PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK APLIKASI CASE TOOL PEMODELAN BASIS DATA KONSEPTUAL UNTUK RDBMS ORACLE BERBASIS WEB

TUGAS AKHIR

Disusun Oleh :

SAENAL MAS'UDI
5198 100 022

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2004

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK APLIKASI CASE TOOL PEMODELAN BASIS DATA KONSEPTUAL UNTUK RDBMS ORACLE 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. Aris Tjahyanto, M.Kom	Ir. Suhadi Lili
NIP : 131 933 299	NIP : 132 048 148

SURABAYA
JUNI, 2004

ABSTRAK

Perancangan desain sistem informasi dengan menggunakan pemodelan basis data secara konseptual telah dapat dilakukan dengan mudah oleh banyak perangkat lunak di pasaran yang khusus melakukan desain sistem. Perangkat-perangkat lunak tersebut merupakan case tool yang handal dalam mendesain sistem, namun pada umumnya perangkat lunak tadi hanya bersifat lokal. Mengingat perkembangan internet yang cepat dan tuntutan untuk efisiensi kerja, maka perlu dibuat suatu aplikasi case tool berbasis web yang mampu melakukan desain sistem dengan tambahan fitur-fitur yang belum ada yang disesuaikan dengan lingkungan web, seperti kemampuan mendesain jarak jauh, fasilitas desain secara multi user, auto refresh dan komunikasi antar pengguna.

Teknologi IIS ,ISAPI, ActiveX Control dan XML dapat digunakan untuk membuat fitur-fitur diatas. Aplikasi yang dibuat ini menggunakan database ORACLE 8i sebagai dasar untuk desain pemodelan data konseptual dan fisik dan untuk penyimpanan file-file data aplikasi.

Proses-proses yang terjadi dalam sistem ini adalah sebagai berikut. Pengguna melakukan editing diagram dan setiap perubahan yang terjadi akan dikirim ke server untuk kemudian diteruskan ke pengguna yang lain. Proses ini terjadi secara otomatis dan auto refresh. Proses yang sama juga terjadi ketika pengguna melakukan chatting. Proses penting lainnya adalah melakukan reverse dan forward engineering. Aplikasi akan mengambil model basis data fisik dari database Oracle di server atau dari file lokal hasil dari perancangan model fisik atau hasil proses forward engineering dan hasil reverse tersebut disimpan dalam repository database.

Aplikasi yang telah dibuat telah mengalami uji coba fungsionalitas dan kinerja. Untuk uji coba fungsional hasil yang ditunjukkan telah memenuhi kebutuhan dari pengguna aplikasi. Sedangkan untuk uji coba kinerja, aplikasi mampu berjalan dengan kecepatan yang stabil selama perangkat keras dan bandwidth yang digunakan cukup memadai.

KATA PENGANTAR

Bismillaahirrohmaanirrohiim. Alhamdulillah, segala puji bagi Allah atas segala nikmat yang telah diberikan. Sholawat dan salam tetap tercurah bagi Nabi Muhammad saw. Segala hal terjadi atas ijin Allah swt. Demikian pula dalam penyelesaian Tugas Akhir yang telah penulis susun dengan mengambil judul :

"PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK APLIKASI CASE TOOL UNTUK MELAKUKAN PEMODELAN BASIS DATA KONSEPTUAL UNTUK RDBMS ORACLE BERBASIS WEB"

Tugas Akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan pendidikan Strata Satu (S-1) di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan Tugas Akhir ini penulis berusaha menerapkan ilmu-ilmu yang telah didapatkan selama masa perkuliahan. Dan dengan selesainya Tugas Akhir ini penulis sangat berharap dapat memberikan manfaat bagi yang memerlukannya.

Dengan kesabaran dan kerendahan hati penulis menerima saran dan kritik apabila Tugas Akhir ini masih terdapat banyak kekurangan.

Surabaya, Juni 2004

Penulis

UCAPAN TERIMA KASIH

Dengan selesainya Tugas Akhir, penulis mengucapkan terima kasih dan penghargaan yang sebesar-besarnya kepada :

1. KHM. Dhiyauddin Qushwandhi yang dengan tulus mendoakan penulis agar senantiasa mendapat hidayah dari Allah. Sahabat-sahabat di Jama'ah Ihya'us Sunnah AnNabawiyyah, terutama buat para Ahlus Shuffah di Masjid Masyithoh Surabaya.
2. Ibu yang ada di rumah yang telah bekerja keras dalam membesarkan penulis.
3. Bapak Ir. Aris Tjahyanto,M.Kom yang dengan penuh kesabaran memberikan waktu dan bimbingan kepada penulis untuk menyelesaikan Tugas Akhir ini.
4. Bapak Ir. Suhadi Lili yang dengan baik hati telah memberikan waktu dan usahanya untuk membimbing serta memberikan dorongan pada penulis sehingga Tugas Akhir ini selesai dengan baik.
5. Ibu Ir. Esther Hanaya,MSc sebagai dosen wali yang telah membantu penulis sejak awal sampai akhir perkuliahan.
6. Dosen-dosen jurusan Teknik Informatika ITS atas segala hal yang telah diberikan selama masa perkuliahan.
7. Mas Saiful dan Mbak Nur beserta keluarga yang telah memberikan dorongan untuk segera menyelesaikan tugas akhir ini.
8. Adhitya Krisna, tetangga dan teman yang sangat banyak memberikan bantuannya dari awal hingga akhir masa kuliah.

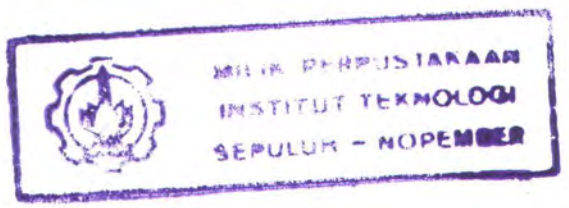
9. Lalu Ayatullah Amarzan, Widyo Wicaksono, Erwin Susanto, Mas Budi, Mas Kun, Pak Arief, Junaidi, Atet dan Adhitya sebagai mantan kolega yang baik dan lucu di IAO.
10. Agung Sedayu, Mas Cahyono, Cak Leman, Gus Nur, Mas Dillah, Cak Wawan, Mas Samsul, Mas Luhur, Mas Ramdhani, Hisyam, Adi, Achmadi, Firman, Into, Minuk, Munir, Irul, Didik 'Simbud', Didik 'Bendon', Amir, Syamson, Hadi Suwirno, Teteng, Deky, Denny, Tri, Widya, Mukhlis, Yayan, Suyuth, Isa serta teman-teman yang lain yang pernah menjadi penghuni di 'Rutan' Asrama Mahasiswa ITS.
11. Dhane Pratignyo, Harry Prasetyo, Imam Artha Kusuma, Nur Cahyo, Rossi Lazuardi, Agus Muliantara, Anang Kunaefi, Kilay, Surateno dan teman-teman C0E lainnya yang terlalu banyak untuk disebutkan satu persatu.
12. Rezin, Khalid, Deddy dan Hengkie dan teman-teman C0D yang lain.
13. Hoirul, Yudha, Eli, Aris 'Ketchu', Wawan dan adik-adik C0F dan C10 yang lain.
14. Mas Puji, Mas Cahyono, Astari, Bagus dan rekan-rekan yang lain mantan pengurus Unit Kegiatan Mahasiswa Judo ITS periode 2000/2001 serta untuk rekan-rekan mantan pengurus Lembaga Minat Bakat ITS periode 2000/2001.
15. Ibu Bakrie sebagai pemilik kos-kosan yang sangat baik. Edi, Agus 'Sa'ad', Udin, Sinchan, Yanto, Gus Mus, Barito, Putra 12 (Mas Billy, Mas Mudakkir, Joko, Thoriq, Rifa'i dan Widi) sebagai mantan teman

satu kos. Ichwan 'Walidji', Somad, Humaidi, Adi 'Gembul', Bagus, Anam, B-Jat, Gypton, Kriwul, Bun-Bun 'Gamer', Dylan, Novi, Dibyo, Timbul, Iwan, Sofyan, Tebe, Teddy, Ayik 'Pinuji', Dain, Ilmi, Mas Dody, Mas Andry, Adi 'Sumanto', Agus 'Dhuren', Nanang, Dikor, Mas Lukman, Wahyu dan Bowo sebagai masyarakat di Dji Sam Soe Apartment.

16. Teman-teman lain yang tidak dapat penulis sebutkan satu persatu yang pernah merasakan suka duka selama kuliah di ITS dan yang telah turut memberikan kontribusi atas selesainya Tugas Akhir ini.

DAFTAR ISI

DAFTAR GAMBAR



DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sejalan dengan semakin pesatnya perkembangan teknologi internet dewasa ini, perkembangan teknologi informasi terus meningkat dengan cepat dan dengan menimbulkan efek yang jelas. Persaingan dunia usaha terutama yang menggunakan teknologi internet tentunya meningkat pula.

Sistem informasi dalam sebuah badan usaha memiliki pengaruh besar terutama dalam efektifitas kinerja karyawan. Sehingga semakin baik sistem informasi yang dibuat dimungkinkan semakin baik pula kinerja dalam perusahaan. Namun perancangan sistem informasi yang baik tentunya lebih baik lagi jika didukung oleh *tools* yang baik pula terutama yang memudahkan bagi analis dalam mendesain sebuah sistem informasi.

Dalam membuat desain sistem dalam hal ini adalah desain pemodelan basis data secara konseptual akan sangat baik jika menggunakan case tool yang memperhatikan efisiensi kerja. Beberapa perangkat lunak case tool pemodelan basis data yang ada di pasaran sudah mampu untuk melakukan pengerjaan desain sistem. Namun belum memanfaatkan teknologi internet untuk membuat desain sistem jarak jauh.

Dengan menambahkan fitur-fitur baru yang memanfaatkan teknologi internet diharapkan pengerjaan desain sistem dapat dioptimalkan dan lebih efisien. Fitur-fitur tersebut adalah kemampuan untuk mendesain jarak jauh,

fasilitas *multi user* dalam melakukan desain. Dengan adanya tambahan fasilitas-fasilitas tersebut seorang analis tidak perlu melakukan banyak mobilitas dalam melakukan desain system dan dengan ini diharapkan pula analis akan merasa lebih fleksibel dalam merancang sistemnya sehingga akan meningkatkan kinerjanya.

1.2. Tujuan Tugas Akhir

Tujuan dari Tugas Akhir ini adalah membangun sebuah *case tool* untuk melakukan pemodelan basis data konseptual untuk RDBMS Oracle dengan berbasis web. Selain melakukan pemodelan basis data konseptual pada editor di Web Browser, aplikasi ini juga menyediakan fasilitas *reverse engineering* dari model fisik ke model konseptual dan *forward* engineering dari model konseptual ke basis data fisik.

1.3. Permasalahan

Permasalahan dalam Tugas Akhir ini adalah:

1. Bagaimana membentuk arsitektur *client server* yang sesuai dengan aplikasi yang dibuat.
2. Bagaimana melakukan *reverse engineering* sehingga dapat mengambil model fisik untuk ditampilkan sebagai model konseptual di editor di *client.*
3. Bagaimana membangun sistem aplikasi yang mampu menangani banyak pengguna secara simultan dalam suatu pekerjaan yang sama dengan tampilan yang mendekati *real time.*

4. Bagaimana mengatasi permasalahan konkurensi antar data yang dikirim oleh tiap pengguna sehingga tidak terjadi kekacauan dalam proses sinkronisasi model diagram antar pengguna.

1.4. Batasan Permasalahan

Aplikasi Case Tool yang akan dirancang dalam Tugas Akhir ini memiliki batasan-batasan sebagai berikut:

1. Obyek pemodelan yang akan ditampilkan pada editor dibatasi pada obyek-obyek umum seperti Entitas, relasi dan turunan.
2. Proses *update* pada editor *client* tidak sepenuhnya real time mengingat hambatan dalam transfer data dan adanya *waktu polling.*
3. *Security* dalam pengiriman data tidak didukung dalam aplikasi ini. *Security* hanya diberikan dengan memanfaatkan password ketika user login.
4. Reverse engireering yang diterapkan dalam aplikasi ini hanya digunakan untuk obyek-obyek yang penting dan sering digunakan, karena RDBMS Oracle yang begitu kompleks akan menyita waktu yang banyak jika semua obyek dalam database dilibatkan.

1.5. Tinjauan Pustaka

Saat ini sudah terdapat beberapa aplikasi *case tool* seperti Power Designer Data Architect yang dapat melakukan pemodelan basis data konseptual. Namun *case tool* ini pada umumnya merupakan aplikasi desktop dan tidak menggunakan teknologi internet. Untuk berinteraksi dengan menerapkan arsitektur client server

memerlukan prosedur yang rumit. Untuk itu diperlukan modifikasi pada *case tool* seperti ini dengan menggunakan teknologi internet.

Aplikasi *case tool* pada tugas akhir ini didukung oleh beberapa teknologi yang mendukung optimasi dalam manajemen aplikasi terdistribusi. Teknologi-teknologi ini antara lain *Internet Information Service* (IIS), *Internet Server API* (ISAPI), *Component Object Model* (COM) dan *Extensible Markup Language*(XML).

iSAPI - http://rampages.onramp.net/~steveg/isapi.html - dikembangkan oleh Process Software bekerja sama dengan Microsoft dan beberapa *vendor* Web server. ISAPI adalah solusi yang bagus bagi pengembang aplikasi dalam membuat web yang dinamis dan handal. Sebelum ada ISAPI, para pengembang web menggunakan teknologi Common Gateway Interface (CGI) untuk membuat aplikasinya. Namun dibanding CGI, ISAPI memiliki banyak kelebihan dalam hal pemakain memori di server yang lebih hemat, kecepatan yang lebih tinggi dan memiliki kontrol lebih baik terhadap koneksi HTTP.

ActiveX Control - yang biasanya dikenal dengan OLE controls atau OCX controls merupakan komponen-komponen atau obyek-obyek yang dapat diletakkan kedalam halaman Web atau aplikasi yang lain untuk menggunakan fungsi-fungsi atau paket yang ada didalam ActiveX tersebut. Misalnya, ActiveX control yang dimasukkan dalam Microsoft Internet Explorer 3.0 atau yang lebih tinggi dapat menjadikan halaman Web mempunyai fitur yang lebih canggih dan animasi yang akan membuat halaman Web lebih menarik.

XML (Extensible Markup Language) merupakan hal baru dalam teknologi internet yang telah melakukan pematangan teknologi dengan aplikasi yang *powerfull*, khususnya untuk manajemen, display, dan pengorganisasian data. Bersamaan dengan bahasa *display*-nya (XSL) dan Document Object Model (DOM), ini merupakan teknologi yang *essential* bagi siapa saja yang menggunakan *markup language* di web. XML dapat digunakan untuk membuat *text document* yang berisi data dalam bentuk yang terformat. Disamping data kita dapat memasukkan detail sekumpulan aturan yang mendefinisikan struktur data. XML dapat memecahkan masalah pertukaran data antar sistem. XML menyelesaikan masalah ini dengan menyediakan *data set* yang menggambarkan dirinya, dengan kata lain xml mendeskripsikan struktur dan tipe data dari elemen yang ada dalam dokumen tersebut dengan memanfaatkan keunggulannya yaitu memiliki tag-tag yang dinamis.

1.6. Manfaat Tugas Akhir

Manfaat dibuatnya aplikasi *case tool* ini adalah adanya efisiensi pekerjaan. Hal ini dapat dijelaskan dengan contoh sebagai berikut, misalnya seorang pengguna akan melakukan analisa terhadap sebuah desain sistem informasi, maka ia tidak memerlukan mobilitas yang tinggi ataupun berinteraksi via telepon atau e-mail. Ia juga dapat memantau perubahan desain sistem informasinya dalam bentuk model konseptual yang terakhir kali ter-*update*.

Manfaat yang lain adalah efisiensi biaya. Karena kita tidak perlu bepergian ke lokasi desain sistem informasi untuk melakukan analisa secara langsung, sehingga biaya dapat ditekan.

Dengan aplikasi ini pula kita dapat melakukan desain bersama. Aplikasi ini mendukung perancangan model konseptual secara *multiuser*. Setiap pengguna dan perancang sistem yang terlibat dalam perancangan model konseptual dapat melakukan perubahan pada desain secara langsung dan dapat diketahui saat itu juga oleh analis dan perancang system yang lain pada perubahan yang terjadi.

1.7. Metodologi Tugas Akhir

Metodologi yang digunakan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1.7.1. Studi Literatur

Pengumpulan literatur mengenai pemrograman internet, ISAPI, ActiveX Control, XML dan Oracle, serta hal-hal lain yang berhubungan dengan Tugas Akhir ini. Mempelajari dan mempersiapkan teknologi dimana aplikasi akan diimplementasikan. Teknologi tersebut antara lain Windows 2000, IIS, ISAPI dan RDBMS Oracle dan lain-lain.

1.7.2. Analisa dan Desain Sistem

Perencanaan dan perancangan algoritma, struktur aplikasi, diagram alur perangkat lunak, struktur ERD, *user interface* dan struktur data yang akan digunakan dalam komunikasi antara *server* dan *client*.

1.7.3. Pembuatan Perangkat Lunak

Penulisan kode program dan inisialisasi komponen pendukung program.

1.7.4. Uji Coba Dan Evaluasi Sistem

Aplikasi yang telah dibuat akan diuji coba dan dievaluasi untuk mengetahui sejauh mana tercapainya tujuan dari pembuatan aplikasi ini.

1.7.5. Pembuatan Buku TA

Pada tahap akhir ini disusun sebuah dokumentasi dari perlaksanaan Tugas Akhir. Dokumentasi ini akan berguna dalam pengembangan aplikasi *case tool* ini.

1.8 Sistematika Penulisan

Buku Tugas Akhir ini dibagi dalam beberapa bab dengan sistematika penyusunan sebagai berikut :

BAB I Pendahuluan

Membahas latar belakang, permasalahan, dan juga manfaat tugas akhir ini serta penjelasan singkat teknologi yg digunakan.

BAB II Teori Penunjang

Menjelaskan teknologi-teknologi yang digunakan dalam pengembangan aplikasi *case tool* ini serperti ISAPI, ActiveX, XML, dan ADO.

BAB III Pemodelan Basis Data Konseptual

Gambaran *case tool* semacam yang sudah ada, konsep pemodelan yang akan dibuat dan fitur-fitur yang akan dibuat dalam aplikasi tugas akhir.

BAB IV Perancangan Perangkat Lunak

Perencanaan dan perancangan algoritma, struktur aplikasi, diagram alur perangkat lunak, struktur ERD, *user interface*, struktur data yang akan digunakan dalam komunikasi *client server*.

BAB V Implementasi Perangkat Lunak

Mengimplementasikan rancangan perangkat lunak.

BAB VI Uji Coba Dan Evaluasi

Uji coba dan evaluasi terhadap sistem perangkat lunak yang dibuat serta penyempurnaan aplikasi jika masih terdapat banyak kekurangan.

BAB VII Penutup

Kesimpulan dan saran untuk pengembangan aplikasi lebih lanjut.

BAB II

TEORI PENUNJANG

BAB II

TEORI PENUNJANG

2.1. Model Basis Data Konseptual

Perancangan model konseptual perlu dilakukan disamping perancangan model fisik. Pada perancangan sebuah basis data, proses desain dimulai dari level konseptual. Model konseptual menunjukkan entitas dan relasinya berdasarkan proses yang diinginkan. Ketika menentukan entitas dan relasinya dibutuhkan analisa data tentang informasi yang ada dalam spesifikasi di masa mendatang. Pada pendekatan model konseptual, beberapa konsep pendekatan Relational digunakan, namun tidak berarti konsep ini nantinya diimplementasikan dalam model Relational saja.

Model konseptual bukanlah pendekatan proses informasi seorang programmer aplikasi, tetapi merupakan kombinasi beberapa cara untuk memproses data untuk beberapa aplikasi. Model konseptual tidak tergantung pada aplikasi individual, tidak tergantung pada DBMS yang digunakan, tidak tergantung pada hardware yang digunakan serta tidak tergantung pada model fisik. Reverse dari model data fisik dengan Data Definition Languange (DDL) yang berbeda dapat menghasilkan model konseptual yang sama. Dalam tugas akhir ini proses forward engineering dari model konseptual ke model fisik dilakukan dengan menggunakan DDL Oracle.

Model konseptual selalu berisi obyek data yang belum diimplementasikan dalam database fisik. Ia memberikan representasi formal dari data yang dibutuhkan untuk menjalankan proses bisnis.

2.1.1 Model Entity Relationship

Model *Entity Relationship* merupakan model data konseptual level tinggi. Model ini sering digunakan untuk membuat konsep aplikasi database, dan digunakan pula oleh beberapa *tool* yang mengaplikasikan konsep dari model ini. Penjelasan mengenai komponen yang terdapat didalam model *Entity Relationship* berikut ini disertai dengan contoh untuk memberikan gambaran.

2.1.1.1 Entitas

Obyek dasar yang terdapat dalam model Entity Relationship adalah *entitas*, yang mewakili 'benda' dalam dunia nyata. Sebuah entitas dapat berupa obyek dalam bentuk fisik, misalnya orang, mobil, rumah atau karyawan. Entitas dapat juga berupa obyek dalam bentuk konsep misalnya perusahaan, pekerjaan, atau mata kuliah.

Tiap entitas memiliki atribut yang merupakan penggambaran secara spesifik dari entitas misalnya entitas Mahasiswa dapat digambarkan dengan atribut NRP, Nama, Umur, Alamat dan Jenis Kelamin misalnya seorang mahasiswa memiliki atribut ('98022',Saenal M', 24,'Keputih','L'). Tiap atribut dalam entitas memiliki nilai yang menjadi bagian utama dari data yang disimpan dalam database.

Gambar 2.1 Sebuah entitas Mahasiswa

Atribut dalam entitas memiliki beberapa tipe yaitu :

1. Simple dan Composite

   Atribut *composite* dapat dibagi menjadi bagian yang lebih kecil yang mewakili beberapa atribut dasar, misalnya atribut Alamat dari entitas Mahasiswa dapat dibagi menjadi Jalan, gang, nomor dan nomor tambahan. Atribut sederhana atau atomik tidak dapat dibagi lagi menjadi bagian yang lebih kecil.

2. Single-valued dan Multivalued

   Sebagian besar atribut mempunyai nilai tunggal untuk entitas tertentu dan atribut ini disebut dengan atribut *single-valued,* misalnya Umur adalah sebuah atribut *single-valued* untuk orang atau manusia. Atribut *multivalued* memiliki beberapa nilai yang mungkin terjadi, misalnya Pendidikan untuk seseorang. Hal ini dapat ditunjukkan dengan adanya orang yang mempunyai pendidikan lebih dari satu misalnya SD, SMP dan SMA.

3. Stored dan Derived

   Sebuah entitas dapat memiliki dua atau lebih atribut yang saling berhubungan. Atribut Umur dan Tanggal Lahir dalam entitas

Mahasiswa saling berhubungan dimana atribut Umur dapat ditentukan dengan melihat atribut Tanggal Lahir dan waktu sekarang. Dalam kasus ini atribut Umur disebut sebagai atribut *derived* dan Tanggal Lahir disebut sebagai atribut *stored.*

4. Nilai Null

   Entitas tertentu dimungkinkan untuk tidak memiliki nilai yang dapat disimpan dalam atribut. Misalnya atribut NomorApartemen dari sebuah sebuah entitas Bangunan hanya dapat digunakan untuk bangunan berupa apartemen sedangkan untuk bangunan rumah atribut tersebut tidak dapat diisi, maka atribut NomorApartemen dapa diisi dengan nilai Null.

2.1.1.2 Tipe Entitas

Sebuah basis data biasanya mempunyai kumpulan entitas yang sama, misalnya entitas-entitas Mahasiswa memiliki atribut yang sama, tapi masing-masing entitas memiliki nilai sendiri untuk masing-masing atribut. Sebuah *tipe entitas* didefinisikan sebagai himpunan atau kumpulan entitas-entitas yang mempunyai atribut yang sama. Tiap-tiap tipe entitas dalam basis data digambarkan dengan nama dan atributnya.

Gambar 2.2 Entitas type

Constraint(batasan) yang penting dalam entitas dari sebuah tipe entitas adalah *key*(kunci) atau *uniqness constraint* pada atribut. Sebuah tipe entitas biasanya memiliki sebuah atribut yang nilainya berbeda untuk masing-masing entitas. Atribut semacam ini disebut dengan atribut kunci, dan nilainya dapat digunakan untuk mengidentifikasi masing-masing entitas secara *unique*(unik). Secara diagramatik tiap-tiap atribut kunci diberi tanda garis bawah.

2.1.1.3 Relasi Dan Tipe Relasi

Satu entitas yang mengacu pada entitas yang lain akan membentuk sebuah relasi misalnya tipe entitas Mahasiswa yang mengacu pada tipe entitas Dosen akan membentuk sebuah relasi yang dapat diberikan nama 'Dosen Wali'. Sebagaimana halnya entitas, relasi juga memiliki tipe relasi yang berisi relasi yang sama antara dua tipe entitas.

Relationship type memiliki *constraint* tertentu yang membatasi kemungkinan kombinasi dari entitas yang mungkin timbul dalam himpunan relationship. Ada dua jenis *constraint* relationship :

1. Cardinality

   Cardinality dalam relationship menunjukkan jumlah relationship dimana sebuah entitas dapat diikutkan didalamnya. Ada 4 jenis cardinality yaitu:

   - 1:1, menunjukkan bahwa satu entitas dalam entitas type berhubungan dengan hanya satu entitas dalam entitas type.
   - 1:N, menunjukkan bahwa satu entitas dalam entitas type berhubungan dengan beberapa entitas dalam entitas type.
   - N:1, menunjukkan bahwa beberapa entitas dalam entitas type berhubungan dengan satu entitas dalam entitas type. Pada prinsipnya jenis ini sama dengan 1:N
   - N:M, menunjukkan bahwa beberapa entitas dalam entitas type berhubungan dengan beberapa entitas dalam entitas type.

2. Participation Constraint

   Participation constraint menunjukkan apakah keberadaan sebuah entitas bergantung pada hubungannya dengan entitas lain melalui relationship. Ada dua jenis *participation constraint* :

   - Total, menunjukkan bahwa satu entitas harus ada dalam hubungan dengan entitas lain melalui relationship.

- Partial, menunjukkan bahwa satu entitas tidak harus ada dalam hubungan dengan entitas lain melalui relatinship.

2.1.1.4 Inheritance

Model *Enhanced ER* (EER) meliputi semua konsep pemodelan dari ER seperti yang telah dijelaskan sebelumnya dan disamping itu EER juga memasukkan konsep *subclass* dan *superclass*. Subclass merupakan jenis *entitas type* yang merupakan spesialisasi dari *entitas type* yang disebut dengan superclass. Hal ini dapat dijelaskan dengan contoh sebagai berikut, *entitas type* Karyawan dapat dibagi lagi menjadi Sekertaris, Manager, Teknisi dan lainnya. Hal ini berarti Sekertaris, Manager dan Teknisi adalah termasuk Karyawan dalam sebuah perusahaan. *Entitas type* Karyawan disebut dengan *superclass* dan *entitas type* Sekertaris, Manager dan Teknisi disebut dengan *subclass*. Konsep *superclass* dan *subclass* menggunakan konsep turunan atau *inheritance*. Entitas yang menjadi *subclass* merupakan turunan atau *inherit* dari entitas yang merupakan anggota *superclass*.

2.1.1.5 Notasi

Beberapa alternatif notasi yang digunakan dalam model Entitas Relationship adalah sebagai berikut :

- Entitas

Gambar 2.3 Notasi untuk sebuah entitas

- Atribut

Gambar 2.4 Notasi untuk atribut

- Relationship

Gambar 2.5 Notasi untuk relationship

- Cardinality

Gambar 2.6 Notasi untuk cardinality dalam sebuah relationship

- Inheritance

Gambar 2.7 Notasi untuk inheritance

Contoh sebuah tipe entitas dalam sebuah diagram ER:

Gambar 2.8 Contoh sebuah entitas beserta atributnya

2.2. Model Basis Data Fisik

Perancangan model basis data fisik secara langsung menggunakan konsep *Relational Data Model* atau Model Data Relasional dan *Relational Constraint*. Model basis data konseptual dapat menghasilkan model basis data fisik melalui sebuah proses yang disebut dengan *mapping*.

2.2.1 Model Data Relational

Model data relasional menggambarkan basis data sebagai sebuah kumpulan relasi. Jika sebuah relasi dianggap sebagai sebuah tabel nilai, maka masing-masing baris dalam tabel mewakili kumpulan data yang saling berhubungan. Secara termonologi, sebuah baris dalam tabel disebut dengan *tuple*, sebuah nama kolom disebut dengan atribut dan sebuah tabel disebut dengan *relation* atau relasi. Tipe data yang menunjukkan tipe dari nilai yang dapat muncul dalam masing-masing kolom disebut dengan domain.

Penjelasan mengenai domain, relasi, tuple, dan atribut akan digambarkan berikut ini melalui ilustrasi atau contoh. Sebuah domain D merupakan sekumpulan nilai atomik dimana masing-masing nilai dalam domain tidak dapat dibagi lagi. Metode umum dalam menentukan domain adalah dengan menentukan sebuah tipe data dari mana data yang membentuk domain digambarkan.

