis sejak sebelum dilahirkan hingga saat ini, dan pasti tetap berlanjut sampai akhir hayat. Tak akan ada yang bisa membalas selain Allah SWT. Serta kakak-kakakku tersayang, Mas Hanif, Mbak Dwi ,Mas Warno, dan Mas Luluk yang selalu mewarnai setiap hari-hari Penulis.
2. Ibu Siti Rochimah, S.Kom, M.Kom sebagai pembimbing I yang selalu sabar membimbing dan menghadapi Penulis.
3. Bapak Faisal Johan Atletiko, S.Kom sebagai pembimbing II yang sekaligus menjadi kakak dan saudara yang baik bagi Penulis.
4. Bapak dan Ibu dosen Jurusan Teknik Informatika – ITS; Pak Fajar, Pak Ruly, Pak Yudhi,Pak Suhadi Lili, Pak Joko, Pak Pak Imam Kuswandayan, Pak Darlis, Bu Ika, Bu Sarwosri, Bu Esther, Bu Ani, Bu Bilqis, Pak Irvan, Pak Agus, Pak Aris, Pak Arif dan para dosen lainnya yang telah berkenan memberikan ilmunya selama Penulis menempuh kuliah.
5. Seluruh staf dan karyawan Jurusan Teknik Informatika – ITS; Mas Yudi, Mas Sugeng, Pak Mu'in, Mbak Fathin, Mbak Eva, Mbak Erna, Pak Karmono, Pak Satpam malam dan para karyawan lainnya.

6. Keponakan-keponakanku yang cakep dan lucu, adik Ilham dan adik Dzaki semoga kalian menjadi anak yang sholeh dan sholihah yang berbakti kepada orang tua dan agama, serta adik-adikku yang baik, dik Ulfa dan Esti.
7. Mas-mas senior yang baik dan sabar, mas Ari Afian, mas Ketchu alias Indra99, serta mas Rono Toga terimakasih telah membantu Penulis sampai detik-detik terakhir 'penentuan' serta untuk segalanya. Hanya Allah yang mampu membalasnya.
8. Sahabat-sahabatku Lufi, Hathfi, Dhewy, Ossy, Melinda, Eddy, Rinda, Yeni, Galuh, Feri Aik, Rizki, Sixteen, Kiky, Robby, Yuni . Semoga persaudaraan ini terus berlanjut sampai kapanpun dan terima kasih atas semua *support* yang diberikan selama ini.
9. Penghuni komputing : Mas Gershom, Mas Cherry, Mas Ray, Mas Indie, Mas Edward, Mbak Puspa, Mas Joko, Adik Asnita, Dimas "Sangeh", mas Hoirul. Makasih atas *support*-nya.
10. Penghuni Lab lainnya : Mas Mumud, mas Ronny, mas Roy, mas Yuonan dan penghuni lainnya yang telah membantu penulis terutama saat-saat pengerjaan TA.
11. Teman-temanku angkatan C11 : Edwin, Wafa, Rolly, Sempal, Phe, Nina, Siti, Dwi, Ria, Bolon, Willy, Rudy, Acong, Bendot, Kenthis, Karpo, bang Deddy, pak Eko, Bulu, bang Ucok, Momon montok, bang Gerar dan semuanya yang nggak bisa disebutkan satu-satu di sini
12. Adik-adik angkatan C12 yang baik, Meiji, Helmi, Habib, Roy, Heri, Santi, Hendra, Huda, Hendry, Surya, Farida, Ratna, Andika, dll

13. Temen-temen yang tidak tahu termasuk golongan yang mana ☺ : Haerul, Dian, Iin, Ervin, dan Bagus, juga yang lainnya jaga terus komunikasi kita n thx 4 all.
14. Penghuni Kos : mbak Yuli, Ita, Likha, mbak Ida, Fitri, Mbak Aya, Abah, mbak Nita, Mbak Anita, mbak Pipit, mbak Puja dan Hikmah yang selalu menemani hari2 penulis di kosan.
15. Rekan-rekan lain yang tidak dapat Penulis sebutkan satu-persatu.

Tiada untaian kata yang cukup yang dapat penulis sampaikan sebagai balasan atas jasa yang penulis terima melainkan hanya harapan semoga Allah SWT membalas semua amal tersebut. Jazakumullah Khairan Katsiran.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL


PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

BAB I

PENDAHULUAN

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

SQL Server telah menyediakan komponen untuk membangun dan mengelola data yang digunakan dalam proses analisa online (*Online Analitycal Processing* atau lebih dikenal dengan OLAP). *Analysis Service* terdiri dari *server Analysis Service*, *English Query*, dan komponen pendukung lain. *Analysis Service* membangun *cube* dari data untuk membantu analisa multidimensi. Istilah "*cube*" digunakan untuk mendeskripsikan agregasi data. Dengan cara agregasi ini dapat diambil kesimpulan dari data yang digunakan untuk analisa query yang kompleks seperti hasil penjualan per bulan dan proyeksi penjualan. Adapun tools yang disediakan oleh *SQL Server* untuk dapat menggunakan fasilitas tersebut dinamakan *Microsoft Analysis Services*, *Services Manager*. Untuk dapat menggunakan fasilitas tersebut terlebih dahulu harus dibuat ODBC (*Open Database Connectivity*), data *source* dan menentukan *cube* atau *fact table* serta dimensi.

Namun dalam melakukan pembuatan *analysis services* untuk sebuah skema *database* tersebut masih kurang dalam pemanggilannya. Dimana pengguna hanya bisa membuat proses analisa pada computer yang ter-*install* aplikasi komponen *database* SQL *Server* baik *server* ataupun klien. Untuk itu dengan memanfaatkan sistem arsitektur terbuka dari SQL *Server* dapat dibuat suatu antar muka grafis berbasis web yang fleksibel dan optimal untuk menggantikan tugas

Cube Editor pada *Analysis Service* dalam pembuatan *cube* dan *schema*. Sehingga seorang administrator *database* dapat melakukan tugas-tugas *administrative* seperti membuat koneksi, membangun *cube(fact table)*, dimensi dan *schema database* dimanapun dia berada melalui jaringan internet.

Terdapat beberapa cara untuk membuat antar muka berbasis web, diantaranya adalah dengan menggunakan *Macromedia Flash MX*. Pada aplikasi ini dicoba untuk menerapkan *Flash MX* untuk sebuah aplikasi berbasis web yang cukup interaktif. Karena pengguna berinteraksi langsung dengan aplikasi klien yang kemudian diteruskan ke sisi *server* untuk diproses. Teknologi *Flash MX* mendukung koneksi dengan bahasa pemrograman lainnya seperti ASP, PHP, CF, XML, dll. Pada aplikasi ini , *Flash MX* akan berhubungan dengan ASP sebagai bahasa *server side*-nya melalui XML.

1.2 PERMASALAHAN

Permasalahan yang diangkat dalam tugas akhir ini adalah :

1. Bagaimana membuat *Flash Movie* yang dilengkapi *ActionScript* dengan menggunakan *Macromedia Flash MX* dapat berkomunikasi dengan *Library* SQL *Server* ?
2. Bagaimana membuat sistem komunikasi antara *Flash* dan *ASP* ?
3. Bagaimana memanfaatkan DLL yang telah disediakan oleh *SQL Server* untuk melakukan pembuatan skema dan *cube* pada *analysis manager* secara *remote* melalui web?

1.3 BATASAN MASALAH

Dari permasalahan di atas, maka batasan dalam tugas akhir ini adalah:

1. Pembuatan aplikasi yang dapat melakukan pembuatan skema *database* dengan *cube* dan *dimensi* yang meliputi koneksi, query, maupun skenario untuk sebuah *Online Analytical Processing* dimana skema untuk dimensinya hanya meliputi *star schema* dan perubahan yang bisa dilakukan meliputi *remove* skema *cube* dan menambah tabel dimensi untuk skema *cube* yang telah dibuat.
2. Pengembangan aplikasi menggunakan *Macromedia Flash MX* sebagai antar muka grafis.
3. Untuk menghubungkan DLL pada *Ms SQL Database Server* dan *Analysis Services* dengan *Macromedia Flash MX* digunakan *Microsoft Active Server Pages (ASP)* dan DLL *user*.
4. Data *Source* yang digunakan hanya yang terdapat pada SQL *Server Enterprise Edition* 2000 .
5. Sistem keamanan dalam komunikasi *client server* merupakan topik yang lepas dari aplikasi ini. Proses pengamanan dalam pengiriman data tidak didukung dalam aplikasi ini. Pengamanan hanya berdasarkan pada authentifikasi *Server SQL Server 2000.*

1.4 TUJUAN PEMBUATAN TUGAS AKHIR

Tujuan pembuatan tugas akhir ini adalah untuk membuat aplikasi berbasis web sebagai alternatif editor skema dan cube pada Ms *Analysis Service* dengan menggunakan *Macromedia Flash MX* 2004 dengan memanfaatkan *library* yang

sudah disediakan oleh SQL *Server* yang dapat menggantikan fungsi dan peran editor skema dan *cube* pada komponen *analysis manager* yang terdapat pada *SQL Server*. Sehingga diharapkan pengguna dapat merancang paket pada sisi *client*. Dengan tujuan akhir dapat mempermudah tugas-tugas administrator dalam membuat dan mengedit skema *cube* untuk *datawarehouse* karena dapat dilakukan secara *remote* dan dimana saja tanpa harus meng-*install* SQL *Analysis Service client*.

1.5 METODOLOGI PEMBUATAN TUGAS AKHIR

Metode penelitian yang dilakukan menggunakan langkah-langkah berikut:

- **Studi literatur, survei data dan observasi**

  Hal yang dilakukan pada tahap awal adalah:

  - Mencari dan mempelajari berbagai macam literatur yang berkaitan dengan rumusan masalah, teori-teori yang berhubungan dengan sistem yang akan dibangun, desain sistem, dan bahasa pemrogramannya.
  - Mengadakan analisa data untuk penyusunan program, serta mengumpulkan data yang dibutuhkan untuk diolah.

- **Perancangan aplikasi**

  Pada tahap ini dilakukan perancangan aplikasi dengan berdasarkan analisa data yang telah dilakukan pada tahap pertama.

- **Pengembangan aplikasi**

Pada tahap ini dilakukan pengembangan perangkat lunak menggunakan *Macromedia Flash MX* dengan *Action Script* dan ASP .

- **Uji coba dan evaluasi**

  Pada tahap ini aplikasi telah selesai dibuat dan siap untuk diuji kebenarannya berdasarkan tujuan pembuatan program tersebut.

- **Penyusunan buku tugas akhir**

  Tahap terakhir digunakan untuk penyusunan buku sebagai dokumentasi lengkap dari pelaksanaan Tugas Akhir.

1.6 SISTEMATIKA PEMBAHASAN

Sistematika pembahasan yang digunakan sebagai berikut:

BAB I : PENDAHULUAN

Menjelaskan mengenai latar belakang, permasalahan, batasan masalah, tujuan, metodologi penelitian, dan sistematika pembahasan.

BAB II : TEORI PENUNJANG

Membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dalam Tugas Akhir ini.

BAB III : ANALISA DAN PERANCANGAN PERANGKAT LUNAK

Pada bab ini dibahas mengenai analisa dan perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, perancangan proses dan perancangan antar muka.

BAB IV : IMPLEMENTASI PERANGKAT LUNAK

Pada tahap ini merupakan implementasi terhadap desain yang dilakukan pada tahap analisa dan perancangan perangkat lunak, termasuk didalamnya disertakan script-script yang berperan penting dalam aplikasi.

BAB IV : UJI COBA DAN EVALUASI SISTEM

Aplikasi yang telah selesai ini nantinya juga akan dievaluasi dan diperbaiki demi kelayakan sistem dan keberhasilan dari sistem ini sesuai dengan tujuannya dibangun.

BAB V : PENUTUP

Berisi kesimpulan yang diambil dari hasil uji coba dan saran untuk pengembangan selanjutnya.

BAB II

TEORI PENUNJANG

Pada bab ini akan dibahas mengenai teori yang menunjang dan berhubungan dalam pembuatan dan perancangan perangkat lunak dalam Tugas Akhir. Pembahasan dimulai dengan penjelasan tentang *Analysis Manager* yang merupakan *Analysis Services* yang terdapat pada Ms SQL *Server* 2000.

2.1 SQL Server 2000 Analysis Services, Analysis Manager

Analysis Services merupakan komponen yang terdapat dalam Ms SQL *Server* 2000 yang merupakan *middle-tier server* untuk *Online Analytical Processing (OLAP)* dan Data *Mining*. Dalam tugas akhir ini hanya dibatasi pada OLAP-nya saja, tidak sampai ke data *mining*-nya. Sistem pada *Analysis Services* melibatkan *server* yang menangani multidimensional *cube* dari data untuk analisa dan menghasilkan klien untuk akses informasi dalam *cube*. *Analysis Services* mengorganisasikan data dari sebuah data *warehouse* kedalam *cube-cube* dengan agregasi data yang sudah dikalkulasikan sebelumnya untuk menghasilkan jawaban dari permasalahan komplek dalam *query analitical*. *Analysis Service* juga menyediakan fasilitas untuk membuat model data *mining* dari multidimensional dan relasional data *source*. *PivotTable® Service*, OLE DB *Provider* yang sudah ter-*include*, digunakan oleh *Microsoft Excel* dan aplikasi lainnya untuk melakukan *retrive* data dari *server* dan menampilkannya untuk pengguna, atau membuat data *cube* lokal untuk *offline analysis*.

Analysis Manager memberikan *service* untuk memanipulasi *object-object* pada OLAP termasuk didalamnya adalah :

- Data *source*
- *Cube wizard*
- *Dimension Wizard*
- *Storage mode*
- *Role*
- *Partition*

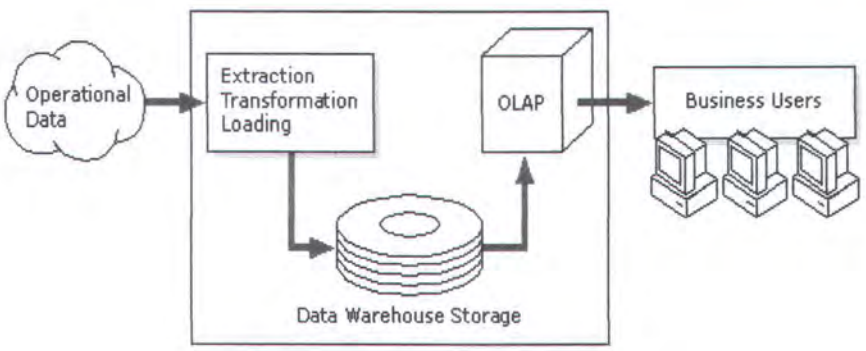
2.2 Data Warehousing dan OLAP

Meskipun terkadang digunakan secara berlawanan, istilah data *warehousing* dan *online analytical processing* (OLAP) diterapkan pada komponen yang berbeda dari sebuah sistem yang lebih mengarah sebagai sistem pendukung pengambilan keputusan (DSS atau *Decission Support System*) atau *bussiness intelligence systems*. Sistem dengan tipe komponen yang seperti ini melibatkan *database* dan aplikasi yang menghasilkan *tool* yang dibutuhkan oleh seorang analis sistem untuk mendukung pengambilan keputusan secara organisasional.

Sebuah *datawarehouse* adalah sebuah *database* yang mengandung data yang biasanya digunakan untuk untuk mereprensetasikan *history* transaksi atau bisnis dari suatu organisasi. Data historis tersebut digunakan untuk analisa yang digunakan sebagai pendukung keputusan bisnis pada beberapa level, dari perencanaan strategis untuk menghasilkan evaluasi dari sebuah organisasi unit diskrit. Data dalam sebuah *datawarehouse* diorganisasikan untuk mendukung

analisa daripada untuk mengolah transaksi *real-time* seperti pada system *Online Transaction Processing* (OLTP).

Teknologi OLAP memungkinkan *datawarehouse* dapat digunakan secara efektif untuk *online analysis* dan menghasilkan jawaban dari analisa iterasi query yang kompleks. Multidimensional data model pada OLAP dan teknik agregasi data, mengorganisasi dan meringkas sejumlah besar dari data sehingga bisa dilakukan evaluasi secara cepat dengan menggunakan analisa *online* dan *tool* grafis. Sistem OLAP menghasilkan kecepatan dan fleksibilitas untuk mendukung analis pada *real-time*.



Gambar II-1 Skema OLAP

Jadi sesungguhnya OLAP adalah teknologi yang memungkinkan aplikasi klien mengakses data *store* (*datawarehouse* untuk analisa data) secara efisien. OLAP memberikan keuntungan untuk pengguna, sebagai contoh :

- Semakin mudah dalam melakukan *select*, navigasi, dan mengeksplore data dari data model multidimensional yang intuitif.
- Bahasa query analisis yang dapat mengksplore relasi data bisnis.

- Pra-kalkulasi untuk query data secara frekuensi menghasilkan waktu respon yang cepat untuk query.

OLAP menghasilkan prosentasi multidimensional dari *datawarehouse*, membuat *cube* yang mengorganisasikan dan merangkum data untuk analisa query yang efisien. Perancangan dari struktur *datawarehouse* dapat mempengaruhi seberapa mudah *cube* tersebut dapat didesain dan dibangun.

Microsoft® SQL *Server*™ 2000 *Analysis Services* membuat data yang dihasilkan pada *datawarehousing* menjadi akurat, stabil, dan mempunyai integritas referensial. Ketika membuat sebuah *datawarehouse* yang digunakan dengan *Analysis Services*, faktor desain berikut ini harus diperhitungkan :

- Diusahakan untuk selalu memakai skema *star*.

  Jika skema *snowflake* dibutuhkan, minimalkan jumlah dari tabel dimensi pada level pertama dari *fact table*.

- Rancangan tabel dimensi untuk *user*.

  Tabel dimensi harus mengandung informasi yang penting tentang *fact table* yang ingin dieksplorasi oleh pengguna.

- Menerapkan normalisasi untuk desain tabel dimensi.

  Data yang tidak memiliki relasi tidak boleh digabungkan ke dalam sebuah tabel dimensi tunggal, dan data tidak boleh diulang pada beberapa dimensi yang berbeda

- Tidak boleh terlalu meringkas isi dari fact table.
- Menggunakan struktur *fact table* yang umum untuk data yang sama.

- Membuat *index* pada *key field.*
- Memastikan *referential integrity.*
- Merancang strategi dalam meng-*update* data.

2.3 Cube (fact table), dimensi dan measure

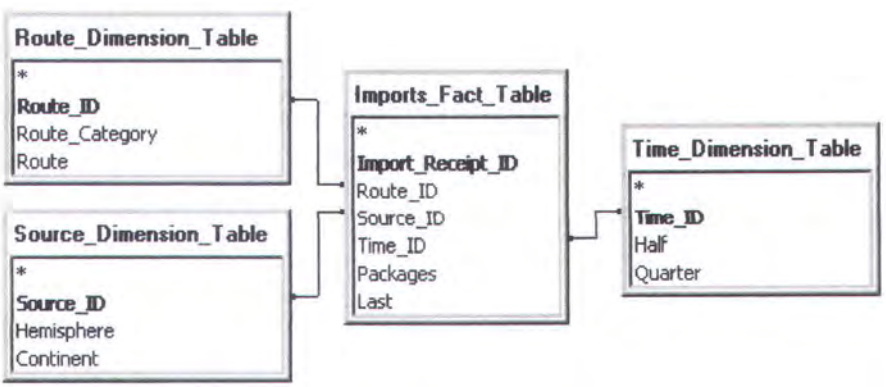
Sebuah *cube* adalah struktur tabel multidimensional yang mengandung dimensi dan *measure* dalam *datawarehouse* yang menyimpan *history* dari data. Dimensi mendefinisikan struktur dari *cube*, sedangkan *measure* menghasilkan nilai *numeric* keuntungan untuk *end user*. Posisi *cell* dalam *cube* didefinisikan oleh interseksi dari anggota dimensi dan nilai *measure* diagregasikan untuk menghasilkan nilai dalam *cell*.

Tabel II-1 Konsep dasar dalam Cube

Topik	Deskripsi
Cube Structure	Detail dari komponen sebuah *cube*, seperti *measure* dan dimensi.
Cube Storage	Mendeskripsikan berbagai teknik dan mode penyimpanan termasuk berbagai penggunaan dari *partition*, *linked cube*, *distributed partitioned cube* dan *real-time cube*.
Cube Processing	Mendeskripsikan proses cakupan *cube*, termasuk informasi pada OLAP *real-time*.
Cube Varieties	Menghasilkan ringkasan *varitis* dari *cube* yang tersedia dalam *Microsoft*® SQL *Server*™ 2000 *Analysis Services*.
Regular Cubes	*Cube* yang berdasarkan tabel , memiliki agregasi sendiri yang menggunakan *space* penyimpanan.
Linked Cubes	*Cube* yang berdasarkan definisi *regular cube* dan disimpan pada *analysis server* yang lain, menggunakan agregasi dari *cube* dasarnya, tidak membutuhkan *space* penyimpanan untuk agregasinya.
Distributed Partitioned Cubes	Merupakan *regular cube* yang *partition*-nya disimpan pada *analysis server* yang lain, menggunakan agregasi *cube* yang bisa berdasarkan *local* atau *remote analysis server* dan lainnya yang dapat diproses berulang kali oleh *analysis server*.

Virtual Cubes	*Logical cube* yang berdasarkan satu atau lebih *regular cube* atau *linked cube*, menggunakan agregasi dari komponen *regular cube* dan *cube* lain yang komponennya digunakan oleh *linked cube*, tidak membutuhkan *space* penyimpanan untuk agregasi.
Local Cubes	*Local cube* berisi file-file yang *portable* dan dapat di-*browse* tanpa harus terhubung dengan analysis server, berdasarkan pada table dan tidak meiliki agregasi.
Real-Time Cubes	*Regular cube* dengan dimensi dan atau *partition*-nya dapat digunakan untuk *real-time* OLAP dan tidak membutuhkan *space* penyimpanan untuk agregasinya.

Contoh struktur sebuah *cube* :



Gambar II-2 Contoh struktur Cube

Sebuah dimensi adalah hirarki yang terorganisasi dari berbagai kategori yang mendeskripsikan data dalam *fact table datawarehouse* atau tabel yang digunakan untuk mendefinisikan *fields* dalam *fact table*, misal: nama penjual, tipe transaksi, atau item barang yang dijual. Dimensi biasanya menggambarkan sekumpulan *set* yang sama dari anggota berdasarkan dasar analisa yang diinginkan oleh pengguna dan merupakan komponen dasar dari *cube*.