Penggunaan nama untuk domain adalah untuk membantu menggambarkan nilainya. Berikut ini adalah contoh domain:

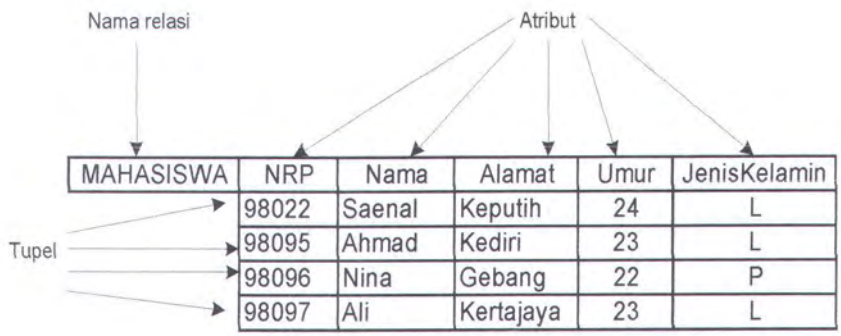
- Sby_phone_numbers. Merupakan kumpulan 10 digit nomor telepon di Surabaya.
- Name. Merupakan kumpulan nama orang.
- Employee_age. Merupakan kemungkinan umur karyawan sebuah perusahaan. Nilainya antara 15 dan 60 tahun.

Tipe data juga ditentukan dalam masing-masing domain, misalnya tipe data untuk domain Sby_phone_numbers dapat dideklarasikan sebagai sebuah *string* dengan format (sss)sss ssss.

Sebuah skema relasi R, ditunjukkan dengan $R(A_1,A_2,\ldots,A_n)$ dibuat dari sebuah relasi dengan nama R dan sekumpulan atribut $A_1,A_2,\ldots,A_n$. Masing-masing atribut A_i memiliki nilai yang ditentukan oleh beberapa domain D dalam skema relasi R. D disebut dengan domain dari A_i dan ditulis dengan $dom(A_i)$. Sebuah skema relasi digunakan untuk menggambarkan sebuah relasi, R disebut dengan nama dari relasi ini. Sebuah contoh skema relasi yang menggambarkan mahasiswa universitas seperti berikut ini:

```
MAHASISWA(NRP, Nama, Alamat, Umur, JenisKelamin)
```

Sebuah relasi r dari skema relasi $R(A_1,A_2,\ldots,A_n)$, yang juga ditulis r(R) merupakan sekumpulan n-tuple $r = \{t_1,t_2,\ldots,t_m\}$. Masing-masing n-tuple *t* merupakan daftar *n* nilai $t = <v_1,v_2,\ldots v_n>$ dimana masing-masing nilai $v_i, 1 \leq i \leq n$, merupakan elemen dari $dom(A_i)$ atau sebuah nilai *null*. Nilai ke-*i* dalam tuple *t* yang berhubungan dengan atribut A_i, mengacu pada $t[A_i]$.



MAHASISWA	NRP	Nama	Alamat	Umur	JenisKelamin
	98022	Saenal	Keputih	24	L
	98095	Ahmad	Kediri	23	L
	98096	Nina	Gebang	22	P
	98097	Ali	Kertajaya	23	L

Gambar 2.9 Contoh atribut dan tupel dari sebuah relasi MAHASISWA

2.2.2 Relational Constraint

Batasan-batasan pada data ditentukan dalam sebuah skema basis data relasional dalam bentuk *constraint*, termasuk didalamnya adalah batasan domain, batasan *key*, integritas entitas, dan batasan integritas referensial.

2.2.2.1 Batasan Domain

Batasan domain menunjukkan bahwa nilai masing-masing atribut A harus merupakan nilai atomik dari domain dom(A). tipe data yang berhubungan dengan domain meliputi tipe data standar numerik untuk integer (misalnya short-integer,integer, long-integer) dan angka real (float dan double-precision float), karakter, string fixed-length, tanggal, dan waktu.

2.2.2.2 Batasan Key

Sebuah relasi didefinisikan sebagai kumpulan tuple. Semua elemen dalam sebuah himpunan berbeda, sehingga semua tuple dalam sebuah relasi juga harus berbeda. Dua tuple tidak dapat mempunyai kombinasi nilai yang sama untuk semua atributnya. Ada bagian dari himpunan atribut dari sebuah skema relasi R dimana tidak ada dua tupel dalam beberapa relasi status r dari R harus mempunyai

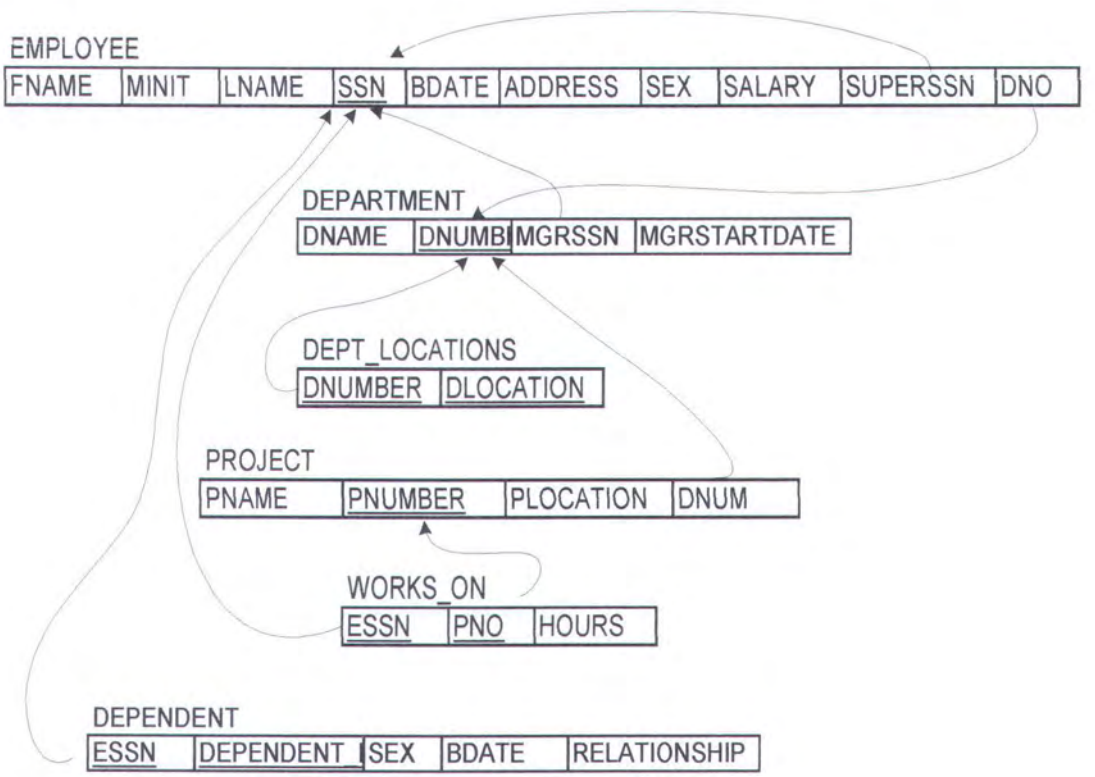
kombinasi nilai yang sama untuk atribut ini dan atribut ini disebut dengan *superkey* dari skema relasi R. Sebuah superkey menunjukkan batasan keunikan dimana tidak ada dua tupel yang berbeda dalam sebuah relasi status r dari R dapat mempunyai nilai sama. Sebuah *key* K dari sebuah skema relasi R merupakan sebuah superkey dari R dengan sifat tambahan dimana membuang beberapa atribut dari K akan membiarkan atau meninggalkan sekumpulan atribut yang bukan superkey dari R. Secara umum, sebuah skema relasi dapat mempunyai lebih dari satu key. Dalam hal ini masing-masing key disebut dengan sebuah *candidate key*. Misalnya relasi Mobil mempunyai key kandidat LicenceNumber dan EngineSerialNumber dan biasanya sebuah key kandidat dijadikan sebagai *primary key* dari relasi. Atribut yang merupakan key primer dari sebuah skema relasi ditandai dengan garis bawah. Batasan yang lain pada atribut menunjukkan apakah nilainya *null* (kosong) atau tidak diperbolahkan, misalnya jika setiap tupel MAHASISWA harus mempunyai nilai valid yaitu bukan nilai nulluntuk atribut NRP, makan NRP dari MAHASISWA diberi batasan yaitu tidak boleh NULL.

2.2.2.3 Integritas Entitas, Integritas Referensial Dan Foreign Key

Batasan integritas entitas adalah tidak ada nilai primary key dapat diberi nilai null. Hal ini dikarenakan nilai primary key digunakan untuk mengidentifikasi tupel dalam sebuah relasi. batasan integritas referensial ditunjukkan antara dua relasi dan digunakan untuk menjaga konsistensi antar tupel dari dua relasi. Secara informal, batasan integritas referensial menunjukkan bahwa sebuah tupel dalam satu relasi mengacu pada relasi yang lain dan harus mengau pada tupel yang ada pada relasi tersebut. Dalam integritas referensial dikenal sebuah konsep yang

disebut dengan *foreign key*. Sekumpulan atribut FK dalam skema relasi R1 merupakan foreign key dari R1 yang mengacu pada relasi R2 jika memenuhi dua syarat berikut:

1. Atribut dalam FK mempunyai domain yang sama dengan atribut primary key PK dari R2, atribut FK dikatakan me-*reference* atau mengacu pada relasi R2.
2. Sebuah nilai dari FK dalam tupel t_1 dari state $r_1(R_1)$ terjadi sebagai sebuah nilai dari PK untuk beberapa tupel t_2 dalam state $r_2(R_2)$ atau nilai null.



Gambar 2.10 Batasan Integritas referensial ditampilkan dalam diagram skema basis data relasional COMPANY

2.3. Tipe Data

Informasi atau data dalam model konseptual dan model fisik memiliki tipe data tertentu. Tipe data yang digunakan dalam model basis data fisik adalah tipe data Oracle. Tipe data oracle merupakan salah satu komponen penting yang digunakan dalam proses *forward* dan *reverse engineering*.

2.3.1. Tipe Data Model Konseptual

Atribut dalam entitas mempunyai tipe data untuk menampung nilai yang diberikan. Pada pemodelan konseptual tipe data yang digunakan adalah tipe data umum yang dapat digunakan untuk semua DBMS. Tipe data yang digunakan dalam pemodelan konseptual ini diadopsi dari salah satu case tool yang umum digunakan yaitu PowerDesigner. PowerDesigner akan dijelaskan pada bab berikutnya. Tipe data model konseptual tersebut adalah sebagai berikut :

1. **Tipe Data Numerik**

Tabel 2.1 Tipe data numerik model konseptual

Tipe Data	Code	Jenis Data
Integer	I	Integer 32-bit
Short Integer	SI	Integer 16-bit
Long Integer	LI	Integer 32-bit
Byte	BT	Nilai 256
Number	N	Angka dengan desimal tertentu
Decimal	DC	Angka dengan desimal tertentu
Float	F	Angka dengan desimal 32-bit
Short Float	SF	Angka dengan desimal kurang dari 32-bit
Long Float	LF	Angka dengan desimal 64-bit
Money	MN	Angka dengan desimal tertentu
Boolean	BL	Berisi dua nilai yang berlawanan (True/False)

2. **Tipe Data Karakter**

Tabel 2.2 Tipe data karakter model konseptual

Tipe Data	Code	Jenis Data
Characters	A	String karakter
Variable Characters	VA	String karakter
Long Characters	LA	String karakter
Long Var Characters	LVA	String karakter
Text	TXT	String karakter
Multibyte	MB	String karakter multibyte
Variable Multibyte	VMB	String karakter multibyte

3. Tipe Data Waktu

Tabel 2.3 Tipe data waktu model konseptual

Tipe Data	Code	Jenis Data
Date	D	Tanggal, bulan dan tahun
Date & Time	DT	*Date* dan *Time*

4. Tipe Data Lain

Tabel 2.4 Tipe data lain model konseptual

Tipe Data	Code	Jenis Data
Binary	BIN	String Biner
Long Binary	LBIN	String Biner
Undefined	<UNDEF>	Tidak didefinikan

2.3.2. Tipe Data Oracle

RDBMS yang digunakan dalam aplikasi yang akan dibuat dalam tugas akhir ini adalah Oracle sehingga tipe data dalam model basis data fisik adalah oracle. Tipe data oracle digunakan pada saat terjadi *forward engineering* dari model konseptual ke model fisik yaitu dengan mengkonversi tipe data konseptual ke tipe data oracle. Tipe data oracle yang digunakan dalam aplikasi tugas akhir ini adalah sebagai berikut:

Tabel 2.5 Tipe data oracle

Tipe Data	Jenis Data
LONG RAW	Digunakan untuk data binary atau byte string. Panjang maksimum data dengan tipe data ini adalah 2 gigabyte.
NVARCHAR2	Digunakan untuk data berupa dan panjang variabel mempunyai ukuran panjang maksimum *size* karakter atau byte, bergantung pada *National character set* yang dipilih. Ukuran maksimum adalah 4000 byte dan ukuran ini harus ditentukan.
RAW	Digunakan untuk data binary atau *byte string*. Panjang maksimum data dengan tipe data ini adalah 2000 byte.
LONG	Data karakter dari tipe data ini dapat mencapai 2 gigabyte.
NCHAR(size)	Digunakan untuk data karakter dengan panjang *size* karakter atau byte, bergantung pada *national character set*. *size* ditentukan antara 1 sampai 2000.
NUMBER(p,s)	Digunakan untuk data numerik dan tipe data ini mempunyai presisi *p* dan skala *s*. Presisi adalah antara 1 sampai 38 dan skala antara -84 sampai 127.
SMALLINT	Digunakan untuk data angka dengan panjang 38
DATE	Digunakan untuk data tanggal atau waktu dengan validitas antara 1 January 4712 SM sampai 31 December 9999 M.
INTEGER	Digunakan untuk data angka dengan panjang 38
FLOAT	Digunakan untuk data angka dengan panjang default126
VARCHAR2	Digunakan untuk data karakter dan panjang variabel mempunyai ukuran byte maksimum 4000 dan minimum 1 dan ukuran ini harus ditentukan dalam deklarasinya
CHAR(size)	Digunakan untuk karakter dengan panjang *size* byte antara 1 sampai 2000 byte.

2.4. Aplikasi Berbasis Web

Ada beberapa komponen yang dapat digunakan dalam pengembangan dan pembuatan aplikasi berbasis web diantarannya adalah Internet Server Application Programming Service(ISAPI), ActiveX Control, ActiveX Data Object(ADO), dan Extensible Markup Language(XML).

2.4.1. Internet Server Application Programming Interface (ISAPI)

Aplikasi berbasis web sudah menjadi bagian yang tidak terpisahkan dari dunia terdistribusi dan platform untuk membuatnya pun tidak terpisahkan pula.

Internet Information Server (IIS) dapat diakses dari browser-browser pada umumnya. Ia menyediakan semua standar service dari Web server yang mensupport HTTP, Secure Sockets Layer (SSL), Common Gateway Interface (CGI), ISAPI.

Internet Server Application Programming Interface (ISAPI) merupakan suatu teknologi baru. ISAPI merupakan sebuah interface ke Hypertext Transfer Protocol (HTTP) server yang memungkinkan developer mengembangkan Web server dan menyediakan tingkat fungsionalitas tertentu. ISAPI extensions (juga disebut *server applications*) dan filter memungkinkan server untuk dikembangkan dengan cara yang CGI atau interface lainnya tidak dapat melakukannya. Beberapa kelebihan yang dimiliki oleh ISAPI dibandingkan dengan CGI antara lain :

1. Membutuhkan *resource* seperti memori server yang lebih sedikit, sehingga server dapat menerima lebih banyak *request.*

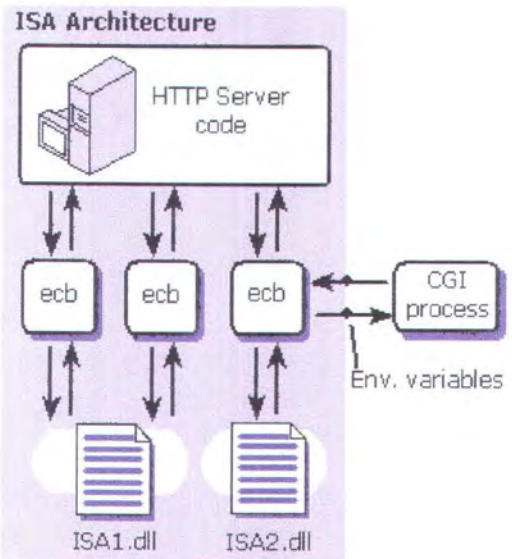
2. Tidak membuat proses baru untuk *thread*-nya tapi dilakukan pada proses yang sama, sehingga memiliki kecepatan proses yang lebih tinggi.

3. memberikan lebih banyak kontrol terhadap koneksi HTTP.

ISAPI sebenarnya dikembangkan untuk menjadi alternative dengan *performance* yang lebih tinggi dari Common Gateway Interface (CGI). Perbedaan utama antara model pemrograman CGI dan ISAPI adalah bahwa dengan CGI system membuat *unique process* setiap kali HTTP server menerima sebuah

request, sedangkan ISAPI *extension* tidak membutuhkan sebuah proses tersendiri. Dengan CGI, setiap kali HTTP server menerima sebuah request, ia harus membuat proses baru. Karena sistem operasi harus menjaga semua proses-proses ini, CGI sangat intensive dalam penggunaan resource. Hal ini membuat CGI sulit untuk digunakan dalam mengembangkan aplikasi berbasis internet yang baik

Diagram berikut menggambarkan perbedaan antara CGI dan ISAPI:



Gambar 2.11 CGI dan ISAPI

Dalam ISAPI, masing-masing request yang diterima oleh HTTP server memulai membuat sebuah struktur data EXTENSION_CONTROL_BLOCK. Pembuatan dan pemeliharaan sebuah struktur data lebih mudah dan lebih cepat daripada membuat proses baru. Disamping itu, karena EXTENSION_CONTROL_BLOCK dan extension keduanya biasanya berjalan dalam proses yang sama, server dapat memproses request lebih cepat dan mengakomodasi request dengan jumlah yang lebih tinggi.

Akhirnya, daripada menggunakan proses isolasi, ISAPI menggunakan *theads* untuk mengisolasi item proses yang berbeda. Karena IIS menggunakan *multiple threads* untuk men-*synchronize* kinerja, maka ISAPI menggunakan resource system dengan lebih efisien dibandingkan dengan CGI atau model yang lain yang menggunakan proses isolasi.

IIS versi 4.0 keatas mensupport proses isolasi untuk ISAPI DLL dan script. IIS menggunakan metode *custom high-speed* untuk membangun komunikasi antara proses server dan proses pengganti yang memblok ISAPI DLL. Ketika IIS menerima sebuah request untuk extension tertentu, ia meletakkan DLL kedalam memori, dimana ia melayani request. Sebagai contoh, HTTP request berikut menyebabkan IIS membuat sebuah *instance* dari MyISAPI.dll.

```
http://IIS/Applications/MyISAPI.dll?paramater1,parameter2
```

Fungsi *TerminateExtension* digunakan untu membebaskan beberapa resource yang di-*lock* atau dialokasikan oleh extension ketika pertama kali diletakkan dalam memory.

2.4.1.1 ISAPI Extension

ISAPI extension diimplementasikan sebagai DLL sebagaimana DLL pada umumnya yang diletakkan kedalam proses IIS atau (jika merupakan bagian dari sebuah aplikasi out-of-process) kedalam sebuah proses tersendiri. Seperti halaman ASP dan HTML, IIS menggunakan lokasi virtual dari file dll dalam file system untuk memetakan ISAPI extension ke namespace URL yang disediakan oleh IIS.

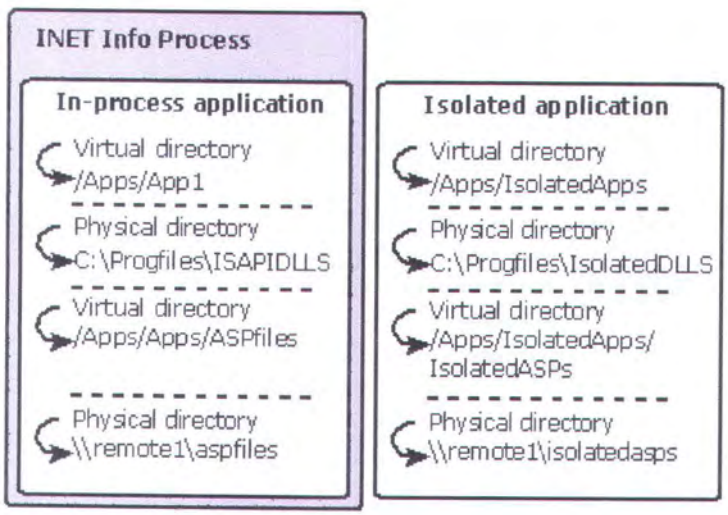
Ketika IIS menerima sebuah request yang memetakan ke sebuah ISAPI extension seperti

```
GET http://localhost/Server/MyISAPI.dll
```

IIS pertama akan memeriksa apakah MyISAPI.dll sudah di-*load* atau belum. Secara default, IIS dikonfigurasi untuk menampung ISAPI extension. Jika diperkirakan bahwa extension belum di-*load*, maka IIS akan me-*load* DLL.

Sekali extension DLL di-*load*, maka seluruh request diatur oleh extension, dan IIS bertindak sebagai perantara dan pembantu yang efisien.

Diagram berikut menggambarkan proses ini.



Gambar 2.12 Proses ISAPI Extension dalam IIS

ISAPI extension dapat mengerjakan berbagai pekerjaan, tapi untuk digunakan dalam IIS, ia harus menyediakan standard interface. Masing-masing extension harus mengimplementasikan dan mengeksport dua fungsi utama yaitu:

1. **GetExtensionVersion**

Merupakan fungsi entry-point pertama IIS. Fungsi ini memungkinkan ISAPI extension untuk me-register informasi dari versi ISAPI dengan IIS.

```
BOOL WINAPI GetExtensionVersion(
HSE_VERSION_INFO* pVer);
```

Parameter pVer menunjuk pada sebuah struktur data HSE_VERSION_INFO yang berisi informasi terntang versi.

Fungsi ini mengembalikan nilai TRUE jika IIS bisa menggunakan ISAPI DLL. Dan mengembalikan nilai FALSE jika sebaliknya. IIS memanggil fungsi ini ketika DLL pertama kali di-*load*.

2. **HttpExtensionProc**

Fungsi ini merupakan entry point utama untuk sebuah ISAPI extension yang dipanggil oleh IIS. Fungsi ini meng-ekspose metode yang digunakan IIS untuk mengakses fungsionalitas yang diekspose oleh extension.

```
DWORD WINAPI HttpExtensionProc(
  LPEXTENSION_CONTROL_BLOCK lpECB);
```

Parameter lpECB menunjuk pada struktur data EXTENSION_CONTROL_BLOCK yang dihubungkan dengan request yang aktif pada saat itu.

Fungsi ini mengembalikan sebuah nilai DWORD yang berisi kode HSE_STATUS dengan kemungkinan nilainya sebagai berikut:

Tabel 2.6 Tabel HSE_STATUS

Nilai	Keterangan
HSE_STATUS_SUCCESS	Extension telah menyelesaikan proses dan server harus men-*disconnect* client dan membersihkan resource yang dialokasikan.
HSE_STATUS_SUCCESS_AND_KEEP_CONN	Extension sudah menyelesaikan proses dan server harus menunggu HTTP request selanjutnya juka client mendukung koneksi *Keep-Alive*. Extension dapat mengembalikan ini jika mampu mengirim header *Content-Length* yang benar ke client.
HSE_STATUS_PENDING	Extension mengantrikan request untuk pemrosesan dan memberitahu server ketika ia sudah selesai.. Jika ISAPI extension menggunakan HSE_STATUS_PENDING, maka response dapat dikirim dari proses yang telah selesai di *thread* yang lain setelah mengembalikan HSE_STATUS_PENDING.
HSE_STATUS_ERROR	Extension menemui error ketika memproses request, jadi server dapat men-*disconnect* client dan membersihkan resource yang dialokasikan. Sebuah HTTP status code 500 ditulis ke IIS log untuk request.

dan satu fungsi opsional yaitu:

TerminateExtension, merupakan fungsi yang meng-*unload* ISAPI DLL. Fungsi ini dipanggil oleh IIS ketika IIS siap untuk meng-*unload* DLL dari prosesnya.

```
BOOL WINAPI TerminateExtension(
  DWORD dwFlags
);
```

parameter dwFlags merupakan DWORD yang menunjukkan apakah IIS harus men-shutdown extension.

Tabel 2.7 Parameter dwFlags

Nilai	Keterangan
HSE_TERM_MUST_UNLOAD	Server mengindikasikan extension bahwa dia akan di-*unload*. Dan extension tidak dapat menolak.

Karena TerminateExtension tidak pernah dipanggil oleh IIS sampai semua request diproses, maka tidak perlu memasukkan kode untuk menunggu request dalam TerminateExtension.

Berikut ini adalah proses yang terjadi ketika IIS menerima request yang dipetakan IIS ke sebuah ISAPI extension:

1. IIS me-*load* DLL, jika belum ada di memori. Ketika DLL di-*load* kedalam memori, *optional DLL entry* / fungsi exit (biasanya DllMain) akan dipanggil secara otomatis oleh windows. Kemudian IIS fungsi extension **GetExtensionVersion.**
2. IIS membentuk proses awal pada request yang dating.

3. IIS membuat dan menyebarkan sebuah EXTENSION_CONTROL_BLOCK, yang akan digunakan IIS untuk melewatkan data request dan pointer fungsi *callback* ke extension.
4. IIS memanggil fungsi ISAPI extension **HttpExtensionProc**, melewatkan pointer ke struktur EXTENSION_CONTROL_BLOCK yang dibuat untuk request ini.
5. ISAPI extension menjalankan perintah apapun yang dirancang untuk dijalankan. Hal ini dapat meliputi membaca data dari client, sebagaimana dalam operasi POST, atau menulis header dan data kembali ke client.
6. Untuk operasi yang sinkron, extension memberitahu IIS bahwa ia sudah menyelesaikan pemrosesan request hanya dengan keluar dari fungsi **HttpExtensoinProc**.
7. IIS menjalankan pembersihan pada koneksi yang digunakan untuk request, dan kemudian menutup koneksi tersebut.
8. Sekali ISAPI extension tidak lagi dibutuhkan, IIS memanggil fungsi **TerminateExtension**, jika extension menyediakannya. Fungsi ini biasanya digunakan oleh extension untuk menjalankan pembersihan.

Fungsi **GetExtensionVersion** tidak dipanggil pada setiap request. Fungsi **HttpExtensionProc** dipanggil sekali pada masing-masing dan setiap request untuk ISAPI extension. Satu struktur **EXTENSION_CONTROL_BLOCK** digunakan untuk masing-masing request yang datang.

Ketika extension memproses sebuah request, ia membutuhkan sebuah cara untuk berkomunikasi dengan IIS. Disamping itu, ada beberapa fungsi umum (seperti menulis melalui socket ke client) yang rumit untuk diimplementasikan. Usaha untuk menduplikat fungsionalitas dalam ISAPI extension yang sudah diimplementasikan oleh IIS tidak akan pernah terwujud.

Fungsi callback ISAPI mengatasi permasalahan ini. Pointer-pointer ke fungsi callback tersedia untuk ISAPI extension melalui struktur EXTENSION_CONTROL_BLOCK, sebuah pointer dimana extension mem-*passing* masing-masing request.

Fungsi callback yang digunakan dengan ISAPI extension adalah :

1. **WriteClient**, mengirim sebuah response balik ke client.
2. **ReadClient**, membaca data, yang disediakan oleh client ke buffer. Fungsi ini berperan dalam menjalankan operasi HTTP POST.
3. **GetServerVariable**, menghasilkan variable-variable server yang berisi informasi tentang request dan server.
4. **ServerSupportFunction**, menyediakan berbagai jenis fungsi yang dapat digunakan oleh extension dalam pemrosesan request.

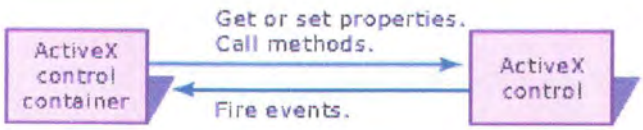
2.4.2. ActiveX Control

ActiveX Control, yang sebelumnya dikenal dengan OLE control atau OCX control, merupakan komponen (atau obyek) yang dibangun dengan menggunakan teknologi Component Object Model (COM) dan dapat dimasukkan kedalam sebuah halaman Web atau aplikasi yang lain untuk menggunakan kembali paket fungsionalitas yang sudah terprogram dalam komponen. Sebuah ActiveX control dapat menggambar dirinya sendiri dalam windownya, merespon terhadap event (misalnya event saat meng-klik mouse), and diatur melalui interface yang meliputi property dan metode, sama dengan yang terdapat dalam obyek *Automation*. Batasan dalam pembuatan ActiveX control ini hanya terletak pada sejauh mana imajinasi pembuatnya.

ActiveX control dapat dikembangkan untuk beberapa kegunaan, seperti akses database, monitoring data, atau grafis. Disamping sifatnya yang portable, ActiveX control men-support fitur yang sebelumnya tidak tersedia pada control tertentu. Disamping itu, ActiveX control sepenuhnya mendukung Automation, yang memungkinkan control untuk meng-ekspose property yang dapat ditulisi dan sekumpulan metode yang dapat dipanggil dengan menggunakan user control.

ActiveX control diimplementasikan sebagai sebuah in-process server yang dapat digunakan dalam beberapa *OLE container*. Fungsionalitas penuh dari ActiveX control tersedia hanya jika digunakan dalam sebuah OLE container yang dirancang untuk mengetahui ActiveX control. Jenis container ini, untuk selanjutnya disebut "control container", dapat mengoperasikan sebuah ActiveX

control dengan menggunakan property dan method dari control, dan menerima pemberitahuan dari ActiveX control dalam bentuk event. Gambar berikut memperlihatkan interaksi ini.