Dalam pembuatan OLAP dari suatu *datawarehouse* ada beberapa langkah yang harus diperhatikan dan komponen yang terlibat didalamnya :

- **Dimensi**

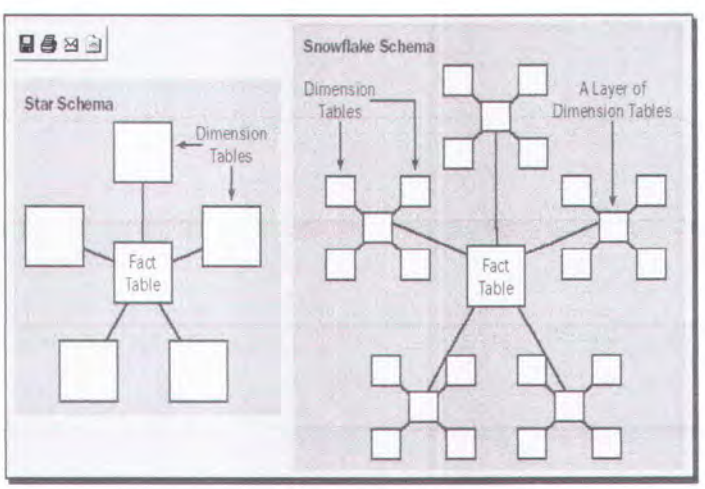
Dalam membuat dimensi dari sebuah *cube* , tipe *source* dari dimensi harus dipilih dari beberapa tipe berikut :

- **Star Schema :**

Star schema merupakan pilihan untuk membuat tipe dimensi yang akan menciptakan sebuah dimensi *regular* berdasarkan tabel dimensi tunggal. *Star schema* lebih sering digunakan dalam *datawarehouse* yang berisi sebuah *fact table* dan beberapa tabel dimensi, yang setiap dimensi berkorespondensi dengan sebuah kolom pada *fact table*.

- **Snowflake Schema :**

Snowflake schema atau skema multidimensional, beberapa tabel dimension digabung ke *fact table*, dengan kata lain beberapa *layer* tabel dimensi dibuat dan setiap *layer* berkorespondensi pada kolom di *fact table*. *Snowflake schema* merupakan pilihan untuk membuat tipe dimensi yang terdiri dari beberapa tabel dimensi. Gambar berikut menunjukkan perbedaan dari *star schema* dan *snowflake schema* :



Gambar II-5 Perbedaan star schema dan sowflake schema

- **Parent Child :**

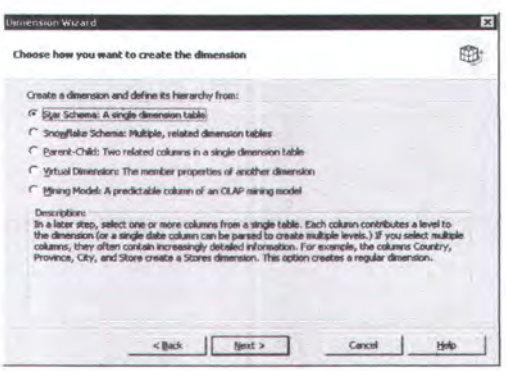
Merupakan pilihan untuk membuat dimensi dengan tingkatan *parent-child* yang didasarkan pada dua kolom dengan tipe data yang sama. Sebagai contoh dalam sebuah table **Employee**, kolom **Employee Number** berisi sebuah *identifier* untuk setiap pegawai dan kolom **Manager Employee Number** berisi jumlah pegawai untuk setiap manager.

- **Virtual dimensi :**

Merupakan pilihan untuk membuat tipe dimensi yang dibuat berdasarkan satu atau lebih anggota *property* dari dimensi lain.

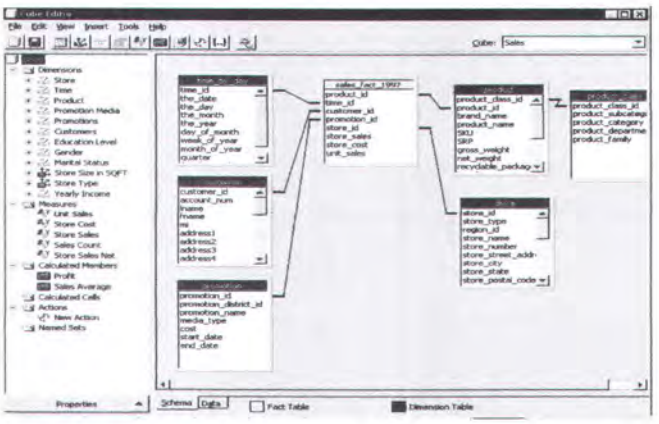
- **Mining model :**

Merupakan pilihan untuk membuat tipe dimensi dari kolom yang dapat diprediksi dari sebuah model OLAP *mining*.



Gambar II-6 Setting pilihan skema dari dimensi

Setelah *cube* dibuat maka partisi dan agregasi adalah *object* berikutnya yang akan dibuat, sehingga hasil eksekusi skema *cube* dapat dilihat dan di-*browse* hasilnya pada jendela Editor Skema dan Cube seperti gambar dibawah ini.



Gambar II-7 Jendela editor skema dan cube pada Analysis Manager

2.4 Microsoft Active X Data Objects (ADO MD)

Microsoft ActiveX Data Objects (*Multidimensional*) (ADO MD) merupakan salah satu cara yang mudah untuk mengakses data multidimensional pada SQL *Server* melalui berbagai bahasa pemrograman seperti *Microsoft Visual Basic*, *Microsoft Visual* C++, and *Microsoft Visual* J++. ADO MD merupakan bagian dari *Microsoft ActiveX Data Objects* untuk meng-*include*-kan *object* yang spesifik kedalam data *multidimensional* seperti *CubeDef* dan *Cellset*. Dengan menggunakan ADO MD, dapat dilakukan *browsing* skema multidimensional, melakukan query sebuah *cube*, dan melihat hasil *output* dari query yang dilakukan. Jadi ADO MD adalah bahasa independen tingkat tinggi untuk mengakses data sekumpulan (*set*) *object* untuk aplikasi data multidimensional. *Visual Basic* dan bahasa pemrograman yang lainnya menggunakan ADO MD sebagai *interface* data *access* untuk penyimpanan data multidimensional.

Seperti halnya ADO, ADO MD menggunakan OLE DB *provider* yang sudah ada untuk mengakses data. Untuk menggunakan ADO MD, *provider* harus merupakan sebuah *provider* data yang multidimensional (MDP (*Multidimensional Data Providers*)) seperti yang sudah didefinisikan oleh OLE DB untk spesifikasi OLAP. MDP menghasilkan data dalam bentuk *multidimensional view* yang merupakan kebalikan dari TDP (*Tabular Data Providers*) yang manghasilkan data dalam bentuk *tabular view*.

ADO dan ADO MD saling berhubungan tetapi, keduanya merupakan model *object* yang terpisah. ADO menghasilkan *object* untuk koneksi ke data *source*, mengeksekusi perintah, *retrieve* data *tabular* dan skema *metadata* dalam format *tabular* serta menghasilkan informasi *error* atau kesalahan dari *provider*. ADO MD menghasilkan object untuk me-*retrieve* data multidimensional dan *viewing* skema *metadata* secara multidimensional.

Ketika menggunakan MDP, dapat dipilih untuk menggunakan ADO, ADO MD atau keduanya untuk aplikasi yang akan dibangun. Dengan menggunakan referensi dari *library* yang ada pada keduanya dalam *project* , dapat dilakukan akses terhadap semua fungsi-fungsi yang telah disediakan oleh MDP.

Untuk menggambarkan data, lebih mudah dengan menggunakan bentuk *tabular view* dari *data set multidimensional*. Hal ini dapat dilakukan dengan menggunakan ADO *Recordset object*, melakukan spesifikasi *Cellset* sebagai *source parameter*.

Tabel II-2 Object yang terdapat dalam ADO MD

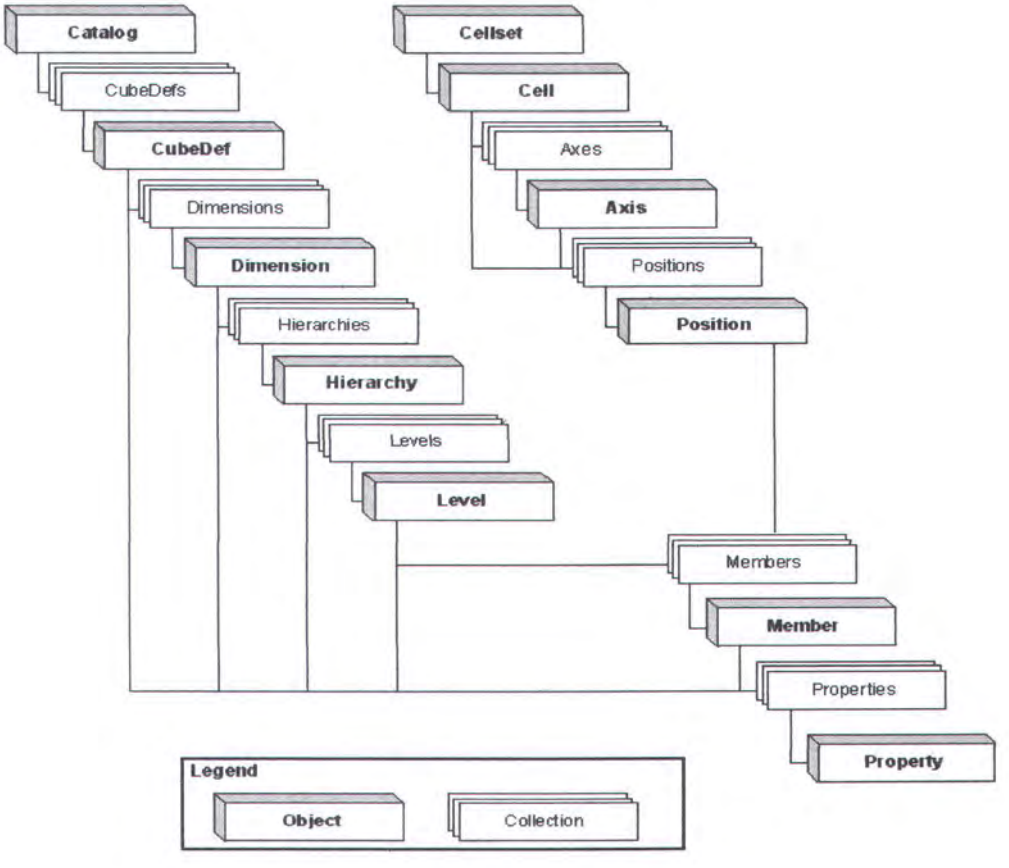
Object	Deskripsi
Axis	Menggambarkan posisi atau *filter axis* dari sebuah *cellset*, berisi member yang dipilih dari satu dimensi atau lebih.
Catalog	Mengandung informasi skema *multidimensional* (termasuk *cube* dan dimensinya, hirarki, level dan member) spesifikasi untuk sebuah data *multidimensional*, khususnya untuk sebuah *Multidimensional Data Provider* (MDP).
Cell	Menggambarkan data pada peropotongan sumbu koordinat, berada dalam sebuah *cellset*.
Cellset	Menggambarkan hasil dari *query multidimensional*. Merupakan sebuah koleksi dari *cell* yang dipilih dari *cube* atau *cellset* yang lain.
CubeDef	Menggambarkan sebuah *cube* dari sebuah skema *multidimensional*, berisi sekumpulan set dimensi.
Dimension	Menggambarkan satu dari sebuah *cube multidimensional*, berisi satu atau lebih hirarki.
Hierarchy	Menggambarkan salah satu cara dimana anggota dari suatu dimensi dapat diagregasikan.
Level	Berisi sekumpulan anggota, yang masing-masing memiliki tingkatan yang sama dengan sebuah hirarki.
Member	Menggambarkan anggota dari sebuah level dalam sebuah *cube*, anak-anak dari anggota level atau anggota dari posisi sepanjang sumbu koordinat dari sebuah *cellset*.
Position	Menggambarkan sekumpulan dari satu atau lebih anggota pada dimensi yang berbeda yang mendefinisikan sebuah titik pada sumbu koordinat.

Selain itu, object **Catalog** terkoneksikan dengan sebuah *object* ADO **Connection**, yang diincludekan dengan standart *library* milik ADO yaitu:

Tabel II-3 Standart Library milik ADO

Object	Deskripsi
Connection	Membuat koneksi dengan sebuah data *source*.

Library Microsoft® ActiveX® Data *Objects* (*Multidimensional*) (ADO MD) berisi sejumlah *object* yang dapat digunakan dengan menggunakan *PivotTable® Service*. *Object Cellset* dapat digunakan untuk melakukan query pada *cube* dalam *Analysis Server* atau file *cube local* (*.cub) dengan menggunakan MDX (*Multidimensional Expressions*). MDX adalah *sintaks* yang digunakan untuk mendefinisikan *object-object* multidimensional, query dan manipulasi data multidimensional. Object **CubeDef** digunakan untuk *retrieve* informasi skema multidimensional. Untuk menggunakan *library* ADO MD secara *independent* dari *library* ADO, dapat digunakan *property* **ActiveConnection** dari *object* **Catalog** atau **Cellset.**



Gambar II-8 Hirarki dari object Catalog dan Cellset

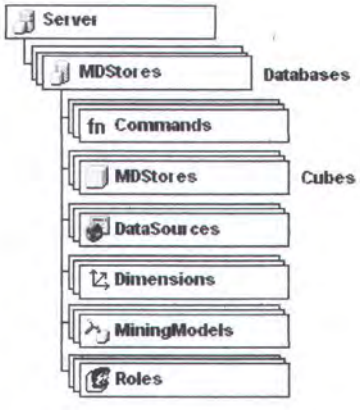
Dalam menggunakan ADO MD *connection* diperlukan *provider* MSOLAP yang merupakan *provider aggregation.*

2.5 DSO (Decision Support Object)

Library DSO (*Decision Support Objects*) dari *Microsoft* SQL *Server* 2000 *Analysis Services* yang menghasilkan sekumpulan *set* dari *object Component Object Model (COM)* dan antar muka yang dapat digunakan untuk membuat aplikasi yang dapat memprogram *object-object* administrator. Dengan menggunakan *library* DSO, dapat dilakukan pengaturan *object Analysis Services* seperti *server*, *database*, data *source*, dimensi, *cube*, *mining model*, dan *role*. DSO adalah *class-class* dalam *library* COM (*Component Object Mode*) dan antar muka yang menghasilkan akses ke *Analysis server*. Ketika *class* dan antar muka tersebut digunakan bersamaan akan membuat sebuah *form object* model yang berkorespondensi dengan struktur internal dari *object* yang diatur oleh *Microsoft* SQL *Server* 2000 *Analysis Services* dan dapat diatur lewat *programming*.

Secara teori, DSO menggunakan aturan pengelompokkan secara hirarki dari *object-object* yang ada untuk mendefinisikan elemen dasar dari penyimpanan data *Analysis Services* seperti yang diterapkan oleh *Analysis Server*. Elemen-elemen dasarnya adalah *database*, data *source*, dimensi, *cube*, model data *mining*, dan *role*. DSO mengelola elemen-elemen dasar berikut dalam struktur hirarki dimana elemen berisi elemen yang lain dalam bentuk *tree*, dengan object *server*-nya berada pada *root* dari *tree*-nya. Object yang lain yang mendukung struktur dasar

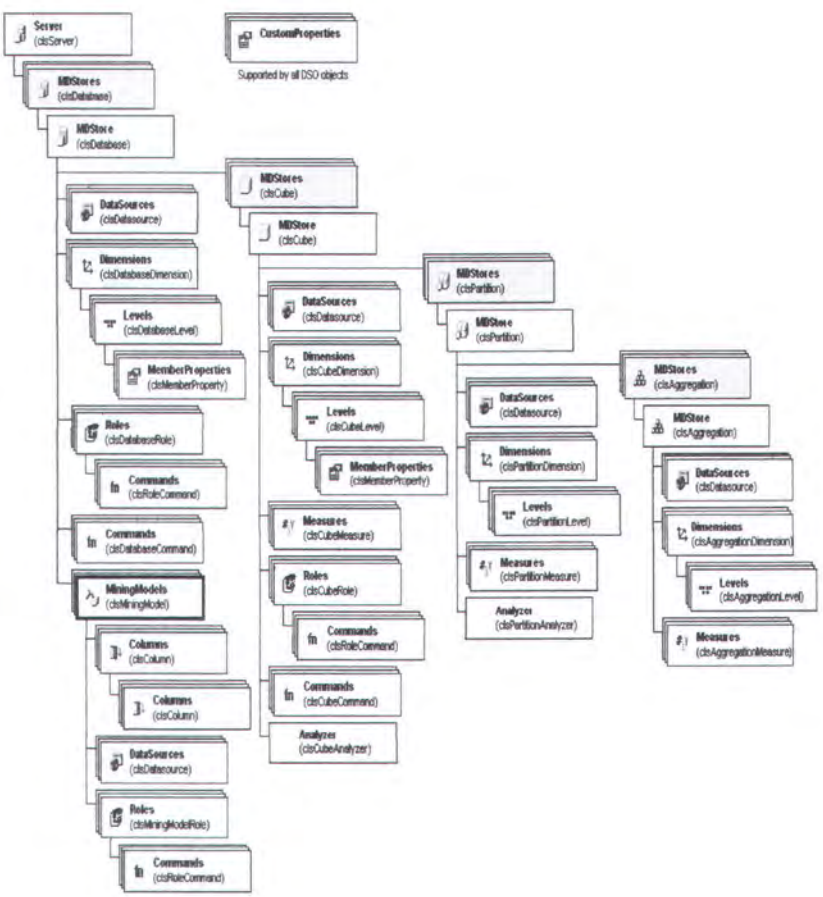
yaitu : *database*, *cube*, *partition* dan *aggregation* yang mendukung dimensi. Diagram berikut menunjukkan rangkuman dari model hirarki object dalam DSO .



Gambar II-9 Hirarki dari object DSO

Object *server* DSO berisi koleksi yang mendefinisikan *database* yang diakses oleh *server*. Setiap *database* dapat mengandung sekumpulan *object* yang mendefinisikan *cube*, *linked cube* atau *virtual cube*. Sebuah *cube* berisi satu atau lebih partisi, yang berisi satu atau lebih agregasi. *Linked cube* dibuat untuk memberikan akses pada *server* lokal ke *server* yang lain. *Remote server* mem-*publish cube*, dan *server* lokal men-*subscribe* ke *remote server* dengan membuat sebuah *linked cube*. *Virtual cube*, merupakan kombinasi dari isi *cube* itu sendiri, hampir mirip dengan sebuah *database* relasional yang dikombinasikan dengan porsi tabel yang dimiliki. Sebuah *database* dapat mengandung satu atau lebih kolom data *mining*. Model data *mining* dapat berisi satu atau lebih kolom data *mining*. *Database* juga mengandung *role*, yang digunakan untuk mengatur *security* pada *database* dan *cube* yang berasosiasi dengannya serta model data *mining*.

Object yang paling penting dalam koleksi *object* pada DSO, *database* dan *cube* yang didukung dengan sebuah koleksi **MDStores** yang menghasilkan *method* dan properti yang dapat digunakan untuk memanipulasi *object*. *Object-object* ini harus merupakan referensi dari *object parent*-nya, dan tidak dapat dibuat secara independen. Cara untuk membuat *database*, *cube*, partisi atau agregasi adalah melalui koleksi **MDStores** dari *object parent*-nya (koleksi dari *object* pada DSO *Server*). Koleksi dari **MDStores** menghasilkan implementasi dengan *method* **add**, **find**, **item**, **addnew** dan **remove** yang menangani relasi *parent-child* antara berbagai *object* dalam DSO.



Gambar II-10 Struktur hirarki koleksi object dalam DSO

Ada dua *object* utama yang diklasifikasikan dalam DSO yaitu *object* yang dapat diakses dan dikelola dengan menggunakan antar muka *default*-nya yang memiliki koleksi, *method* dan propertinya sendiri (misalnya: DSO *DataSource*), dan object yang mengimplementasikan antar muka DSO yang lain sebagai antar muka *default*-nya.

Tabel II-4 antar muka DSO dan nilai properti ClassType-nya

Interface	ClassType property value
Column	clsColumn
CubeAnalyzer	clsCubeAnalyzer
DataSource	clsDataSource
MemberProperty	clsMemberProperty
MiningModel	clsMiningModel
PartitionAnalyzer	clsPartitionAnalyzer
Server	clsServer

Object yang diimplementasikan dengan satu atau lebih antar muka menggunakan *subset* dari koleksi *method* yang ada dan properti yang berasosiasi dengan antar muka untuk variasi yang diimplementasikan secara berbeda untuk *command* pada *database*, *cube* dan *role*. Setiap deskripsi dari koleksi, *method* dan properti berisi nama dari *object* yang ada atau yang diimplementasikan.