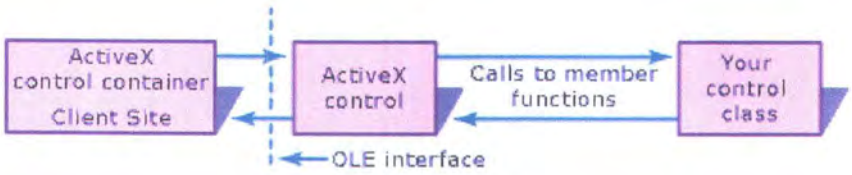


Gambar 2.13 Interaksi ActiveX Control dengan container

ActiveX control menggunan beberapa element pemrograman untuk berinteraksi secara efisien dengan control container dan dengan user. Elemen tersebut adalah class **COleControl**, sekumpulan fungsi *evet-firing* dan pengiriman pemetaan.

Setiap obyek ActiveX control yang dikembangkan menurunkan sekumpulan fitur yang handal dari *base class*-nya, yaitu **COleControl**. Fitur ini meliputi *in-place activation* dan *Automation logic*. COleControl dapat menyediakan obyek control dengan fungsionalitas yang sama dengan obyek window, ditambah dengan kemampuan untuk menimbulkan event.

Ketika ActiveX control digunakan dalam sebuah control container, maka digunakan dua mekanisme untuk berkomunikasi yaitu pertama ActiveX control mengekpose property dan method dan yang kedua ActiveX control menjalankan event. Gambar berikut menunjukkan bagaimana dua mekanisme ini diimplementasikan.



Gambar 2.14 Mekanisme komunikasi pada ActiveX Control

Gambar diatas juga menunjukkan bagaimana OLE interface yang lain (disamping automation dan event) ditangani oleh control.

Semua komunikasi control dengan container dilakukan oleh COleControl. Untuk menanganin beberapa request container, COleControl akan memanggil fungsi member yang diimplementasikan dalam control class. Semua method dan beberapa property ditangani dengan cara ini.

Satu keuntungan dari ActiveX control adalah ActiveX Control juga dapat digunakan di applikasi yang ditulis dengan beberapa bahasa pemrograman, termasuk semua pemrograman Microsoft dan database.

ActiveX control dapat ditambahkan ke dalam halaman Web dengan menggunakan tag HTML <OBJECT>. Tag <OBJECT> memasukkan sekumpulan parameter yang digunakan untuk menentukan data yang mana yang harus digunakan dan untuk mengontrol tampilan dan karakteristik dari control.

Jika ActiveX control telah terinstal dalam system operasi, maka ia akan secara otomatis dijalankan ketika halaman web yang berisi control tersebut ditampilkan.

2.4.3. ActiveX Data Object (ADO)

ActiveX Data Object (ADO) dirancang untuk menyediakan model obyek untuk mengakses, mengedit dan mengupdate secara programatikal berbagai jenis datasource melalui *OLEDB system interface*. Kegunaan umum dari ADO adalah untuk meng-*query* sebuah tabel atau table-tabel dalam sebuah basis data relasional, mengambil dan menampilkan hasilnya dalam sebuah aplikasi, dan memungkinkan user untuk membuat dan menyimpan perubahan data. Hal-hal lain yang dapat dilakukan secara programatikal oleh ADO adalah :

1. Meng-*query* sebuah basis data dengan menggunakan SQL dan menampilkan hasilnya.
2. Mengakses informasi dalam sebuah file melalui internet.
3. Memanipulasi pesan dan folder dalam sebuah system e-mail.
4. Menyimpan data dari sebuah basis data kedalam sebuah file XML.
5. Meng-*execute* command yang digambarkan dengan XML.
6. Menyimpan data kedalam sebuah binary atau XML stream.
7. Memungkinkan user untuk me-review dan membuat perubahan pada data dalam tabel database.
8. Membuat dan menggunakan kembali database command.
9. Menjalankan *stored procedures*.
10. Secara dinamis membuat struktur yang fleksibel, disebut dengan **Recordset**, untuk mengambil, menavigasi, dan memanipulasi data.
11. Menjalankan operasi transaksional basis data.

12. Mem-filter dan men-sort copy dari informasi basis data berdasar criteria run-time.
13. Membuat dan memanipulasi hasil hierarki dari basis data.
14. Mem-bind field basis data ke komponen *data-aware.*
15. Membuat remote, dan men-disconnect beberapa Recordset.

Berikut ini adalah elemen dari model pemrograman ADO:

1. Connection

Sebuah aplikasi dapat mengakses sebuah data source melalui sebuah *connection* yang sangat diperlukan dalam transaksi data. Aplikasi dapat mengakses sebuah data source secara langsung (biasanya disebut dengan *two-tier system*) atau secara tidak langsung (biasanya disebut *thre-tier system*) melalui sebuah media misalnya Microsoft Internet Informaition Service (IIS).

Obyek model menggunakan obyek **Connection** untuk menciptakan sebuah koneksi. Sebuah *transaction* membatasi awal dan akhir dari rangkaian operasi data akses yang berlangsung melalui sebuah koneksi. ADO memastikan bahwa perubahan terhadap data source yang dihasilkan dalam operasi dari sebuah transaksi berlangusung dengan sukses atau gagal.

Jika transaksi dibatalkan atau satu dari operasinya gagal, maka operasi-operasi tidak akan menghasilkan apa-apa sebagaimana jika

tidak terjadi operasi dalam transaksi. Data source akan kembali sebagaimana keadaan sebelum proses transaksi terjadi.

Obyek model tidak menerapkan konsep transaksi secara eksplisit, tapi cukup diwakili dengan method obyek **Connection**.

ADO mengakses data dan service dari OLEDB provider. Obyek **Connection** digunakan untuk menunjuk pada sebuah provider tertentu dan beberapa parameter. Misalnya, Remote Data Service (RDS) dapat diakses secara eksplisit atau dengan Microsoft OLEDB Remoting Provider jika secara tidak eksplisit.

Data source yang menjadi target sebuah *connection* dapat ditunjukkan dengan sebuah *connection string* atau *Uniform Resource Locator* (URL).

2. Command

Sebuah *command* yang dikeluarkan melalui *connection* yang sudah dibuat dapat memanipulasi data source dengan beberapa cara. *Command* menambah, menghapus atau meng-update data dalam data source, atau mengambil data dalam bentuk *row* dalam sebuah tabel.

Model obyek menggunakan obyek **Command** untuk menciptakan sebuah *command.* Keberadaan obyek **Command** memberikan keuntungan bagi ADO untuk mengoptimalkan jalannya *command.*

3. Parameter

Seringkali, *command* membutuhkan variable atau parameter, yang dapat diubah sebelum *command* digunakan. Misalnya, *command* yang digunakan sama tapi informasi yang diambil tiap waktu berbeda.

Parameter sangat berguna untuk menjalankan *command* yang bersifat seperti fungsi. Dalam hal ini diketahui apa yang dikerjakan *command* tapi tidak diketahui bagaimana *command* berkerja.

Model obyek menggunakan objek **Parameter** untuk membuat parameter.

4. Recordset

Jika *command* merupakan sebuah query yang mengembalikan data sebagai *row* yang berisi informasi dalam sebuah table, maka *row-row* ini ditempatkan di *storage* lokal.

Model obyek menggunakan obyek **Recordset** untuk membuat *storage* ini.

Recordset merupakan sarana utama dalam pemeriksaan dan pengolahan data dalam *row*. Dengan obyek **Recordset** dimungkinkan:

- Menunjukkan *row-row* mana yang tersedia untuk pemeriksaan.
- Melintasi *row-row*.

- Menunjukkan order dimana *row* di lewati.
- Menambah, mengubah, atau menghapus *row*.
- Meng-update data source dengan row yang sudah berubah.
- Mengatur seluruh status Recordset.

5. Field

Sebuah *row* dari **Recordset** berisi satu atau lebih *field*. Jika **Recordset** dibayangkan sebagai sebuah grid dua dimensi, maka *field* tampil dalam bentuk *column* (kolom). Masing-masing *field (column)* mempunyai atribut nama, tipe data, dan nilai. Inilah nilai yang mengisi data yang sebenarnya dari sebuah data source.

Model obyek menggunakan obyek **Field** untuk membuat sebuah *field*.

Untuk memodifikasi data dalam data source, yang dimodifikasi adalah nilai dari obyek **Field** dalam *row* Recordset. Akhirnya perubahan pada sebuah Recordset diteruskan ke data source. Metode pengaturan *transaction* pada obyek Connection dapat menjamin bahwa perubahan sukses atau gagal.

6. Error

Error dapat terjadi pada setiap saan dalam aplikasi, biasanya diakibatkan oleh gagalnya pembuatan sebuah *connection*, menjalankan *command*, atau menjalankan operasi pada sebuah obyek (misalnya, berusaha menggunakan obyek Recordset yang belum diinisialisasi).

Model obyek menggunakan obyek **Error** untuk membuat instance error.

Error-error yang terjadi menghasilkan satu atau lebih obyek **Error**. Error berikutnya yang terjadi akan menghapus nilai obyek **Error** sebelumnya.

7. Property

Masing-masing obyek ADO mempunyai property yang unik yang menggambarkan atau mengontrol karakteristik dari obyek tersebut.

Ada dua tipe property yaitu *built-in* dan *dynamic*. Property *built-in* adalah bagian dari obyek ADO dan selalu ada. Sedangkan property *dynamic* ditambahkan ke *collection* property dari obyek ADO dengan mendasari *data provider* atau *service provider*, dan hanya ada jika *provider* sedang digunakan..

Model obyek menggunakan obyek **Property** untuk membuat sebuah property.

8. Record

Semua data source ada dalam bentuk tabel dalam sebuah basis data. Informasi system *storage* seperti file dan system e-mail, berisi komponen *container* dan *content*. Sebuah *container* bisa mengambil *content* atau yang lain, *subordinate container*.

Dalam sebuah file system, *container* dan *content* adalah direktori dan file, dalam sebuah e-mail system, *container* dan *content* adalah folder dan pesan.

Model obyek menggunakan obyek **Record** untuk membuat sebuah *container* atau *content*. Sebuah *row* dari Recordset dapat dijadikan sebagai sebuah obyek **Record**.

Obyek **Record** menyediakan sarana untuk:

- Meng-copy, menghapus, atau memindahkan item yang diwakilinya.
- Membuat sebuah Record baru yang sesuai dengan item yang diwakili, misalhya sebuah direktori atau file, atau sebuah row dari Recordset.

Sebuah obyek **Record** digunakan sejalan dengan obyek ADO yang lain, misalnya obyek **Connection** dan **Recordset**.

9. Stream

Content dari sebuah informasi system *storage*, misalnya file dalam sebuah file system, berisi byte *stream*. Juga sebuah buffer dalam memory yang berisi byte *stream*.

Model obyek menggunakan obyek **Stream** untuk menampung byte *stream*.

Obyek **Stream** menyediakan sarana untuk :

- Membaca dan menulis serangkaian byte atau baris text.
- Meletakkan dirinya dari atau ke file.

Obyek **Stream** digunakan hanya sejalan dengan obyek Record.

10. Collection

ADO menyediakan *collection*, sebuah tipe obyek yang berisi type tertentu dari obyek yang lain. Obyek-obyek dalam *collection* dapa diambil melalui property *collection* dengan nama, sebagai string text, atau dengan integer.

ADO menyediakan empat tipe *collection* :

- Obyek **Connection** mempunyai *Error collecton*, yang berisi semau obyek Error yang dibuat dalam merespon kegagalan dalam data source.

- Obyek **Command** mempunyai *collection* Parameter, yang berisi semua obyek parameter yang digunakan dalam obyek Command.

- Obyek **Recordset** dan **Record** mempunyai *collection* Fields, yang berisi semua obyek *field* yang mendefinisikan kolom dari obyek Recordset.

- Disamping itu, obyek **Connection, Command, Recordset,** dan Field semuanya mempunyai collection Properties, yang berisi semua obyek Property yang dipakai pada obyek tersebut.

Obyek ADO memiliki property-properti yang di set atau diambil nilainya dengan tipe data umum seperti INTEGER, CHARACTER, atau BOOLEAN. Bagaimanapun sangat berguna untuk memikirkan property tertentu sebagai nilai kembalian dari tipe data "COLLECTION OBJECT". Obyek *collection* mempunyai *method* untuk menyimpan dan mengambil obyek lain yang sesuai untuk *collection.*

11. Event

Event merupakan pemberitahuan bahwa operasi tertentu akan terjadi atau sudah terjadi. Event dapat digunakan untuk memerintahkan aplikasi untuk melakukan beberapa perintah.

Model obyek tidak secara eksplisit membentuk *event*, tapi mewakilkannya ke *event-handler routines.*

Event handler yang dipanggil sebelum operasi dimulai memberikan keuntungan untuk memeriksa dan mengolah parameter operasi, kemudian membatalkan atau meneruskan operasi tersebut sampai selesai. *Event handler* yang dipanggil sesudah selesainya operasi memberitahukan tentang selesainya operasi.

2.4.4. Extensible Markup Language (XML)

Sekilas Extensible Markup Language (XML) tampak seperti HTML, dimana keduanya sama-sama merupakan sub dari Standard Generalized Markup Language (SGML). Perbedaan antara XML dan HTML terletak pada dua hal yaitu sintak dan semantic-nya.

HTML dan XML menggunakan <, >, dan & untuk membuat struktur element dan attribute. Dokumen XML harus mengikuti aturan-aturan untuk mengidentifikasi bagian-bagian dokumen dan membuat struktur elemen yang bercabang. Elemen-elemen dalam documen XML tidak dapat saling melingkupi (*overlap*). Jika tag awal untuk sebuah elemen muncul dalam elemen yang lain, ia harus diakhiri dalam isi elemen yang sama. Misalnya, kode HTML berikut menunjukkan kombinasi antara bold dan italic dengan meng-*overlap* strukturnya.

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

dalam beberapa browser HTML, kode diatas akan tampil sebagai berikut:

This is bold text. ***This is bold italic text.*** *This is italic text.*

Dalam parser XML, semua pemrosesan berhenti setelah menemukan karena parser XML mencari </i>, dan tidak akan menerima . Untuk mencapai format yang sama dalam XML, maka digunakan sintak sebagai berikut:

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

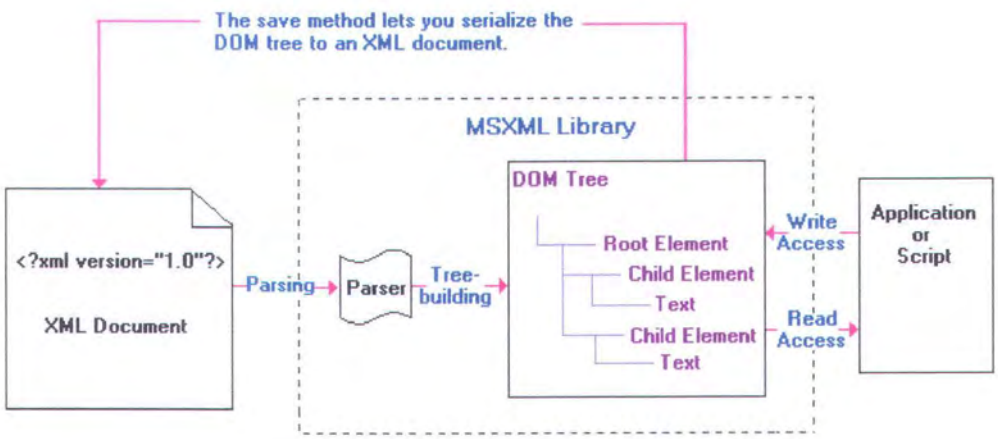
Extensible Markup Language (XML) menyediakan suatu cara untuk mendeskripsikan data yang terstruktur. Tidak seperti tag-tag HTML, yang digunakan terutama untuk mengontrol display dan tampilan data, tag-tag XML digunakan untuk mendefinisikan struktur dan tipe data dari data itu sendiri.

XML menggunakan sekumpulan tag untuk menggambarkan elemen-elemen data. Masing-masing elemen meng-*encapsulate* sekian data yang mungkin sangat sederhana atau bahkan sangat kompleks. XML sangat sederhana, tidak tergantung pada platform, dan merupakan standar yang banyak dipakai secara luas. Kelebihan XML bila dibandingkan dengan HTML adalah bahwa XML memisahkan user interface dari struktur data. Pemisahan data dari tampilan memungkinkan adanya integrasi data dari source yang berbeda. Misalnya informasi customer, order barang, hasil riset, pembayaran rekening, catatan medis, data catalog, dan informasi lain dapat dikonversikan kedalam bentuk XML.

2.4.4. XML Document Object Model (XMLDOM)

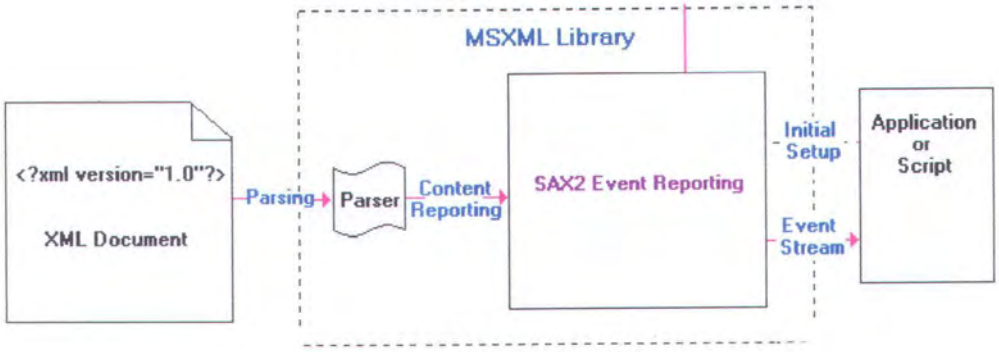
Document Object Model (DOM) memberikan standar interpretasi yang mudah dari dokumen XML untuk aplikasi dan skrip. Implementasi DOM dalam MSXML memungkinkan untuk mengambil dan membuat dokumen, menghimpun error-error, jika ada, mengakses dan memanipulasi informasi dan struktur yang ada dalam dokumen, dan menyimpannya kembali dalam bentuk XML file.

Implementasi DOM hanya satu bagian dari parser MSXML. Diagram berikut menunjukkan langkah yang dimasukkan dalam *parsing* sebuah dokumen XML dan memberikan informasinya ke aplikasi atau skrip.



Gambar 2.15 Langkah parsing dokumen XML

Untuk pendekatan yang lain, MSXML parser juga menyediakan sebuah API yang berbeda, *Simple API for XML* (SAX2).



Gambar 2.16 Simple API for XML

Pendekatan DOM membuat sebuah obyek tree yang diatur dengan parser MSXML. Hal ini memungkinkan developer untuk mengambil keuntungan dalam mengatur isi XML, daripada harus membuatnya sendiri.

DOM menyediakan sebuah interface untuk mengambil, mengakses, memanipulasi, dan menyambung dokumen XML. DOM menyediakan sebuah representasi dari dokumen XML yang disimpan dalam memory, yang menyediakan *random access* pada isi dari seluruh dokumen. DOM memungkinkan aplikasi untuk membiarkan *logic* yang disediakan oleh parser MSXML untuk mengambil informasi yang berbasis XML, menggunakan fasilitasnya daripada menulis kode untuk membaca dan memproses.

Ketika parser MSXML meletakkan sebuah dokumen XML kedalam DOM, ia membacanya dari awal sampai akhir dan membuat model node *logical* dari struktur dan isi dalam dokumen XML. Dokumen itu sendiri merupakan node tunggal yang berisi semua node-node yang lain, termasuk node yang mewakili elemen *root*, yang berisi semua elemen, attribute, dan node teks dalam dokumen.

DOM memungkinkan aplikasi untuk bekerja dengan struktur dan informasi dokumen XML sebagai struktur program dan bukan *text stream*. Aplikasi dan skrip dapat membaca dan memanipulasi struktur ini tanpa mengetahui detail dari sintak XML, dengan memanfaatkan fasilitas yang dibuat dalam DOM API dari XML.

DOM menggunakan dua abstaksi: *tree-like hierarchy* dan *node* yang mewakili isi dan struktur dokumen. Hirarki disusun dari node-node ini, yang mungkin berisi atau diisi oleh node-node yang lain. Hal ini menunjukkan bahwa banyak kinerja pemrosesan XML yang membutuhkan navigasi struktur *tree* ini untuk mencari dan memodifikasi informasi didalamnya.

DOM memperlakukan node-node sebagai obyek, yang membuatnya mungkin untuk membuat skrip yang mengambil dokumen dan kemudian melewati semua node, memberitahukan apa yang ditemukan dalam *tree*.

Berikut ini diantara obyek yang dimiliki oleh XML DOM:

- DOMDocument

  Obyek DOMDocument merupakan dasar dari XML Document Object Model (DOM). Ia mempunyai property dan method yang memungkinkan untuk menavigasi, meng-query, dan memodifikasi isi dan struktur dari dokumen XML.

- IXMLDOMNode

IXMLDOMNode merupakan obyek utama dalam Document Object Model (DOM). Elemen, atribut, *comment*, instruksi pemrosesan, dan setiap komponen dokumen yang lain dapai di wakili dengan sebuah IXMLDOMNode. Obyek DOMDocument merupakan XMLDOMNode itu sendiri.

- IXMLDOMNodeList

  Obyek IXMLDOMNodeList dihasilkan oleh *collection* **childNodes** dan method **selectNodes** dan **getElementsByTagName**.

- IXMLDOMNamedNodeMap

  Obyek IXMLDOMNamedNodeMap dihasilkan oleh *attributes property*. Obyek **IXMLDOMNamedNodeMap** berbeda dengan daftar node dalam *collection* node yang dapat diakses dengan nama maupun angka. Karena order atribut tidak penting dalam XML, hal ini dapat membuat pemrosesan atribut lebih mudah.

2.5. Pemodelan Visual Dengan Rational Rose

Pemodelan visual merupakan penggambaran secara grafikal dari proses-proses dunia nyata. Model digunakan untuk memahami masalah, berkomunikasi dengan siapapun yang masuk dalam proyek, memodelkan sistem yang komplek, menyiapkan dokumentasi dan mendesain program dan basis data.

Rational Rose merupakan perangkat lunak pemodelan visual yang dapat digunakan untuk membuat, menganalisa, mendesain, melihat, memodifikasi dan

memanipulasi komponen dan mengimplementasikan sistem dengan cara yang membuatnya mudah untuk berkomunikasi.

2.5.1. Use Case Diagram

Diagram use case menggambarkan bagaimana sistem digunakan dilihat dari luar sistem. Diagram Use case berisi :

- Actor, mewakili pengguna sistem. Actor merupakan seseorang atau sesuatu yang :
  - Berinteraksi dengan atau menggunakan sistem
  - Memasukkan data ke dan mengambil informasi dari sistem.
  - Merupakan hal diluar sistem dan tidak mempunyai kontrol terhadap use case.

Gambar 2.17 Notasi Actor

- Use case, merupakan serangkaian transaksi atau aksi yang dibentuk oleh sistem sebagai respon dari aksi yang dilakukan oleh actor. Use case berisi semua kejadian yang dapat terjadi antara pasangan actor dan use case.

Gambar 2.18 Notasi Use Case

- Interaction atau relationship, menunjukkan interaksi antara actor dan use case.

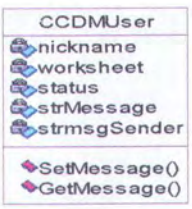
Gambar 2.19 Notasi Relationship

Diagram use case digunakan selama analisa untuk mendapatkan kebutuhan sistem dan mengerti bagaimana sistem harus bekerja.

2.5.2. Class Diagram

Sebuah Obyek merupakan sebuah representasi dari sebuah entitas baik konsep maupun dunia nyata. Sebuah obyek dapat mewakili sesuatu yang nyata misalnya mobil atau komputer, atau suatu konsep misalnya proses kimia atau transaksi bank.

Sebuah Class merupakan penjelasan dari sekumpulan obyek dengan atribut, operasi , dan relasi yang umum. Setiap obyek merupakan isi dari beberapa class dan obyek tidak dapat menjadi isi lebih dari satu class.



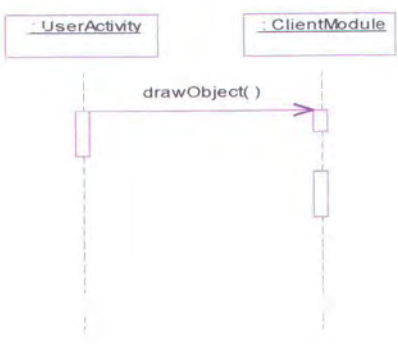
Gambar 2.20 Notasi Class

Class diagram dibuat untuk menyediakan gambar atau *view* dari beberapa atau semua class dalam model.

2.5.3. Sequence Diagram

Sequence diagram menunjukkan interaksi obyek yang disusun dalam rangkaian waktu. Sequence diagram menggambarkan obyek dan class yang termasuk dalam skenario dan rangkaian pesan yang diberikan antar obyek diperlukan untuk melaksanakan skenario. Sequence diagram berhubungan dengan use case dalam model dari sistem yang masih dalam pengembangan.

Masing-masing obyek memiliki *timeline* yang digambarkan dengan garis putus-putus dibawah obyek. Pesan antar obyek digambarkan dengan tanda panah yang menunjuk dari pengirim pesan ke penerima pesan.



Gambar 2.21 Contoh sequence diagram

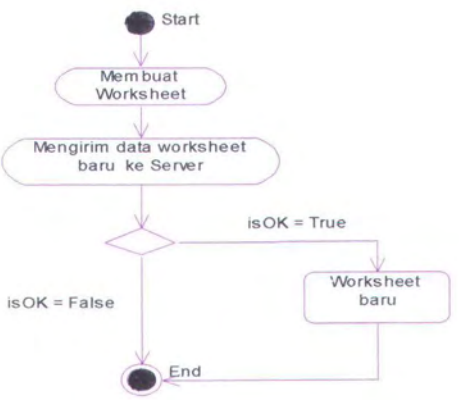
2.5.4. Statechart Diagram

State merupakan kondisi yang terjadi selama obyek masih ada ketikaobyek berada dalam suatu kondisi, melakukan aksi, atau menunggu kejadian. State dari obyek dapat dikarakteristikkan dengan nilai dari satu atau lebih atribut dari class.

Diagram transisi state mencakup semua pesan dimana obyek dapat mengirim dan menerimanya. Skenario mewakili satu alur diagram transisi. Interval antara dua pesan yang dikirim oleh obyek mewakili sebuah state.

2.5.5. Activity Diagram

Sebuah *activity* mewakili pelaksanaan tugas atau aktifitas dalam *workflow* dan juga mewakili pelaksanaan statemen dalam sebuah prosedur. Sebuah activity mirip dengan state, tapi dalam activity tidak ada proses menunggu.



Gambar 2.22 Contoh activity diagram

BAB III

PEMODELAN BASIS DATA KONSEPTUAL

BAB III

PEMODELAN BASIS DATA KONSEPTUAL

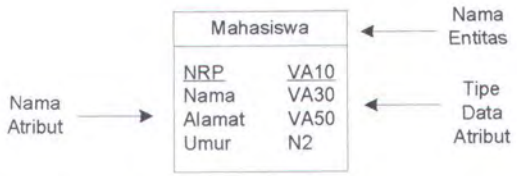
3.1. Pemodelan PowerDesigner DataArchitect

Case Tool yang umum digunakan oleh analis untuk melakukan pemodelan basis data salah satunya adalah PowerDesigner. PowerDesigner memiliki kegunaan diantaranya adalah untuk membuat model basis data konseptual atau disebut *Conceptual Data Model* (CDM) dan model basis data fisik atau disebut *Physical Data Model* (PDM). CDM mewakili seluruh struktur logika dari sebuah basis data yang tidak bergantung pada perangkat lunak dan struktur penyimpanan data. Sebuah model konseptual selalu berisi obyek data yang belum diimplementasikan dalam basis data fisik.

Model data konseptual dalam PowerDesigner memiliki beberapa obyek yang digunakan untuk memodelkan suatu sistem informasi yaitu :

- Entity atau entitas

  Entitas mewakili obyek yang didefinisikan dalam sistem informasi sebagai tempat dimana informasi akan disimpan. Misalnya dalam sebuah model mengenai mahasiswa dan dosen maka entitasnya adalah Mahasiswa dan Dosen. Sebuah entitas memiliki elemen dasar yaitu atribut yang merupakan data atau detail dari entitas. Notasi yang digunakan untuk menggambarkan sebuah entitas dalam diagram adalah :

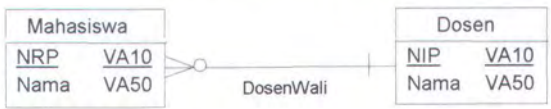


Gambar 3.1 Contoh sebuah entitas

Atribut yang digambarkan dengan garis bawah merupakan atribut pengenal.

- Relationship atau relasi

  Relasi merupakan hubungan antara dua entitas. Misalnya dalam sebuah CDM Universitas ada sebuah relasi dengan nama AnakWali yang menghubungkan entitas Mahasiswa dan entitas Dosen dimana Mahasiswa mempunyai hubungan sebagai anak wali dari Dosen. Relasi tersebut juga memperlihatkan adanya hubungan timbal balik yaitu mahasiswa mempunyai dosen wali dan dosen mempunyai anak wali.

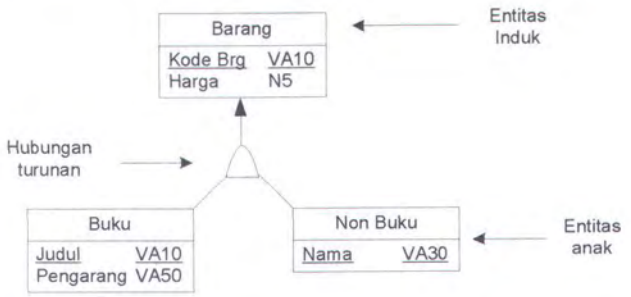


Gambar 3.2 Contoh sebuah relasi dari dua entitas

- Inheritance atau turunan

  Inheritance atau turunan digunakan untuk mendefinisikan sebuah entitas sebagai sebuah spesialisasi dari sebuah entitas lainnya yang bersifat lebih general. Entitas yang dimasukkan dalam sebuah turunan mempunyai karakteristik yang hampir sama tapi berbeda. Entitas yang bersifat general merupakan entitas induk dan berisi semua karakteristik yang umum.