Tabel II-5 Daftar antar muka DSO dan nilai properti ClassType-nya

Interface	ClassType property value
Command	clsDatabaseCommand clsCubeCommand clsRoleCommand
Dimension	clsDatabaseDimension clsCubeDimension clsPartitionDimension clsAggregationDimension
Level	clsDatabaseLevel clsCubeLevel clsPartitionLevel clsAggregationLevel

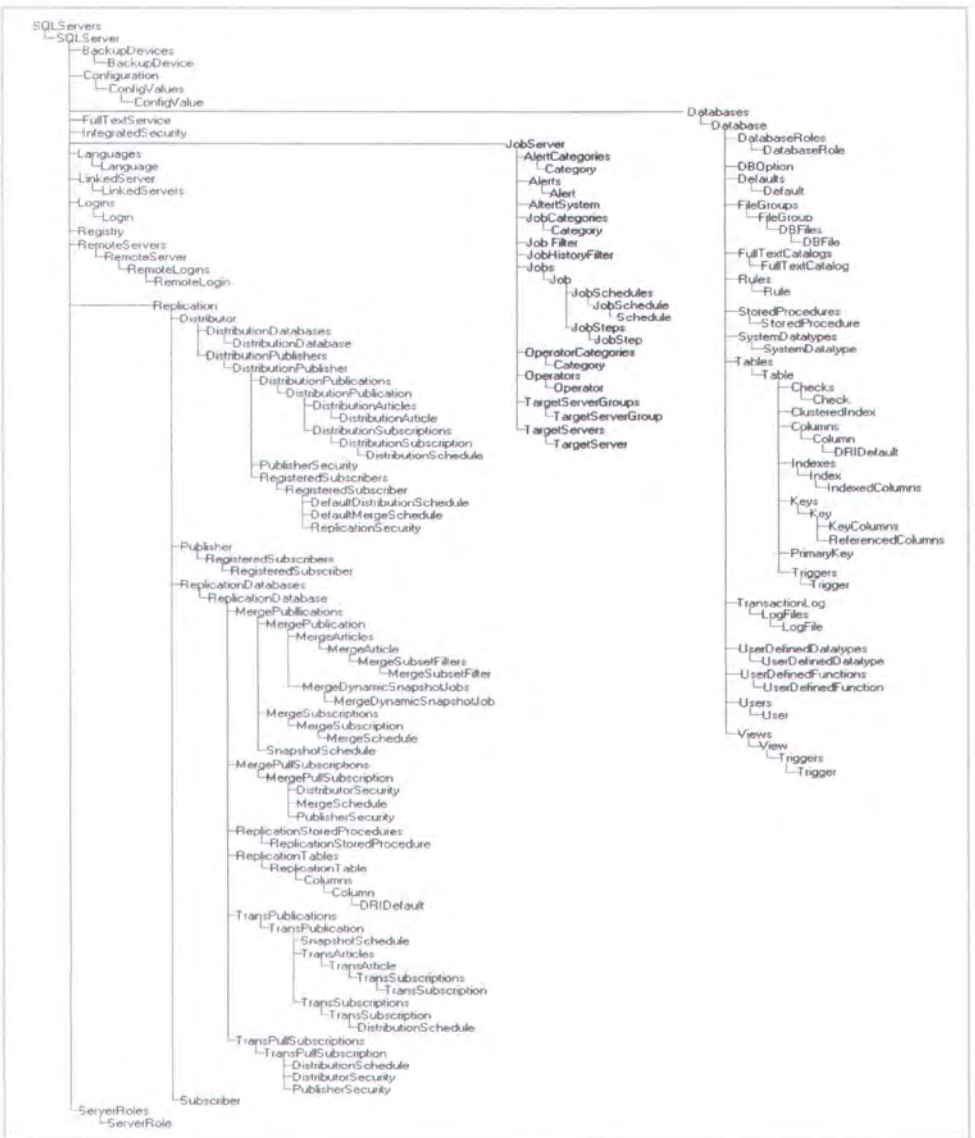
MDStore	clsDatabase clsCube clsPartition clsAggregation
Measure	clsCubeMeasure clsPartitionMeasure clsAggregationMeasure
Role	clsDatabaseRole clsCubeRole clsMiningModelRole

Library DSO tidak dibuat untuk *scripting* yang aman. Object dalam library yang dibuat aman untuk *scripting* akan mengambil *account* dari *security* konteksnya ketika *object* tersebut dibuat. Jika sebuah kontrol atau *library* yang tidak aman untuk *scripting* di *load* pada *Microsoft Internet Explorer* 4.0 atau lebih, *browser* hanya dapat menjalankan *scipt* pada mode *security* yang rendah dari IE. Jika DSO digunakan sebagai bagian dari komponen *middle-tier* pada aplikasi *client/server*, maka pola desain yang melewatkan secara langsung referensi dari *object* DSO harus digunakan.

2.6 SQL DMO (Database Management Object)

SQL Server Enterprise Manager adalah program yang dibuat untuk pengaksesan *Object-Object* SQL *Server* secara *native.* Apa yang dimaksud dengan *Object-Object* SQL Server? Pada konteks ini, istilah Object mengacu pada arsitektur *service* SQL *Server*, contohnya adalah *Object server, database*, *tabel, stored procedure, role, login, user, job*, dan sebagainya. Object-Object dari service SQL Server tersebut secara umum disebut SQL Distributed Management Object (SQL-DMO).

SQL DMO adalah sebuah model Object komponen (*Component Object Model,* atau COM) yang memungkinkan programmer memanipulasi Object-Object SQL Server secara *native* melalui kode program. Seorang programmer dapat melakukan itu menggunakan bahasa MS Visual Basic, VBScript, MS Visual C++, Delphi atau PowerBuilder. Bahkan sesungguhnya *Microsoft* mengembangkan SQL *Server Enterprise Manager* menggunakan komponen SQL-DMO sebagai pondasi dasarnya.



Gambar II-11 Struktur komponen SQL-DMO

2.7 Active Server Pages

Active Server Pages (ASP) bukan sebuah bahasa pemrograman karena masih menggunakan teknologi *script* pemrograman lain, yaitu *VBScript* dan *Jscript*. Beberapa keunggulan ASP, antara lain:

- Dapat mengubah isi halaman web secara dinamis.
- Memberikan respon terhadap query atau data yang dikirim dari suatu *form* HTML.
- Mengakses data atau *database* kemudian mengembalikan hasilnya ke *browser*.
- Lebih sederhana dan cepat dibandingkan dengan CGI dan *Perl*.

Pada dasarnya suatu file ASP terdiri dari *tag* HTML seperti halnya file HTML. Sedangkan untuk menuliskan *script* pemrograman dalam ASP, harus diawali dan diakhiri dengan tanda *delimiter*, berupa <% dan %>. *Script* pemrograman yang dieksekusi oleh *server* ini dapat berisi ekspresi, pernyataan, prosedur atau operator-operator yang sesuai dengan *script* pemrograman tersebut. *Script* pemrograman *default* dalam ASP adalah *VBScript*.

ASP adalah aplikasi *server* terbuka dimana kita bisa menggabungkan HTML, *script* yang berjalan di *server* dan komponen COM untuk menciptakan solusi yang bagus dan dinamis untuk aplikasi bisnis yang berbasis Web.

2.8 VBScript

VBScript adalah bagian dari *Visual Basic* yang membawa penulisan *script* yang aktif ke sebagian besar lingkungan. *Script* ini mencakup penulisan *script*

pada sisi *client* yaitu *browser*, maupun pada sisi *server* yaitu *Internet Information Server* (IIS). Ketika berdiri sendiri sebagai *script* pemrograman yang dikenal dengan *VBScript*, maka berwujud *client-side*. Akan tetapi, jika dimanfaatkan sebagai pemrograman ASP maka berwujud *server-side*.

2.9 Macromedia Flash MX 2004

Macromedia Flash merupakan teknologi multimedia yang berbasiskan vektor yang penggunaannya secara khusus pada web. *Flash* menggunakan vektor yang berarti sebuah gambar dapat diperbesar sampai manapun tanpa kehilangan kualitas gambar.

2.9.1 ActionScript

ActionScript adalah bahasa skrip dari *Macromedia Flash*. *ActionScript* digunakan untuk menentukan jalannnya *movie*. Seperti halnya bahasa skrip lainnya, ActionScript mempunyai aturan *sintaks*, *keyword* dan *operator*. Selain itu, *ActionScript* juga dapat menggunakan variabel-variabel untuk menyimpan informasi yang kemudian dapat ditampilkan kembali. Kita dapat menggunakan *Object-Object* dan *function* (fungsi) yang dihasilkan *ActionScript*, atau membuat *Object-Object* dan fungsi sendiri.

Sintaks dan jenis *AcrionScript* mirip dengan yang terdapat didalam bahasa pemrograman *JavaScript*. Dibawah ini terdapat beberapa perbedaan antar *ActionScript* dengan *JavaScript*.

- *ActionScript* tidak mendukung *Object* untuk menentukan *browser*, seperti *Document, Window,* dan *Anchor*.

- *ActionScript* tidak sepenuhnya mendukung semua *Object-Object* yang disediakan *JavaScript*.
- *ActionScript* mendukung bentuk *sintaks* yang tidak dapat digunakan didalam *JavaScript*. Sebagai contoh, sintaks *telTarget, ifFrameLoaded,* dan *slash.*
- *ActionScript* tidak mendukung beberapa bentuk *sintaks* didalam *JavaScript*, misalnya *try,catch,* dan *thrown.*

2.9.2 Penulisan ActionScript

Penulisan skrip di dalam *ActionScript* bersifat *object-oriented.* Hal ini berarti setiap informasi dikelola ke dalam group yang biasa disebut dengan *class*. Dari satu *class*, dapat dibuat banyak *instance* (atau bisa disebut dengan *Object*). *Instance*-lah yang digunakan di dalam skrip. *Class* yang disediakan oleh *ActionScript* dimasukkan kedalam kategori *Objects* pada panel *Actions*.

Object di dalam *ActionScript* dapat berisi data, atau grafis yang didalam *stage* direpresentasikan dengan *movieClip*, *button* atau *text fileds*. Semua *movie clips* adalah *instance* dari *class movieClips* yang disediakan oleh *ActionScript*. Semua *button* adalah *instance* dari *class Button* yang disediakan oleh *ActionScript*. Setiap *instance movie clip* mempunyai keseluruhan *properti* (seperti *_height*, *_rotation*, *_totalframes*) dan mempunyai keseluruhan metode (seperti *gotoAndPlay, loadMovie, startDrag)* dari class *movieClip.*

Ada beberapa aturan yang digunakan di dalam *ActionScript* diantaranya :

- Menggunakan *sintaks* titik (*Dot Syntax*)
- tanda titik (.) digunakan untuk mengindikasikan *properti* atau method yang berkaitan dengan suatu *Object* atau *movie clip*. Tanda titik juga digunakan untuk mengidentifikasikan *target path* suatu *movie clip*, variabel, fungsi, atau *Object*.

- Menggunakan tanda kurung kurawal
- *Statement ActionScript* digroupkan didalam blok yang ditandai dengan kurung kurawal ({}).
- Menggunakan tanda titik toma { ; }
- tanda titik koma merupakan tanda berakhirnya statement *ActionScript*.
- Menggunakan tanda kurung

2.10 XML (Extensible Markup Languange)

XML secara sepintas mirip dengan HTML. Keduanya berasal dari *Standar Generalized Markup Language* (SGML). Namun XML berbeda dari HTML dalam hal *sintaks* dan *semantik*.

XML harus mengikuti aturan dalam mengidentifikasi bagian-bagian dokumen dan dalam membuat struktur elemen yang bersarang. Elemen-elemen di dokumen XML tidak boleh saling tumpang tindih. Jika ada *tag* awal untuk sebuah elemen dan *tag* tersebut muncul lagi di elemen lain, maka *tag* yang sama tersebut harus ditutup dulu dengan *tag* penutup.

Sebagai contoh:

```
<b>Ini huruf tebal. <i> Ini huruf tebal miring. </b> Ini
huruf miring </i>
```

Ini huruf tebal. *Ini huruf tebal miring.* *Ini huruf miring.*

HTML masih bisa menerima, namun tidak demikian jika diimplementasikan di XML. Untuk XML maka haruslah seperti berikut:

```
<b>This is bold text.</b><i><b>This is bold italic text.</b>
This is italic  text.</i>
```

Semua struktur yang mengandung ambiguitas dihilangkan dalam dokumen XML, semua *parser* XML melihat dokumen sebagai struktur elemen bersarang yang sama. Untuk semua atribut harus diberi tanda petik, boleh tanda petik satu (') atau tanda petik dua ("). Tidak boleh ada karakter <, >, atau & dalam teks di dokumen

UNIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

XML. Untuk menggunakannya dipakai entitas < , > atau &amp. Jika di HTML tanda <b> berarti tanda untuk huruf tebal, tidak demikian di XML. Di XML, pengembang boleh mendefinisikan sendiri menurut deskripsinya.

Berikut merupakan keuntungan yang bisa didapat jika menerapkan teknologi XML:

1. Ekstensibilitas, bebas menentukan tag-tag sendiri.
2. Data dan presentasi dipisahkan.
3. Fungsi pencarian yang lebih baik.
4. Kemampuan untuk ditukar dan digabung dengan dokumen XML yang lain.
5. Cukup ditulis sekali dan dapat digunakan di mana saja.
6. Penyederhanaan aplikasi.
7. Penyimpanan data dalam format tree yang memudahkan dalam pengambilan datanya.

2.11 DOM (Document Object Model)

Untuk kebutuhan standarisasi bagi representasi XML, DOM (*Document Object Model*) sangat baik untuk kebutuhan pemrograman. Dasar dari DOM adalah konsep dan abstraksi *node interface* yang mendefinisikan kumpulan metode-metode dan *properti-properti* umum bagi *Object* DOM. *Interface* memiliki turunan dalam bentuk susunan tipe *node* seperti *element nodes* dan *attribute nodes*.

DOM berisi informasi mengenai struktur sebuah dokumen XML. Dalam penampilannya, dokumen XML berbentuk pohon *node* yang artinya setiap elemen merupakan *node* dan setiap *node* memungkinkan untuk memiliki *subtree* lagi. XML dan DOM diharapkan dapat menjadi *platform* yang netral dan standar.

Object dari XML DOM yang cukup penting adalah *Node. Interface* node mendefinisikan sekumpulan metode dan atribut umum, namun tidak semua jenis *node* memilikinya, contohnya adalah *childnodes* yang mewakili kumpulan *node*.

Pada Tabel II.6 akan dijelaskan mengenai tipe-tipe node yang terdapat di DOM. Atribut dari sebuah elemen adalah *read only,* dan *nilai nodeName* dan *nodeValue* tergantung dari tipe *node* yang digunakan. Contohnya, *nilai* dari *node* elemen selalu *null* sedangkan *nilai* dari *Attr Node* adalah teks. Tabel II.7 adalah tabel yang menjelaskan tentang atribut-atribut yang dimiliki oleh *node*.

Tabel II-6 Tipe Node dalam DOM

Tipe	Node Anak	Konstanta	N
Element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference	ELEMENT_NODE	
Attr	Text, EntityReference	ATTRIBUTE_NODE	
Text	-	TEXT_NODE	
CDATASection	-	CDATA_SECTION_NODE	
Entity Reference	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference	ENTITY_REFERENCE_ NODE	
Entity	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference	ENTITY_NODE	
Comment	-	COMMENT_NODE	
Processing Instruction	-	PROCESSING_ INSTRUCTION_NODE	
Document	Element(max 1), Comment, ProcessingInstruction, DocumentType	DOCUMENT_NODE	
DocumentType	-		0
Document Fragment	Element, Text Comment, ProcessingInstruction, CDATASection, EntityReference	DOCUMENT _FRAGMENT_NODE	1
Notation	-	NOTATION_NODE	2

Tabel II-7 Atribut yang dimiliki oleh Node

Nama atribut	Deskripsi
NodeName	Nama node tergantung dari tipenya
NodeValue	*Nilai* node tergantung dari tipenya
NodeType	Kode yang menjelaskan tipe Object
ParentNode	Node Parent. Boleh dimiliki semua node kecuali Document, DocumentFragment dan Attr
ChildNodes	Nodelist yang berisi semua node anak
FirstChild	Anak pertama dari sebuah node. Jika tidak memiliki anak maka *nilai*nya adalah *null*
LastChild	Anak terakhir dari sebuah node. Jika tidak memiliki anak maka *nilai*nya adalah *null.*
PreviousSibiling	Node sebelumnya yang selevel
NextSibling	Node sesudahnya yang selevel
Attributes	Sebuah *NameNodeMap* yang berisi atribut dari sebuah node
OwnerDocument	Object dokumen yang diasosiakan ke node ini

BAB III

ANALISA DAN PERANCANGAN PERANGKAT LUNAK

BAB III

ANALISA DAN PERANCANGAN PERANGKAT LUNAK

3.1 DESKRIPSI UMUM SISTEM

Pada bab ini dijelaskan mengenai analisa dan perancangan aplikasi dan pembuatan aplikasi. Analisa dan perancangan aplikasi itu sendiri ditujukan untuk memberikan gambaran secara umum terhadap aplikasi yang dibuat. Hal ini berguna untuk menunjang pembuatan aplikasi sehingga kebutuhan akan aplikasi tersebut dapat diketahui. Dengan analisa dan perancangan aplikasi, maka akan mempermudah untuk mengadakan pengembangan lebih lanjut terhadap aplikasi yang dibuat.

Untuk membuat aplikasi pada tugas akhir ini terlebih dahulu dilakukan perancangan proses, serta perancangan data. Perancangan proses berguna untuk mengintegrasikan semua proses yang terjadi dalam aplikasi. Sedangkan perancangan data berguna untuk mengetahui data apa saja yang dibutuhkan dalam proses-proses pada aplikasi ini.

Aplikasi yang akan dibangun merupakan editor atau *tools* untuk melakukan pemodelan skenario pembuatan *cube* untuk kebutuhan OLAP *datawarehouse*. Data yang diolah merupakan data multidimensional pada SQL Server. *Tools* yang dibangun merupakan aplikasi *client*. Proses-proses yang dilakukan di *client* akan diteruskan ke aplikasi *server* untuk diolah berdasarkan *input* yang diberikan.

3.1.1 Arsitektur Sistem

Sistem aplikasi yang akan dibangun adalah aplikasi *client-server* yang berbasis web yang memiliki arsitektur 3-*tier* yang terdiri atas *User Interface Layer*, *Business Logic Layer* dan *Database Layer.*

Microsoft® SQL Server™ 2000 Analysis Services meliputi server *Analysis* dan *PivotTable® Service*. *Server Analysis* membuat dan mengatur *cube* untuk data multidimensional untuk proses analisa *online* (OLAP) dan menghasilkan data multidimensional untuk *Service Pivot Table*, yang akan menghasilkan data kepada klien melalui *Microsoft Decision Object* dan OLE DB untuk *service provider* OLAP.

Server menyimpan meta data milik *cube* (*data definition specification*) dalam sebuah media penyimpanan. *Cube* yang lengkap dapat disimpan dalam berbagai mode penyimpanan :

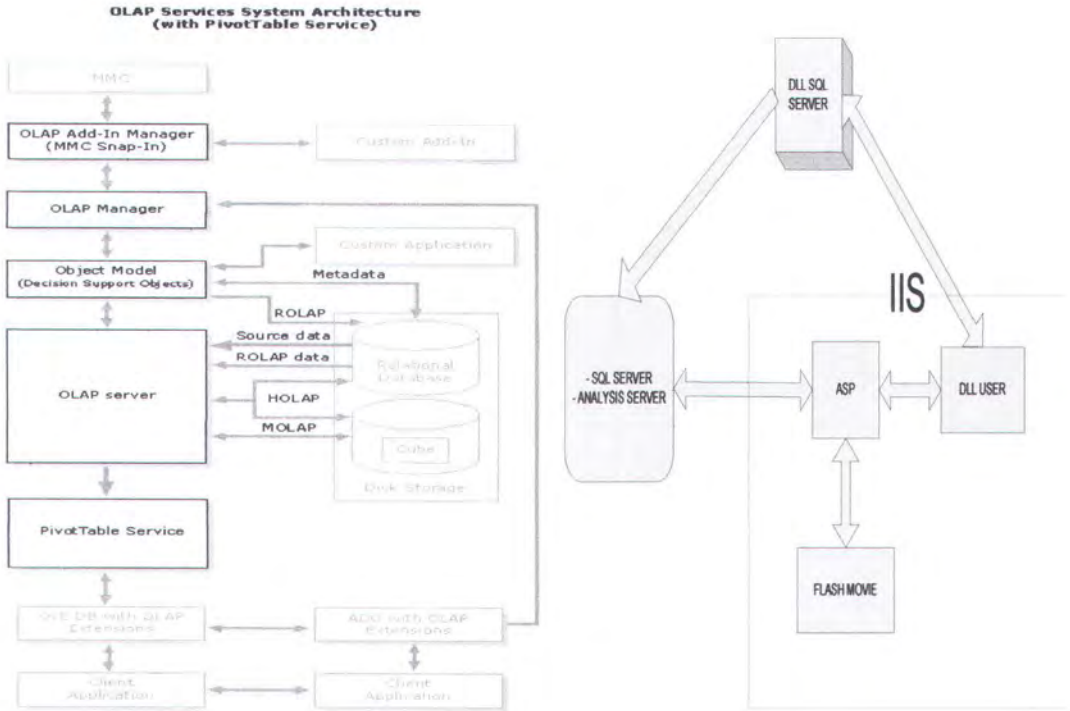
- MOLAP (*Multidimensional Database Files*)
- ROLAP (*Relational Database*), atau
- HOLAP (*Hybrid of Multidimensional Database and Relational Tables*)

Sumber data untuk *cube* multidimensional terdapat pada *database* relasional dimana data sudah ditransformasikan kedalam sebuah skema *star* atau *snowflake* yang umumnya digunakan dalam sistem *datawarehouse* OLAP. *Analysis Services* dapat bekerja dengan banyak *database* relasional yang

mendukung koneksi ODBC atau OLE DB. Ketika digunakan sebagai bagian dari SQL *Server* 2000, *Analysis Services* menawarkan keamanan yang seimbang dan kemampuan yang lainnya. Akan tetapi, karena keterbatasan dari fasilitas yang diberikan *DSO* untuk mengakses object-object dalam Analysis Services, dalam Tugas Akhir ini hanya terbatas pada penyimpanan skema MOLAP dan *Star Schema.*

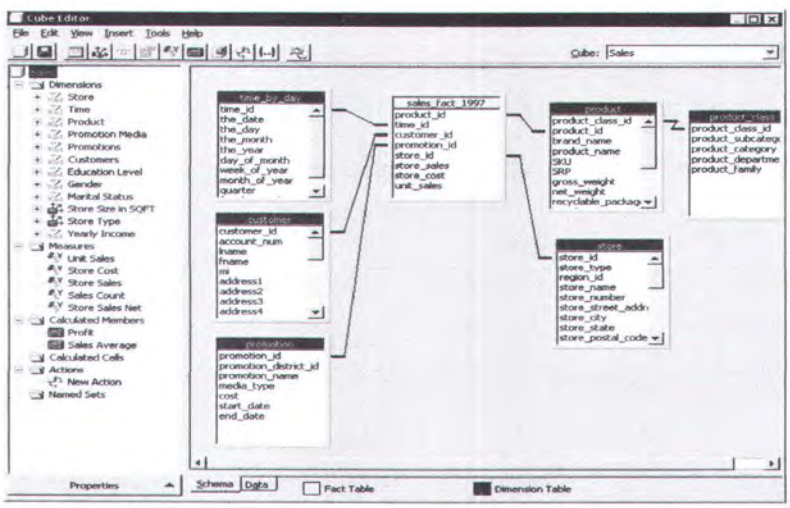
Kontrol dari *server* dilakukan melalui *user interface* dari *Analysis Manager* atau melalui aplikasi *custom* yang dibuat dengan menggunakan model DSO (*Decision Support Objects*). DSO mengontrol pembuatan dan manajemen dari *object* oleh *server* dan mengatur meta data *object* dalam media penyimpanan. Model *object* digunakan oleh program *Analysis Manager* yang menghasilkan *user interface* melalui sebuah *snap-in* ke *Microsoft Management Console* (MMC). Model *Object* DSO dapat digunakan oleh aplikasi yang ditulis dengan menggunakan *Microsoft Visual Basic* untuk menghasilkan kontrol program pada *server.* Selain itu dapat juga digunakan untuk membangun aplikasi untuk berinteraksi dengan *user interface* dari *Analysis Manager.*

Diagram berikut mengilustrasikan elemen dan fungsi dari *Analysis Server* dan menggunakan *service PivotTable* untuk menghasilkan data multidimensional kepada aplikasi *client* konsumen. *User interface Analysis Manager* menggunakan *service PivotTable* untuk menghasilkan data multidimensional dari *server* untuk *browsing* oleh administrator *server.*



Gambar III-1 Arsitektur Sistem

3.1.2 Kemampuan Yang Dimiliki Aplikasi



Gambar III-2 Editor pada SQL Analysis Manager

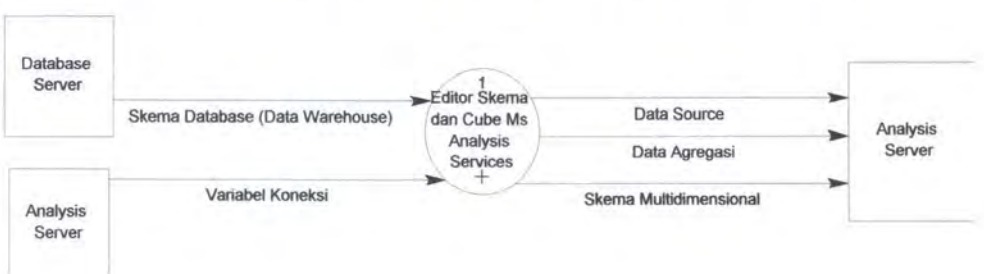
Aplikasi editor *tools* untuk melakukan permodelan skenario pembuatan skema *cube* dan dimensi OLAP untuk data *warehousing* memiliki kemampuan sebagai berikut :

Pembuatan *cube* , yaitu menentukan tabel yang akan dijadikan *fact table* dari skema OLAP yang akan dibuat. Didalam membuat *cube* , maka *measure* juga harus ditentukan, termasuk didalamnya adalah dimensi dari *cube*. Sebelum *cube* dibuat maka *user* harus menentukan data *source* yang akan digunakan untuk OLAP. Data *source* dibuat dengan menggunakan *provider* yang telah tersedia dalam SQL *Server*.

Menampilkan skema dari *cube* yang telah dibuat. Ketika *user* selesai melakukan *setting* dalam pembuatan *cube*, maka data dari *client* akan dikirimkan ke *server*. Jika proses di *server* telah selesai maka skema akan ditampilkan berdasarkan data yang diambil lagi dari *server* setelah proses eksekusi *cube* selesai.

Melakukan perubahan pada skema *cube* yang sudah dibuat. *User* dapat merubah struktur *cube* dengan menambahkan table dimensi. Perubahan dilakukan pada sisi *client* baru kemudian dikirimkan ke *server*. Jika *server* telah selesai mengeksekusinya maka data akan ditampilkan lagi pada jendela editor skema *cube*.

Gambar III.3 berikut ini merupakan penggambaran proses-proses global yang dapat dilakukan oleh pengguna terhadap aplikasi yang dirancang.



Gambar III-3 DAD Level 0 Sistem Editor Skema dan Cube

Proses global yang dilakukan dalam aplikasi editor *cube* dan skema yang dibuat berbasis web dengan menggunakan *Macromedia Flash* MX 2004 dengan bahasa pemrograman ASP adalah me-*retrieve database* dari SQL *Enterprise Manager* sebagai data *source* yang nantinya digunakan sebagai *source table* dan *source column* dalam pembuatan skema *cube* dari sebuah *datawarehouse*, membuat koneksi dengan *Analysis Manager* untuk menghubungkan antara aplikasi dengan *Analysis Server*, menyimpan skema *cube* yang telah dibuat kedalam *Analysis Manager* dan menyimpan skema *database* aslinya kedalam *database Server*, serta menampilkan kembali hasil skema *Cube* yang telah dibuat tersebut supaya pengguna dapat melakukan perubahan.

3.2 DESAIN APLIKASI CLIENT

Aplikasi *client* dibangun dengan menggunakan Macromedia Flash MX 2004, yang merupakan suatu bentuk antar muka grafis yang cukup menarik dan dinamis, dimana pengguna bisa berinteraksi dengan aplikasi maupun dengan pengguna lain. Aplikasi *client* memiliki tampilan yang hampir sama dengan *Editor Cube* yang disediakan oleh SQL *Server* sehingga pengguna tidak perlu belajar terlalu lama dalam menggunakan Aplikasi ini.

Sebagai aplikasi perancangan dan pembuatan skema *cube* untuk sebuah *datawarehouse*, proses utama yang dilakukan oleh pengguna adalah membuat suatu *fact table* dan dimensi-dimensinya, menetapkan *level* serta *measure* dari *fact table*, kemudian memproses agregasi dari skema yang telah dibuat supaya hasilnya dapat di-*browse* oleh pengguna, dimana dalam aplikasi ini hanya dibatasi

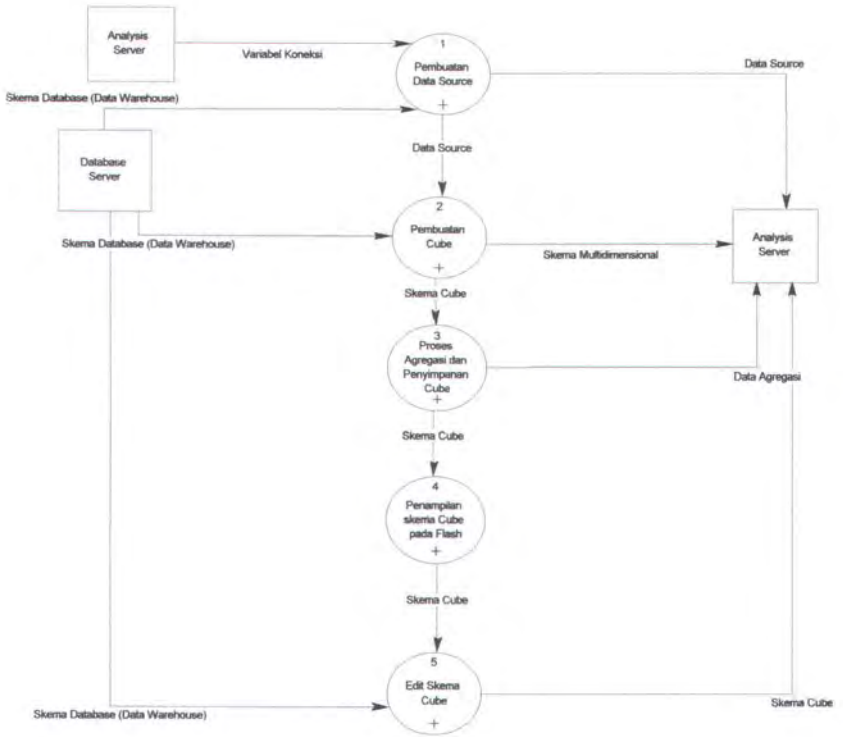
dengan menggunakan metode desain *storage* MOLAP (Multidimensional OLAP). Setelah skema tersebut berhasil diproses dengan sukses, maka skema keseluruhan akan ditampilkan kembali oleh aplikasi kepada pengguna dalam bentuk diagram relasi antar tabel yang aslinya (*source table*-nya) supaya pengguna dapat melakukan perubahan di kemudian hari. Aplikasi dapat menampilkan kembali skema yang telah dibuat ke dalam *object-object Movie Clip*, dengan mengambil data skema yang telah dibuat dari *database* yang dibuat sendiri karena SQL Server tidak memberikan fasilitas dan metode untuk mengambil skema *database source* yang berhubungan dengan skema cube yang telah dibuat (*source table* dan *source columnnya* tidak dapat di-*retrieve*). *Database* yang dibuat dalam *Enterprise Manager* ini berfungsi untuk menyimpan skema *source table* dari *Cube* yang telah dibuat supaya Aplikasi dapat menampilkannya kembali dengan terlebih dahulu membuat *object* XML dari data skema yang terdapat pada *database*. Desain arsitektur Aplikasi *Client* memiliki Object-Object sebagai berikut:

1. **Object Koneksi,** merupakan *Object* yang berisi properti-properti dari sebuah koneksi. Pada aplikasi ini hanya digunakan sebuah *object* koneksi saja sebagai penghubung antara *Analysis Manager* dengan data *source*nya yang dibatasi hanya meliputi SQL *Server* saja.
2. **Object SavedData,** merupakan *object* yang menyimpan semua data *input*-an user yang diperlukan untuk membangun sebuah skema *cube*. Data-data tersebut disimpan ke dalam *database Server Enterprise Manager* untuk digunakan ketika skema ditampilkan kembali kepada *user* jika *user* ingin melakukan perubahan pada skema yang telah dibuat.
3. **Object BuildCube,** object ini berisi perintah untuk memanggil CubeEditor.dll yang merupakan *dll* dari *Active* X yang dibuat sendiri

untuk mengatasi keterbatasan *security* pada web sehingga ASP tidak bisa menjalankan fungsi-fungsi DSO untuk membangun skema *cube* ke dalam *Analysis Manager*. Didalam *object* ini terdapat banyak fungsi diantaranya adalah *create dimension, create level dimension, create cube* dan *measure* serta proses agregasi desain *storage*-nya dengan menggunakan MOLAP

4. **Object LoadSkema,** merupakan *object* yang berisi *query* sql untuk me-*retrieve* skema *cube* yang telah dibuat dari *database* dan menampilkannya ke *stage*, sehingga *user* dapat melihat skema yang telah dibuat.
5. **Object Dimension,** merupakan *object* yang berfungsi untuk melakukan perubahan dengan menjalankan *query* dan DSO *object* untuk menambah measure pada skema *cube* yang telah dibuat.
6. **Object Remove,** merupakan *object* yang berisi perintah untuk menghapus *cube*.
7. **Object RebuildCube,** merupakan *object* yang berisi fungsi dan prosedur untuk membangun skema *cube* setelah dilakukan perubahan pada skema *cube* yang telah dibuat pada tahap sebelumnya. *Object* ini perlu dibuat dalam *class* tersendiri karena proses edit skema *cube* sebenarnya tidak bisa dilakukan secara langsung sehingga *object* yang sudah ada dihapus terlebih dahulu baru kemudian dibuat lagi . Step-step dalam *rebuild cube* harus dilakukan secara berurutan dan tidak bias bersamaan.
8. **Object ClearStage,** merupakan *object* yang berfungsi untuk membersihkan semua *movie clip* yang ada pada *stage*.
9. **Object GambarGaris,** merupakan *Object* yang menangani dan menggambarkan garis relasi pada skema *Cube* yang telah dibuat dengan menggunakan *source table* dan *source column*-ya. Garis ini menunjukkan hubungan atau relasi antara skema *cube* dengan skema *database* aslinya.

Untuk menjelaskan proses apa saja yang terdapat dalam sistem aplikasi ini digunakan DAD level 1 seperti terlihat pada gambar III.4. Ini merupakan detail dari sistem aplikasi yang akan dibuat.



Gambar III-4 DAD Level 1 Sistem Editor Skema dan Cube

Terdapat lima proses utama yang dapat dilakukan aplikasi yaitu, pembuatan data *source*, pembuatan *cube*, proses agregasi dan penyimpanan *Cube*, penampilan skema *cube* pada *Flash*, dan edit skema *cube*.

3.2.1 DAD Level 2 : Proses Pembuatan Data Source

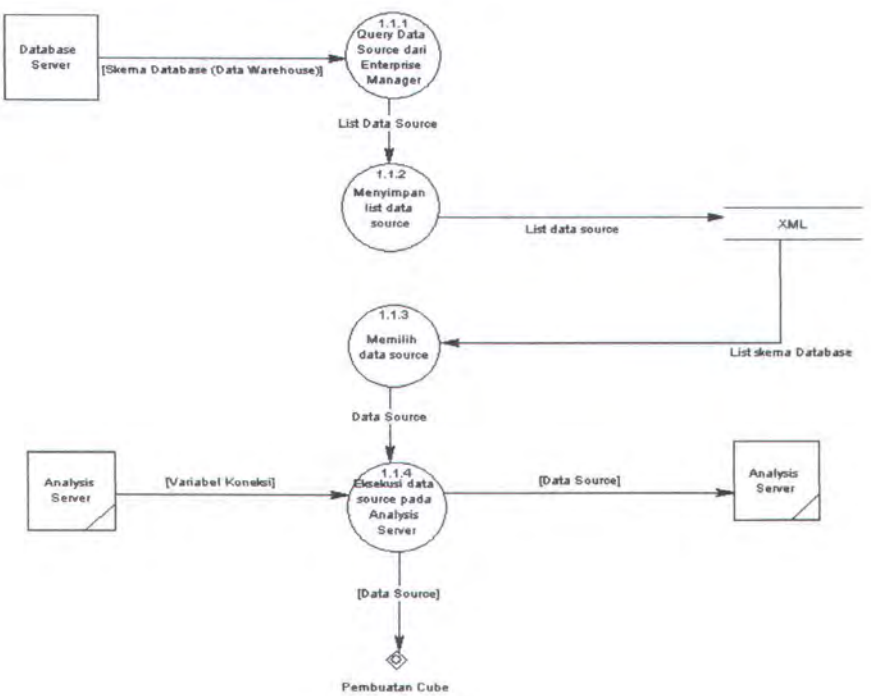
Pembuatan data *source* merupakan fungsi terpenting yang harus dibuat, karena data *source* berfungsi untuk menghubungkan *Analysis Server* dengan *Database Server*, sehingga pengguna dapat membuat suatu skema *cube* seperti layaknya yang dibuat pada Editor *Cube* pada *Analysis Manager* miliknya SQL *Server* 2000. Proses-proses yang dilakukan dalam pembuatan data *source* adalah :

1. Query data *source* dari *Enterprise Manager*
2. Menyimpan list data *source*

3. Memilih data *source*
4. Eksekusi data *source* pada *Analysis Server*

Proses pertama yang dilakukan dalam pembuatan data *source* adalah mengambil (me-*retrieve*) *database* yang terdapat dalam *database server* SQL *Enterprise Manager* dengan melakukan *query* menggunakan *object-object database* yang terdapat pada SQL DMO. Supaya hasilnya dapat ditampilkan kedalam *form* aplikasi dimana pengguna dapat memilih *database* mana yang akan dijadikan data *source*, maka *list* data *source* yang sudah diperoleh dari hasil query tersebut disimpan ke dalam object XML. Proses selanjutnya adalah pengguna memilih data *source* yang akan dibuat skema *cube*-nya, dan hasilnya akan disimpan kedalam *database* kemudian dieksekusi pada *Analysis Manager*.

Untuk lebih jelasnya, dapat dilihat gambar III.5 yang merupakan gambaran proses dari pembuatan data *source*.



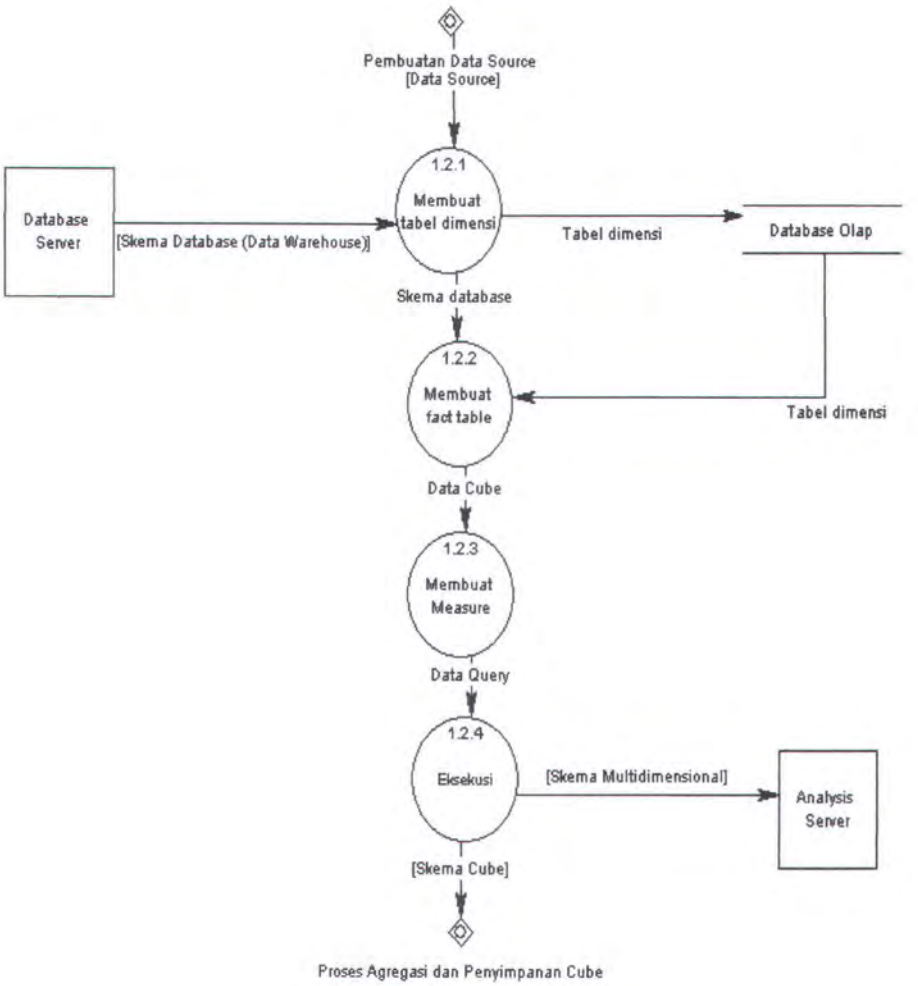
Gambar III-5 DAD Level 2 dari Proses Pembuatan Data Source

3.2.2 DAD Level 2 : Pembuatan Cube

Pada proses pembuatan *cube* yang dilakukan setelah data *source* selesai dibuat, terdapat beberapa tahapan proses yang harus dilakukan, antara lain:

1. membuat tabel dimensi
2. membuat *fact table*
3. membuat *measure*
4. eksekusi

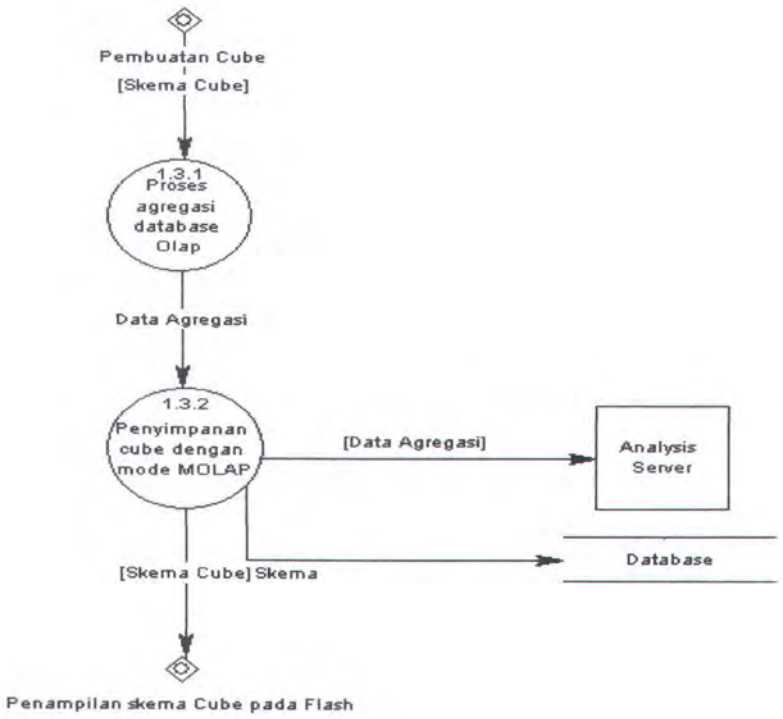
Proses pertama yang harus dilakukan dalam melakukan pembuatan *cube* adalah membuat tabel dimensi dengan menggunakan *table* yang sudah tersedia pada *Source Table* dari hasil koneksi data *source* yang telah dibuat pada proses sebelumnya. Didalam membuat tabel dimensi sekaligus dibuat level yang digunakan sebagai acuan dalam tabel dimensi tersebut. Langkah selanjutnya adalah membuat *measure* dari kolom yang terdapat pada *fact table*. Setelah proses pembuatan *measure* selesai, maka proses selanjutnya adalah eksekusi dari *cube* yang telah dibuat dengan struktur tabel dimensi yang menyertainya pada *Analysis Manager*. Langkah-langkah yang dilakukan untuk meng-create skema cube pada *Analysis Server* berbeda dengan langkah yang ada pada sisi klien. Di sisi klien pengguna menentukan terlebih dahulu *fact table* dan *measure* yang akan dijadikan sebagai cube, kemudian baru menentukan table dimensi yang memiliki *referential integrity* dengan *fact table*- nya. Sedangkan di sisi *server*, sebaliknya. Dimensi harus dibuat terlebih dahulu untuk dihubungkan dengan fact table dalam cube. Sebagai gambaran proses yang lebih jelas, dapat dilihat pada diagram alur data pada gambar III.6 berikut ini :



Gambar III-6 DAD Level 2 Proses Pembuatan Cube

3.2.3 DAD Level 2 : Proses Agregasi dan Penyimpanan Cube

Proses agregasi dan penyimpanan *cube* dilakukan dengan menggunakan metode MOLAP (Multidimensional OLAP) supaya data multidimensional dalam skema *Cube* yang telah dibuat pada proses sebelumnya dapat di-*browse* hasilnya oleh pengguna. Agregasi dan peyimpanan *Cube* dilakukan oleh *object BuildCube* yang akan mengeksekusi skema tersebut ke *Analysis Server* dan menyimpannya secara bersamaan kedalam *database* yang sudah dibuat sebelumnya.