Entitas anak yang menjadi turunannya berisi semua karakteristik yang khusus. Satu atau lebih entitas anak dapat diturunkan dengan menggunakan sebuah hubungan turunan.



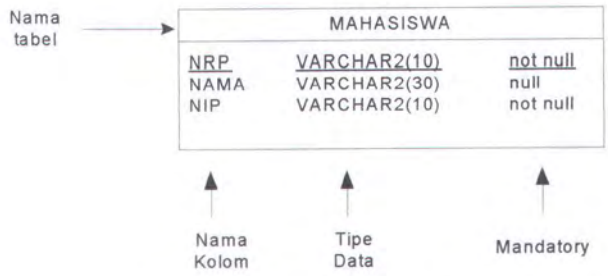
Gambar 3.3 Contoh sebuah entitas

PDM merupakan gambaran dari implementasi basis data secara fisik. PDM bergantung pada aplikasi dan struktur penyimpanan data. CDM yang telah dibuat dapat di-*generate* ke dalam bentuk PDM sesuai dengan DBMS yang diinginkan. Pembuatan PDM dapat juga dilakukan melalui pembuatan secara langsung melalui PDM editor dan melalui proses *reverse engineering* dari basis data ke dalam bentuk diagram PDM. DBMS untuk PDM yang akan dijelaskan berikut ini adalah Oracle 8i.

Ada beberapa obyek yang dimiliki oleh PDM dalam PDDA yaitu diantaranya :

- Tabel

  Tabel merupakan kumpulan data yang disusun dalam kolom dan baris. Tabel dalam model ini merupakan padanan dari tabel yang ada dalam basis data. Sebuah tabel terdiri dari satu atau lebih kolom yang berisi data dalam bentuk baris.



Gambar 3.4 Contoh sebuah tabel

- Referensi

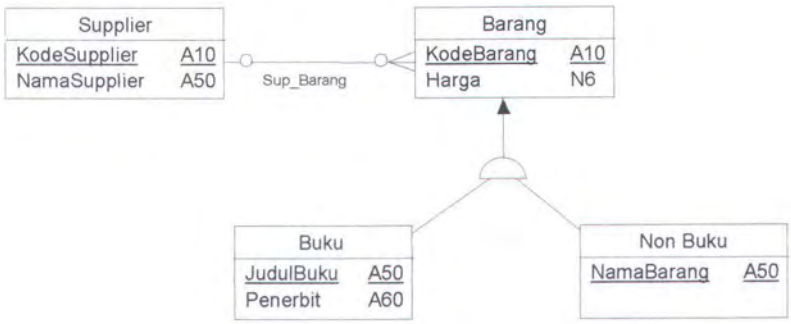
  Referensi menghubungkan primary key dalam tabel induk ke foreign key dalam tabel anak. Integritas referensial mengacu pada aturan yang mengatur konsistensi data, khususnya interaksi antara primary key dan foreigh key dalam tabel yang berbeda. Integritas referensial mendikte apa yang terjadi jika terjadi proses *update* dan *delete* sebuah nilai dalam kolom yang direferensi dalam tabel induk dan ketika terjadi proses *delete* sebuah baris data yang berisi kolom yang direferensi dari tabel induk.

Gambar 3.5 Contoh referensi dari dua table

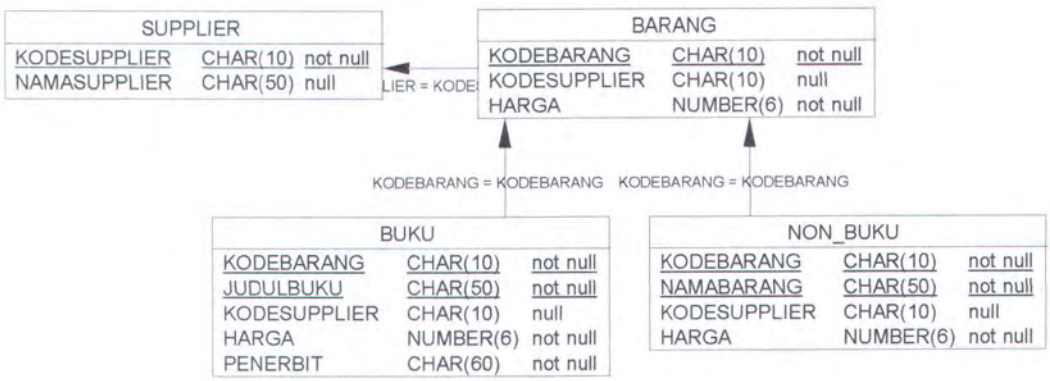
Beberapa fitur yang terdapat dalam case tool PowerDesigner ini diantaranya adalah :

- Memodelkan sebuah sistem informasi menggunakan diagram *Entity Relationship* yang disebut dengan Conceptual Data Model (CDM).

- Membuat Physical Data Model (PDM) yang berhubungan dengan diagram CDM yang telah dibuat dengan memilih target Database Management Sistem (DBMS)-nya.
- Membuat *script* pembuatan database dengan DBMS yang telah ditentukan.
- Melakukan Reverse Engineering dari bentuk PDM ke dalam bentuk CDM.



Gambar 3.6 Contoh diagram CDM PowerDesigner



Gambar 3.7 Contoh diagram PDM PowerDesigner

Selama ini aktifitas pemodelan yang dilakukan dengan menggunakan PowerDesigner tidak dapat dipantau atau dilihat oleh pengguna yang lain yang menggunakan file atau worksheet yang sama. Sehingga data terakhir yang dilihat oleh beberapa pengguna file yang sama tidak dijamin validitasnya yaitu tidak ada

jaminan apakah diagram yang sedang dipakai oleh satu pengguna sama strukturnya dengan diagram yang sedang dipakai oleh pengguna yang lain. Untuk mengetahui data terakhir dari sebuah diagram adalah dengan menutup diagram tersebut terlebih dahulu dan membukanya kembali. Hal ini tentunya tidak efisien. Dengan demikian ada beberapa fitur yang tidak dimiliki oleh PowerDesigner yaitu:

1. Tidak memiliki kemampuan untuk menampilkan struktur diagram terakhir yang digunakan oleh beberapa orang secara bersamaan.
2. Tidak dapat digunakan untuk melakukan presentasi jarak jauh dimana pengguna yang lain dapat memantau segala perubahan yang terjadi pada sebuah diagram.
3. Tidak dapat mengetahui siapa saja pengguna yang sedang mengakses sebuah diagram.

3.2. Pemodelan Basis Data Fisik Dengan Aplikasi PDM Editor

PDM editor disini merupakan case tool yang berbasis web. Aplikasi ini digunakan untuk memodelkan basis data fisik untuk RDMBS Oracle. PDM Editor dibangun dengan menggunakan teknologi ISAPI, XML, XMLDOM, and ActiveX Control. Case tool ini menerapkan sistem *client server* dengan arsitektur sistem *three-tier* dan terdiri dari dua aplikasi utama yaitu aplikasi pada sisi client dan aplikasi yang ada pada sisi server. Ada beberapa kemiripan dalam hal visualisasi diagram dengan case tool PowerDesigner dalam hal pemodelan basis data fisik. Beberapa fitur yang ada dalam PDM editor adalah sebagai berikut :

2. Memiliki kemampuan untuk menampilkan data terakhir yang digunakan oleh beberapa pengguna secara bersamaan.
3. Dapat digunakan sebagai media untuk presentasi jarak jauh.
4. Menggunakan konsep client server dalam hal komunikasi datanya.
5. Memiliki kemampuan untuk melakukan *forward engineering* dari bentuk diagram fisik menjadi sebuah basis data Oracle.
6. Memiliki kemampuan untuk melakukan *reverse engineering* dari basis data Oracle kedalam bentuk diagram fisik.
7. Memiliki kemampuan untuk melakukan komunikasi dengan pengguna yang lain.

Pemodelan sebuah basis data secara langsung dalam bentuk model basis data fisik memiliki tingkat kesulitan yang lebih bila dibandingkan dengan pemodelan dalam bentuk model konseptual. Pada umumnya para analis merancang sebuah basis data dengan mendesain sebuah diagram basis data konseptual terlebih dahulu untuk kemudian melakukan proses *forward engineering* ke dalam model fisiknya sesuai dengan DBMS yang akan digunakan. Dengan mempertimbangkan hal diatas, ada beberapa fitur yang tidak dimiliki oleh PDM editor:

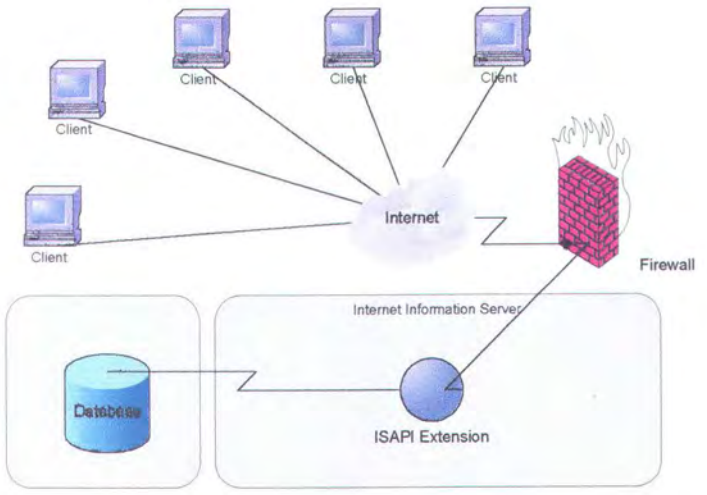
1. Tidak dapat melakukan proses *reverse engineering* dari model basis data fisik menjadi model basis data konseptual.
2. Tidak dapat melakukan pemodelan basis data konseptual.

3. Tidak dapat mengetahui pengguna yang sedang aktif dan sedang menggunakan diagram yang sama walaupun dapat berkomunikasi dengan pengguna lain.

3.3. Konsep Umum Pemodelan Basis Data Konseptual Berbasis Web

Data terakhir yang dihasilkan oleh proses seorang pengguna diperlukan untuk mengatasi permasalah keakuratan dan kesamaan data yang digunakan oleh beberapa pengguna. Salah satu solusi untuk mengatasi permasalahan validitas data terakhir yang digunakan oleh pengguna adalah penerapan aplikasi berbasis web. Pada aplikasi yang berbasis web ini digunakan beberapa teknologi untuk mendukungnya yaitu Internet Information Server (IIS), Internet Server Application Programming Interface (ISAPI) Extension, ActiveX Control, XML dan XMLDOM.

ISAPI diimplementasikan dalam bentuk DLL yang diletakkan dalam proses IIS yang berada pada sisi server. IIS bertugas untuk menerima setiap *request* dari client untuk kemudian diteruskan ke ISAPI yang sudah di-*load*. ISAPI memiliki fungsi yang digunakan untuk memproses semua *request* tadi. Dalam aplikasi ini fungsi-fungsi yang dimiliki oleh ISAPI antara lain untuk menyimpan data terakhir yang terupdate oleh beberapa pengguna dan menyimpan data ke dalam basis data repository. Teknologi ActiveX Control digunakan pada sisi client yaitu dengan meracang sebuah komponen sehingga dapat digunakan dalam sebuah halaman web. Pemodelan yang dilakukan pada sisi client ditangani oleh komponen ActiveX Control dimana pengguna dapat berinteraksi secara langsung dengan diagram yang dibuat.



Gambar 3.8 Arsitektur Sistem

Komunikasi antara client dan server memanfaatkan teknologi XML dan XMLDOM. Data yang dihasilkan dari sebuah proses dari client akan di ubah kedalam format XML. File XML tersebut kemudian di kirimkan ke server dengan menggunakan XMLDOM. Demikian juga sebaliknya data yang akan dikirimkan dari sisi server ke client dikonversi kedalam bentuk XML.

3.4. Konsep Teknis Pemodelan Basis Data Konseptual Berbasis Web

Case tool pemodelan basis data konseptual berbasis web yang akan dibuat ini memiliki dua aplikasi utama yaitu aplikasi pada sisi server yang untuk selanjutnya disebut aplikasi server dan aplikasi pada sisi client yang untuk selanjutnya disebut dengan aplikasi client. Sebagaimana disebutkan sebelumnya case tool ini menggunakan teknologi Internet Server Application Programming Interface (ISAPI) Extension dalam komunikasi data dengan menerapkan konsep *client server* yang disesuaikan dengan aplikasi yang akan dibuat ini.

Aplikasi client merupakan aplikasi yang digunakan untuk melakukan pemodelan basis data konseptual dimana aplikasi ini berinteraksi secara langsung dengan pengguna.. Pada aplikasi ini ada beberapa kemiripan dengan PowerDesigner yaitu dari segi notasi pemodelannya. Hal ini dilakukan karena saat ini masih banyak para analis sistem informasi yang menggunakan PowerDesigner. Untuk menampilkan data terbaru yang terupdate dalam server, aplikasi ini melakukan *auto refresh* yaitu dengan mengirimkan request untuk mengambil data terbaru ke server kemudian server akan mengembalikan nilai data terbaru jika memang ada data yang terupdate. Interval waktu untuk melakukan auto refresh ini adalah 3 detik.

Beberapa obyek pemodelan yang dimiliki oleh case tool yang akan dibuat ini adalah entitas, relasi dan turunan. Obyek-obyek tersebut mewakili obyek-obyek yang sama pada model *entity relationship* yang telah dijelaskan pada bab sebelumnya. Penggambaran obyek-obyek ini dilakukan oleh komponen ActiveX Control yang dirancang pada aplikasi client. Proses yang terjadi pada obyek-obyek ini adalah penambahan, penghapusan dan perubahan. Setiap kali proses tersebut dilakukan oleh pengguna, aplikasi client akan mengirimkan data untuk diproses dan disimpan oleh aplikasi server.

Aplikasi server dibangun dengan menerapkan teknologi ISAPI Extension untuk menangani dan memproses *request* yang dikirimkan oleh aplikasi client. Aplikasi *server* ini mengolah data yang diterima dari *client* dalam bentuk XML dan mengembalikan nilai ke *client* dalam bentuk XML pula. Sebagaimana aplikasi client, aplikasi server juga memiliki obyek-obyek yaitu entitas, relasi dan turunan.

Akan tetapi obyek-obyek ini tidak berinteraksi secara langsung dengan pengguna karena berada pada aplikasi server yang berupa DLL. Obyek-obyek ini digunakan untuk menampung data terakhir yang ter-*update* yang dikirimkan oleh aplikasi client.

Disamping dua aplikasi diatas ada sebuah basis data yang digunakan sebagai penyimpanan data atau *repository*. Proses penyimpanan data obyek ini secara langsung dilakukan oleh aplikasi server setelah menerima request dari aplikasi client. Repository ini dibangun dengan menggunakan RDBMS Oracle. Ada dua mode penggunaan case tool ini yaitu mode single user dan mode multi user. Pada mode single user pengguna hanya bekerja seorang diri dan datanya hanya dapat disimpan dalam file lokal. Pada mode multi user, pengguna dapat bekerja atau menggunaka worksheet secara bersama-sama dengan pengguna yang lain dan dapat menyimpannya yaitu dalam file lokal atau ke dalam basis data repository.

Untuk mendukung kesempurnaan dari aplikasi yang akan dibuat maka perlu memasukkan aplikasi PDM Editor ke dalam aplikasi ini. Hasil proses forward engineering dari model konseptual ke model fisik dapat dilihat di aplikasi PDM Editor. PDM editor juga dapat digunakan untuk melakukan perubahan data dari model fisik yang dihasilkan dari proses forward engineering. Model fisik yang ada di aplikasi PDM editor dapat diubah kembali dengan proses reverse engineering ke dalam bentuk konseptual. Dengan demikian nantinya ada dua jenis proses reverse engineering pada PDM editor yaitu dari basis data ke dalam bentuk model fisik dan dari model fisik menjadi model konseptual.

3.5. Kebutuhan Sistem Pemodelan Basis Data Konseptual

Dengan memperhatikan beberapa fitur yang tidak dimiliki oleh case tool semacam yang telah disebutkan sebelumnya maka case tool yang akan dibuat harus memiliki kriteria sebagai berikut :

1. Dapat memodelkan basis data konseptual.

   Pembuatan struktur basis data langsung dengan menggunakan pemodelan basis data fisik adalah tidak umum dilakukan. Dibutuhkan aplikasi yang juga dapat digunakan untuk memodelkan basis data konseptual. Untuk memudahkan pembuatan perangkat lunak maka notasi untuk obyek dalam case tool yang akan dibuat mengadopsi notasi obyek dalam case tool PowerDesigner.

2. Bersifat Online-Real Time

   Data yang tampil pada satu pengguna harus sama dengan data yang ada pada pengguna yang lain. Perubahan yang dilakukan oleh pengguna yang satu dapat dipantau oleh pengguna yang lain. Untuk itu diperlukan sebuah sistem yang real time. Untuk memenuhi kebutuhan akan adanya fitur multi user maka aplikasi harus online sehingga data dapat diakses dari berbagai tempat.

3. Dapat berkomunikasi dengan pengguna lain

   Untuk mendukung kelancaran dalam melakukan pemodelan maka dibutuhkan komunikasi dengan pengguna yang lain. Hal ini dapat dilihat jika sistem yang akan dibuat melibatkan sekian banyak orang yang membutuhkan pemikiran yang rumit tentu akan membuang

waktu jika harus melakukan mobilitas hanya untuk mengkonfirmasikan atau mendiskusikan sebuah model dengan pengguna yang lain.

4. Manajemen Model

   Pengguna yang menggunakan aplikasi ini dapat berjalan secata single user dan multi user sehingga perlu adanya pengaturan untuk menambah atau menghapus pengguna. Diagram basis data konseptual yang dibuat oleh pengguna dibuat dalam sebuah worksheet. Worksheet ini dapat disimpan dan dibuka kembali baik dalam mode single user maupun multi user.

5. Konkurensi data

   Aplikasi yang menggunakan sistem client server adakalanya mempunyai kendala dalam hal konkurensi data yaitu perubahan data oleh seorang pengguna tidak menimbulkan kerancuan data yang sedang diakses oleh pengguna yang lain. Sehingga diperlukan sistem yang dapat menjaga konkurensi data yang sedang diakses oleh beberapa pengguna.

6. Dapat melakukan forward dan reverse engineering

   Case tool yang akan dibuat haruslah memiliki kemampuan untuk melakukan proses forward engineering dari model konseptual ke dalam model fisik dan sebaliknya melakukan proses reverse engineering dari model fisik ke dalam model konseptual. Pada proses

forward engineering, DBMS yang digunakan oleh case tool ini adalah Oracle.

BAB IV

PERANCANGAN PERANGKAT LUNAK

BAB IV

PERANCANGAN PERANGKAT LUNAK

Bab ini menjelaskan tentang tahapan proses perancangan perangkat lunak mulai dari penggambaran aplikasi secara umum sampai perancangan data, perancangan proses dan perancangan antar muka dengan pengguna.

4.1 Desain Fitur

Sebelum membahas perancangan proses maupun perancangan data, analisa kebutuhan sistem yang dibahas pada bab III dengan poin-poin berikut:

a. Dapat memodelkan basis data konseptual.
b. Bersifat Online-Real Time
c. Manajemen model dan berkomunikasi dengan pengguna lain
d. Konkurensi data
e. Dapat melakukan forward dan reverse engineering

akan dijelaskan fitur-fitur yang akan dibuat untuk memenuhi dari kebutuhan tersebut. Perancangan fitur sendiri akan dapat menjelaskan secara global gambaran tentang perangkat lunak yang akan dibuat.

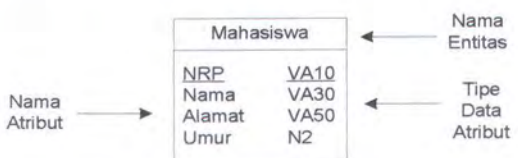
4.1.1 Memodelkan Basis Data Konseptual

Salah satu tujuan dalam perancangan aplikasi dalam tugas akhir ini adalah untuk membuat aplikasi yang dapat memodelkan basis data konseptual. Pemodelan basis data konseptual ini dilakukan pada sisi client yang berinteraksi

langsung dengan penguna. Model data konseptual yang akan dibuat dalam aplikasi ini mengadopsi model basis data konseptual dari case tool PowerDesigner. Model basis data konseptual yang ada dalam aplikasi ini memiliki beberapa obyek. Berikut ini penjelasan mengenai obyek-obyek tersebut dengan sedikit mengulang penjelasan pada bab sebelumnya :

- Entitas

  Entitas mewakili obyek yang digunakan untuk menyimpan informasi. Misalnya entitas Mahasiswa untuk menyimpan data tentang mahasiswa. Sebuah entitas memiliki elemen dasar yaitu atribut. Entitas memiliki informasi tentang posisi dalam model yaitu panjang, tinggi, posisi kiri atas, nama, code, warna (tulisan, *background*, dan garis tepi) serta apakah entitas itu akan digenerate kedalam model fisik atau tidak. Contoh sebuah entitas yang digambarkan dalam sebuah model basis data konseptual adalah sebagai berikut:



Gambar 4.1 Contoh sebuah entitas

  Obyek entitas juga memiliki informasi tentang kolom atau atribut yang terdapat didalamnya. Setiap kolom menyimpan informasi tentang nama, code, tipe data, *primary key*, boleh diisi null atau tidak (*mandatory*).

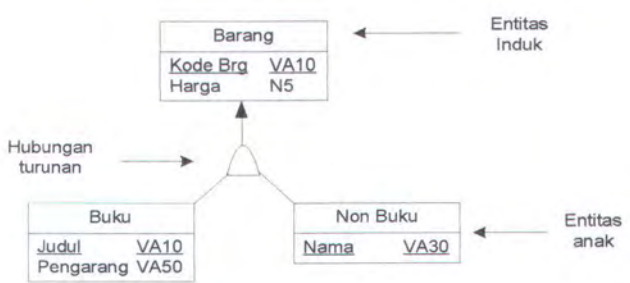
- Relasi

Relasi merupakan hubungan antara dua entitas. Obyek relasi dalam model memiliki informasi tentang nama, code, *cardinality*, entitas induk dan anak, *dependent*(dependency entitas yang satu terhadap yang lain), *dominant*(dominasi entitas yang satu terhadap yang lain), *mandatory* dan posisi relasi terhadap entitas yang dihubungkan.

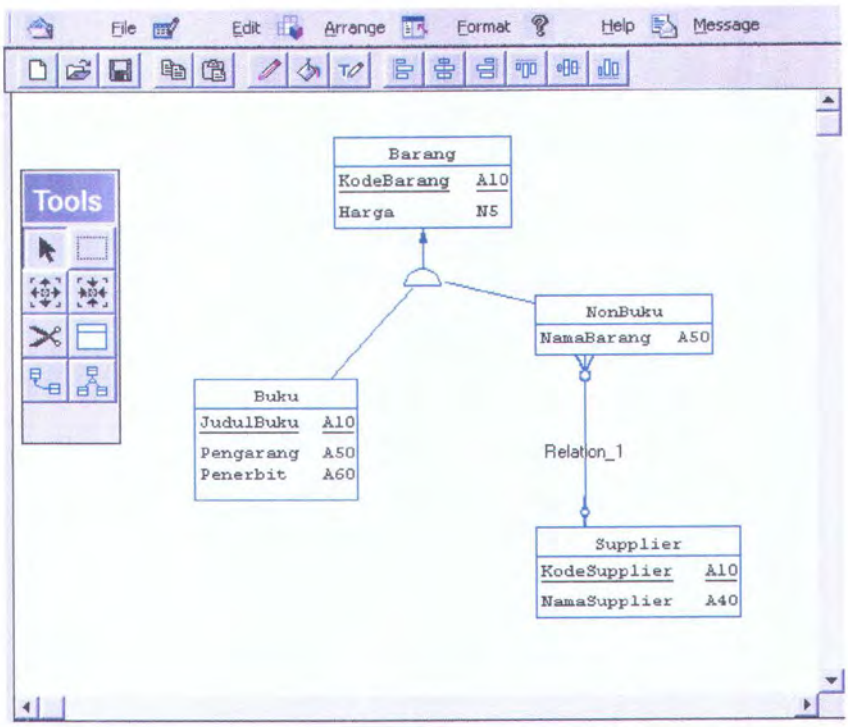
Gambar 4.2 Contoh sebuah relasi dari dua entitas

- Turunan

Turunan digunakan untuk mendefinisikan sebuah entitas sebagai sebuah spesialisasi dari sebuah entitas lainnya yang bersifat lebih general. Obyek turunan dalam model memiliki informasi tentang nama, code, entitas induk, posisi, *mutually exclusive*, jenis turunan yaitu semua atribut entitas induk atau hanya atribut primer entitas induk yang diturunkan. Obyek ini juga menyimpan informasi tentang entitas anak dalam turunan dimana obyek turunan anak ini memiliki informasi tentang entitas induk dan entitas anak itu sendiri.



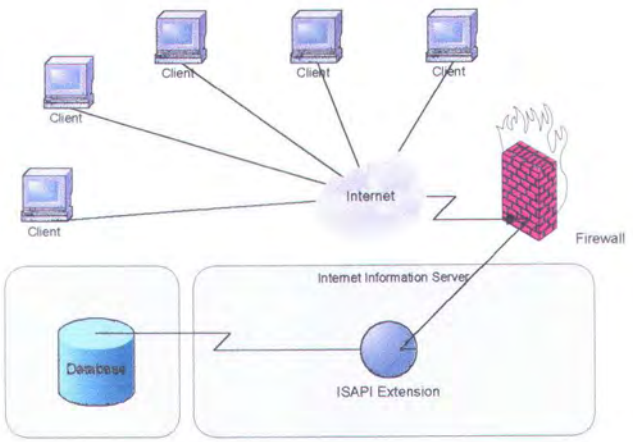
Gambar 4.3 Contoh sebuah turunan



Gambar 4.4 Contoh antar muka sebuah diagram

4.1.2 Bersifat Online Realtime

Bersifat online realtime disini berarti perubahan data oleh satu pengguna dalam pemodelan dapat dilihat oleh pengguna yang lain pada saat itu juga dengan selisih waktu kurang lebih 3 detik. Penggunaan selisih waktu ini dikarenakan aplikasi membutuhkan waktu untuk memproses perubahan data dalam server sedangkan aplikasi server sendiri menggunakan metode *polling* dimana server menunggu *request* dari *client*. Untuk memenuhi kebutuhan ini maka digunakan sistem client server yang disesuaikan dengan aplikasi dimana server memanfaatkan teknologi Internet Information Server (IIS) dan Internet Server Application Programming Interface (ISAPI).



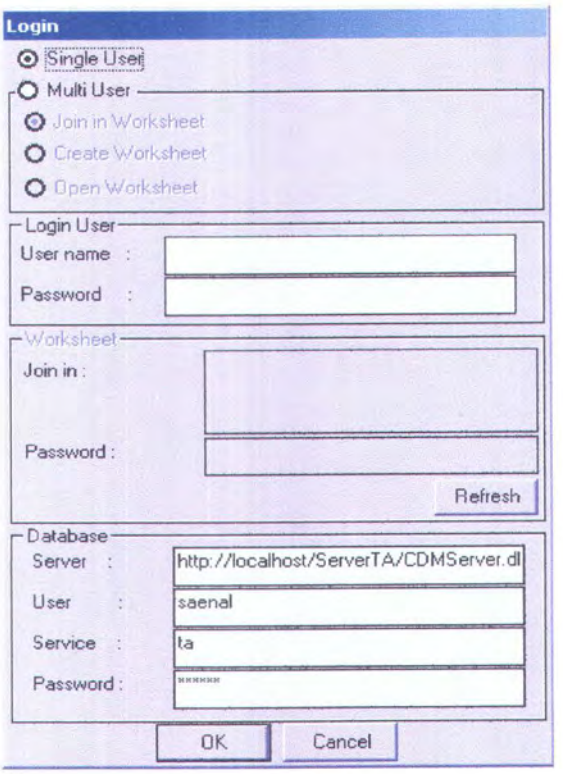
Gambar 4.5 Arsitektur Sistem

Segala aktifitas yang terjadi pada client akan dikirimkan ke server berupa *request*. Server akan menyimpan aktifitas tersebut dan mengupdate data sesuai dengan aktifitas dalam *request*-nya. Jika client mengirimkan request untuk proses penyimpanan maka server akan menyimpan data ke dalam media penyimpanan sesuai dengan mode penguna yang mengirimkan. Setiap 3 detik aplikasi client akan memeriksa apakah terjadi perubahan pada data yang ada diserver, jika ada perubahan maka client akan mengirimkan *request* untuk meminta data yang terbaru.

4.1.3 Manajemen Model Dan Berkomunikasi Dengan Pengguna Lain

Manajemen model disini adalah pembuatan worksheet, penambahan dan penghapusan pengguna yang online, penambahan obyek, penghapusan obyek, perubahan obyek, penyimpanan data dan pengambilan data. Tempat pembuatan model pada aplikasi client disebut dengan worksheet. Pengguna yang membuat worksheet merupakan *administrator* dari worksheet tersebut dimana administrator dapat menghapus pengguna lain yang sedang online dalam worksheetnya. Setiap

pengguna yang ingin menggunakan aplikasi tugas akhir ini harus terdaftar dalam repository dengan memiliki *user name* dan *password*. Pengguna aplikasi ini dibagi menjadi dua mode yaitu *single user* dan *multi user* dan ini ditentukan pada saat pengguna *login*.