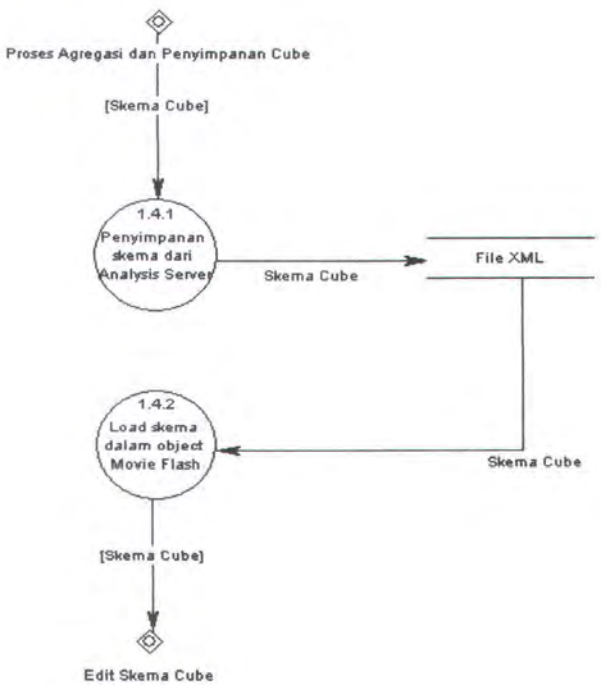
Gambar III-7 DAD Level 2 Proses Agregasi dan penyimpanan Cube

3.2.4 DAD Level 2 : Penampilan skema cube pada Flash

Setelah penyimpanan agregasi *cube* , maka proses-proses lain yang dijalankan selanjutnya adalah penyimpanan skema *cube* ke dalam *Analysis Server* dan kedalam *database* supaya hasilnya dapat di-*load* kembali oleh aplikasi dengan terlebih dahulu mengubahnya menjadi *object-object* multidimensional yang disimpan dalam XML.

Awalnya object XML pada aplikasi *client* dikirim melalui fasilitas *fscommand* pada flash, yang kemudian dituliskan oleh *server* dalam hal ini dengan bantuan bahasa *VBScript*, ke bentuk file XML. Dari bentuk file XML kemudian

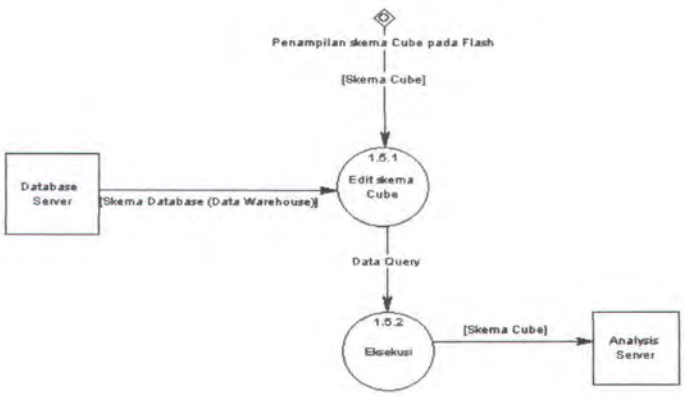
menggunakan teknologi *Microsoft* XML DOM, file tersebut dibaca dan kemudian di-*parsing* menjadi *object-object* koneksi, *cube, dimensi, source table, source column* dan sebagainya. Sehingga apabila dibaca secara keseluruhan akan terbentuk skema relasional *cube* dalam OLAP. Yang kemudian dengan bantuan SQL DSO akan disimpan pada *Analysis Manager* sebagai *Cube Schema.*



Gambar III-8 DAD Level 2 Proses Penampilan skema Cube pada Flash

3.2.5 DAD Level 2 : Edit skema cube

Proses ini merupakan proses *reverse engineering* dari file XML yang pernah kita buat. File XML dibaca dan kemudian diterjemahkan kedalam *object – object movieClip* yang memiliki properti-properti tertentu sesuai dengan isi dari hasil pembacaan file XML tersebut. Untuk lebih jelasnya dapat dilihat dari gambar III.9 yang merupakan DAD Level 2 dari proses mengedit skema *Cube*.



Gambar III-9 DAD Level 2 Proses Edit skema cube

Untuk proses edit, pengguna hanya terbatas bisa melakukan perubahan untuk menambah tabel dimensi pada skema *cube* tersebut. Sebelum data file XML tersebut dapat dibuka dan diparsing kebentuk *movieClip,* terlebih dahulu harus melewati proses *authentifikasi* dimana pengguna harus memasukkan *username* dan *password.*

3.3 DESAIN APLIKASI SERVER

Aplikasi *server* dibangun menggunakan bahasa pemrograman *Active Server Pages* (ASP) dan VB *Active* X dll. Yang mana dalam implementasinya, aplikasi yang dibuat banyak mempergunakan ADO MD, DSO, SQLDMO dan tentu saja ODBC sebagai alat untuk menciptakan koneksi.

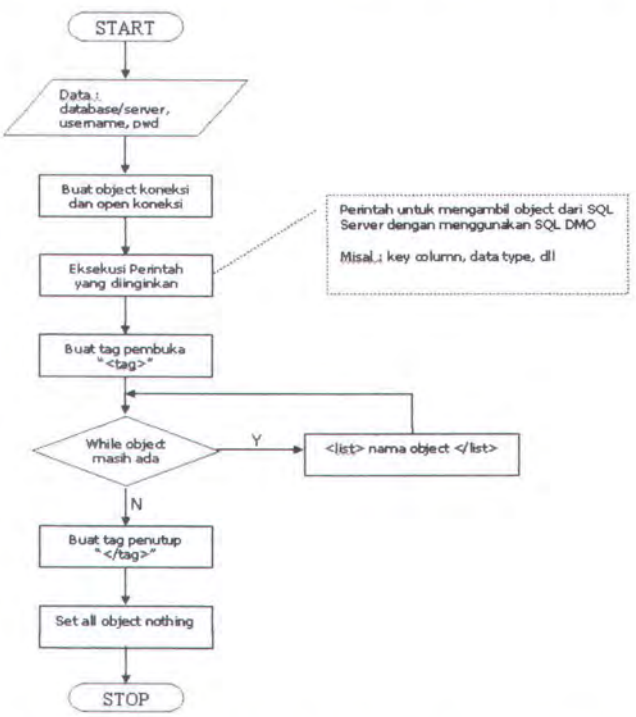
Fungsi – fungsi utama dari aplikasi server :

1. Mengambil informasi (*schema*) dari suatu server seperti : daftar *database*, daftar tabel, daftar *field*, daftar *column*, daftar *datatype*, daftar *primary key*, daftar *measures*, daftar *dimensi*, daftar *level*, dan daftar *cube*.

2. Membuat skema *cube* dengan menggunakan *object-object* yang ada pada DSO (*Decission Support Object*) dengan menggunakan informasi yang dikirim dari sisi *client*.
3. Menyimpan segala variabel dan properti dari *object-object* yang digunakan selama proses pembuatan *cube* ke dalam *database* Cube Editor.
4. Mengeksekusi skema yang dibuat secara *remote* berdasarkan permintaan

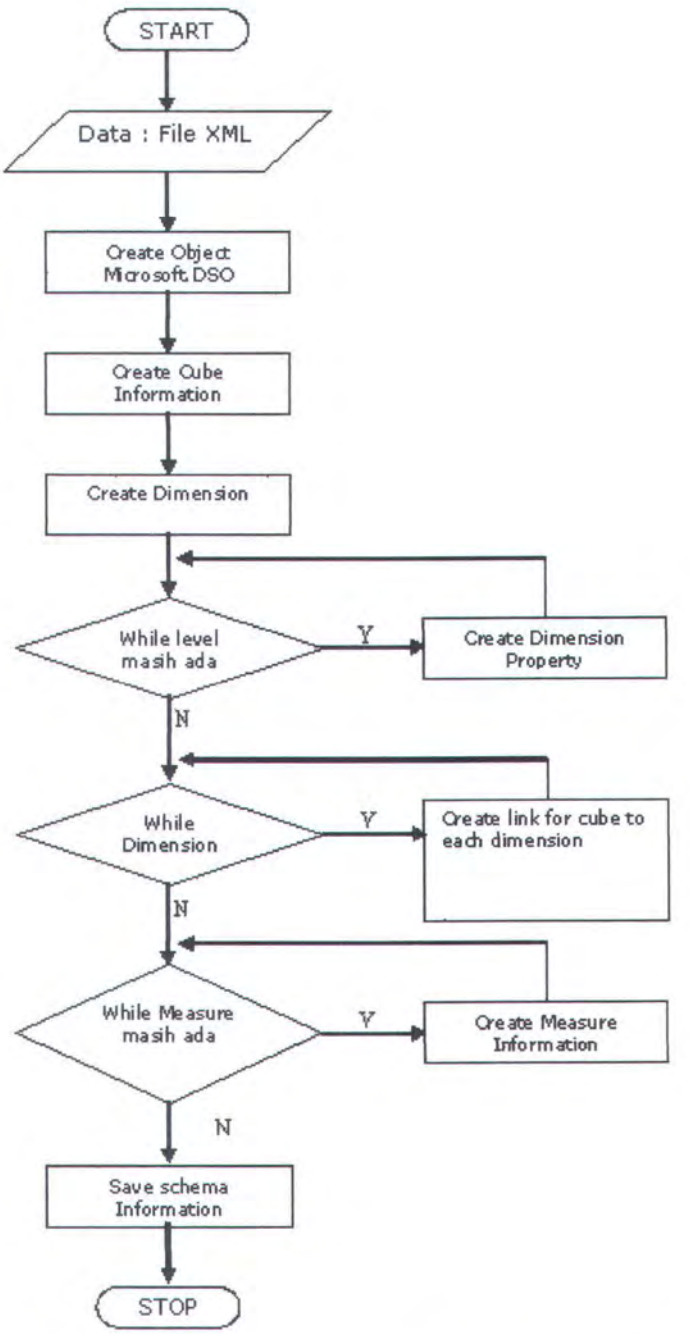
3.3.1 Mengambil Informasi (schema) dari database Server

Agar informasi yang didapat bisa dibaca oleh *Flash* maka, hasil keluaran dari file *Server* dibuat dalam bentuk *tag* XML. Gambar 3.9 merupakan Alur program untuk pengambilan informasi *schema* dari suatu *server*, dimana penerapan algoritma ini terdapat pada file *dataSource.asp, dimTable.asp, skema.asp, dimensionList.asp, dimensionTable.asp, cubeList* dan *tableList.asp*



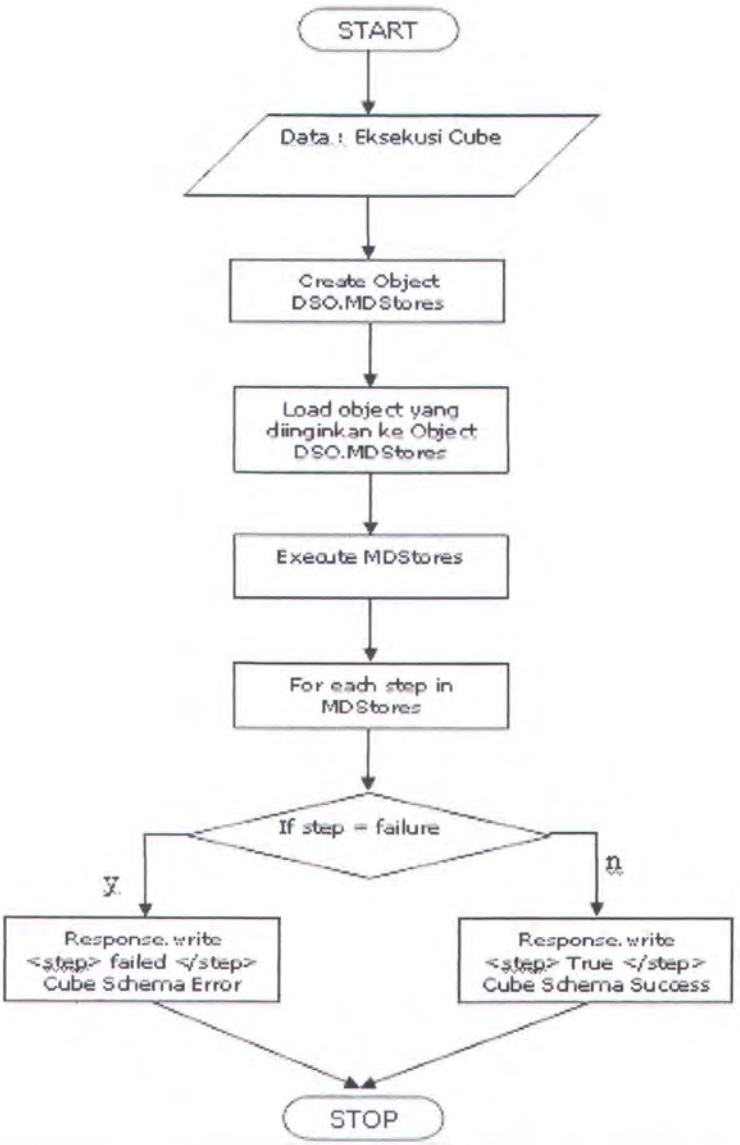
Gambar III-10 Algoritma Pengambilan Informasi (schema) Database Server

3.3.2 Algoritma Pembuatan Skema Cube



Gambar III-11 Flow chart Algoritma Pembuatan Skema Cube

3.3.3 Algoritma menampilkan skema Cube



Gambar III-12 Flow Chart Algoritma menampilkan Schema Cube

3.4 STRUKTUR PENYIMPANAN

Pada saat fungsi penyimpanan dijalankan maka suatu *object* yang dibuat dari sisi *client* akan disimpan pada *server* dalam tiga bentuk struktur penyimpanan yaitu penyimpanan pada *Analysis Server*, *Database SQL Server* dan

penyimpanan dalam bentuk file XML. Ketiga bentuk penyimpanan tersebut dimaksudkan agar pengguna dapat mengedit dari sisi *client* maupun dari sisi *server*.

Object-object DSO yang dibuat selama proses pembuatan sutau skema *Cube* akan disimpan kedalam *database Cube* Editor karena *Analysis Server* tidak menyediakan metode untuk mengakses relasi *source table* dengan skema *cube*-nya untuk ditampilkan pada jendela skema *Cube*. *Autentifikasi* pengguna dilakukan di awal , ketika pengguna membuka aplikasi ini.

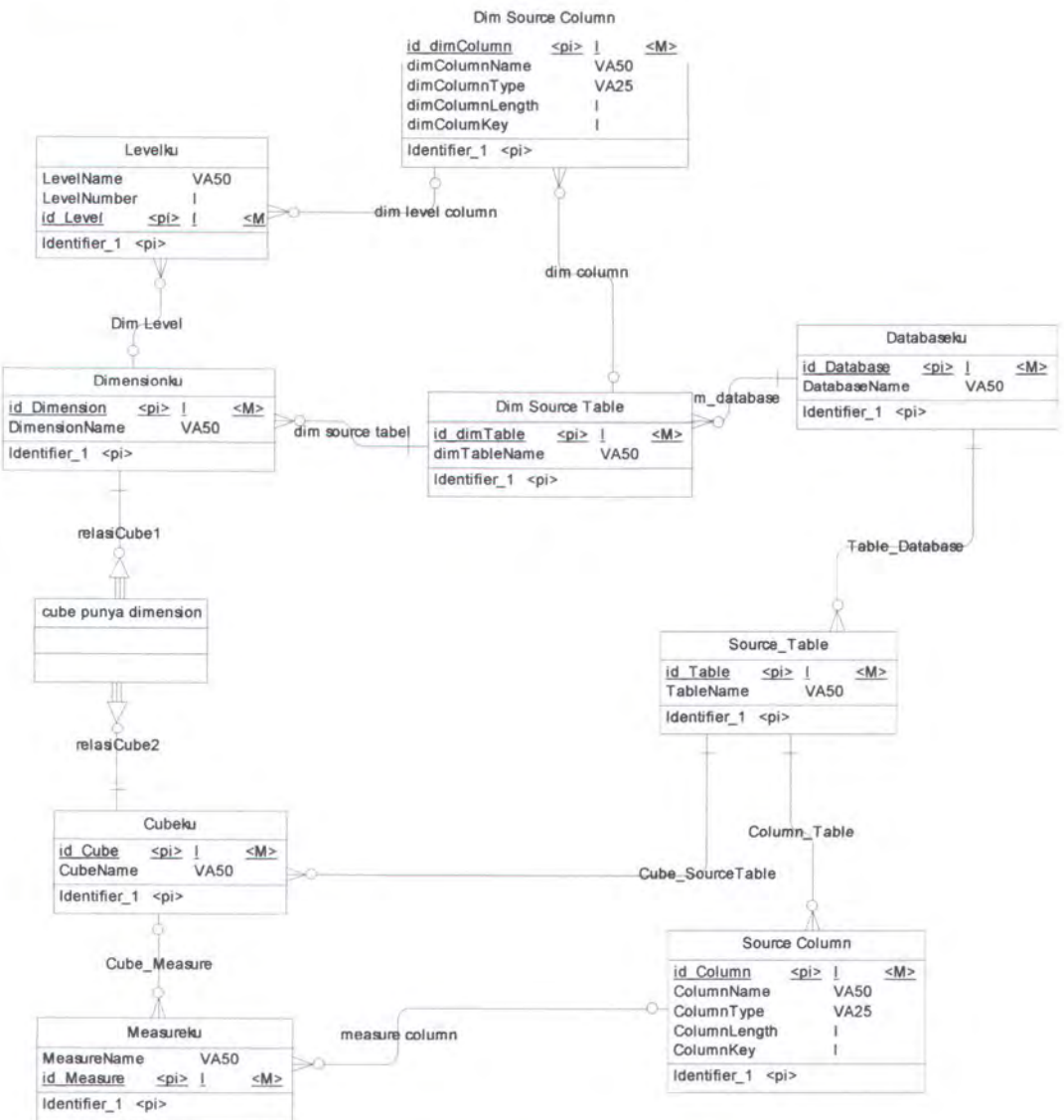
Untuk dapat melakukan perubahan skema *Cube* yang telah tersimpan pada *Analysis Server*, pengguna dapat membuka jendela Editor Skema *Cube* pada *Console Tree* di *Analysis Server* seperti terlihat pada gambar 2.10. Sedangkan untuk melakukan perubahan pada skema *Cube* di sisi *client* (pada aplikasi) digunakan skema *Cube* yang telah tersimpan pada *Database Cube* Editor yang telah dibuat sebelumnya. Kemudian dari *database* tersebut, disimpan ke dalam *object* XML supaya *Flash* dapat menampilkannya dalam bentuk *movie clip* berdasarkan *object* pada XML tersebut.

3.4.1 Skema Database Cube Editor

Microsoft SQL *Server* tidak menyediakan suatu metode untuk menampilkan atau me-*retrieve* data skema *cube* yang berhubungan dengan skema aslinya, dimana ketika *cube* selesai diproses , skema relasional data *source* dengan skema *cube* harus ditampilkan. Oleh karena itu, dalam membuat aplikasi editor skema *cube* ini diperlukan sebuah *database* untuk menyimpan relasional antara

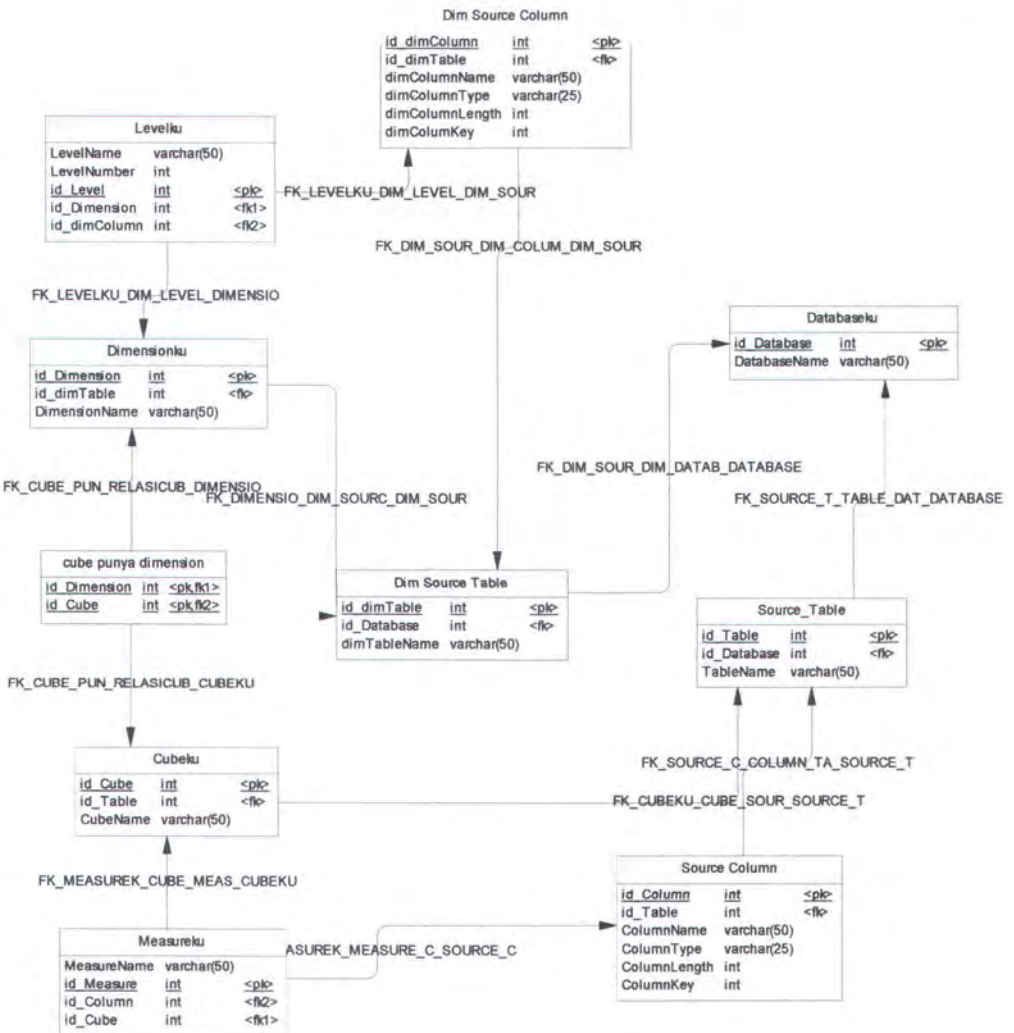
skema *datasource* dengan skema *cube*-nya, sehingga dapat ditampilkan kembali pada aplikasi ini. Jadi aplikasi hanya bisa menampilkan skema *cube* yang dibuat pada aplikasi ini karena data skema *source table* dari *Analysis Manager* tidak dapat di-*retrieve*.