Gambar 4.6 Fasilitas Login

- Pembuatan worksheet dan penghapusan worksheet

  Pada mode single user worksheet dibuat pada saat pengguna login kedalam aplikasi. Worksheet pada mode ini tidak dapat diakses oleh pengguna yang lain karena informasi tentang worksheet ini tidak dikirimkan ke server. Pada mode multi user worksheet dibuat pada saat pengguna login dan memilih untuk membuat worksheet baru dan worksheet ini akan dikirimkan ke server untuk ditambahkan ke dalam

obyek worksheet yang ada pada aplikasi server. Dengan demikian worksheet ini akan dapat diakses oleh pengguna yang lain. Penghapusan worksheet dilakukan pada saat pengguna logout dari aplikasi untuk kedua mode pengguna (single user atau multi user).

- Penambahan dan penghapusan pengguna

  Penambahan pengguna terjadi pada mode multi user. Penambahan pengguna disini berarti penambahan pengguna yang melakukan join pada sebuah worksheet. Pengguna yang join dalam sebuah worksheet akan ditambahkan dalam data pengguna di aplikasi server. Segala aktifitas perubahan data pada worksheet yang dilakukan akan dapat dilihat oleh pengguna yang terdapat dalam daftar pengguna dalam workshet tersebut. Penghapusan pengguna dapat terjadi karena dua hal yaitu pengguna logout dan pengguna yang dihapus oleh administrator worksheet. Jadi pembuat (administrator) worksheet mempunyai hak untuk menghapus pengguna yang bergabung dalam worksheet yang sedang aktif yang dibuatnya.

- Penambahan, penghapusan dan perubahan obyek model

  Perancangan diagram entity relationship dengan menggunakan aplikasi ini melibatkan beberapa proses yaitu penambahan, penghapusan dan perubahan obyek model (entitas, relasi, dan turunan). Proses-proses ini dilakukan melalui diagram editor secara langsung. Namun jika pengguna sedang berada dalam mode multiuser, maka penambahan, penghapusan dan perubahan obyek diagram juga dapat ditimbulkan oleh proses yang

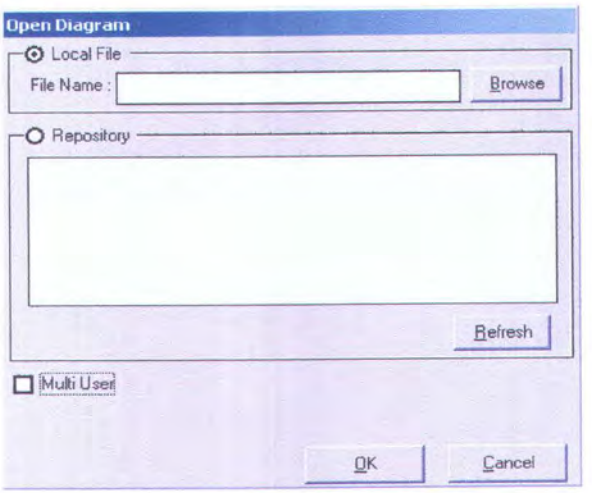
sama pada server dimana proses pada server ini terjadi karena adanya aktifitas penambahan, penghapusan dan perubahan oleh pengguna yang lain yang tergabung dalam worksheet yang sama. Perubahan obyek diagram dilakukan pada tampilan dari obyek, misalnya warna, letak dan ukuran obyek. Perubahan ini juga dapat dilakukan pada property dari obyek tersebut, seperti nama, code, kolom, tipe data pada obyek entitas atau merubah jenis relasi pada sebuah obyek relasi.

- Penyimpanan dan pengambilan data dalam repository

  Diagram yang sudah dibuat dalam sebuah worksheet dapat disimpan dalam file. Penyimpanan ini dapat dilakukan dalam komputer lokal berupa file XML atau dalam database server dengan format yang disesuaikan dengan aplikasi ini. Penyimpanan ini dapat dilakukan oleh semua pengguna. Pengguna yang berada pada mode single user hanya dapat menyimpan diagram ke dalam file lokal dan tidak dapat menyimpan diagram ke dalam reptository.

  Diagram yang telah tersimpan dapat dibuka kembali dengan mode single user atau multi user dengan catatan apabila pengguna berada dalam mode single user kemudian akan membuka diagramnya dalam mode multi user maka aplikasi akan membuat worksheet baru. Pengguna yang dapat membuka diagram dari repository hanya pengguna yang berstatus sebagai administrator dari worksheet yang akan dibuka. Pengguna yang berada pada mode multi user yang membuka worksheet dari repository yang berbeda dengan worksheet yang sedang dibuka akan logout secara

otomatis dan mengirimkan request ke server untuk menghapus worksheet yang sedang dibuka.

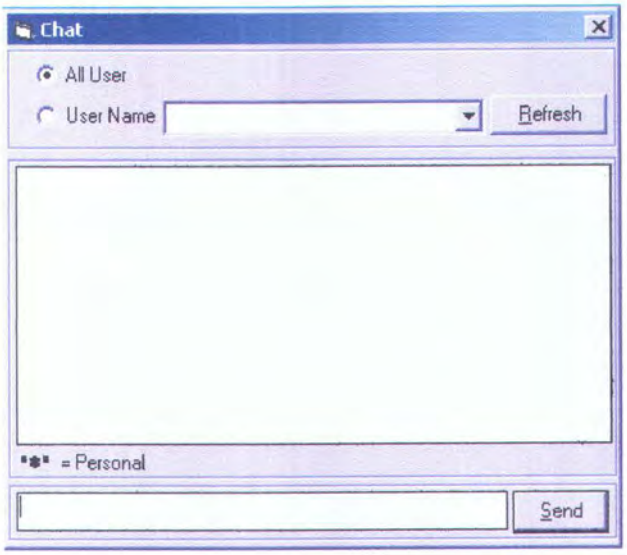


Gambar 4.7 Fasilitas Membuka file di lokal atau repository

Setiap pengguna yang sedang online dalam sebuah worksheet dengan mode multi user akan disimpan dalam aplikasi server sebagai pengguna yang sedang aktif. Untuk mengetahui pengguna yang sedang online maka client mengirimkan requestnya ke server dan server akan mengembalikan nilai yang berisi informasi tentang pengguna-pengguna yang sedang online dalam worksheet yang sama dengan pengguna yang mengirimkan request.

Aplikasi *client* dalam tugas akhir ini dilengkapi dengan sebuah fasilitas *chatting* yang digunakan untuk komunikasi antar pengguna dalam worksheet yang sama. Pengguna dapat memilih apakah pesan itu akan disampaikan pada seluruh pengguna atau hanya untuk seorang pengguna tertentu.

Dengan adanya fasilitas ini diharapkan pengguna dapat membuat sebuah forum diskusi antar pengguna dalam pembuatan sebuah diagram tanpa memerlukan mobilitas yang tinggi.



Gambar 4.8 Fasilitas Komunikasi dengan Pengguna lain

4.1.4 Konkurensi Data

Konkurensi data dalam aplikasi ini sangat diperlukan untuk menjaga validitas data yang sedang digunakan oleh masing-masing pengguna. Salah satu kunci penting yang digunakan dalam konkurensi untuk aplikasi ini adalah proses yang akan ditampilkan dalam editor adalah proses yang pertama kali dilakukan pada saat tertentu. Untuk lebih jelasnya akan diberikan ilustrasi berikut ini.

Jika terdapat pengguna yang melakukan penghapusan obyek. Di lain tempat ada pengguna yang sedang *join* dalam worksheet yang sama dengan pengguna pertama yang akan meng-*update* obyek yang sama dalam sebuah worksheet, maka server akan memberitahukan pengguna kedua bahwa obyek tersebut telah dihapus oleh pengguna yang lain. Sedangkan jika *action* yang dilakukan oleh pengguna adalah proses Update maka server akan memakai hasil update dari pengguna yang pertama kali melakukan proses update dengan mengunci obyek yang terupdate oleh proses pertama di server.

4.1.5 Forward Dan Reverse Engineering

Aplikasi ini dilengkapi dengan fasilitas untuk melakukan *forward engireering* dari model konseptual ke model fisik database dengan tipe data Oracle. Berikut ini adalah algoritma dalam proses *forward engineering* :

1. Melakukan pengecekan apakah terjadi *looping* atau siklus dalam model konseptual dari relasi yang memiliki *dependency*. Jika terdapat *looping* tersebut maka proses forward engineering tidak dapat dilakukan.
2. Untuk setiap entitas dalam diagram dibuat tabel dengan atribut atau kolom sama dengan semua atribut yang ada dalam entitas yang bersangkutan. Tipe data dari atribut atau kolom entitas dikonversi kedalam bentuk tipe data Oracle.
3. Untuk setiap turunan dimana entitas induk di-*generate* dan entitas anak tidak di-*generate* dan entitas anak bukan merupakan entitas induk dari turunan yang lain, maka kolom dalam tabel dari entitas anak dimasukkan ke dalam tabel dari entitas induk.
4. Untuk setiap turunan dimana entitas induk tidak di-*generate* dan entitas anak di-*generate*:
    - Jika semua attibut entitas induk diturunkan maka masukkan semua kolom dalam tabel dari entitas induk dan buat relasi yang menghubungkan tabel dari entitas anak dengan tabel dari entitas yang digenerate sesuai dengan alur turunannya.
    - Jika semua attibut entitas induk diturunkan maka masukkan kolom primer dalam tabel dari entitas induk dan buat relasi yang

menghubungkan tabel dari entitas anak dengan tabel dari entitas yang digenerate sesuai dengan alur turunannya.

5. Untuk setiap relasi dengan *cardinality* 1 : 1 (*One to One*) yang menghubungkan antara 2 entitas misalnya entitas A dengan entitas B :
   - Jika salah satu *mandatory*-nya bernilai *True* (misalnya entitas A), maka masukkan *primary key* dari entitas B sebagai *foreign key* dalam entitas A dan *foreign key* ini tidak boleh diisi NULL. Kemudian masukkan *primary key* dari entitas A sebagai *foreign key* dalam entitas B dan *foreign key* ini boleh diisi dengan NULL.
   - Jika dua-duanya *mandatory*-nya bernilai *True* maka masukkan *primary key* dari entitas B sebagai *foreign key* dalam entitas A dan begitu juga sebaliknya. Dan masing-masing *foreign key* tidak boleh diisi dengan NULL.
6. Untuk setiap relasi dengan *cardinality* 1 : N (*One to Many*) yang menghubungkan antara 2 entitas misalnya entitas A dan entitas B, maka masukkan *primary key* dari entitas A sebagai *foreign key* dalam entitas B:
   - Jika entitas A *mandatory*-nya adalah *True* maka *foreign key* harus diisi (tidak boleh NULL).
   - Jika entitas A *mandatory*-nya adalah *False* maka *foreign key* boleh diisi dengan NULL.
7. Untuk setiap relasi dengan *cardinality* 1 : N (*One to Many*) dan memiliki *dependency* yang menghubungkan antara 2 entitas misalnya entitas A dan

entitas B. maka masukkan *primary key* dari entitas A sebagai *foreign key* dalam entitas B dan *foreign key* ini menjadi *primary key* dalam entitas B.

8. Untuk setiap relasi dengan *cardinality* N : M (*Many to Many*) yang menghubungkan antara 2 entitas misalnya entitas A dan entitas B, maka buat tabel baru (misalnya C). Masukkan *primary key* dari kedua entitas (A dan B) sebagai *foreign key* dalam entitas baru (entitas C) dan *foreign key* ini tidak boleh diisi dengan NULL.
9. Untuk setiap turunan dalam diagram, setiap *subclass* atau anak misalnya entitas E dan entitas D dari *superclass* atau induk misalnya entitas A maka masukkan setiap *primary key* dalam entitas A sebagai *foreign key* dalam entitas E dan entitas D dan *foreign key* ini menjadi *primary key* dalam entitas E dan D.

Dalam tugas akhir ini dilakukan pengembangan atau penambahan fitur dalam aplikasi PDM Edtor yang sudah ada untuk melakukan reverse engineering dari model fisik ke dalam model konseptual. Pada aplikasi PDM Editor terdapat obyek tabel dan obyek referensi dengan definisi sama seperti pada Powerdesigner. Untuk mempermudah dan mempersingkat penjelasan mengenai entitas dalam hubungannya dengan konversi dari referensi menjadi relasi maka entitas dijelaskan dengan format *nama_entitas(mandatory, dependent, dominant)* dimana masing-masing variabel tersebut berisi nilai ya atau tidak (*True(T)* atau *False(F)*). Berikut ini adalah algoritma untuk melakukan reverse engineering:

1. Mencari tabel yang mempunyai spesifikasi untuk digenerate menjadi relasi dengan cardinality N:M (Many to Many). Spesifikasi disini adalah

terdapat hanya dua referensi dengan tabel anak yang sama dimana tiap kolom dalam tabel tersebut merupakan primary key dan foreign key semua.

2. Untuk setiap tabel selain tabel dengan spesifikasi cardinality N:M dibuat entitas dimana setiap kolomnya dibuat menjadi atribut dan tipe datanya dikonversi menjadi tipe data umum yang digunakan dalam aplikasi perancangan model konseptual.
3. Untuk setiap referensi yang memiliki spesifikasi dependency dalam model fisik dibuat relasi:
    - Jika cardinalitas-nya 0..1, dibuat relasi dengan cardinalitas 1:1. Entitas yang satu mempunyai format A(F, F, F) dan entitas yang lain mempunyai format B(T, T, T).
    - Jika cardinalitas-nya 0..n dibuat relasi dengan cardinalitas 1:N. Entitas yang satu mempunyai format A(F, F, F) dan entitas yang lain mempunyai format B(T, T, F).
    - Jika cardinalitas-nya 1..1 dibuat relasi dengan cardinalitas 1:1. Entitas yang satu mempunyai format A(T, F, F) dan entitas yang lain mempunyai format B(T, T, T).
    - Jika cardinalitas-nya 1..n dibuat relasi dengan cardinalitas 1:N. Entitas yang satu mempunyai format A(T, F, F) dan entitas yang lain mempunyai format B(T, T, F).
4. Untuk setiap referensi yang tidak memiliki spesifikasi dependency dalam model fisik dibuat relasi:

- Jika cardinalitas-nya 0..1, dibuat relasi dengan cardinalitas 1:1. Entitas yang satu mempunyai format A(F, F, F) dan entitas yang lain mempunyai format B(T, F, T).
- Jika cardinalitas-nya 0..n dibuat relasi dengan cardinalitas 1:N. Entitas yang satu mempunyai format A(F, F, F) dan entitas yang lain mempunyai format B(T, F, F).
- Jika cardinalitas-nya 1..1 dibuat relasi dengan cardinalitas 1:1. Entitas yang satu mempunyai format A(T, F, F) dan entitas yang lain mempunyai format B(T, F, T).
- Jika cardinalitas-nya 1..n dibuat relasi dengan cardinalitas 1:N. Entitas yang satu mempunyai format A(T, F, F) dan entitas yang lain mempunyai format B(T, F, F).

5. Untuk setiap referensi yang memiliki informasi turunan maka dibuat sebuah turunan.
6. Untuk setiap tabel dengan spesifikasi untuk digenerate menjadi relasi dengan kardinalitas N:M sebagaimana point ke 1 dibuat relasi dengan kardinalitas N:M.

4.2 Perancangan Proses

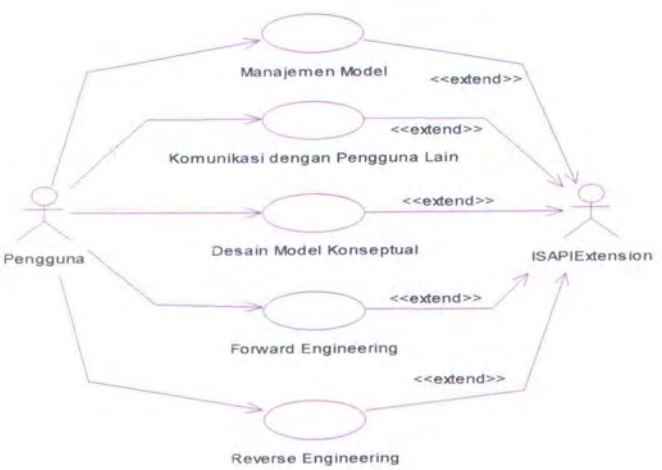
Desain proses digunakan untuk mengetahui proses apa saja yang ada dalam sistem. Desain proses dari perangkat lunak ini menggunakan pemodelan visual dengan tool Rational Rose. Desain proses akan digambarkan dalam bentuk diagram use case, diagram class, diagram state dan diagram sequence.

Proses yang terjadi dalam aplikasi ini akan dibagi menjadi dua yaitu proses dalam aplikasi client dan proses dalam aplikasi server yang akan dijelaskan pada subbab berikutnya.

4.3. Aplikasi Client

Aplikasi *client* adalah aplikasi yang memungkinkan interaksi secara langsung antara aplikasi dengan pengguna maupun antar pengguna. Pengguna dapat melakukan *diagram editing* yang dapat berupa penambahan, penghapusan dan modifikasi obyek dalam sebuah area kerja atau *worksheet*. Aplikasi *client* ini dibangun dengan menggunakan teknologi ActiveX Control (OCX).

Interaksi antara pengguna dengan aplikasi client serta antara aplikasi client dengan aplikasi server dapat digambarkan dalam diagram use case sebagai berikut:



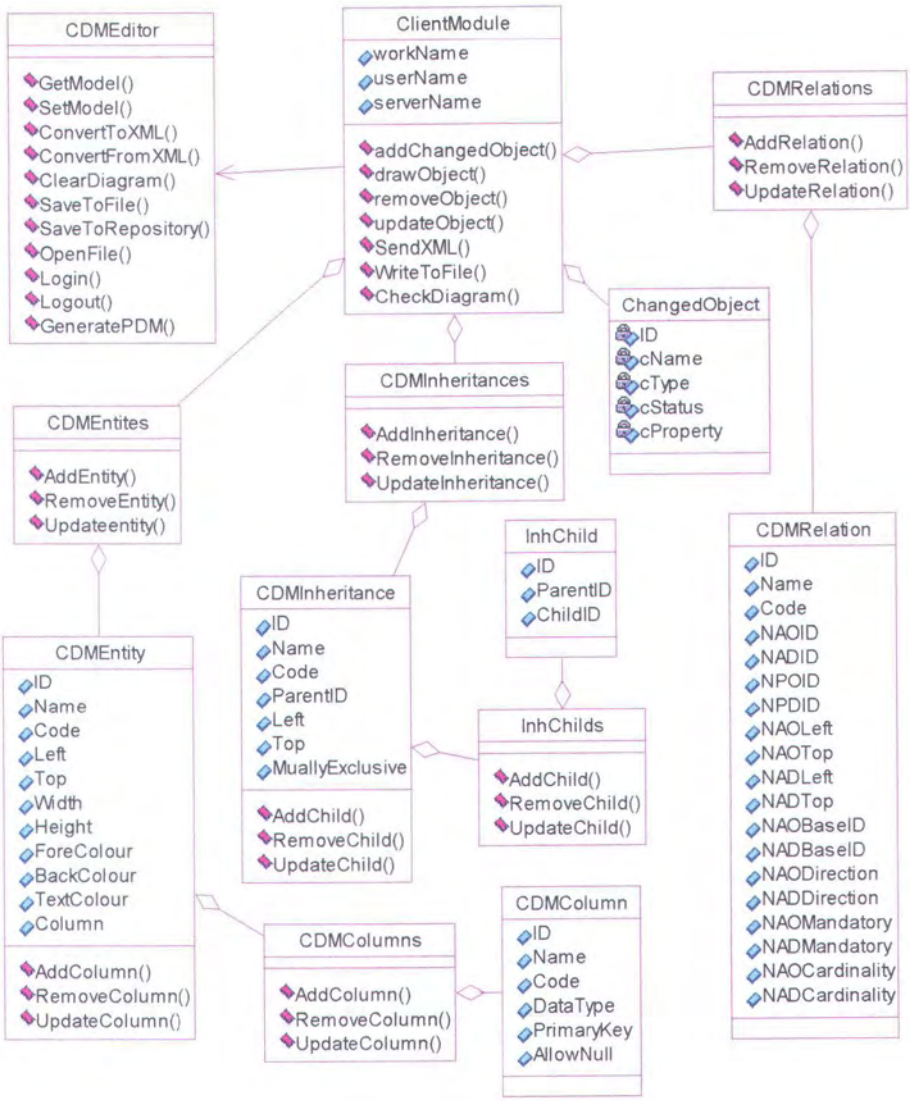
Gambar 4.9 Interaksi antara Aplikasi client, pengguna dan aplikasi server

Aplikasi ini mempunyai beberapa komponen utama yaitu:

1. **CDMEntity**, merupakan obyek diagram yang berupa entitas yang dapat mempunyai beberapa kolom (berupa obyek **CDMColumn**). Beberapa obyek CDMEntity berada dalam satu obyek *collection* (kumpulan dari beberapa obyek) yaitu **CDMEntities**.
2. **CDMRelation**, merupakan obyek diagram yang berupa relasi yang menghubungkan satu atau dua entitas. Beberapa obyek CDMRelation berada dalam satu obyek *collection* yaitu **CDMRelations**.
3. **CDMInheritance**, merupakan obyek diagram yang berupa relasi yang menghubungkan satu atau beberapa entitas dengan sebuah entitas yang bersifat lebih *general* atau entitas induknya. CDMInheritance mempunyai sebuah obyek yaitu **CDMInhChilds** yang merupakan kumpulan dari satu atau beberapa obyek **CDMInhChild** yang merupakan obyek diagram yang berisi informasi hubungan entitas yang spesifik sebagai turunan dengan entitas yang lebih umum sebagai induknya. Beberapa obyek CDMInehritance berada dalam satu obyek *collection* yaitu **CDMInheritances**.
4. **ChangedObject**, merupakan obyek yang menyimpan informasi tentang perubahan yang terjadi pada editor diagram untuk kemudian dikirimkan ke server.
5. **PDMTable,** merupakan obyek yang digunakan untuk menyimpan data tabel hasil proses *forward* dan *reverse engineering*. Dalam setiap obyek ini terdapat obyek **PDMColumns** yang berisi kolom-kolom dimana setiap kolomnya disimpan dalam obyek **PDMColumn**. Dalam **PDMColumn**

terdapat obyek **PDMColLists** yang berisi daftar kolom-kolom dari objek PDMTable yang lain yang mengacu pada kolom ini. Seluruh obyek PDMTable disimpan dalam obyek **PDMTables**.

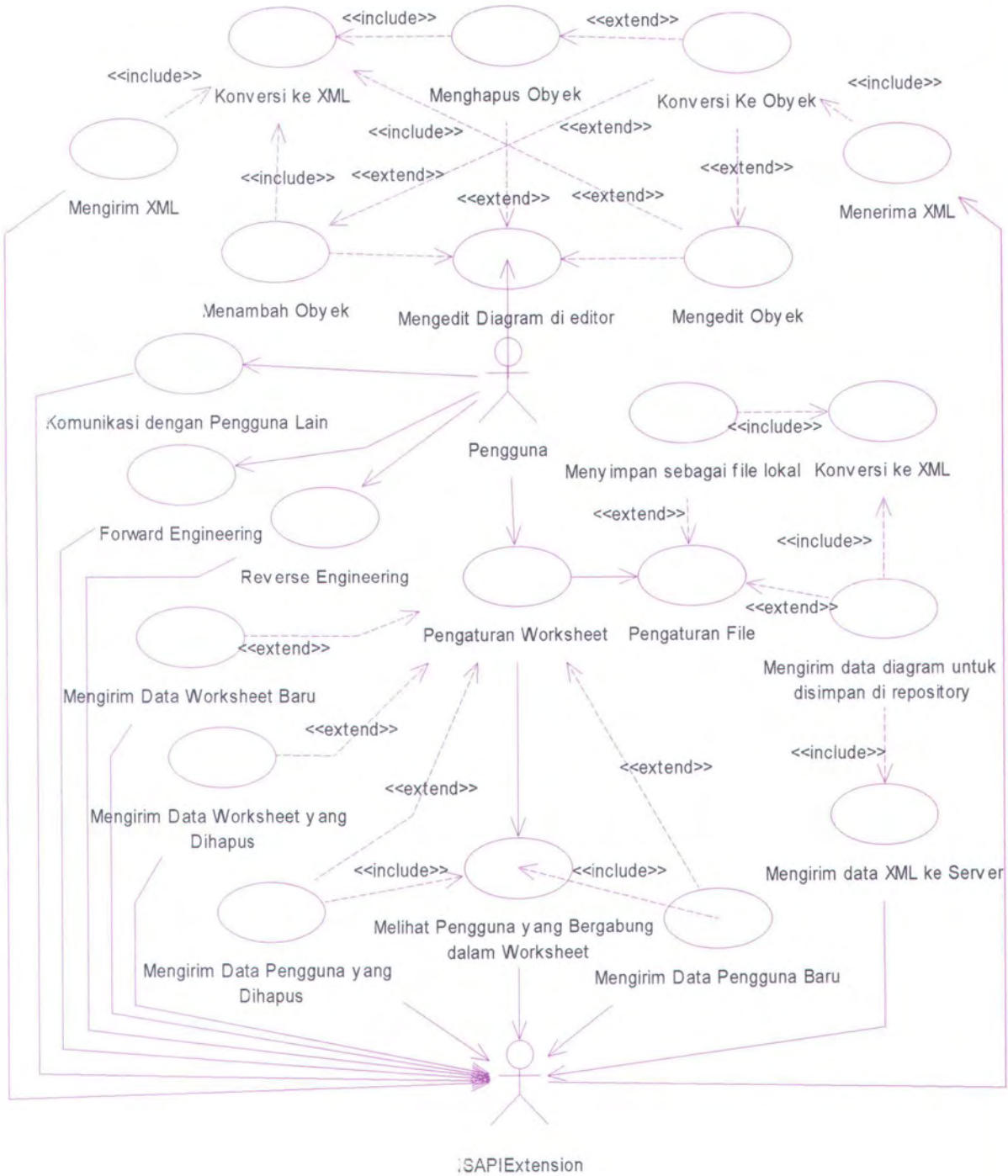
6. **PDMRelation**, merupakan obyek yang digunakan untuk menyimpan data relasi hasil proses *forward* dan *reverse engineering*. Dalam setiap obyek ini terdapat obyek **PDMRelColumns** yang berisi kolom-kolom yang saling berelasi antara dua tabel.



Gambar 4.10 Arsitektur aplikasi Client (a)

Gambar 4.11 Arsitektur aplikasi Client (b)

Proses-proses yang terjadi pada aplikasi client dapat digambarkan lebih detail dalam diagram use case berikut ini:



Gambar 4.12 Detail Proses yang terjadi dalam Aplikasi Client

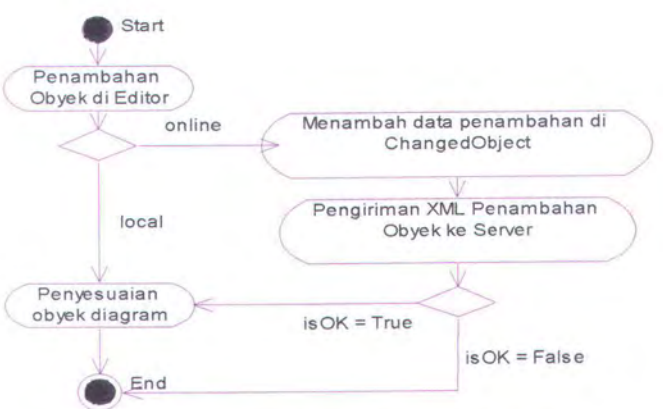
4.3.1 Editing Diagram

Proses-proses yang terjadi pada saat editing diagram adalah sebagai berikut:

1. Penambahan obyek diagram, seperti entitas, relasi atau turunan.
2. Perubahan obyek diagram yang meliputi perubahan bentuk tampilan dari obyek seperti warna, letak, maupun ukurannya. Perubahan baik berupa penambahan, penghapusan maupun perubahan ini juga dapat dilakukan pada properti dari obyek, seperti nama, *code*, kolom dan tipe data dalam entitas.
3. Penghapusan obyek diagram.

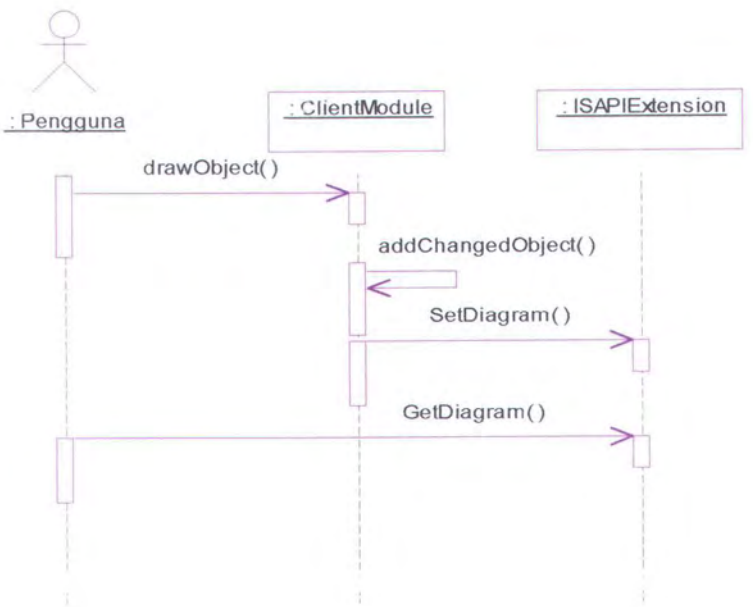
4.3.1.1 Penambahan Obyek

Penambahan obyek diagram dilakukan melalui diagram editor secara langsung. Namun jika pengguna sedang berada dalam mode *multiuser*, maka penambahan obyek diagram juga dapat ditimbulkan oleh penambahan obyek pada server dimana penambahan obyek pada server ini terjadi karena adanya aktifitas penambahan obyek oleh pengguna yang lain.