Berikut ini adalah gambar diagram dari CDM (*Conceptual Data Model*) *database Cube* Editor :



Gambar III-13 CDM Database Cube Editor

Dari CDM tersebut jika di-generate kedalam PDM (*Physical Data Model*) untuk mendapatkan relasi *database* yang sesungguhnya, hasilnya adalah sebagai berikut:



Gambar III-14 PDM Database Cube Editor

Dari diagram PDM diatas dapat diketahui alur penyimpanan skema dari *source table* ke dalam *database Cube* Editor, dimana ketika *object* dibuat maka secara otomatis proses penyimpanan juga harus dilakukan bersamaan dengan *step by step* pembuatan skema *cube*.

3.4.2 Struktur XML

Hal yang perlu diperhatikan dalam komunikasi *client server* adalah masalah pengiriman dan penerimaan *string* XML. Hal yang mempengaruhi kinerja sistem adalah ukuran *file* transfer dan jarak yang ditempuh dari *client* ke *server*. Semakin besar ukuran *file* dan jarak yang ditempuh, semakin kurang optimal kinerja sistem. Struktur kerangka file XML yang digunakan pada aplikasi secara umum hampir sama, karena fungsi XML disini hanya sebagai *object* untuk mem-*parsing* semua data dan variabel yang dibutuhkan oleh aplikasi.

Berikut ini adalah beberapa struktur XML yang digunakan sebagai media penyimpanan sementara untuk menghubungkan aplikasi dari *client* ke *server* :

- Struktur XML untuk menampilkan data source dari database server

```
<DBList label = "Data Source in SQL Server 2000">
    <Database label = "">
      <Tabel id="" label="">
            <Column id="" label="" type="">
            </Column>
      </Tabel>
    </Database>
</DBList>
```

- Struktur XML untuk menampilkan source column dari database server

```
<ColumnList>
      <Column id="" label="" type="">
      </Column>
</ColumnList>
```

- Struktur XML untuk menampilkan skema cube dari Analysis Manager

```
<Cube name="" column=" |  | ">
      <Dimension name="" column=" | | " key=""
position="">
      </Dimension>
</Cube>
```

- Struktur XML untuk menampilkan cube list dari analysis server

```
<CubeList>
      <Cube label="">
      </Cube>
</CubeList>
```

- Struktur XML untuk menampilkan skema cube dari analaysis server ke aplikasi Flash

```
<CubeList>
      <Cube name="" column=" | | |">
            <Dimension name="" column="" key=""
position="">
            </Dimension>
      </Cube>
</CubeList>
```

BAB IV

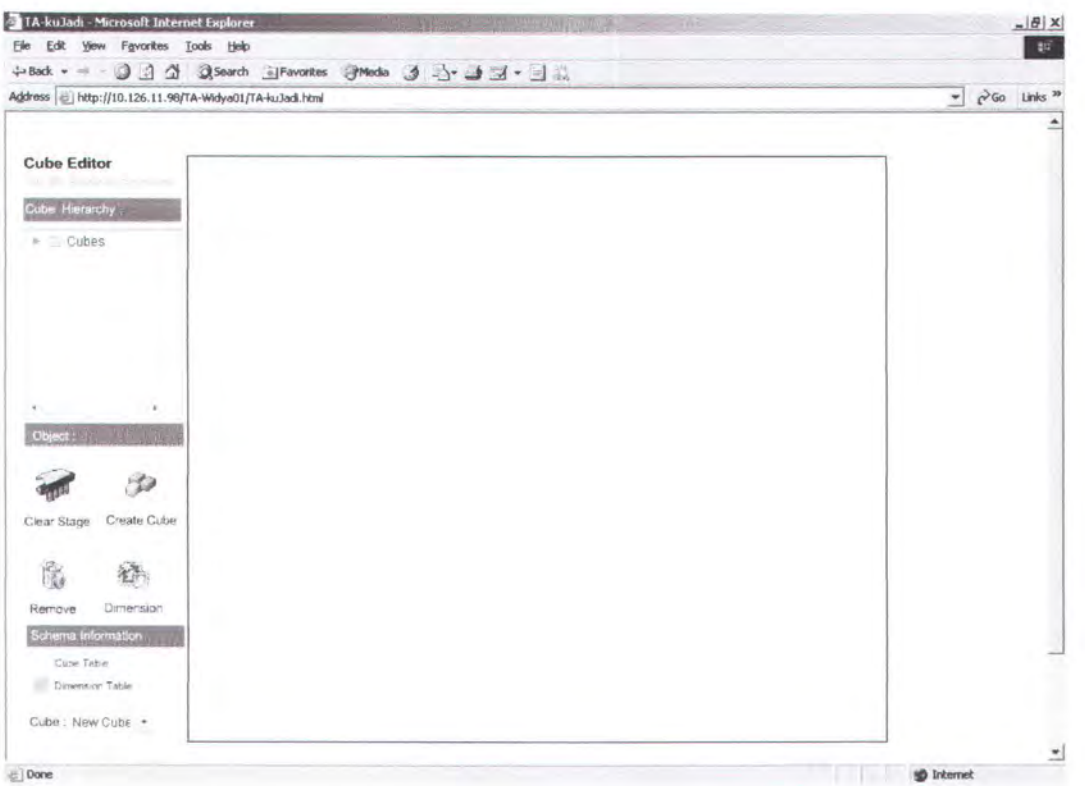
IMPLEMENTASI PERANGKAT LUNAK

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

Pada pembahasan perancangan perangkat lunak telah digambarkan secara umum mengenai rancangan sistem aplikasi pemodelan database fisik berbasis web. Pada bab ini akan dijelaskan implementasi dari rancangan yang telah dibuat.

4.1 GUI FLASH MX SEBAGAI CLIENT

Antar muka adalah sarana yang berfungsi sebagai perantara antara pengguna dengan sistem. Antar muka diharapkan dapat mempermudah pengguna dalam memakai sistem. Oleh karena itu, antar muka yang baik sangat dibutuhkan untuk setiap sistem.



Gambar IV-1 Tampilan utama Editor Skema dan Cube

PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Aplikasi *client* ini dibuat dengan menggunakan Macromedia Flash MX 2004, untuk menampilkan aplikasi yang dinamis dengan besar file yang cukup kecil. Bahasa pemrograman yang digunakan berupa *actionScript* yang mana pada dasarnya bentuk penulisannya hampir sama dengan struktur penulisan bahasa pemrograman *Java.*

Fungsi dan prosedur tidak dijelaskan secara detail karena akan panjang sekali, namun dijelaskan sebagian sebagai gambaran untuk fungsi yang lain.

4.1.1 Fungsi Pembuatan Object Koneksi

Untuk membuat suatu *object* maka pada *workspace* dibuat *movieClip* dimana didalamnya terdapat *object* XML. Dimana bentuk dari *tag* XML sudah dijelaskan sebelumnya pada bab 3. Jadi apabila kita menghapus *movieClip* tersebut, secara otomatis *object* XML dari *movieClip* tersebut akan ikut hilang. Terdapat dua fungsi utama yang dijalankan, yaitu *function* attachMovie[*movieClip*]. Berfungsi untuk menciptakan *movieClip* pada *workspace.* Yang kemudian dilanjutkan dengan pemanggilan fungsi yang bertujuan untuk membuat Object XML untuk *movieClip* tersebut. Yaitu *function* loadCube[*movieClip*] untuk gambar tabel *Cubenya* dan loadDimension[*movieClip*] untuk gambar tabel dimensi.

Fungsi loadCube*[movieClip]* dan loadDimension[*movieClip*] pada dasarnya adalah fungsi pembuatan object XML untuk *movieClip* yang diduplikat pada *workspace*. Bentuk dari struktur *object* XML yang dibuat telah dijelaskan sebelumnya pada bab 3.

4.1.2 Fungsi yang Menangani Gambar Garis

Untuk membuat skema relasi antara *fact table* dengan tabel-tabel dimensinya, diperlukan garis relasi yang menghubungkan antara *primary key* dengan *foreign key*-nya. Fungsi yang berisi *action script* untuk menangani garis tersebut adalah sebagai berikut:

```
aff_createLine("cb"+xml1.firstChild.attributes.name,"dim"+xm
lDim.childNodes[i].attributes.name,xmlDim.childNodes[i].attribute
s.position,1);
```

Karena untuk menghubungkan antara tabel dimensi dengan tabel *cube*-nya perlu diketahui posisi index dari masing-masing *primary key*-nya pada *listbox*, maka fungsi aff_createLine memiliki parameter-parameter sebagai berikut:

Tabel IV-1 Keterangan fungsi gambar garis

Parameter	Keterangan
"cb"+xml1.firstChild.attributes.name	Nama *movie clip* yang menggambarkan tabel *Cube* berisi dengan *listBox* dan *dynamic text*
dim"+xmlDim.childNodes[i].attributes.name	Nama *movie clip* yang menggambarkan tabel dimensi, bersifat dinamis karena dalam skema jumlah dimensi bisa berubah
xmlDim.childNodes[i].attributes.position	Koordinat kolom pada tabel *Cube* yang memiliki *reference* ke tabel dimensi (*foreign key*)
1	Koordinat *primary key* pada tabel dimensinya, dimana dalam aplikasi selalu diset berada pada kolom teratas dengan nilai index = 1

Garis yang dibuat dengan fungsi ini dapat mengikuti *object-object movie clip* yang dihubungkannya yaitu tabel *cube* dan tabel Dimensi ketika di-*drag* ataupun ketika *listBox*-nya di *scroll* naik atapun turun. Ketika *object movie clip* tersebut di-*drag* maka *line* atau garisnya akan di-*redraw* ulang, menghitung posisi koordinat *object* dan bisa berpindah secara dinamis.

4.1.3 Fungsi yang Menangani Workspace

- *Fungsi advPopup*

  Fungsi ini berfungsi untuk membuat *popup window* yang meminta masukan dari pengguna dengan menge-*load movie clip* berupa *form* yang telah dibuat berurutan dalam *frame* sesuai urutan langkah pengerjaan untuk membuat sebuah skema *cube* dari *database*.

- Fungsi *Remove*

  Fungsi ini memanggil *form* yang meminta masukan dari pengguna, jika ingin menghapus sebuah skema *cube* yang sudah dibuat sebelumnya.

- Fungsi *Clear Stage*

  Fungsi ini berfungsi untuk membersihkan *stage* dari *movie clip- movie clip* atau dengan kata lain sebagai *clear screen.*

```
for(var i in stage_mc){
        stage_mc[i].removeMovieClip();
        };
```

- Fungsi *Refresh*

  Fungsi ini akan melakukan *loading* ulang data XML yang sudah ter-*update* karena proses masukan yang telah dilakukan pengguna. Untuk setiap komponen yang berbeda maka masing-masing dibuat fungsi sendiri yang terpisah.

```
function refresh_skema()
{
  skema_int = setInterval(function(){
  xml1.load("http://10.126.11.98/TA-Widya01/cubeList.asp");
  clearInterval(skema_int);
  },1000);
}
```

- Fungsi *Add Dimension*

  Fungsi ini berfungsi untuk menambahkan dimensi baru kedalam skema *cube* yang sudah ada. Jika id session cube dalam *root*-nya kosong maka fungsi

akan mengembalikan nilai *false* berupa *warning* bahwa belum ada skema *cube* yang dibuka. Jadi fungsi ini hanya berjalan jika pengguna sudah membuka sebuah skema *cube* pada *stage* melalui *comboBox*.

4.1.4 Fungsi Yang Menangani Form

Form-form yang dibuat dalam bentuk *movie clip* diletakkan dalam *frame-frame* yang berada dalam sebuah *movie clip* lagi secara berurutan sesuai urutan langkah-langkah yang diperlukan untuk membuat sebuah skema *cube* dari *database*.

Di setiap *form* terdapat tiga fungsi utama yaitu :

1. Fungsi *Loading*, Berguna untuk menginisialisasi nilai dari properti-properti yang terdapat pada *form* tersebut. Nilai inisialisasi tersebut didapat dari membaca atau mem-*parsing object* XML dari *Object movieClip* yang me-*reference form* tersebut.
2. Fungsi *Ok*, Merupakan fungsi yang berguna untuk menyimpan masukan dari pengguna menjadi isi dari *object* XML yang sudah diciptakan sebelumnya.
3. Fungsi *Cancel*, dengan fungsi ini maka *movieClip* form yang ditampilkan oleh *popup window* akan dihapus beserta semua masukan pengguna.

```
_root.deletePopup () ;
```

Untuk dapat berhubungan dengan aplikasi *server flash* menyediakan beberapa fasilitas. Pada aplikasi ini digunakan 2 fasilitas yaitu :

1. [XML].**Load** (url)

   pemanggilan fungsi *[xml].load* akan mengirim variabel ke suatu url di server dengan metode *GET* , kemudian menerima hasil dari pemanggilan *url* tersebut dalam bentuk tag XML. Setiap fungsi *[xml].load* diikuti dengan pemanggilan fungsi *[xml].onload,* yang

berfungsi untuk mem-*parsing* hasil keluar *url* yang berbentuk *tag* XML untuk dipergunakan lebih lanjut.

Pada aplikasi, fungsi ini digunakan untuk mendapatkan informasi skema *database* dari *server*, *list of data source*, *list of cube*, dan semua variabel yang di-*load* dari *database* sesuai dengan pilihan pengguna.

2. **fscommand** (*command, parameter*)

   pemanggilan fungsi *fscommand* akan memanggil suatu fungsi tertentu pada aplikasi *server* yang sesuai dengan *command* dan *parameter* yang diberikan. Hasil keluarannya hanya dapat mempengaruhi variabel tertentu pada *flash*.

   Pada aplikasi, fungsi ini digunakan untuk pembuatan skema pada SQL *Analysis Manager*, penyimpanan data dan pengambilan variable dari SQL *Enterprise Manger*, *Create File* XML, *window Pop Up*, dan untuk mengeluarkan peringatan apabila terjadi *error* pada penggunaan aplikasi.

4.2 IMPLEMENTASI APLIKASI SERVER

Pada aplikasi server terdapat beberapa hal utama yaitu:

1. Aplikasi untuk menampilkan semua skema dari *database* yang dibutuhkan sebagai data *source*.
2. Aplikasi untuk menyimpan semua variabel yang diinputkan pengguna selama proses pembuatan skema *cube*.

3. Pengambilan informasi skema *cube* dari *database server*.
4. Penulisan XML.
5. Pembuatan skema *cube*
6. Fungsi untuk *remove object* pada *server*.

4.2.1 Aplikasi untuk menampilkan semua skema dari database yang dibutuhkan sebagai data source

Langkah pertama dalam membuat skema *cube* dari sebuah *datawarehouse* adalah menampilkan semua skema *database* yang dapat dijadikan data *source*, dalam hal ini hanya dibatasi pada SQL *Server* saja. Berikut ini adalah *script* atau *pseudocode* untuk menampilkan *list database* yang ada pada SQL *Server* yang dapat digunakan sebagai data *source* oleh pengguna yang dibuat dengan menggunakan object SQL DMO.

```
<%
deklarasi variable dan object SQL DMO

'create object SQL DMO
set dmosqlsvr = createobject("sqldmo.sqlserver")

dmosqlsvr.Connect strserver,strloginID,strpassword
      for each dmodatabase in dmosqlsvr.Databases
            if not dmodatabase.name = "Cube Editor" then
            for each dmotable in dmodatabase.Tables
                  If not dmotable.systemobject then
                        For Each dmocolumn in dmotable.Columns

                        Next
                  End if
            Next
            End If
      next

dmosqlsvr.Close
set dmoFilefroup = nothing
set dmodbfile = nothing
set dmologfile = nothing
set dmodatabase = nothing
set dmotable = nothing
set dmocolom  = nothing
set dmosqlsvr = nothing

%>
```

4.2.2 Aplikasi untuk menyimpan semua variable yang diinputkan user selama proses pembuatan skema Cube

Aplikasi ini terdiri dari banyak fungsi dan *procedure*, dimana setiap fungsi dan *procedure* tersebut dibuat dalam halaman *asp* tersendiri untuk menjalankan perintah yang berbeda-beda karena proses penyimpanan variabel dilakukan setiap kali pengguna berpindah dari *form* ke *form* berikutnya maka variabel yang sudah dimasukkan dari *form* sebelumnya akan disimpan kedalam *database*. Proses penyimpanan ini dibuat terpisah karena *object* yang disimpan berbeda-beda dan variabel pada *form* sebelumnya menentukan nilai dari variabel yang akan ditampilkan pada *form* berikutnya.

Beberapa fungsi dan *procedure* penyimpanan yang digunakan untuk menyimpan variabel-variabel pengguna, yaitu simpanCube.asp, simpanTable.asp, simpanMeasure.asp, simpanLevel.asp, simpanDimension.asp, dan simpanTableDim.asp. Karena fungsi dan *procedure* yang lain hampir sama hanya objectnya saja yang berbeda maka contoh *pseudocode* yang cukup untuk mewakilinya adalah simpanCube.asp :

```
<%
->Deklarasi variable
->Create object ADODB Connection dan Recordset

->connect ke database server
->ambil variabel dari flash
->update cube name pada table Cubeku di database Cube Editor

->Create Object(process.classCube)
Object.BuildCube id,namaDataSource,idCube,namaCube,namaTabel

Set object=nothing
Server.close
%>
```

4.2.3 Pengambilan informasi skema cube dari database server

Pengambilan informasi skema *database* dari *server* digunakan *Object* SQL DMO dan DSO serta *object* ADO MD yang terdiri dari beberapa fungsi dan *procedure* dalam halaman skema.asp, cubeList.asp, listCube.asp, dan listDimension.asp.

Berikut ini adalah contoh *script* atau *pseudocode* dari cubeList.asp untuk menampilkan skema *cube* yang telah dibuat pengguna ke *tree* supaya terlihat susunan hirarki dari *cube-cube* yang ada di *server Analysis Manager* :

```
<%
->Deklarasi variable dan object ADOMD
->Connect ke Analysis Server
OLAPServerName = "SERVER-TA"
cat.ActiveConnection = "Data Source=" & OLAPServerName &
";Initial
Catalog=CubeEditor;Provider=msolap;Uid=widya;Pwd=hanifku;"

   ➔ query object ADO MD dari server
   for i=0 to cube.count-1
      ->tampilkan dalam bentuk XML
      For a=0 to measure.count-1
            ->tampilkan dalam bentuk XML
      Next
      For j=0 to dimension.count-1
            ->tampilkan dalam bentuk XML
            For k=0 to level.count-1
                  ->tampilkan dalam bentuk XML
            Next
      Next
   Next

%>
```

4.2.4 Penulisan XML

Penulisan XML dilakukan oleh halaman asp yang me-*retrieve* setiap informasi dari dan ke *database* dengan menggunakan *contentype* XML yang disediakan oleh *asp*. Jadi meskipun filenya adalah .asp tetapi *output*-nya berupa

tag-tag XML. Hampir semua halaman asp yang menampilkan *output* ke layar *stage* pada GUI *Flash*, memberikan *output* berupa XML karena untuk komunikasi antara GUI *Flash* dan *database server* maupun *Analysis Manager* hanya bisa dilakukan dengan dijembatani oleh XML karena keterbatasan pada aplikasi *Flash* yang hanya bisa membaca struktur data berupa *text* ataupun XML. Berikut ini adalah salah satu contoh penulisan XML oleh halaman *.asp* yang menghasilkan *tag-tag* XML untuk digunakan sebagai informasi pada *comboBox* untuk menampilkan daftar *Cube* yang telah dibuat di *Analysis Manager* yaitu listCube.asp :

```
<%@ Language=VBScript %>
<%
dim serverku, constr , rec, sql, maxdb, max,
databaseName,sqlku, sqlmu, saya, gue, maxtbl
Dim xml
response.ContentType = "text/xml"
xml = "<?xml version='1.0' encoding='windows-1252'?>"

set serverku = server.CreateObject("ADODB.Connection")
set rec = server.CreateObject("ADODB.Recordset")

constr = "Provider=SQLOLEDB.1;Persist Security
Info=False;User ID=sa;Initial Catalog=Cube Editor;Data
Source=SERVER-TA;Password=widyacakep"
serverku.ConnectionString = constr
serverku.open

xml = xml + "<CubeList>"
xml = xml + ("<Cube label=""New Cube"">")
xml = xml + "</Cube>"
'ambil nilai maksimal dari id_Database
sql="select CubeName from Cubeku"
rec.Open ""& sql &"",serverku

while not rec.EOF
      xml = xml + ("<Cube label=""" + rec(0) + """>")
rec.movenext
xml = xml + "</Cube>"
wend
xml = xml + "</CubeList>"

response.Write(xml)

rec.close
%>
```

Halaman listCube.asp diatas jika di-*run* akan menghasilkan *output* berupa *tag-tag* XML sebagai berikut :

```
<?xml version="1.0" encoding="windows-1252" ?>
 <CubeList>
      <Cube label="New Cube" />
      <Cube label="penawaran" />
 </CubeList>
```

dimana *tag-tag* tersebut akan dibaca oleh komponennya *Flash* yaitu *Combo Box* UI *Component*.

4.2.5 Pembuatan skema cube

Untuk membuat skema *cube* , langkah awal yang dilakukan adalah memanggil *class* yang telah dibuat pada VB *Active* X dll yaitu *CubeEditor.dll* karena keterbatasan security pada asp yang tidak memungkin untuk melakukan penulisan *object* DSO pada *Analysis Manager*. *Class* yang dipanggil untuk membangun skema *cube* adalah *buildCube.cls*. *Class* tersebut dipanggil oleh halaman .asp. Berikut ini merupakan isi dari *class buildCube.dll* yang didalamnya terdiri dari banyak fungsi dan *procedure* untuk menjalankan proses pembuatan skema dan *cube* pada *Ms. Analysis Manager* :

1. fungsi *connect* ke *server Analysis Manager*

```
Public Sub ConnectToServer()

    Set m_dsoServer = New DSO.Server
    m_dsoServer.Connect "server-TA"

Exit Sub
```

2. fungsi untuk *connect* ke *database* yaitu *CubeEditor*

```
Public Sub ConnectDatabase()
    Dim sDatabaseName As String
    sDatabaseName = "CubeEditor"
        Set m_dsoDatabase =
m_dsoServer.MDStores(sDatabaseName)
Exit Sub
```

3. fungsi untuk *connect* ke *datasource* yang dipilih oleh pengguna

```
Public Sub ConnectDatasource()

    Dim dsoDatasource As DSO.DataSource

    If m_dsoDatabase.DataSources.Find(dataSourceName)
Then
        Set dsoDatasource =
m_dsoDatabase.DataSources(dataSourceName)
    Else

        Set dsoDatasource =
m_dsoDatabase.DataSources.AddNew(dataSourceName)
        dsoDatasource.ConnectionString = _
        "Provider=SQLOLEDB.1;Persist Security
Info=False;Initial Catalog=" + dataSourceName + ";Data
Source=Server-TA;Connect
Timeout=15;Trusted_Connection=yes"

        dsoDatasource.Update
    End If

Exit Sub
```

4. fungsi untuk membangun tabel-tabel dimensi dan level *member*-nya

```
Public Sub CreateDimension()

    Deklarasi semua variable dan object ADO, DSO
    ->Query dimension dari database
    ->Hitung jumlah dimensinya
    ->Deklarasi ulang isi array DsoDimension(k)
    Do While Not rec.EOF
        Set dsoDimension(i) =
m_dsoDatabase.Dimensions.AddNew(rec(0))
        With dsoDimension(i)
            ➔ create description dan propertinya
            ➔ join dengan Data Source
        end With
      -> query level dari database
      -> Deklarasi Ulang isi dsoLevel(kl)
      Do While Not rec1.EOF
          Set dsoLevel(j) =
      dsoDimension(i).Levels.AddNew(rec1(0))
          With dsoLevel(j)
          ->create member key column
          ->create property dari level dimension
      rec.movenext
   Loop

 Exit sub
```

5. fungsi untuk membangun cube dan menghubungkan dengan dimensi serta membuat measure untuk cube-nya

```
Public Sub CreateCube()
 deklarasi semua variable dan object DSO serta ADO
    Set m_dsoCube =
m_dsoDatabase.MDStores.AddNew(CubeName)
    With m_dsoCube
      ->create description
      ->add cube dengan data source
      ->ambil source table dari cubenya
      ->hitung EstimatedRows
      ->ambil nama dari dimension
      ->itung jumlah recordnya/ berapa banyak
dimensinya
        ReDim dsoCubeDim(k)
        Do While Not rec3.EOF
           Set dsoCubeDim(n) =
.Dimensions.AddNew(rec3(0))
           n = n + 1
           rec3.MoveNext
        Loop
        ->list semua tabel dimensinya
        ->join semua table dimension dengan cubenya
    End With
->Create measurenya
        ReDim dsoMeasure(k)
        Do While Not rec6.EOF
        Set dsoMeasure(j) =
m_dsoCube.Measures.AddNew(rec6(0))
            With dsoMeasure(j)
                  -> joinkan dengan source column dan
primary column
                  -> create property Measure
            End With
        rec6.MoveNext
            Loop
    m_dsoCube.Update
Exit Sub
```

4.2.6 Fungsi untuk remove object pada server

Untuk menghapus semua atau *object-object* tertentu dari *server*, dibedakan antara *Analysis Server* dengan *Database Server* karena metode dan fungsi untuk me-*remove object* tersebut tidak sama. Untuk itu ada 2 fungsi *remove*, yaitu :

1. *remove object* pada *Analysis Manager*

untuk menghapus *object* pada *server Analysis* digunakan *object* DSO yang dibuat *class* tersendiri dengan menggunakan *Active* X yaitu *classRemove*.cls sebagai berikut :

→ untuk menghapus tabel dimensi

```
m_dsoDatabase.Dimensions.Remove (object)
```

→ untuk menghapus *cube*

```
m_dsoDatabase.MDStores.Remove (Name)
```

2. *remove object* pada *Enterprise Manager*

   untuk menghapus *object* pada *server database* digunakan *object* ADO biasa dengan perintah query "DELETE FROM ... " untuk menghapus isi *database* yang menyimpan data masukan pengguna selama proses pembuatan skema *cube*.

```
delete from databaseku where id_database='idnya'
```

4.3 LINGKUNGAN IMPLEMENTASI

Dalam implementasi untuk aplikasi editor skema dan *cube* ini, sistem perangkat keras yang digunakan untuk aplikasi *client* tidaklah perlu terlalu besar. Dalam pengembangan aplikasinya digunakan : *Pentium* 4 1,7 GHz, *RAM* 1 GB dan *Hardisk* 20 GB. Namun dengan spesifikasi yang lebih tinggi (dan sekarang sudah umum), aplikasi dipastikan berjalan dengan lebih cepat.

Aplikasi hanya dapat berjalan dalam lingkungan *Microsoft Windows 2000 Advanced Server* sebagai sistem operasi *server*. Untuk sistem operasi yang berjalan di *client*, *Microsoft Windows* 98 sudah mencukupi dengan *Internet Explorer* yang sudah memiliki *Flash Plug In*. Untuk web *server* digunakan *Microsoft Internet Information Services* (IIS).

Aplikasi yang dipakai dalam pengembangan perangkat lunak :

1. *Macromedia Flash MX Professional* 2004 versi 7.2

2. *Active Server Page*
3. *Microsoft Visual Basic versi* 6.5 (*Service Pack* 5) untuk membuat dll melalui VB *Active* X dll
4. *Power Designer* 6.0 untuk pembuatan alur diagram proses pada aplikasi.
5. *Power Designer* 11.0 untuk pembuatan CDM dan PDM *database Cube Editor*.
6. *Adobe Photoshop* 5.0 untuk pembuatan grafik-grafik yang dibutuhkan oleh aplikasi.

BAB V

UJI COBA DAN EVALUASI

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai uji coba aplikasi *case tool* yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui apakah aplikasi dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba.

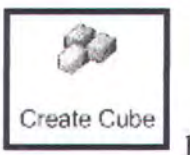
5.1 UJI COBA

Pada uji coba ini akan dibuat sebuah skenario analisa data dari sebuah *datawarehouse* yaitu *Northwind*. *Database Northwind* adalah *database* yang merupakan bawaan dari SQL *Server Enterprise Manager*. Pada *database Northwind* akan dilakukan analisa tentang pemesanan produk dari kastemer dan pegawai, dengan menggunakan tabel master *Orders* sebagai *fact table*-nya dan tabel dimensinya yaitu *Customers* dan *Employee*. *Database Northwind* hanya mempunyai tabel master *Orders* yang bisa dijadikan *fact table* yang memiliki *referential integrity* dengan tabel *Customers* dan *Employees*.

Proses uji coba yang akan dilakukan terbagi dalam 5 tahap :

1. Pembuatan skema *cube* Pemesanan dari sisi klien dan perbandingannya dengan pembuatan skema *cube* dari sisi *server*.
2. Membuka skema *cube* yang telah dibuat.
3. Melakukan perubahan dari skema *cube* yang telah dibuat dengan menambahkan tabel dimensi.
4. Membandingkan hasil perubahan skema *cube* dari sisi klien dan dari sisi *server*.

5.1.1 Pembuatan Skema Cube

Untuk membuat skema cube maka langkah pertama yang dilakukan adalah membuka *wizard* dengan mengklik tombol 

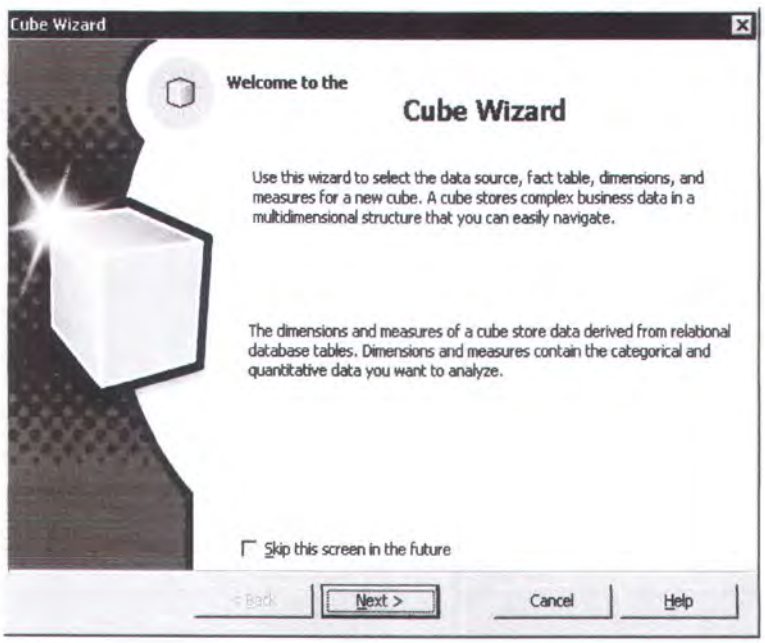
pada *toolbox Object*, kemudian akan keluar halaman *wizard* untuk membuat *fact table* dari *cube* untuk memilih properti-properti yang diperlukan. *Form-form* yang terlibat dalam *cube wizard* ini adalah sebagai berikut:

5.1.1.1 Form Cube Wizard

Gambar V-1 Form Cube Wizard

Kemudian klik tombol *next* untuk melakukan proses selanjutnya atau tombol *cancel* untuk membatalkan/menutup jendela *popup cube wizard.*

Sebagai perbandingan, dapat dilihat *from* selamat datang pada *cube wizard* milik SQL *Analysis Manager* sebagai berikut:



Gambar V-2 Form Cube Wizard pada server

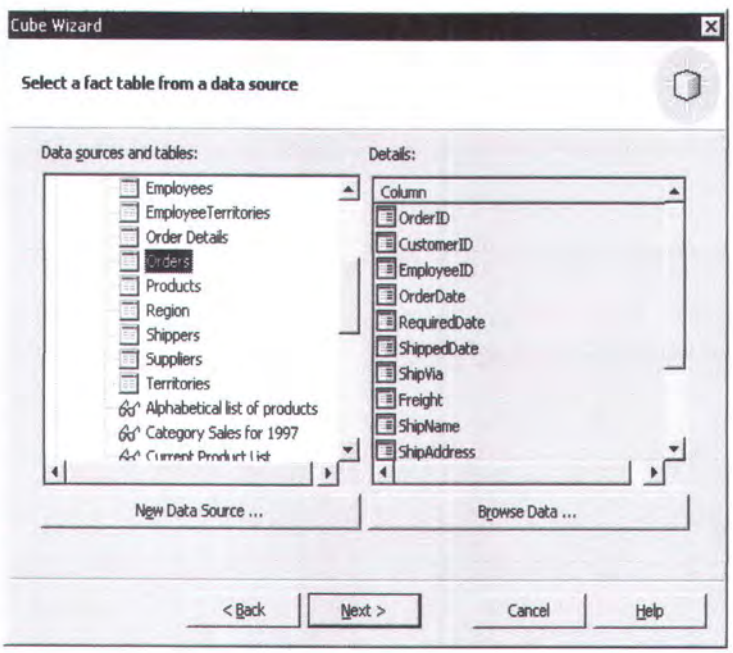
5.1.1.2 Form data sources and tables

Setelah mengklik tombol *next*, maka *form* selanjutnya adalah *form* data *sources and tables* yang merupakan *form* untuk mengambil masukan tabel yang dipilih pengguna untuk dijadikan *fact table* dari skema *cube* yang akan dibuat beserta data *source*-nya. Pada *form data source and tables* akan ditampilkan *list* dari *database* yang bisa digunakan sebagai data *source* untuk membuat skema *cube* (skema *datawarehouse*).

Sesuai dengan skenario yang direncanakan , maka *database* yang dipilih adalah *Northwind* dan tabelnya adalah *Orders*.

Gambar V-3 Form data sources and tables

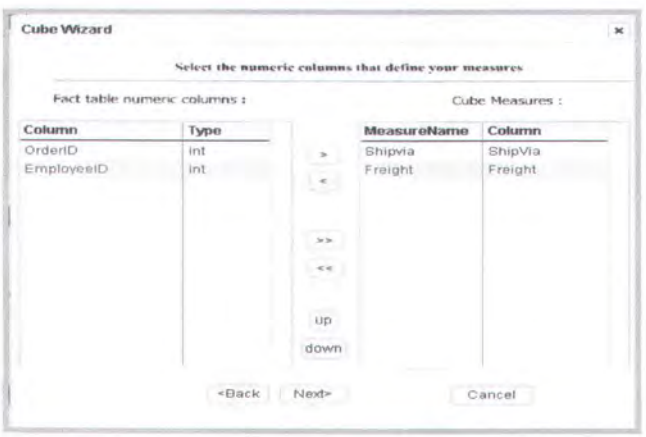
Terdapat perbedaan antara *form data sources and tables* pada aplikasi klien dengan *server*. Karena pada klien, data *source* yang ditampilkan hanyalah data *source* yang terdapat pada SQL *Enterprise Manager* saja sehingga menu untuk *add new data source* ditiadakan.



Gambar V-4 Form data sources and tables pada server

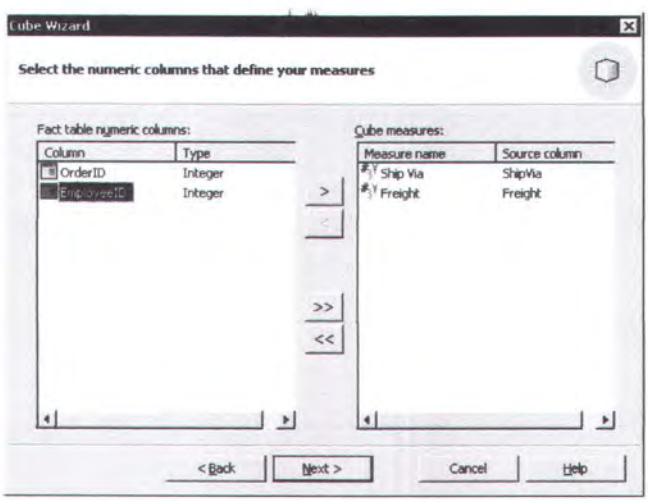
5.1.1.3 Form fact table numeric columns

Setelah diklik *next*, maka *form* selanjutnya adalah *form fact table numeric columns* yang digunakan untuk memilih *measure* dari *cube* yang akan dibuat. Sesuai dengan skenario yang direncanakan, maka *column* numerik yang dipilih adalah *ShipVia* dan *Freight*.



Gambar V-5 Form fact table numeric columns

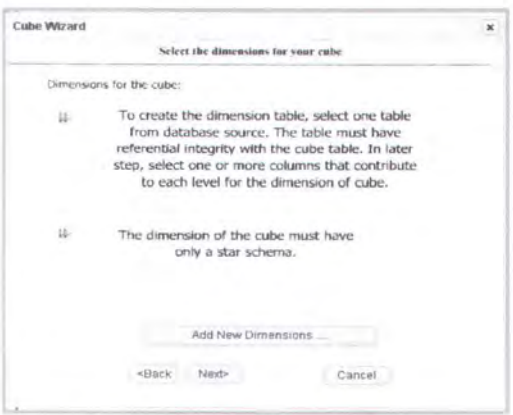
Sebagai perbandingan, berikut ini adalah *form fact table numeric column* pada sisi *server* :



Gambar V-6 Form fact table numeric column

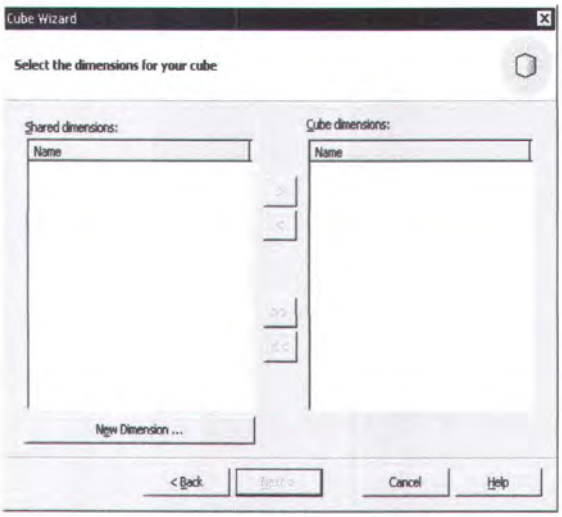
5.1.1.4 Form add new dimension

Setelah pengguna mengklik tombol *next* maka *form* selanjutnya adalah *form add dimension* yang berfungsi untuk memanggil *dimension wizard.* Didalam *form* ini tidak terdapat data *grid* untuk menampilkan dimensi yang sudah dibuat untuk *cube* tersebut, karena komponen dalam *flash movie* tidak bisa di-*refresh* dan tidak mempunyai *event* untuk meng-*handle* antara *form* satu dengan *form* yang lain.



Gambar V-7 Form add new dimension

Sebagai perbandingan, berikut ini adalah *form add new dimension* dari sisi *server* :



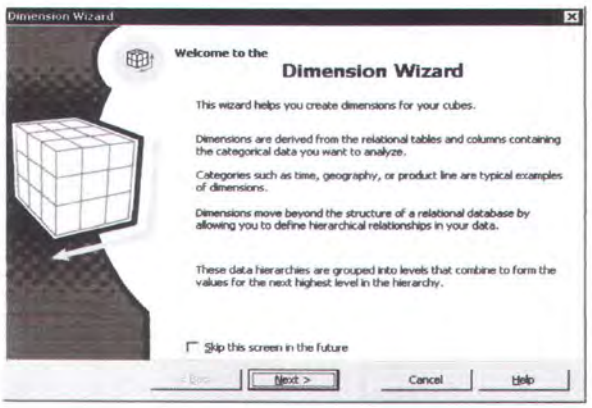
Gambar V-8 Form add new dimension pada sisi server

5.1.1.5 Form dimension wizard

Kemudian setelah pengguna mengklik tombol *add new dimension* , maka *form dimension wizard* yang berisi halaman selamat datang akan terlebih dahulu ditampilkan.

Gambar V-9 Form dimension wizard

Sebagai perbandingan, berikut ini adalah form dimension wizard pada sisi server.



Gambar V-10 Form dimension wizard pada sisi server

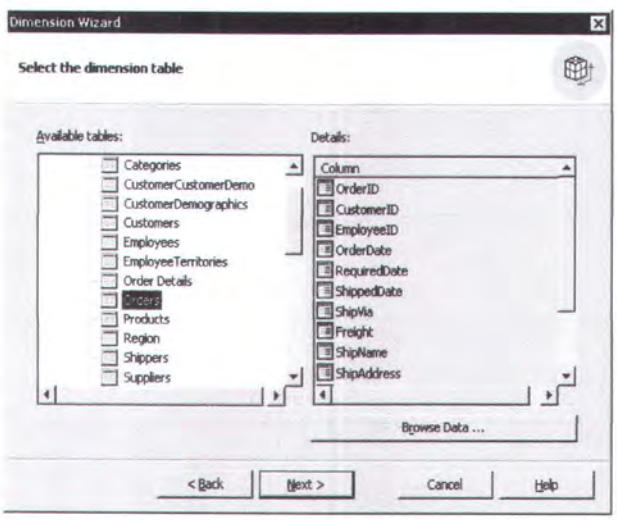
Sesuai dengan skenario yang telah direncanakan maka diperlukan dua tabel dimensi untuk *cube* Pemesanan yaitu tabel dimensi *Customers* dan *Employees*.

5.1.1.6 Form dimension table

Setelah diklik next, dari halaman selamat datang akan ditampilkan semua tabel dari *database* yang menjadi data *source* dari skema *cube* yang sedang dibuat. Sesuai dengan skenario yang direncanakan maka tabel dimensi yang dipilih adalah *Customers*.

Gambar V-11 Form dimension table

Berikut ini adalah perbandingan dengan *form dimension table* di sisi *server* :

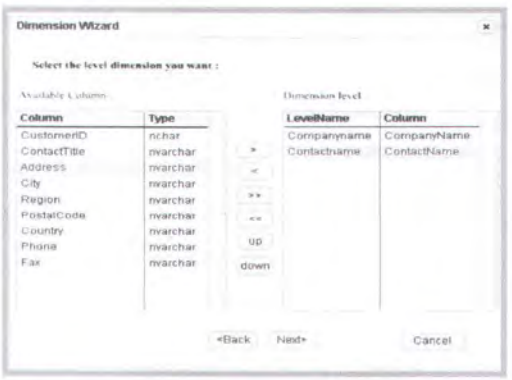


Gambar V-12 Form dimension table pada sisi server

5.1.1.7 Form level dimension

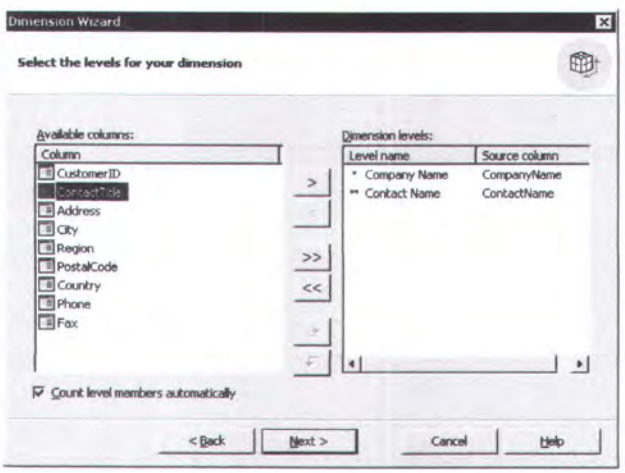
Form selanjutnya, setelah pengguna mengklik *next* adalah *form* level *dimension* yang merupakan *form* untuk menginputkan *column* yang digunakan sebagai level dari tabel dimensi. Sesuai dengan skenario yang telah direncanakan

maka level dimensi untuk tabel *Cutomers* adalah *CompanyName* dan *ContactName*.