Gambar 4.13 Activity Diagram Penambahan Obyek

Setiap penambahan obyek diagram yang dilakukan oleh pengguna yang sedang berada dalam mode *multiuser* maka obyek tersebut akan disimpan dalam obyek ***ChangedObject*** seperti yang telah dijelaskan diatas dan dikirimkan ke server untuk disimpan dan diproses apakah penambahan obyek tersebut berhasil atau tidak.

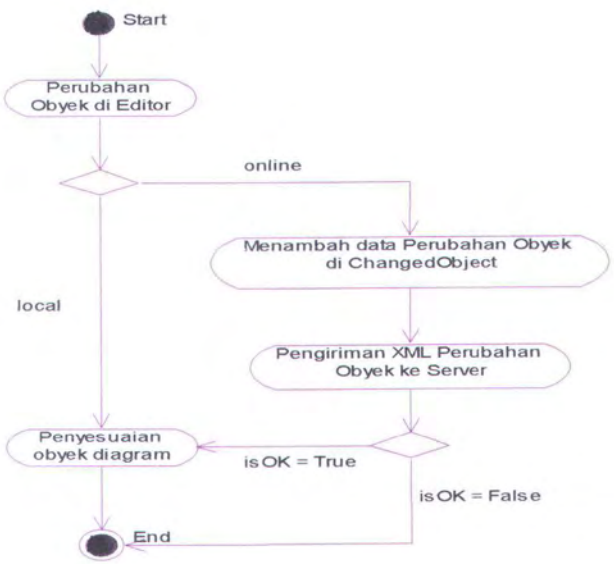


Gambar 4.14 Sequence Diagram Penambahan Obyek

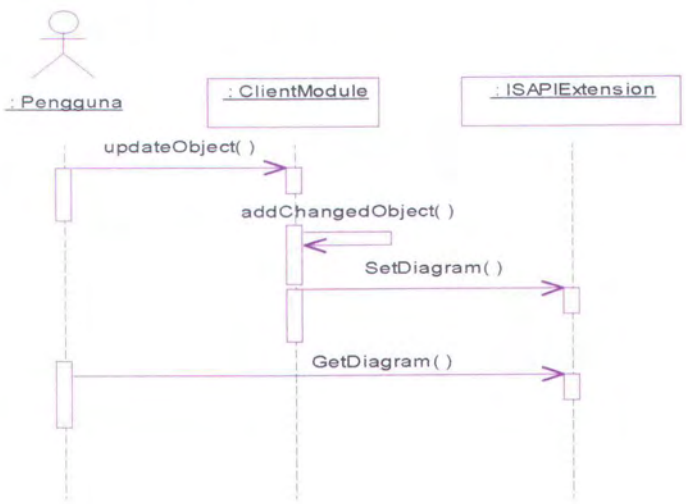
4.3.1.2 Perubahan Obyek

Perubahan obyek diagram dilakukan pada tampilan dari obyek, misalnya warna, letak dan ukuran obyek. Perubahan ini juga dapat dilakukan pada property dari obyek tersebut, seperti nama, code, kolom, tipe data pada obyek entitas atau merubah jenis relasi pada sebuah obyek relasi. Seperti halnya pada proses penambahan obyek, perubahan ini dapat terjadi karena adanya perubahan data

pada server sebagai akibat dari perubahan yang dilakukan oleh pengguna yang lain. Jika pengguna sedang berada dalam mode *multiuser* maka perubahan tersebut akan disimpan dalam obyek ChangedObyek dan dikirim ke server untuk diproses.



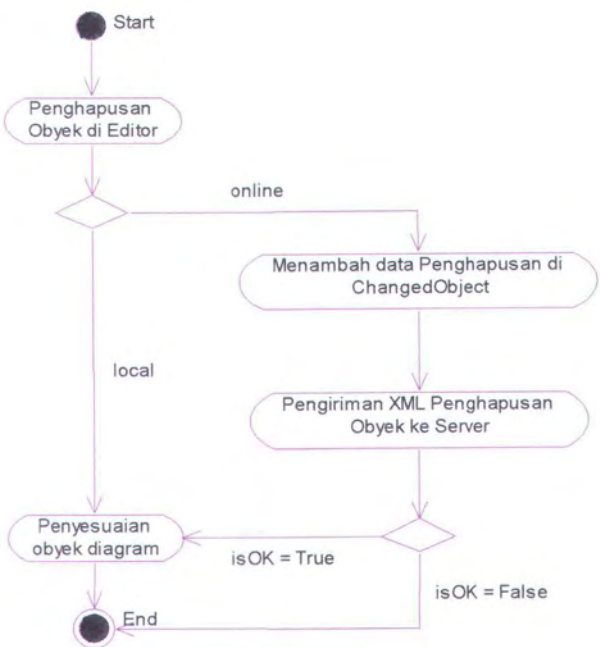
Gambar 4.15 Activity Diagram Perubahan Obyek



Gambar 4.16 Sequence Diagram Perubahan Obyek

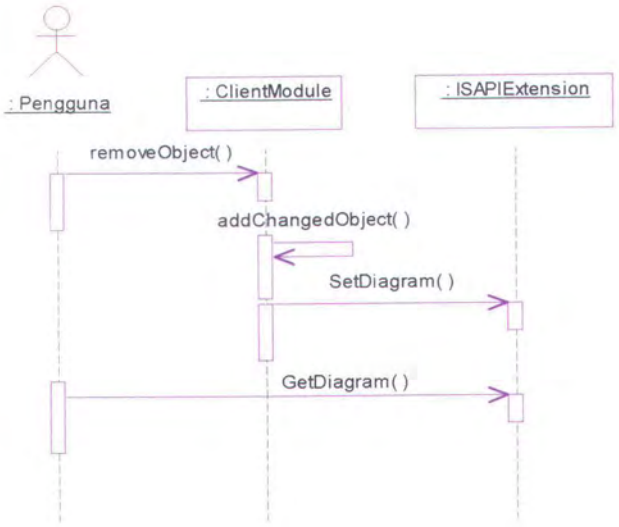
4.3.1.3 Penghapusan Obyek

Proses penghapusan obyek diagram tidak jauh berbeda dengan proses penambahan obyek dimana penghapusan obyek diagram dilakukan pada editor diagram dan penghapusan ini dapat pula terjadi karena adanya penghapusan obyek oleh server.



Gambar 4.17 Activity Diagram Penghapusan Obyek

Alur penghapusan obyek dapat digambarkan sebagai berikut :



Gambar 4.18 Sequence Diagram Penghapusan Obyek

4.3.2 Manajemen Model

Manajemen model berfungsi untuk mengatur bagaimana sebuah diagram dibuat, disimpan, dan dibuka.

4.3.2.1 Pengaturan Worksheet

Dalam mode multi user maka diagram disusun dalam bentuk sebuah worksheet. Sedangkan untuk mode single user, secara eksplisit tidak mempunyai worksheet, tapi editor diagram dapat dianggap sebagai worksheet-nya.

Pada mode multi user, pengguna dapat membuat sebuah worksheet baru dan pengguna yang membuat worksheet akan bertindak sebagai administrator dari worksheet tersebut. Disamping pilihan untuk membuat sebuah worksheet baru, pengguna juga dapat melakukan *join* dengan pengguna yang lain dalam sebuah worksheet yang sudah ada.

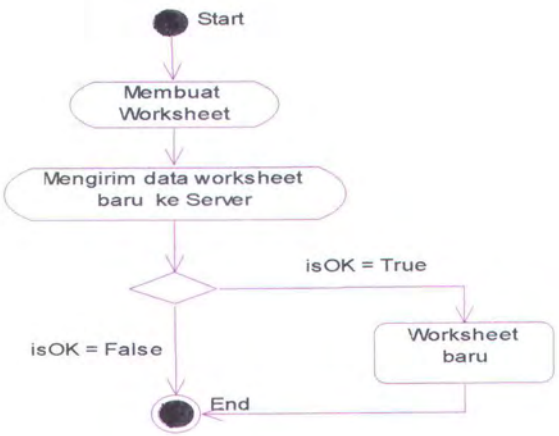
Proses-proses yang terjadi pada pengaturan worksheet ini adalah sebagai berikut:

1. Pembuatan atau penambahan worksheet.
2. Penghapusan worksheet.
3. Penambahan pengguna worksheet.
4. Penghapusan pengguna worksheet.
5. Melihat pengguna yang sedang aktif dalam worksheet.

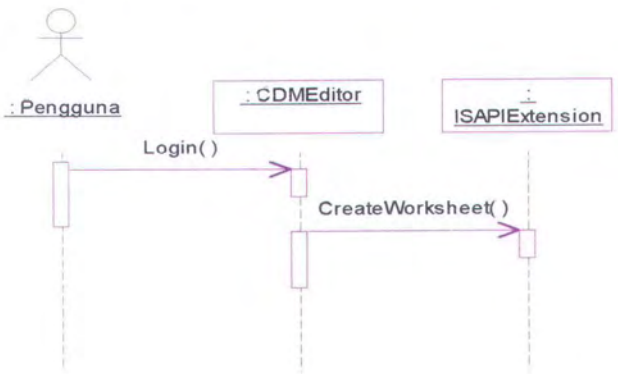
4.3.2.1.1. Penambahan Worksheet

Seperti yang telah dijelaskan diatas bahwa pada mode single user, secara eksplisit pengguna tidak mempunyai worksheet. Jadi yang dimaksud dengan penambahan worksheet disini adalah penambahan worksheet pada mode multi user.

Penambahan worksheet terjadi pada saat pengguna yang berada pada mode multi user login dan menggunakan pilihan untuk membuat worksheet. Worksheet yang dibuat harus mempunyai nama yang *uniqe* yaitu tidak boleh sama dengan worksheet yang sudah ada dan sedang aktif. Pengguna yang memutuskan untuk membuat sebuah worksheet, maka pengguna tersebut akan bertindak sebagai administrator untuk mengatur segala sesuatu yang berhubungan dengan worksheet-nya. Pengguna yang tidak ingin membuat worksheet baru dapat bergabung dalam worksheet lain yang sedang aktif dan dapat memilih worksheet mana yang akan dimasuki dengan password yang telah ditentukan oleh administrator worksheet yang bersangkutan.



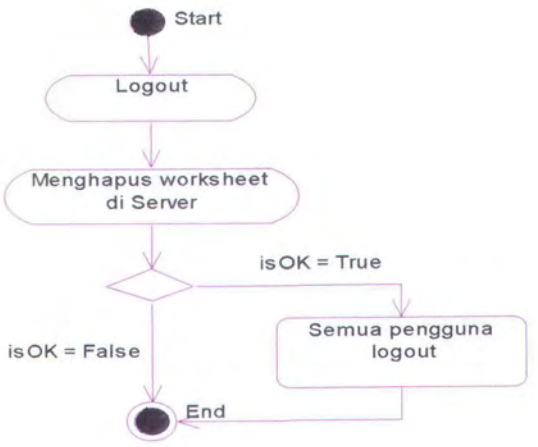
Gambar 4.19 Activity Diagram Penambahan Worksheet



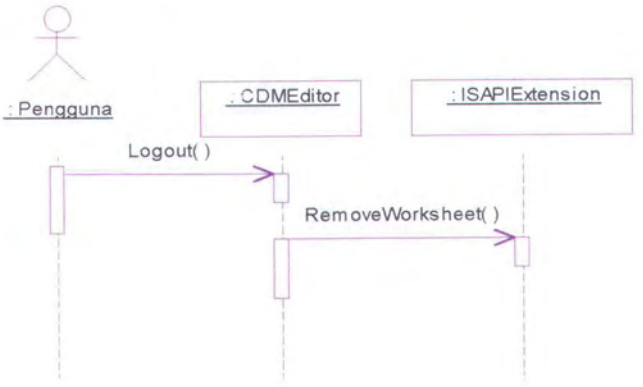
Gambar 4.20 Sequence Diagram Penambahan Worksheet

4.3.2.1.2. Penghapusan Worksheet

Penghapusan worksheet hanya dapat dilakukan oleh administrator atau pengguna yang membuat worksheet tersebut. Sehingga pengguna yang *join* dalam worksheet tidak mempunyai hak untuk menghapus. Penghapusan worksheet terjadi jika administrator worksheet yang bersangkutan *logout*. Penghapusan worksheet akan mengakibatkan seluruh pengguna yang bergabung dalam worksheet tersebut akan logout secara otomatis.



Gambar 4.21 Activity Diagram Penghapusan Worksheet



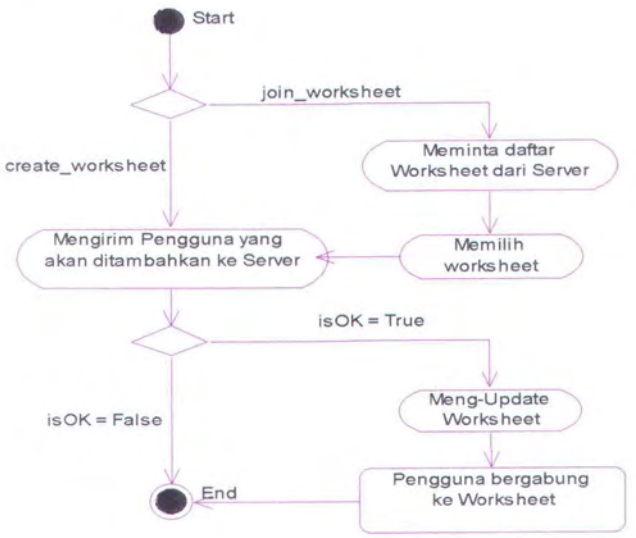
Gambar 4.22 Sequence Diagram Penghapusan Worksheet

4.3.2.1.3. Penambahan Pengguna

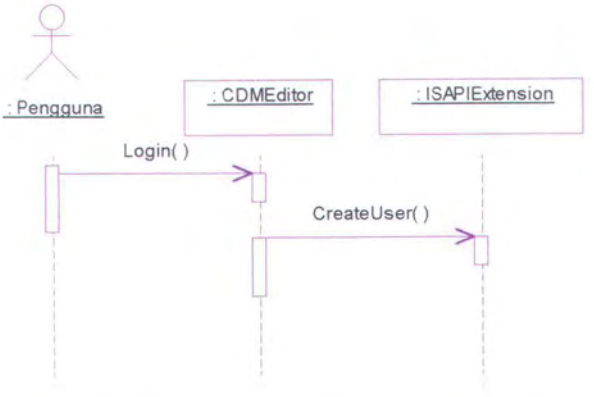
Penambahan pengguna yang dimaksud disini adalah penambahan pengguna untuk bergabung dalam sebuah worksheet. Pengguna yang bertindak sebagai administrator secara otomatis menjadi pengguna dalam worksheet yang dibuat. Hanya pengguna yang terdaftar sebagai user yang dapat ditambahkan dan pengguna tidak dapat login lebih dari sekali dalam worksheet yang sama. Pengguna yang akan bergabung dapat memilih worksheet mana yang akan

dimasuki. Proses penambahan ini terjadi ketika pengguna melakukan login dalam worksheet baik bertindak sebagai pembuat worksheet atau administrator maupun sebagai pengguna yang bergabung dalam worksheet.

Berikut ini adalah diagram yang menggambarkan proses yang terjadi pada saat penambahan pengguna dalam worksheet:



Gambar 4.23 Activity Diagram Penambahan Pengguna



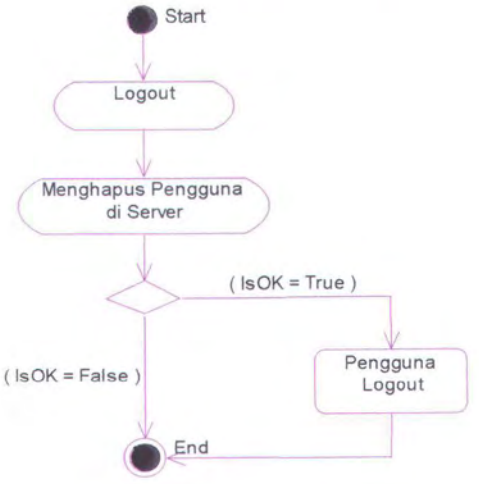
Gambar 4.24 Sequence Diagram Penambahan Pengguna

4.3.2.1.1. Penghapusan Pengguna

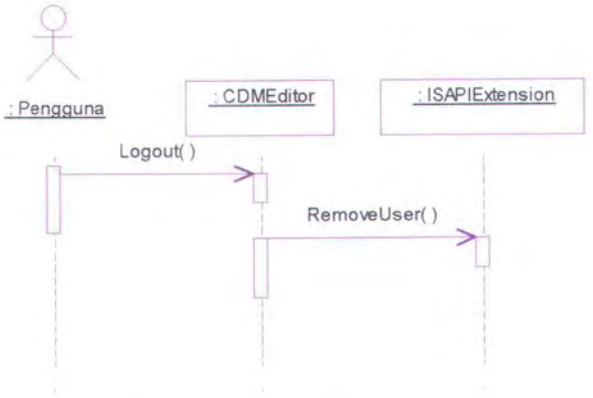
Pengguna yang tergabung dalam sebuah worksheet dapat dihapus oleh administrator atau pembuat worksheet. Penghapusan ini akan mengakibatkan pengguna yang dihapus akan keluar dari worksheet.

Gambar 4.25 Activity Diagram Penghapusan Pengguna

Penghapusan pengguna juga terjadi pada saat pengguna tersebut logout.



Gambar 4.26 Activity Diagram Logout



Gambar 4.27 Sequence Diagram Penghapusan Pengguna

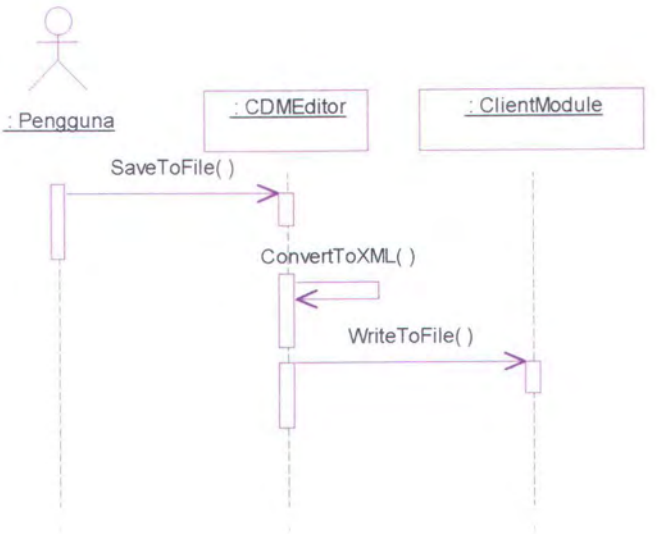
4.3.2.2. Pengaturan File

Diagram yang sudah dibuat dalam sebuah worksheet dapat disimpan dalam file. Penyimpanan ini dapat dilakukan dalam komputer lokal berupa file XML atau dalam database server dengan format yang disesuaikan dengan aplikasi ini. Penyimpanan ini dapat dilakukan oleh semua pengguna. Diagram yang telah tersimpan dapat dibuka kembali dengan catatan apabila pengguna berada dalam mode multi user maka pengguna tersebut akan beralih ke mode single user. Hal ini dimaksudkan untuk menghindari terjadinya *inconsistency* dalam pembuatan diagram dalam worksheet yang sama.

4.3.2.2.1. Penyimpanan Dalam File Lokal

Obyek-obyek yang ada dalam diagram yang akan disimpan dalam file lokal akan dikonversi ke dalam bentuk *string*. Data string yang dihasilkan akan disimpan dalam sebuah file lokal dengan format XML (berekstensi .xml). Struktur XML pada file lokal ini akan dijelaskan pada sub bab yang lain.

Proses penyimpanan diagram ke dalam file lokal digambarkan dalam bentuk diagram berikut ini :

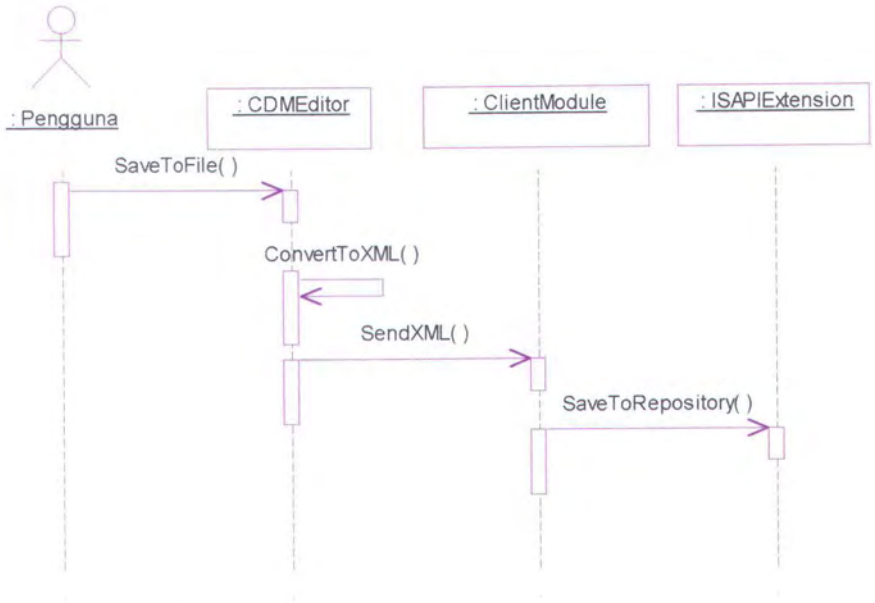


Gambar 4.28 Sequence Diagram Penyimpanan File Lokal

4.3.2.2.1. Penyimpanan Dalam Database

Sebagaimana halnya pada penyimpanan dalam file lokal, penyimpanan dalam database akan mengkonversi obyek-obyek dalam diagram menjadi data *string* dengan format XML. Struktur XML dalam proses penyimpanan dalam database ini sama dengan format pada file lokal dan akan dijelaskan pada bab yang lain. Data XML ini akan dikirim ke server untuk diproses dan disimpan dalam database. Proses yang terjadi pada server akan dijelaskan pada pembahasan terserndiri.

Proses penyimpanan diagram ke dalam database digambarkan dalam bentuk diagram sebagai berikut :



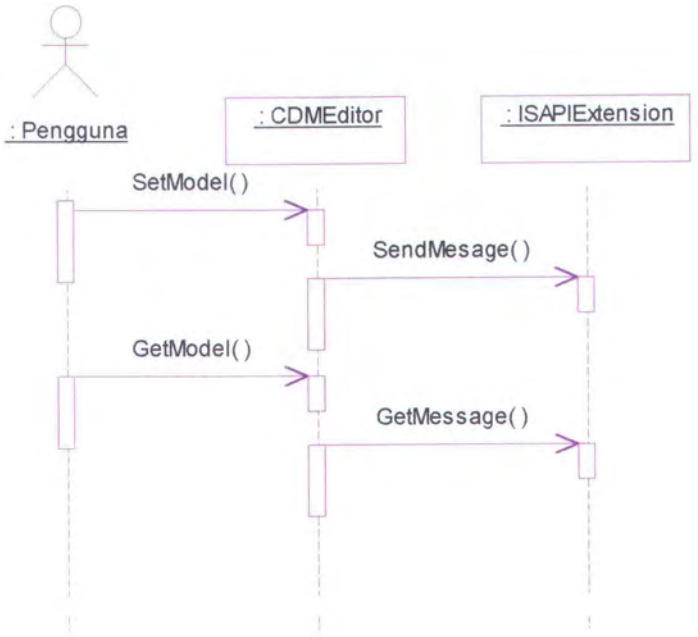
Gambar 4.29 Sequence Diagram Penyimpanan ke Database

4.3.3 Komunikasi Antar Pengguna

Aplikasi *client* dalam tugas akhir ini dilengkapi dengan sebuah fasilitas *chatting* yang digunakan untuk komunikasi antar pengguna dalam worksheet yang sama. Pengguna dapat memilih apakah pesan itu akan disampaikan pada seluruh pengguna atau hanya untuk seorang pengguna tertentu.

Dengan adanya fasilitas ini diharapkan pengguna dapat membuat sebuah forum diskusi antar pengguna dalam pembuatan sebuah diagram tanpa memerlukan mobilitas yang tinggi.

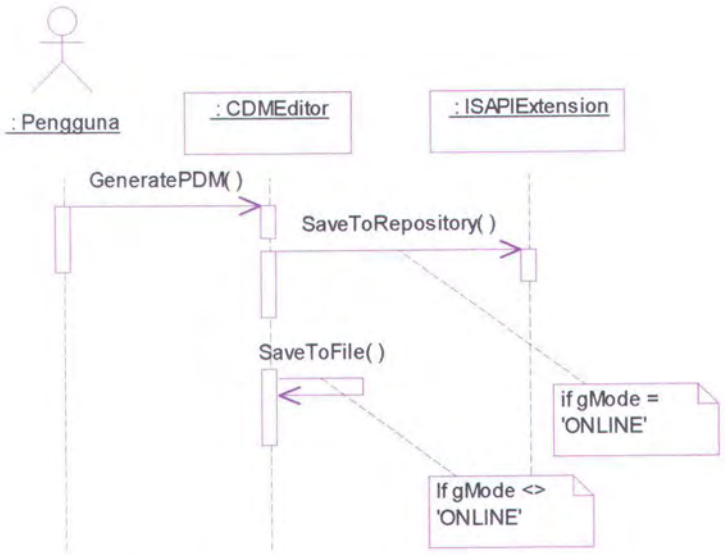
Alur proses yang terjadi pada komunikasi antar pengguna digambarkan seperti diagram dibawah ini :



Gambar 4.30 Sequence Diagram Komunikasi Antar Pengguna

4.3.4 Forward Engineering

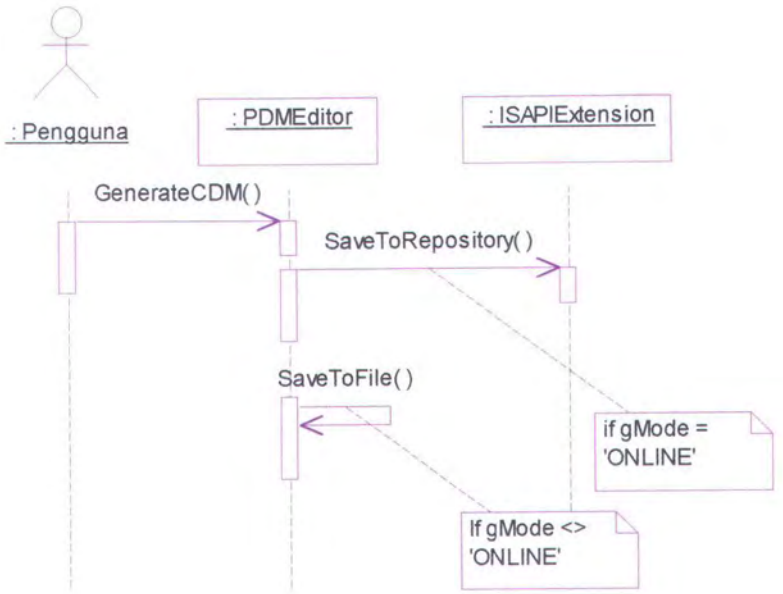
Proses *forward engineering* ini melibatkan obyek PDMTables dan PDMRelations seperti yang sudah dijelaskan pada subbab sebelumnya. Obyek-obyek ini akan dikonversi dengan menyimpan informasi obyek asalnya misalnya obyek tabel yang berasal dari entitas akan menyimpan informasi tentang entitas tersebut. Hasil konversi ini disimpan dalam bentuk XML. Untuk mengetahui hasil dari *forward engineering* ini, file XML hasil konversi dibuka dengan menggunakan PDM Editor. Format XML hasil konversi akan dijelaskan pada subbab yang lain. Jika pengguna berada pada mode single user maka hasil konversi akan disimpan dalam file lokal. Jika pengguna berada pada mode multi user maka hasil konversi akan dikirim ke server untuk disimpan dalam repository.



Gambar 4.31 Sequence Diagram Forward Engineering

4.3.5 Reverse Engineering

Proses *reverse engineering* dilakukan pada aplikasi PDM Editor. Hal ini dilakukan untuk memudahkan pengguna yang melakukan proses ini untuk melakukan perubahan pada diagram model fisik. Proses ini dimulai dengan mengkonversi obyek-obyek yang ada dalam PDM editor kedalam format XML. Konversi obyek-obyek ini memperhatikan informasi obyek model konseptual yang dimiliki. Jika obyek model fisik memiliki informasi obyek model konseptual yang menjadi asalnya makan obyek model fisik tersebut akan dikonversi menjadi obyek model konseptual asal. Jika pengguna berada pada mode single user maka hasil konversi akan disimpan dalam file lokal dan jika pengguna berada pada mode multi user maka hasil konversi dikirim ke server untuk disimpan dalam repository.



Gambar 4.32 Sequence Diagram Reverse Engineering

4.4. Aplikasi Server

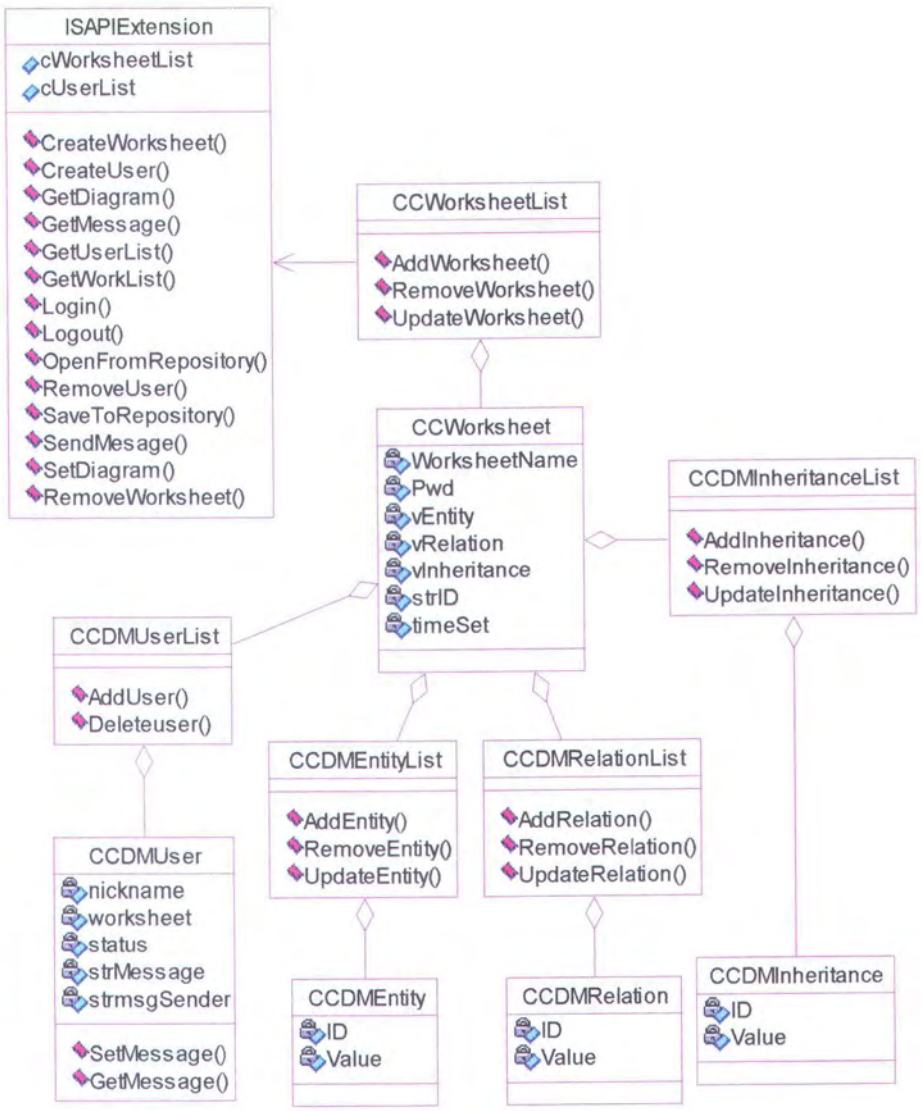
Aplikasi server dibuat dengan menggunakan teknologi ISAPI yaitu ISAPI extension. Karena itu aplikasi server yang dibuat akan memiliki extension DLL dan diletekkan dalam proses IIS.

Sebagaimana aplikasi *client*, aplikasi *server* ini memiliki beberapa obyek yang digunakan untuk menyimpan data. Obyek-obyek tersebut adalah :

1. **CCDMEntity**, merupakan obyek yang menyimpan informasi tentang entitas. Beberapa obyek CCDMEntity berada dalam satu obyek *collection* (kumpulan dari beberapa obyek) yaitu **CDMEntityList**.
2. **CCDMRelation**, merupakan obyek yang menyimpan informasi tentang relasi yang menghubungkan satu atau dua entitas. Beberapa obyek

CCDMRelation berada dalam satu obyek *collection* yaitu **CCDMRelationList**.

3. **CCDMInheritance**, merupakan obyek yang menyimpan informasi tentang relasi yang menghubungkan satu atau beberapa entitas dengan sebuah entitas yang bersifat lebih *general* atau entitas induknya. Beberapa obyek CCDMInehritance berada dalam satu obyek *collection* yaitu **CCDMInheritanceList**.
4. **CCWorksheet**, merupakan obyek yang menyimpan informasi tentang worksheet yang sedang aktif saat itu. Beberapa obyek CCWorksheet berada dalam satu obyek *collection* yaitu **CCWorksheetList**.
5. **CCDMUser**, merupakan obyek yang menyimpan informasi tentang data pengguna yang sedang aktif dan sedang berada dalam mode *multi user*. Beberapa obyek CCDMUser berada dalam satu obyek *collection* yaitu **CCDMUserList**.
6. **CCDMID**, merupakan obyek yang menyimpan data counter ID terakhir dari masing-masing obyek diagram. Beberapa obyek CCDMID berada dalam satu obyek *collection* yaitu **CCDMIDList**.



Gambar 4.33 Arsitektur Aplikasi Server

Secara garis besar aplikasi server ini berfungsi untuk mengatur permintaan yang dikirimkan oleh *client*. Fungsi tersebut dapat di bagi menjadi :

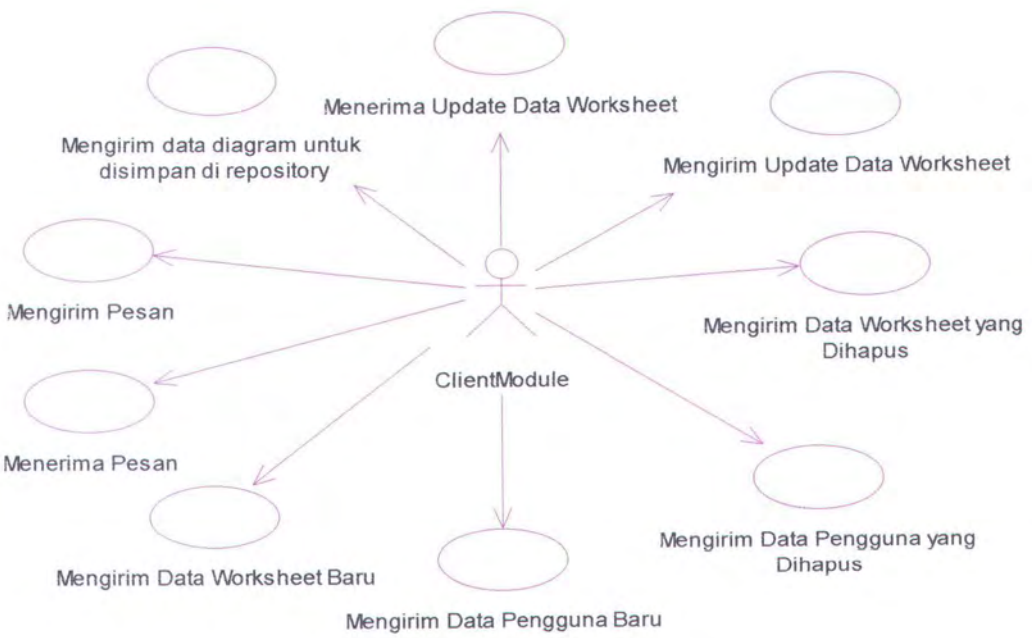
1. Mengatur permintaan *client*.
2. Mengatur penggunaan *repository* dalam database.

3. Mengatur konkurensi data yang dikirimkan oleh *client*.
4. Mengatur komunikasi antar pengguna.

Proses-proses yang terjadi dalam aplikasi server adalah sebagai berikut:

1. Membuat atau menambah dan menghapus worksheet.
2. Menambah dan menghapus pengguna.
3. Menyimpan data diagram dalam repository.
4. Menambah, menghapus dan meng-update data diagram.
5. Menerima dan mengirim pesan yang digunakan dalam komunikasi antar pengguna

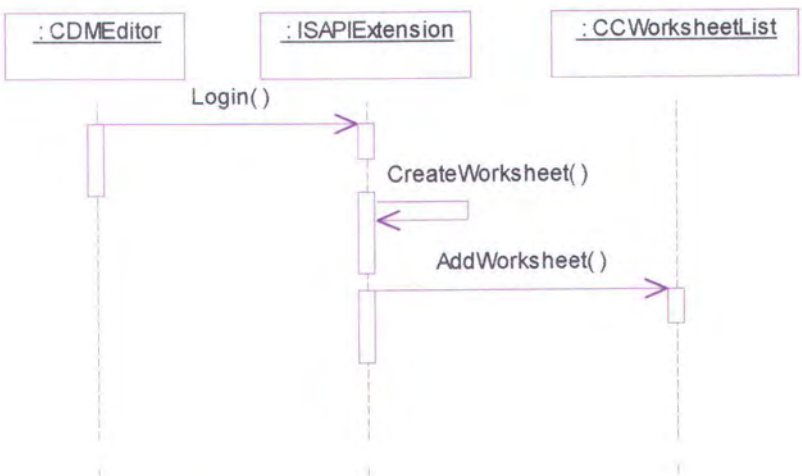
Proses-proses diatas merupakan hasil interaksi atau komunikasi dengan aplikasi client. Interaksi dengan aplikasi cilent dapat digambarkan dalam bentuk diagram berikut ini :



Gambar 4.34 Proses interaksi aplikasi server dan aplikasi client

4.4.1. Menambah Dan Menghapus Worksheet

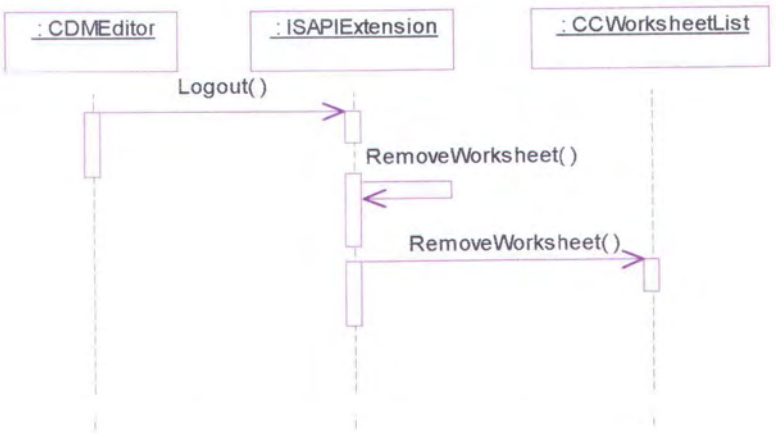
Aplikasi server memiliki *collection* **CCWorksheetList** yang menampung semua worksheet yang sedang aktif pada server. Aplikasi server melakukan proses penambahan worksheet pada CCWorksheetList pada saat server menerima request dari *client* pada saat pengguna login dan membuat worksheet baru pada *client*. Server akan memeriksa apakah worksheet yang diminta untuk dibuat mempunyai nama yang sama dengan worksheet yang sedang aktif pada saat itu. Jika memiliki nama yang sama maka aplikasi server akan mengirimkan informasi bahwa worksheet yang diminta sudah ada.



Gambar 4.35 Sequence Diagram Penambahan Worksheet

Penghapusan worksheet dilakukan oleh server pada saat pengguna yang berstatus sebagai *administrator* logout dari worksheet yang bersangkutan. Penghapusan worksheet ini mengakibatkan semua pengguna yang bergabung dalam worksheet tersebut secara otomatis akan *logout* dari worksheet.

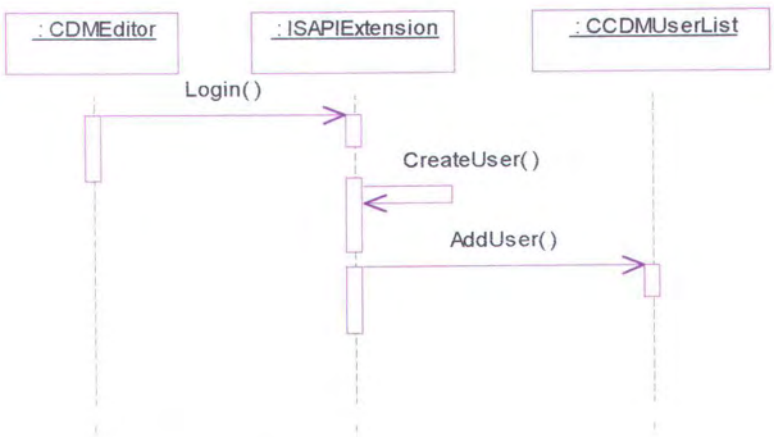
Diagram penghapusan worksheet dalam aplikasi server adalah sebagai berikut :



Gambar 4.36 Sequence Diagram Penghapusan Worksheet

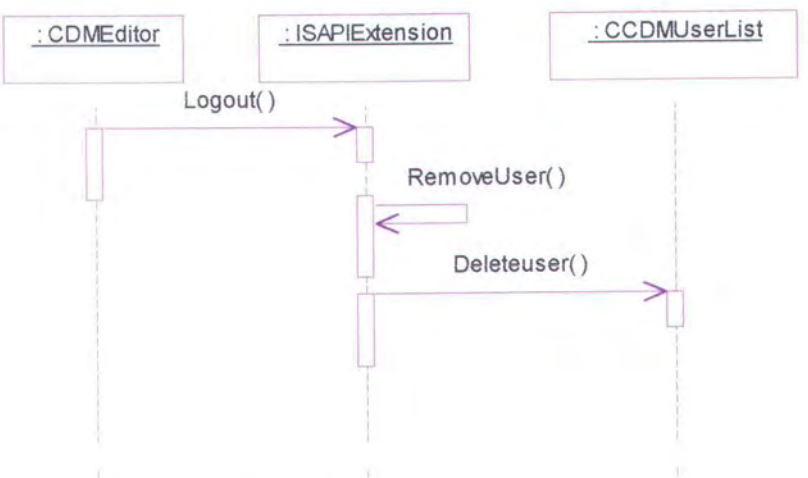
4.4.2. Menambah Dan Menghapus Pengguna

Pengguna yang sedang *online* dengan server akan dimasukkan dalam daftar pengguna yaitu berupa obyek *collection* **CCDMUser**. Penambahan pengguna dilakukan oleh server setelah menerima *request* dari *client* pada saat pengguna login, baik pengguna tersebut membuat worksheet baru maupun *join* terhadap worksheet yang telah ada. Pengguna tidak boleh login lebih dari satu kali dalam worksheet yang sama.



Gambar 4.37 Sequence Diagram Penambahan Pengguna

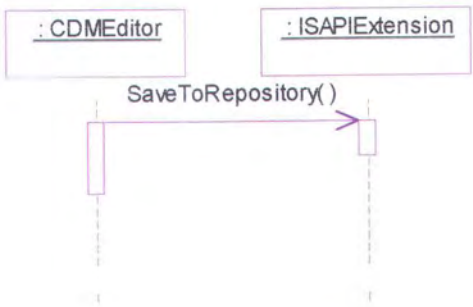
Server melakukan penghapusan pengguna dari *collection* CCDMUser ketika pengguna tersebut logout dari worksheet. Penghapusan pengguna juga dilakukan pada saat pengguna yang berstatus sebagai *administrator* logout dari worksheet. Jika hal ini terjadi maka seluruh pengguna yang tergabung dalam worksheet tersebut akan terhapus.



Gambar 4.38 Sequence Diagram Penghapusan Pengguna

4.4.3. Menyimpan Data Diagram Dalam Repository

Penyimpanan yang dilakukan oleh server adalah penyimpanan data diagram dalam *database* repository. Pada proses ini server mengubah obyek-obyek diagram seperti entitas, relasi, dan turunan kedalam bentuk XML. Masiing-masing obyek diagram mempunyai format XML tersendiri. Data dalam bentuk XML tersebut kemudian disimpan dalam database. Masing-masing pengguna yang berada dalam mode *multi user* dapat menyimpan diagramnya dalam database.

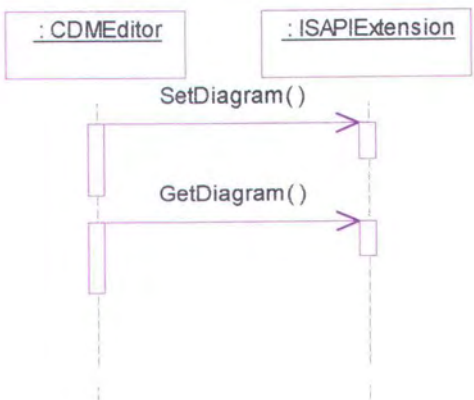


Gambar 4.39 Sequence Diagram Penyimpanan Data Diagram dalam repository

4.4.4. Menambah, Menghapus Dan Meng-update Data Diagram

Aplikasi server memiliki beberapa obyek untuk menyimpan data diagram. Obyek-obyek tersebut berupa *collection* atau kumpulan dari obyek diagram. Request penambahan, penghapusan maupun update data diagram yang dikirimkan oleh pengguna akan mempengaruhi data diagram dari pengguna yang lain yang sedang memakai worksheet yang sama.

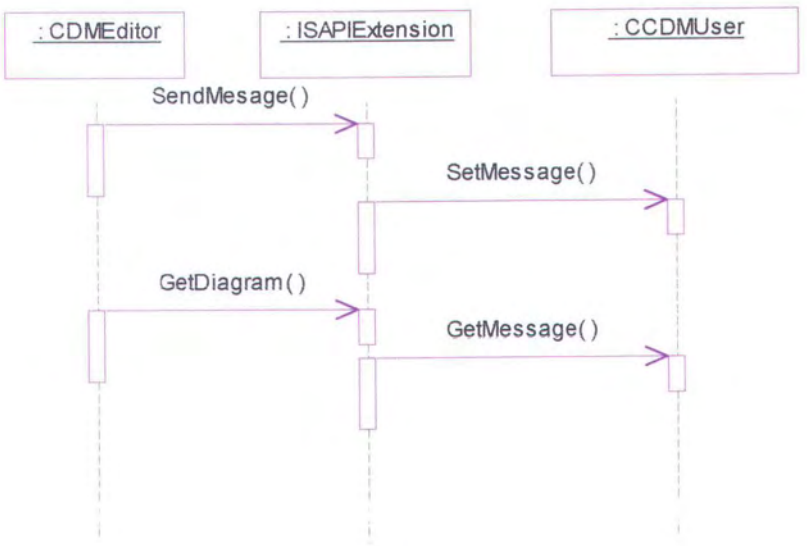
Proses diatas digambarkan seperti diagram berikut ini :



Gambar 4.40 Sequence Diagram Update Data Diagram

4.4.5. Menerima Dan Mengirim Pesan

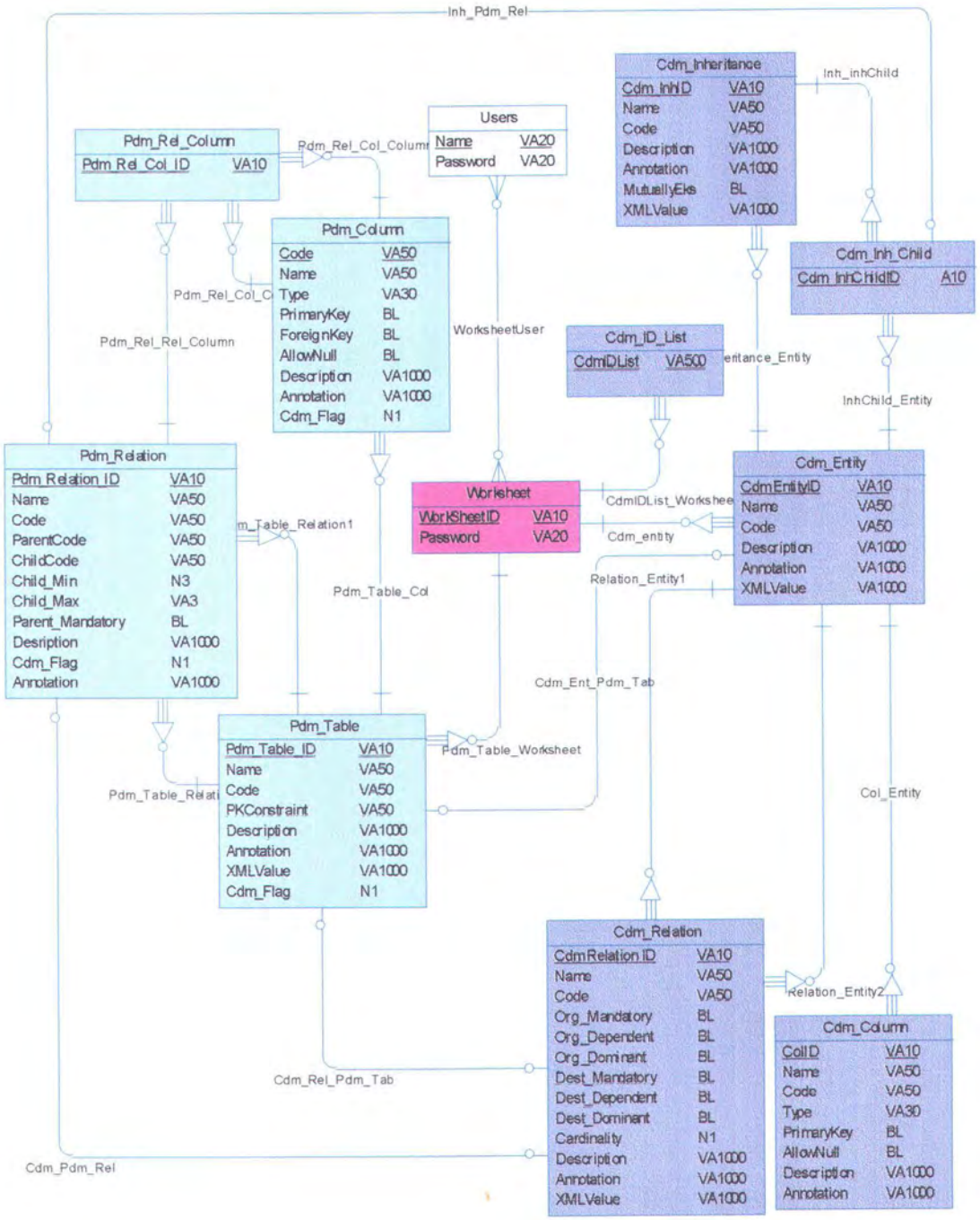
Pesan yang dikirimkan oleh pengguna akan dikirimkan oleh *client* ke *server* melalui *form* dialog komunikasi. Pengiriman pesan yang terlah diterima oleh server dilakukan bersamaan dengan pengiriman update data obyek model ke *client*.



Gambar 4.41 Sequence Diagram Komunikasi antar Pengguna

4.5. Struktur Repository

Rancangan struktur repository dalam database disusun seperti model diagram konseptual berikut ini :



Gambar 4.42 Struktur model konsep Repository

Model konseptual diatas kemudian dikonversi kedalam bentuk model fisik menghasilkan model yang dapat dilihat pada lampiran A dengan tabel-tabel sebagai berikut:

1. Tabel **Users**, merupakan tabel untuk menympan data pengguna. Setiap pengguna yang akan menggunakan aplikasi ini harus memiliki *login* yang disimpan dalam kolom Name dan password yang disimpan dalam kolom Password.
2. Tabel **Worksheet**, merupakan tabel untuk menyimpan data worksheet yang digunakan oleh masing-masing pengguna.
3. Tabel **Cdm_Entity**, merupakan tabel yang digunakan untuk menyimpan data obyek-obyek entitas.
4. Tabel **Cdm_Column**, merupakan tabel yang digunakan untuk menyimpan data kolom dalam obyek entitas.
5. Tabel **Cdm_Relation**, digunakan untuk menyimpan obyek-obyek yang berupa relasi.
6. Tabel **Cdm_Inheritance**, digunakan untuk menyimpan obyek-obyek turunan.
7. Tabel **Cdm_Inh_Child**, digunakan untuk menyimpan data obyek entitas yang menjadi turunan dari obyek dalam tabel Cdm_Inheritance.
8. Tabel **Cdm_ID_List**, pada saat pengguna melakukan penyimpanan dalam repository maka *client* akan mengirimkan data *counter* ID terakhir dari obyek-obyek dalam worksheet. Data *counter* ID ini akan disimpan dalam tabel Cdm_ID_List.
9. Tabel **Pdm_Table**, merupakan tabel yang digunakan untuk menyimpan data obyek tabel baik yang berasal dari proses forward

engineering maupun dari hasil pemodelan dengan menggunakan PDM Editor.

10. Tabel **Pdm_Column**, merupakan tabel yang digunakan untuk menyimpan data kolom dari obyek tabel yang ada pada tabel Pdm_Table.
11. Tabel **Pdm_Relation**, merupakan tabel yang digunakan untuk menyimpan data relasi model fisik baik yang berasal dari proses forward engineering maupun dari proses pemodelan fisik dengan menggunaan PDM Editor.
12. Table **Pdm_Rel_Column**, merupakan tabel yang digunakan untuk menyimpan data kolom yang merupakan foreign key sesuai dengan relasi model fisik yang ada pada tabel Pdm_Relation.

4.6. Komunikasi Client Server

4.6.1. Polling Data

Metode yang digunakan dalam komunikasi client server dalam aplikasi ini adalah metode *poll*. Dalam metode ini client akan mengirimkan request secara periodik ke *server* dan *server* segera mengirimkan sebuah nilai kembalian yang akan digunakan oleh *client*. Frekuensi pengiriman request dalam aplikasi ini di set selama 3 detik. Format data yang digunakan dalam komunikasi client server disini adalah XML.

4.6.1. Struktur XML Aplikasi

Ukuran file XML yang digunakan dalam komunikasi *client server* mempengaruhi akselerasi dari aplikasi. Semakin besar ukuran file XML yang dikirim maka akan menambah beban kinerja sistem.

4.6.1.1. Struktur XML Diagram

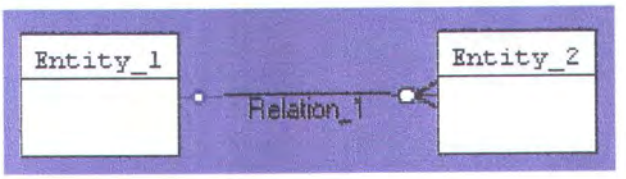
Obyek-obyek dalam diagram (seperti entitas, relasi dan turunan) dikonversi kedalam bentuk XML sebelum disimpan kedalam *file* lokal atau ke *repository*. Format masing-masing obyek adalah sebagai berikut :

1. **Entitas**

```
<entitylist>
    <ent>
        <property>
            <name> nama_entity </name>
            <code> code_entity </code>
            <id> id_entity </id>
            <gen> flag_generate </gen>
            <desc> description_entity </desc>
            <annot> annotation_entity </annot>
            <interface>
                  <top> posisi_top_dari_entity </top>
                  <left> posisi_left_dari_entity </left>
                  <height> tinggi_entity </height>
                  <width> panjang_entity </width>
                  <backcolor> warna_entity </backcolor>
                  <forecolor> warna_border_entity </forecolor>
                  <textcolor> warna_tulisan_entity </textcolor>
            </interface>
            <columns>
                <column>
                    <id> id_kolom </id>
                    <name> nama_kolom </name>
                    <code> code_kolom </code>
                    <datatype> tipe_data_kolom </datatype>
                    <primarykey> flag_primary_kolom </primarykey>
                    <allownull> flag_null_kolom </allownull>
                </column>
            </columns>
        </property>
    </ent>
</entitylist>
```

2. Relasi

Terdapat beberapa komponen yang membangun sebuah relasi dimana dalam aplikasi *client* direpresentasikan dalam beberapa node. Berikut ini adalah gambaran sebuah relasi:



Pada Relation_1 terdapat 4 *node* dan 1 *link*, 2 *node* melekat pada Entity_1 dan 2 *node* yang lain melekat pada Entity_2. 2 *node* tersebut adalah pertama *node* untuk menampilkan gambar jenis relasi atau yang dalam aplikasi ini disebut dengan *cardinality* (seperti one to one, one to many dan many to many) dan yang kedua adalah *node* untuk menghubungkan *node* pertama dengan *link*.

Dalam struktur script dibawah ini terdapat beberapa istilah, diantaranya :

1. **Node_arrow_org**, merupakan node yang digunakan untuk menampilkan gambar jenis relasi atau *cardinality* yang terletak pada *node original* atau *node* asal (yang dimaksud asal disini adalah node pertama pada waktu *mouse* di *click* untuk membuat relasi atau turunan).

2. **Node_arrow_dest**, merupakan node sama dengan Node_arrow_org tapi node ini berada pada *node destination* atau *node* tujuan (yang

dimaksud dengan *node* tujuan disini adalah *node* yang menjadi tujuan pada saat *mouse* di *drag* untuk membuat relasi atau turunan).

3. **Node_arrow_pos_org**, merupakan node yang menghubungkan Node_arrow_org dengan *link*.

4. **Node_arrow_pos_dest**, merupakan node yang menghubungkan Node_arrow_dest dengan *link*.

```
<relationlist>
    <rel>
        <property>
            <name> nama_relasi </name>
            <code> code_relasi </code>
            <id> id_relasi </id>
            <desc> description_relasi </desc>
            <annot> annotation_relasi </annot>
            <cardinal> kode_cardinality </cardinal>
            <naobid> id_node_arrow_org_base</naobid>
            <naodep> flag_dependency_node_arrow_org</naodep>
            <naoman> flag_mandatory_node_arrow_org </naoman>
            <nadbid> id_node_arrow_dest_base </nadbid>
            <naddep> flag_dependency_node_arrow_dest </naddep>
            <nadman> flag_mandatory_node_arrow_dest </nadman>
            <interface>
              <naoid> id_node_arrow_org</naoid>
              <naodir> kode_arah_node_arrow_org </naodir>
              <naoija> index_jenis_gambar_node_arrow_org</naoija>
              <naoja> jenis_gambar_node_arrow_org </naoja>
              <naoleft> posisi_left_node_arrow_org </naoleft>
              <naotop> posisi_top_node_arrow_org</naotop>
              <nadid> id_node_arrow_dest </nadid>
              <naddir> kode_arah_node_arrow_dest </naddir>
              <nadija> index_jenis_gambar_node_arrow_dest </nadija>
              <nadja> jenis_gambar_node_arrow_dest </nadja>
              <nadleft> posisi_left_node_arrow_dest </nadleft>
              <nadtop> posisi_top_node_arrow_dest </nadtop>
              <npoid> id_node_arrow_pos_org </npoid>
              <npdid> id_node_arrow_pos_dest  </npdid>
            </interface>
        </property>
    </rel>
</relationlist>
```

3. Turunan

```
<inheritancelist>
    <inh>
        <property>
            <name> nama_inheritance </name>
```

```
            <code> kode_inheritance </code>
            <id> id_inheritance </id>
            <desc> description_inheritance </desc>
            <annot> annotation_inheritance </annot>
            <mutually> flag_mutually </mutually>
            <parentid> id_parent </parentid>
            <interface>
            <left> posisi_left_inheritance </left>
            <top> posisi_top_inheritance </top>
            </interface>
            <childs>
                <child>
                    <id> id_link_child </id>
                    <childid> id_entity_child </childid>
                    <parentid> id_entity_parent </parentid>
                </child>
            </childs>
        </property>
    </inh>
</inheritancelist>
```

4. IDObject

Script IDObject berisi tentang informasi *counter* ID terakhir dari entitas, relasi, dan turunan.

```
<idobj>
      <property>
            <inhid> counter_id_inheritance </inhid>
            <entid> counter_id_entity </entid>
            <relid> counter_id_relation </relid>
      </property>
</idobj>
```

4.6.1.2. Struktur XML Update Diagram

Pada saat pengguna melakukan suatu aktifitas atau *action* dalam aplikasi *client*, maka aplikasi *client* akan menyimpan *action* tadi dalam sebuah obyek yang telah dijelaskan pada subbab sebelumnya yaitu ChangedObject. Selanjutnya *action* tersebut akan dikonversi kedalam bentuk XML. *Action* yang di lakukan pengguna terdiri dari :

1. **Add**, *action* ini menunjukkan adanya proses penambahan obyek.