Gambar V-13 Form level dimension

Sebagai perbandingan, berikut ini adalah *form* level *dimension* yang terdapat pada sisi *server*:



Gambar V-14 Form level dimension pada sisi server

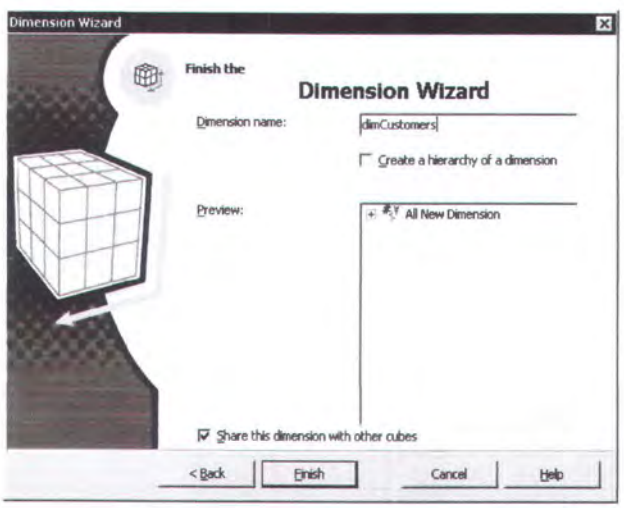
5.1.1.8 Form simpan dimension

Setelah menentukan level untuk tabel dimensi *Customers* maka langkah selanjutnya adalah menyimpan dimensi tersebut kedalam *database* dengan mengklik tombol *next* yang akan menampilkan *form* simpan *dimension*.

Kemudian sesuai dengan skenario yang telah direncanakan maka pada *inputBox*, dimensi dari tabel *Customers* diisi dimCustomers.

Gambar V-15 Form simpan dimension

Berikut ini adalah *form* simpan dimension dari sisi *server* sebagai perbandingan *form* pada sisi klien:



Gambar V-16 Form simpan dimension pada sisi server

Terdapat perbedaan pada tampilan *form* simpan *dimension*, yaitu pada sisi klien tidak terdapat menu untuk *preview* struktur hirarki dari dimensi yang akan dibuat pada *server*.

5.1.1.9 Form simpan cube

Setelah pengguna mengklik *finish* pada jendela *popup Dimension wizard*, maka jendela *popup* akan dihapus dari *layer*, dan hasil inputan nama dimensi akan ditampilkan pada data *grid* di *form add new dimension*. Kemudian, untuk menguji proses *add dimension* maka langkah berikutnya pengguna mengklik tombol *next* dan akan ditampilkan *form* simpan *cube*. Pada *form* simpan *cube* terdapat *inputBox* yang meminta inputan nama *cube* untuk skema yang akan dibuat.

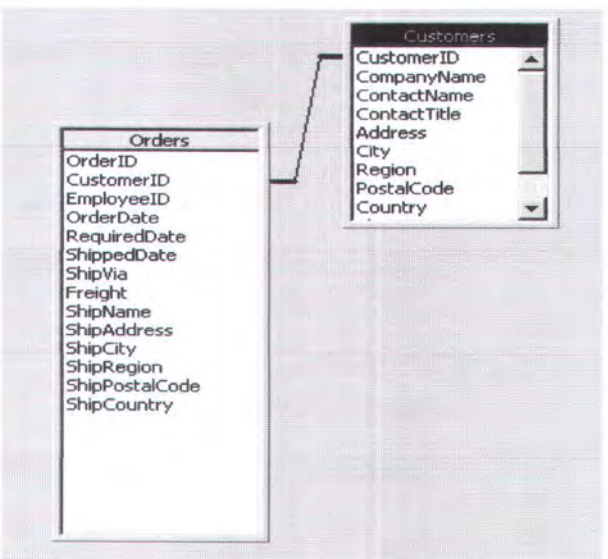
Sesuai dengan skenario yang telah direncanakan, maka pada *inputBox* diisi "Pemesanan".

Gambar V-17 Form simpan cube

Setelah pengguna mengklik tombol *finish* maka hasil skema *cube* akan ditampilkan sebagai berikut:

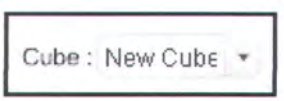
Gambar V-18 Hasil skema cube pada sisi klien

Sebagai perbandingan, maka hasil skema *cube* yang ditampilkan pada sisi *server* adalah sebagai berikut:



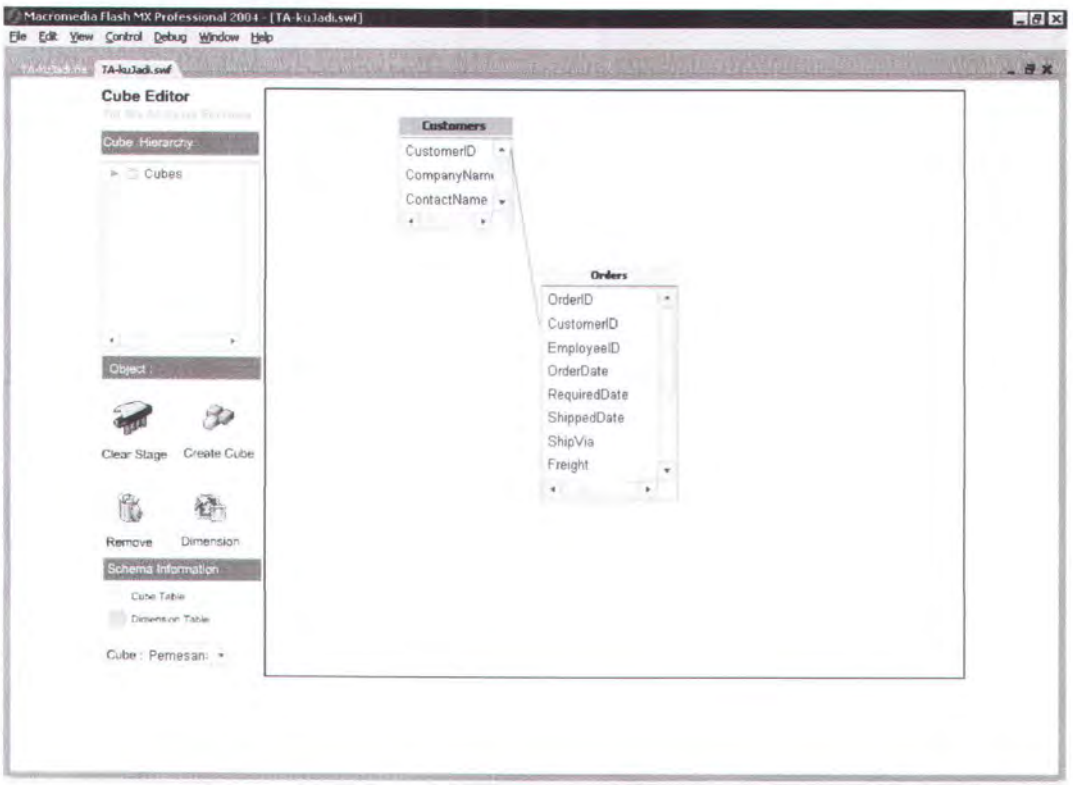
Gambar V-19 Hasil skema cube pada sisi server

5.1.2 Membuka skema cube yang telah dibuat

Jika pengguna ingin menampilkan kembali skema *cube* yang sudah dibuat sebelumnya, maka klik pada *comboBox* 

. Sesuai dengan

skenario yang telah direncanakan maka klik *item* pada *comboBox* yaitu Pemesanan maka skema *cube* akan ditampilkan kembali seperti berikut ini:



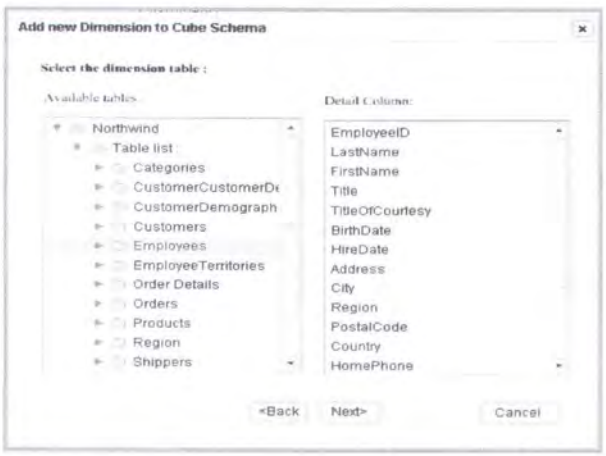
Gambar V-20 Tampilan skema cube setelah diklik dari combo box

5.1.3 Melakukan perubahan dari skema cube yang telah dibuat dengan menambahkan table dimensi

Untuk melakukan perubahan pada skema *cube* yang telah dibuat sebelumnya, maka pengguna harus membuka kembali skema tersebut melalui *comboBox*. Kemudian untuk menambahkan tabel dimensi ke dalam skema *cube*, klik pada tombol 

, yang akan menampilkan *form add table dimension.*

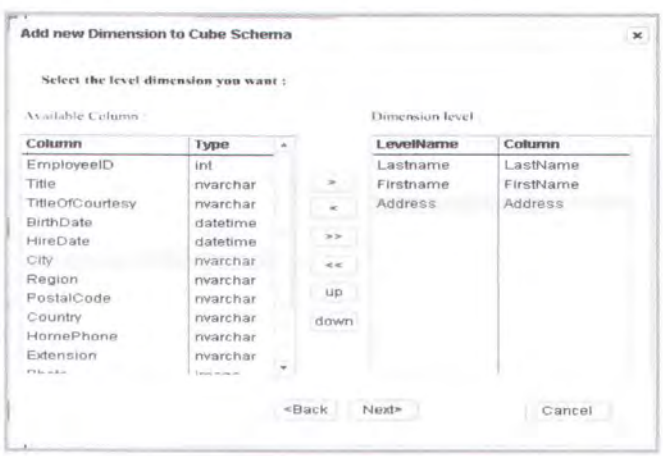
Sesuai dengan skenario yang telah direncanakan maka tabel dimensi yang dipilih adalah *Employees*.



PERPUSTAKAAN INSTITUT TEKNOLOGI SEPULUH - NOPEMBER

Gambar V-21 Form dimension table

Kemudian klik *next* untuk menentukan level dari tabel dimensi *Employee*. Sesuai dengan skenario yang telah dirancang maka level *column* untuk tabel dimensi *Employee* adalah *LastName*, *FirstName*, dan *Address*.

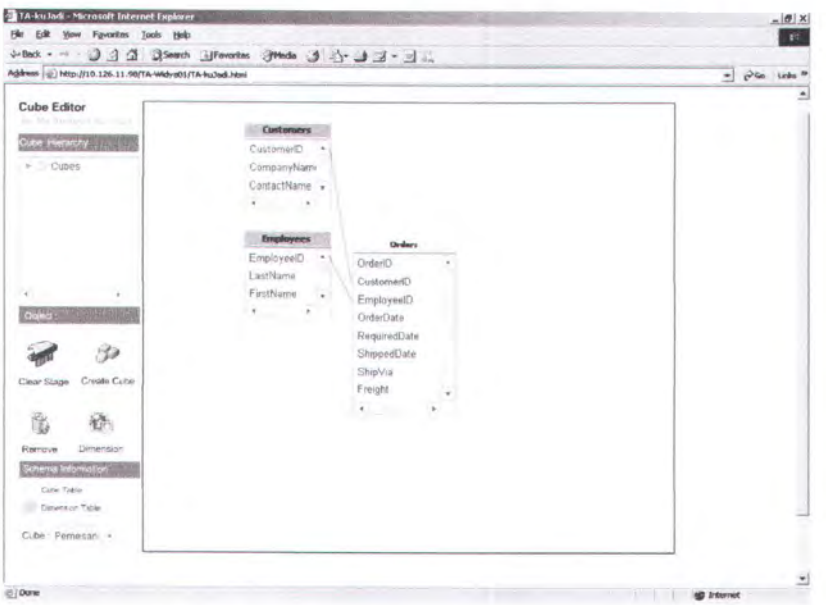


Gambar V-22 Form level dimension

Kemudian klik *next*, maka akan tampil *form* simpan *dimension* yang meminta input dari pengguna berupa nama dimensi. Sesuai dengan skenario yang telah dirancang maka *inputBox* diisi dimEmployees.

Gambar V-23 Form simpan dimension

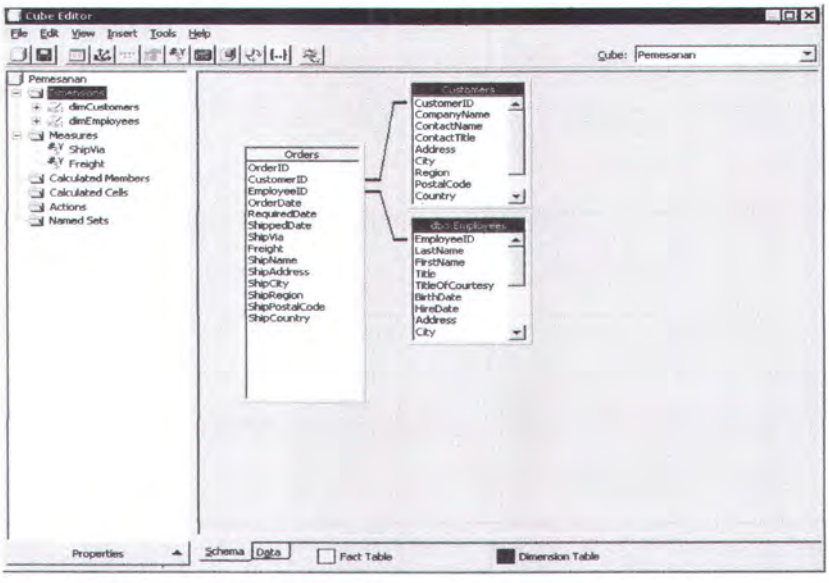
Setelah pengguna mengklik tombol *finish* maka skema *cube* akan ditampilkan kembali sesuai dengan perubahan yang dilakukan pengguna.



Gambar V-24 Tampilan akhir skema cube setelah di edit

5.1.4 Membandingkan hasil perubahan skema cube dari sisi klien dan dari sisi server

Sebagai perbandingan, berikut ini adalah hasil skema *cube* dilihat pada sisi *server* setelah dilakukan perubahan pada sisi klien :



Gambar V-25 Tampilan skema cube seteleh di edit pada sisi server

5.2 RESUME HASIL UJI COBA

Berikut ini merupakan hasil resume uji coba yang telah dilakukan sesuai dengan skenario yang telah direncanakan :

Tabel V-1 Hasil Resume Uji coba

No	Butir uji coba	Prosedur	Kriteria berhasil	Hasil uji coba
1.	*Create* data *source*	- pada *form* data *source and tables* pilih *database Northwind*	- tidak ada pesan error - *database* terkoneksi - data *source* ter-*create* - data source tersimpan pada *database*	sesuai
2.	Membuat *cube* (*fact table*)	- pada *form* data *source and tables* pilih tabel *Orders* - pada *form fact table numeric column* pilih kolom *ship via* dan *freight* sebagai *measure* dari *cube* - *measure* yang dipilih harus merupakan kolom *numeric* yang ada pada *fact table*-nya	-tidak ada pesan error - data *measure* yang dipilih user tersimpan dalam *database*	sesuai
3.	Membuat dimensi dari	- klik pada *button add*	- tidak ada pesan	sesuai

	cube	*new dimension*, maka *form dimension wizard* akanmuncul - lakukan berurutan sesuai dengan langkah-langkah untuk membuat dimensi dari *cube* - pada *form dimension table* pilih tabel *Customers* sebagai tabel dimensi dari cube *Orders* - pilih level dari dimensinya pada *form* level dimension, yaitu kolom *companyname* dan *contactname* - beri nama dimensinya dengan nama dimCustemers pada textbox di *form* simpan dimension	error - data tabel dimensi dan levelnya tersimpan pada *database*	
4.	Menyimpan *cube* dan mengeksekusinya pada *analysis server*	- isi nama *cube* "Pemesanan" pada textbox yang terdapat pada form simpan *cube* - klik *button finish*	- tidak ada pesan error - data *cube* tersimpan dalam database - *cube* ter-*create* pada *Analysis Server* - skema *cube* ditampilkan pada *stage* di aplikasi	sesuai
5.	Mengedit skema *cube*	- buka skema *cube* melalui combo box pada halaman utama - klik pada *button* dimension untuk melakukan perubahan pada skema *cube* dengan menambahkan table dimensi - pada *form dimension table* pilih tabel *Employees* - pada form level dimension pilih kolom *LastName*, *FirstName* dan *Address* - pada form simpan dimension, beri nama dimensinya dengan nama dimEmployees - klik tombol *finish*	- tidak ada pesan error - skema akan ditampilkan kembali sesuai dengan perubahan yang dilakukan yaitu dengan bertambahnya tabel dimensi dimEmployees	sesuai

5.3 EVALUASI

Evaluasi dilakukan dengan mengacu pada tahap uji coba dan dari tujuan dari pembuatan perangkat lunak. Pada sub bab ini hal-hal yang akan dievaluasi meliputi:

1. Pengeditan Pada *Workspace*
2. Pembuatan skema *cube*
3. Perubahan skema *cube* dengan menambahkan tabel dimensi baru ke dalam skema *cube*.
4. Hasil skema *cube* yang telah dibuat dari sisi klien.

5.3.1 Pengeditan Pada *Workpsace*

Evaluasi dilakukan dengan teliti dan berulang-ulang pada proses menggambar, meng-*Delete,* meng-*Cut* dan meng-*Copy movieClip*. Berikut merupakan hasil evaluasi dari uji coba yang telah dilakukan:

1. Tidak ada kesalahan struktur XML setelah dilakukan proses *cut, delete, copy* maupun *paste.*
2. Melakukan *Drag Drop* dan Scroll pada object *movieClip* yang diikuti dengan perubahan garis.

5.3.2 Pembuatan skema Cube

Dari hasil uji coba dimana pengguna membuat skema *cube* pada *workspace* tidak ditemukan adanya kesalahan. Pembuatan diagram berupa *movie clip* pada *stage* untuk menampilkan skema *cube* berjalan dengan baik. Semua *form* properti dari koneksi maupun *task* dapat di-*load* dengan sempurna. Hanya saja waktu yang dibutuhkan *request object* skema dari sebuah *server* tidak dapat

diperkirakan dengan pasti. Tergantung kelancaran jaringan dan kecepatan kerja dari masing-masing *server*.

5.3.3 Perubahan skema cube dengan menambahkan table dimensi baru ke dalam skema cube

Hasil evaluasi yang didapat dari prubahan skema *cube* yang telah dibuat sebelumnya dengan menambahkan tabel dimensi baru yang dibuat pada *workspace* di *client,* positif. Hal ini dapat langsung dilihat dari munculnya tabel dimensi baru pada *Client* dan *Server*. Hasil skema *cube* yang terbentuk pada dasarnya sama dengan skema *cube* pada *client* tetapi dengan bentuk diagram yang agak berbeda tetapi memiliki logika yang sama. Aliran data berjalan sesuai dengan *workflow* yang dibuat.

5.3.4 Hasil skema cube yang telah dibuat dari sisi klien

Setelah dilakukan proses pembuatan skema cube secara *remote* dari sisi *client,* hasil yang diinginkan adalah sama apabila dilakukan proses pembuatan skema *cube* secara langsung pada *server*. Dari sini dapat disimpulkan bahwa uji coba untuk pembuatan skema *cube* secara *remote* melalui web berhasil.

BAB VI

PENUTUP

BAB VI
PENUTUP

6.1 KESIMPULAN

Berdasarkan pada perancangan dan pembuatan sistem terhadap permasalahan yang diangkat, maka dapat diambil kesimpulan sebagai berikut :

1. Untuk dapat menggunakan object-object yang terdapat dalam DSO, tidak bisa dilakukan secara langsung melalui ASP karena masalah security sehingga perlu dibuat DLL sendiri yang menangani semua fungsi dan *method* dalam *object* DSO tersebut.
2. DLL yang mendukung untuk *Analysis programming* pada SQL *Server* tidak disediakan untuk web *scripting*.
3. *Movie clip* yang digunakan pada aplikasi Cube Editor membutuhkan banyak *flash movie library* dan *function-function* untuk menampilkan skema cube secara dinamis.

6.2 SARAN

Berikut ini beberapa saran pengembangan yang mungkin dilakukan :

1. Untuk memaksimalkan Aplikasi ini dapat dibuat *tools* tambahan berupa *browse* meta data dengan *pivot* tabel untuk SQL *Analysis Manager* yang kemudian diintegrasikan pada aplikasi ini. Sehingga akan lebih memudahkan pengguna untuk melakukan analisa data pada skema *cube* untuk *datawarehousing*.

2. Pengembangan menu *item* untuk editing skema *cube* dengan menggunakan metode agregasi yang lain seperti ROLAP (*Relational* OLAP) ataupun HOLAP (*Hybrid* OLAP).

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. Young Michael J. 2000. *Step By Step XML*. PT. Elex Media Komputindo. Jakarta
2. Frank Miller and Rachelle Reese, *Professional SQL Server 7.0 Development using SQL DMO, SQL NS and DTS*
3. Palinski, John Adolph. "*Oracle SQL and PL/SQL Handbook: A Guide for Data Administrators, Developers, and Business Analysts*". Pearson Education, Inc. *Indianapolis, IN* 2003
4. Wijaya, Didik & Andar Parulian Hutasoit. "*Macromedia Flash MX dengan ActionScript: Tip dan Trik*". Elex Media Komputindo. Jakarta. 2003.
5. Feri Djuandi, MCDBA, MCSE. *"SQL Server 2000 untuk Professional"*. PT Elex Media Komputindo. Jakarta 2002.
6. Lukmanul Hakim. *"111 Rahasia dan Trik Kreatif Macromedia"* ,Elex Media Komputindo. Jakarta 2003.
7. Garcia, et al, *"Microsoft SQL Server 2000 Administrator's Companion Tools"*, Microsoft Press , 2000
8. Robert Reinhardt, Joey Lott, *"Flash™ MX 2004 ActionScript Bible"*, Willey Publishing, Inc, 2004