Berikut ini adalah contoh script yang dibuat pada saat menambahkan sebuah entitas.

```
<add>
	<ent>
		<id>1</id>
		<property>
			<name>Entity_1</name>
			<code>ENTITY_1</code>
			<id>1</id>
			<gen>True</gen>
			<desc></desc>
			<annot></annot>
			<inteface>
				<top>1725</top>
				<left>4965</left>
				<height>800</height>
				<width>900</width>
				<backcolor>16777215</backcolor>
				<forecolor>16711680</forecolor>
				<textcolor>0</textcolor>
			</inteface>
			<columns>
			</columns>
		</property>
	</ent>
	<idobj>
		<property>
			<inhid>0</inhid>
			<entid>1</entid>
			<relid>0</relid>
		</property>
	</idobj>
</add>
```

Setiap penambahan obyek akan selalu diikuti oleh penambahan counter id yang diikutsertakan dalam script.

2. **Delete**, merupakan *action* yang menunjukkan bahwa terjadi penghapusan obyek pada diagram. Contoh script untuk *action* ini adalah sebagai berikut:

```
<delete>
	<ent>
		<id>1</id>
		<property></property>
	</ent>
</delete>
```

Penghapusan obyek hanya memerlukan informasi ID. Sedangkan informasi tentang property tidaklah diperlukan dalam *action* ini.

3. **Update**, *action* ini menunjukkan adanya proses *update* data obyek. Contoh script *action* update adalah sebagai berikut:

```
<update>
        <ent>
                <id>2</id>
                <property>
                        <name>Entity_2</name>
                        <code>ENTITY_2</code>
                        <id>2</id>
                        <gen>True</gen>
                        <desc></desc>
                        <annot></annot>
                        <inteface>
                                <top>2895</top>
                                <left>4470</left>
                                <height>825</height>
                                <width>1260</width>
                                <backcolor>16777215</backcolor>
                                <forecolor>16711680</forecolor>
                                <textcolor>0</textcolor>
                        </inteface>
                        <columns>
                        </columns>
                </property>
        </ent>
</update>
```

4.6.1.3. Struktur XML Hasil Forward Engineering

```
<file>
   <tablelist>
        <table>
             → properti table
             <columnlist>
                <column>
                   → properti column
                   <collists>
                      <collist>
                         → properti collist
                      </collist>
                   </collists>
                </column>
             </columnlist>
        </table>
   </tablelist>
   <rellist>
        <relation>
```

```
                → properti relation
                <relcolumns>
                    <relcolumn>
                        → properti relcolumn
                    </relcolumn>
                <relcolumns>
            </relation>
        </rellist>
    </file>
```

Tanda → ***properti*** menunjukkan isi dari properti obyeknya yang sudah dikonversi menjadi *string*.

4.6.1.4. Struktur XML Hasil Reverse Engineering

Struktur XML yang di hasilkan pada saat proses penyimpanan sebagai file lokal sama dengan struktur XML hasil proses penyimpanan seperti yang telah dijelaskan pada subbab sebelumnya.

BAB V

IPLEMENTASI PERANGKAT LUNAK

BAB V

IMPLEMENTASI PERANGKAT LUNAK

5.1. Pembuatan Aplikasi Client

Aplikasi *client* yang dibangun dalam tugas akhir ini adalah sebuah komponen ActiveX Control yang digunakan dalam browser Internet Explorer. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman Visual Basic 6.0. Dalam pembuatan komponen ini digunakan sebuah komponen bantu yaitu Addflow. Addflow digunakan untuk memberikan kemudahan dalam penggambaran obyek-obyek diagram seperti entitas,relasi dan turunan.

Berikut ini adalah penjelasan tentang program atau *code* yang dibuat untuk membangun aplikasi *client*. Penjelasan ini akan dilakukan secara singkat dan hanya mengambil point-point penting dalam aplikasi dengan menunjukkan penulisan *code* secara VB-like karena untuk menjelaskan semua *code* akan memakan waktu dan penjelasan yang sangat panjang.

5.1.1. Variabel Global

1. eEntities, merupakan koleksi dari obyek CDMEntity.
2. rRelations, merupakan koleksi dari obyek CDMRelation.
3. iInheritances, merupakan koleksi dari obyek CDMInheritance.
4. iInhChilds, merupakan koleksi dari obyek CDMInhChild.
5. chgObjCol, merupakan koleksi dari obyek ChangedObject.
6. EntityID, berisi counter ID CDMEntity.

7. RelationID, berisi counter ID CDMRelation.
8. NodeArrowOrgID, berisi counter ID Node Arrow Org.
9. NodeArrowDestID, berisi counter ID Node Arrow Dest.
10. NodeArrowPosOrgID, berisi counter ID Node Arrow Pos Org.
11. NodeArrowPosDestID, berisi counter ID Node Arrow Pos Dest.
12. counterNodeInherit, berisi counter ID CDMInheritance.
13. Conn, merupakan flag apakah pengguna berada pada mode *single user* dengan nilai "LOCAL" atau mode *multi user* dengan nilai "ONLINE".
14. StatusDiagram, merupakan flag apakah pengguna sudah login atau logout.
15. userName, berisi nama pengguna dalam worksheet.
16. userPwd, berisi password pengguna dalam worksheet.
17. workName, berisi nama worksheet yang sedang digunakan.
18. workPwd, berisi password worksheet yang sedang digunakan.
19. strSchema, berisi nama schema database yang digunakan.
20. strPwdSchema, berisi password schema database.
21. strService, berisi nama service untuk database.
22. strServer, berisi alamat *url* aplikasi server.

5.1.2. Fungsi Dan Prosedur Yang Menangani Komunikasi Data.

```
'========== Fungsi untuk mengirim request ke server ============
Public Function SendXML(url As String, xmlData As String, bAsync
   As Boolean) As String
   Dim mXMLReq As New XMLHTTP30
   Dim X As String
   On Error GoTo Error1

   mXMLReq.Open "POST", url, bAsync
   mXMLReq.send xmlData
```

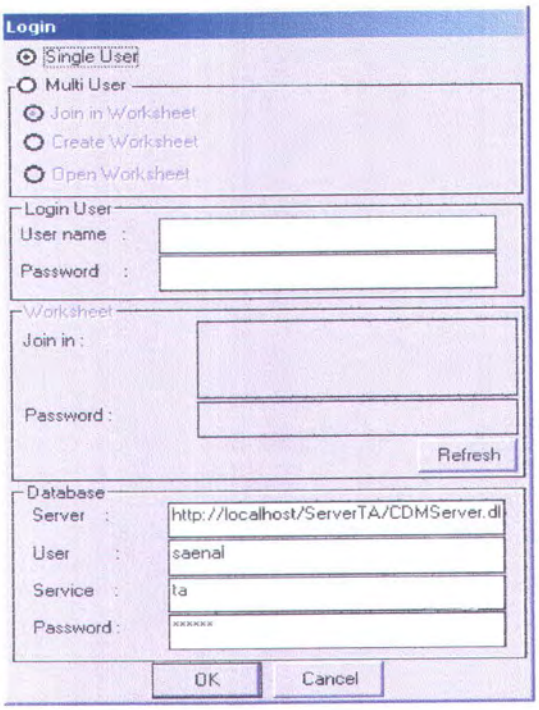
```
    SendXML = mXMLReq.responseText
    Exit Function
Error1:
    SendXML = Err.Number & " " & Err.Description
End Function
```

5.1.3. Fungsi Dan Prosedur Yang Berhubungan Dengan Worksheet.

```
'============== Fungsi untuk login ke worksheet ===============
Function Login() As String
    Dim Str As String
    param = strSchema & "&" & strPwdSchema & "&" & strService &
            "&" & userName & "&" & userPwd
    Str = SendXML(strServer & "?Login", param, False)
    Login = Str
End Function

'============== Prosedur untuk logout dari worksheet =============
Sub Logout()
    Dim Str As String
    param = strSchema & "&" & strPwdSchema & "&" & strService &
            "&" & userName & "&" & userPwd
    Str = SendXML(strServer & "?Login", param, False)
End Sub

'========= Fungsi untuk membuat user baru dalam worksheet ========
Function CreateUser() As String
    Dim Str As String
    Str = SendXML(strServer & "?CreateUser", workName & "&" &
          workPwd & "&" & userName, False)
    CreateUser = Str
End Function

'========= Fungsi untuk membuat worksheet ========
Function CreateWorksheet()As String
    Dim Str As String
    Str = SendXML(strServer & "?CreateWorksheet", workName & "&" &
          workPwd & "&" & userName, False)
    CreateWorksheet = Str
End Function

'========= Fungsi untuk mengambil daftar worksheet ========
Function GetWorksheetList() As String
    Dim Str As String
    Str = SendXML(strServer & "?GetWorkList", "", False)
    GetWorksheetList = Str
End Function

'========= Fungsi untuk mengambil daftar user ========
Function GetUserList() As String
    Dim Str As String
    Str = SendXML(strServer & "?GetUserList", workName, False)
    GetUserList = Str
End Function

'========= Fungsi untuk mengirim data ke server ========
Function SetModel() As String
    Dim Str As String
    Str = SendXML(strServer & "?SetDiagram", cdmWorksheet & "&" &
          userName & "&" & strXML, False)
```

```
    SetModel = Str
End Function

'========= Fungsi untuk mengambil data dari server ========
Function GetModel() As String
    Dim Str As String
    Str = SendXML(strServer & "?GetDiagram", cdmWorksheet & "&" &
          userName & "&" & "ok", False)
    GetModel = Str
End Function

'========= Fungsi untuk menghapus user dari worksheet online =====
Function RemoveUser() As String
    Dim Str As String
    Str = SendXML(strServer & "?RemoveUser", cdmWorksheet & "&" &
          userName, False)
    RemoveUser = Str
End Function
```

Pengguna yang akan menggunakan aplikasi ini harus mengisi form seperti login berikut ini:



Gambar 5.1 Form Login

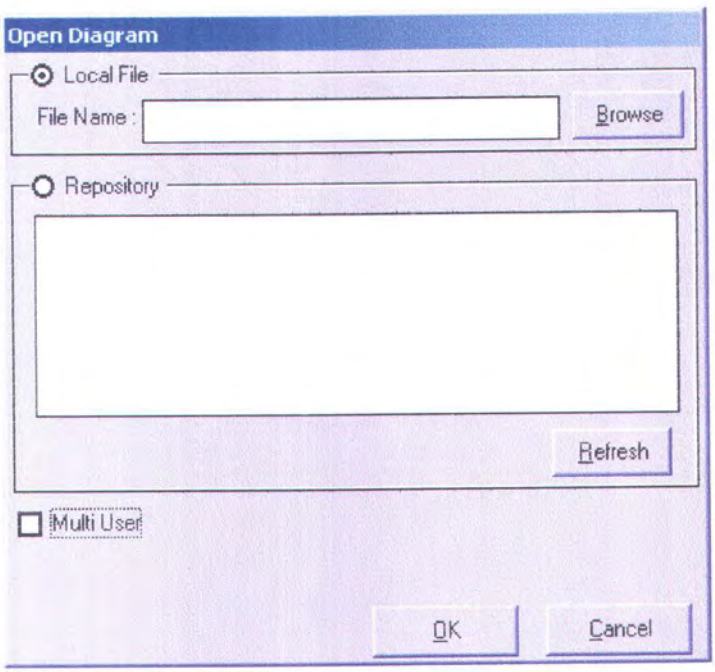
Field Server pada Database merupakan lokasi aplikasi server, skema database repository diisi pada fiel User dengan nama service dan passwordnya.

Form ini kemudian akan menjalankan fungsi Login() seperti yang telah ditulis diatas.

5.1.4. Fungsi Yang Berhubungan Dengan Penyimpanan.

```
'========= Fungsi untuk menyimpan data ke repository =====
Function SaveToRepository() As String
    Dim Str As String
    Str = SendXML(strServer & "?SaveToRepository", strService &
          "&" & workName & "&" & userName & "&" & strXML, False)
    SaveToRepository = Str
End Function

'========= Fungsi untuk mengambil data dari repository =====
Function OpenToRepository() As String
    Dim Str As String
    Str = SendXML(strServer & "?OpenFromRepository", strService &
          "&" & workName & "&" & userName, False)
    OpenToRepository = Str
End Function
```

Pengguna yang telah login kedalam aplikasi dapat menyimpan atau membuka file baik lokal maupun di repository. Pengguna membuka file yang telah tersimpan dengan menggunakan form:



Gambar 5.2 Form Membuka file

5.1.5. Fungsi Dan Prosedur Yang Berhubungan dengan Data XML.

```
'==== Fungsi untuk konversi data obyek ke dlm bentuk XML =====
Function GetScriptObject() As String
    Dim i As Integer
    Dim strEnt As String, strRel As String, strInh As String,
        strXML As String, strID As String
    Dim myEntity As CDMEntity
    Dim myRelation As CDMRelation
    Dim myInheritance As CDMInheritance

    strXML = "<?xml version=" & """1.0""" & "?>" & vbCrLf
    strXML = strXML & "<object>" & vbCrLf

    strEnt = "<entitylist>" & vbCrLf
    For i = 1 To eEntities.Count
        Set myEntity = eEntities.Item(i)

        strEnt = strEnt & "<ent>" & vbCrLf
        strEnt = strEnt & "<property>" &
                  GetPropertyEntity(myEntity) & "</property>" &
                  vbCrLf
        strEnt = strEnt & "</ent>" & vbCrLf
    Next i
    strEnt = strEnt & "</entitylist>" & vbCrLf

    strRel = "<relationlist>" & vbCrLf
    For i = 1 To rRelations.Count
        Set myRelation = rRelations.Item(i)

        strRel = strRel & "<rel>" & vbCrLf
        strRel = strRel & "<property>" &
                 GetPropertyRelation(myRelation) & "</property>" &
                 vbCrLf
        strRel = strRel & "</rel>" & vbCrLf
    Next i
    strRel = strRel & "</relationlist>" & vbCrLf

    strInh = "<inheritancelist>" & vbCrLf
    For i = 1 To iInheritances.Count
        Set myInheritance = iInheritances.Item(i)

        strInh = strInh & "<inh>" & vbCrLf
        strInh = strInh & "<property>" &
                 GetPropertyInheritance(myInheritance) &
                 "</property>" & vbCrLf
        strInh = strInh & "</inh>" & vbCrLf
    Next i
    strInh = strInh & "</inheritancelist>" & vbCrLf

    strID = "<idobj>" & vbCrLf
    strID = strID & "<propertyid>" & vbCrLf
    strID = strID & "<inhid>" & counterNodeInherit & "</inhid>" &
            vbCrLf
    strID = strID & "<entid>" & EntityID & "</entid>" & vbCrLf
    strID = strID & "<relid>" & RelationID & "</relid>" & vbCrLf
    strID = strID & "</propertyid>" & vbCrLf
    strID = strID & "</idobj>" & vbCrLf
```

```
    strXML = strXML & strEnt & strRel & strInh & strID

    strXML = strXML & "</object>" & vbCrLf

    GetScriptObject = strXML
End Function


'==== Prosedur untuk mengkonversi data XML ke dlm bentuk obyek ===
Sub ConvertFromXML(strXML As String)
    Dim XMLDoc As New FreeThreadedDOMDocument30
    Dim XMLNode As IXMLDOMNode, entNode As IXMLDOMNode
    Dim XMLRel As IXMLDOMNode, entRel As IXMLDOMNode
    Dim XMLInh As IXMLDOMNode, entInh As IXMLDOMNode

    Dim myEntity As New CDMEntity
    Dim myRelation As New CDMRelation
    Dim myInheritance As New CDMInheritance

    Dim i As Integer

    XMLDoc.async = False
    XMLDoc.validateOnParse = True
    XMLDoc.loadXML strXML

    Set XMLNode = XMLDoc.selectSingleNode("//idobj")
    If Not XMLNode Is Nothing Then
        Set entNode = XMLNode.childNodes.Item(i)
        ConvertIdOBj entNode
    End If

    Set XMLNode = XMLDoc.selectSingleNode("//entitylist")
    If Not XMLNode Is Nothing Then
        For i = 0 To XMLNode.childNodes.Length - 1
            Set entNode = XMLNode.childNodes.Item(i)
            Set myEntity = ConvertToEntity(entNode)
            eEntities.Add , , , , , , , , myEntity
        Next i
    End If

    Set XMLRel = XMLDoc.selectSingleNode("//relationlist")
    If Not XMLRel Is Nothing Then
        For i = 0 To XMLRel.childNodes.Length - 1
            Set entRel = XMLRel.childNodes.Item(i)
            Set myRelation = ConvertToRelation(entRel)
            rRelations.Add , , , , , , , , , , , , , , ,
                           myRelation
        Next i
    End If

    Set XMLRel = XMLDoc.selectSingleNode("//inheritancelist")
    If Not XMLRel Is Nothing Then
        For i = 0 To XMLRel.childNodes.Length - 1
            Set entInh = XMLRel.childNodes.Item(i)
            Set myInheritance = ConvertToInheritance(entInh)
            iInheritances.Add , , , , , myInheritance
        Next i
    End If
End Sub
```

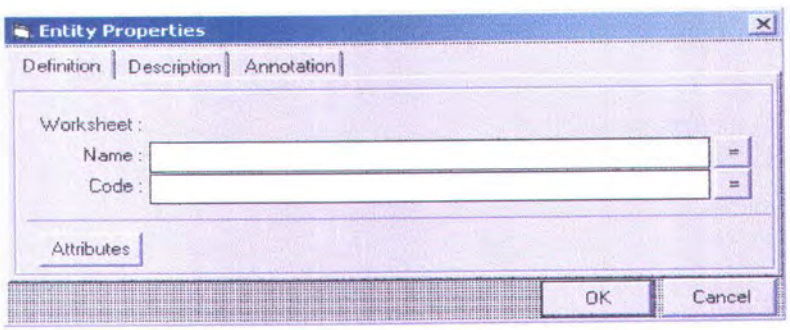
5.1.5. Fungsi Dan Prosedur Yang Berhubungan Dengan Obyek Diagram.

Operasi yang terjadi pada obyek diagram adalah penambahan, penghapusan dan perubahan dimana obyek diagram disini dapat berupa entitas, relasi, dan turunan. Penambahan obyek diagram dilakukan dengan memilih obyek yang akan ditambahkan melalui *toolbox* seperti berikut ini:

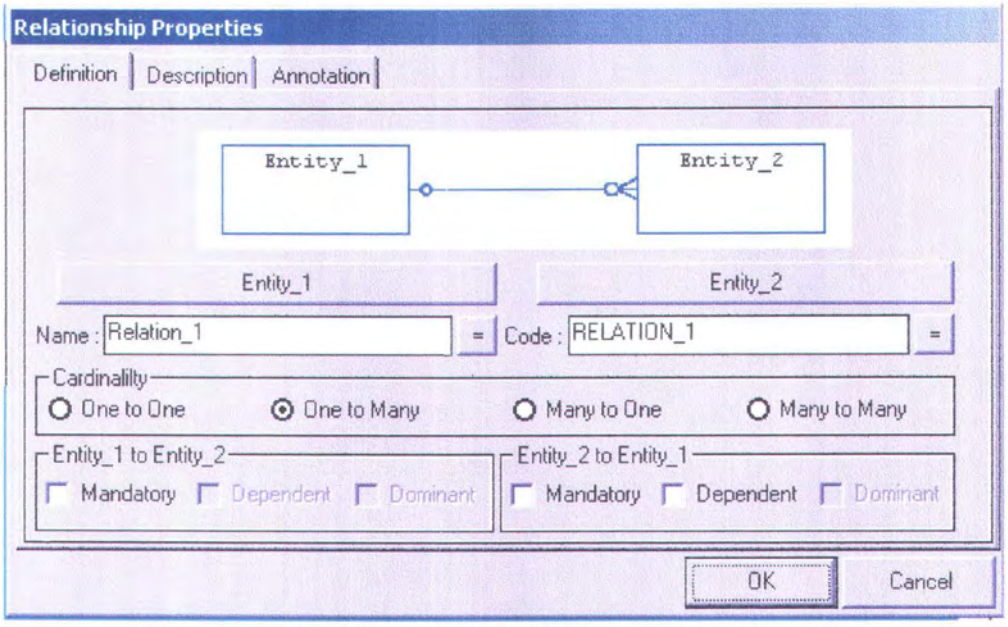
Gambar 5.3 Toolbox

Penghapusan obyek dilakukan dengan menggunakan *toolbox* diatas atau dengan menggunakan tombol Delete pada Keyboard dengan memilih obyek yang akan dihapus terlebih dahulu.

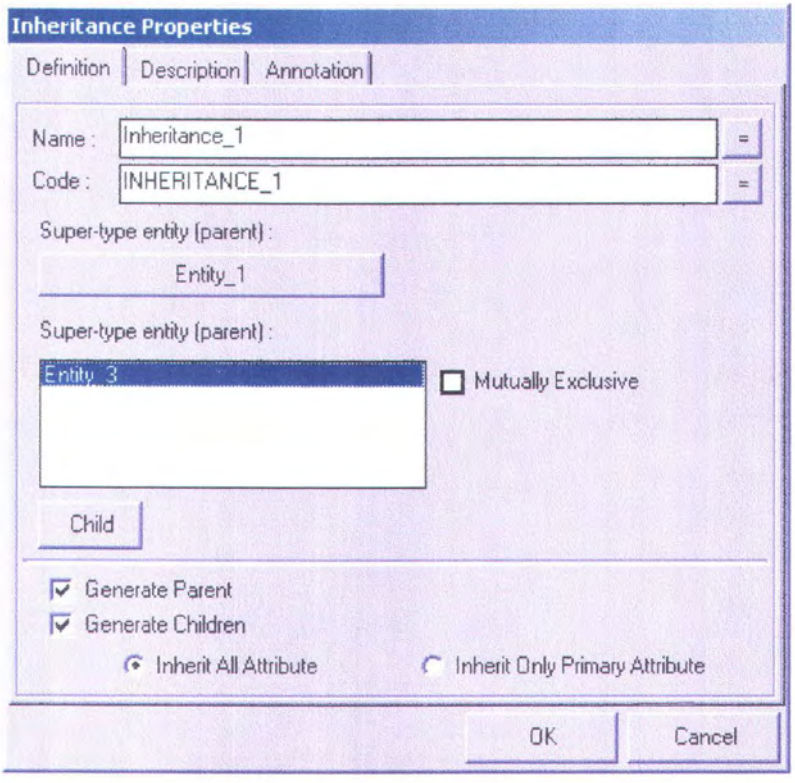
Perubahan obyek diagram dapat berupa perubahan *property*, posisi, maupun warna serta informasi tentang obyek lain yang mempunyai hubungan. Proses perubahan data property masing-masing obyek dilakukan dengan menggunakan form berikut ini:



Gambar 5.4 Form Perubahan Entitas



Gambar 5.5 Form Perubahan Relasi



Gambar 5.6 Form Perubahan Turunan

```
'========= Prosedur untuk menambah sebuah obyek entity =====
Sub AddEntity()
     EntityID = EntityID + 1
     eEntities.Add EntityID, Y, X

     drawEntity AddFlow, eEntities.Item(CStr(EntityID))

     addChangedObject CStr(EntityID), "ENTITY", "ADD",
                      eEntities.Item(CStr(EntityID)).Name
                      'menambah info perubahan data
     If Conn = "ONLINE" Then
         If Not SetModel Then Exit Sub
     End If
End Sub

'========= Prosedur untuk menambah sebuah obyek relationship =====
Sub AddRelation()
     vArahMouse = GetDirectionMouse(nNodeOrg, nNodeDest)

     RelID = addLink(AddFlow, imglstArrow, vArahMouse, nNodeOrg,
             nNodeDest)

     addChangedObject RelID, "RELATION", "ADD",
             rRelations.Item(RelID).Name 'menambah info perubahan
             data
End Sub

'========= Prosedur untuk menambah sebuah obyek inheritance =====
Sub AddInheritance()
    counterNodeInherit = counterNodeInherit + 1

    iInheritances.Add CStr(counterNodeInherit), vTopInherit,
                      vLeftInherit, Mid(nNodeOrg.Key, 2)
    IndexChildInherit = 1

     addChangedObject CStr(counterNodeInherit), "INHERITANCE",
               "ADD",
             iInheritances.Item(CStr(counterNodeInherit)).Name
             'menambah info perubahan data
End Sub

'========= Prosedur untuk menghapus sebuah obyek entity =====
Sub RemoveEntity()
    addChangedObject CStr(Index), "ENTITY", "DELETE"
    eEntities.Remove Index
End Sub

'======== Prosedur untuk menghapus sebuah obyek relationship =====
Sub RemoveRelation()
    aAddflow.Nodes.Remove ("NAO_" & vIndex)
    aAddflow.Nodes.Remove ("NAD_" & vIndex)
    aAddflow.Nodes.Remove ("NAPO" & vIndex)
    aAddflow.Nodes.Remove ("NAPD" & vIndex)
    rRelations.Remove vIndex

    addChangedObject CStr(vIndex), "RELATION", "DELETE"
End Sub

'========= Prosedur untuk menghapus sebuah obyek inheritance =====
Sub RemoveInheritance()
    aAddflow.Nodes.Remove ("NI_" & vIndex)
```

```
    iInheritances.Remove vIndex

    addChangedObject CStr(vIndex), "INHERITANCE", "DELETE"
End Sub

'========= Prosedur untuk mengupdate sebuah obyek entity =====
Sub UpdateEntity()
    ResizeEntity AddFlow, imglstArrow, nNodeSelected, "SIZE"
End Sub

'====== Prosedur untuk mengupdate sebuah obyek relationship =====
Sub UpdateRelation()
    Dim vRelation As CDMRelation

    For Each vRelation In rRelations
        If "NAD_" & vRelation.NADID = nNodeUpdate.Key Then
            vRelation.NADLeft = nNodeUpdate.Left
            vRelation.NADTop = nNodeUpdate.Top
            vRelation.NADDirection = vDirection
            addChangedObject CStr(vRelation.ID), "RELATION",
                        "UPDATE", vRelation.Name 'menambah info
                        perubahan data
        ElseIf "NAO_" & vRelation.NAOID = nNodeUpdate.Key Then
            vRelation.NAOLeft = nNodeUpdate.Left
            vRelation.NAOTop = nNodeUpdate.Top
            vRelation.NAODirection = vDirection
            addChangedObject CStr(vRelation.ID), "RELATION",
                        "UPDATE", vRelation.Name 'menambah info
                        perubahan data
        End If
    Next
End Sub

'======= Prosedur untuk mengupdate sebuah obyek inheritance =====
Sub UpdateInheritance()
    Dim vInheritace As CDMInheritance

    For Each vInheritace In iInheritances
        If "NI_" & vInheritace.ID = nNodeUpdate.Key Then
            vInheritace.Left = nNodeUpdate.Left
            vInheritace.Top = nNodeUpdate.Top
            addChangedObject CStr(vInheritace.ID), "INHERITANCE",
                        "UPDATE", vInheritace.Name
        End If
    Next
End Sub
```

5.1.6. Fungsi Dan Prosedur Yang Berhubungan Forward Engineering.

```
'======= Fungsi untuk memeriksa apakah diagram bisa dikonversi ke
'======= model fisik atau tidak =====
Function CheckDiagram() As Boolean
        For i = 0 To UBound(vEntityListDep)
            vEntityListDepTemp = vEntityListDep
            If IsLoopDependent(vEntityListDepTemp, i) Then
                MsgBox "Error : Entity depend on one another!!!",_
```

```
                        vbExclamation
                CheckDiagram = False
                Exit Function
            End If
        Next i
End Function

'======= Prosedur untuk mengkonversi obyek diagram yang
'=======digenerate ke model fisik =====
Sub CreatePDMObject()
    For Each eEntity In eEntities
        PDMTableID = PDMTableID + 1

        Set tPDMTable = CopyEntityToTable(eEntity)
        tPDMTable.ID = PDMTableID
        tPDMTables.Add , , , , , , , , , , , , , , tPDMTable
    Next
    For Each rRelation In rRelations
        With rRelation
            If .Cardinality = OnetoMany Then
                Set tTableP = GetPDMTable_ID(rRelation.NAOBaseID)
                Set tTableC = GetPDMTable_ID(rRelation.NADBaseID)

                If eEntities.Item(rRelation.NAOBaseID).IsGenerate_
                    And _
             eEntities.Item(rRelation.NADBaseID).IsGenerate_
                    Then
                   rPDMRelations.Add rPDMRelations.Count + 1,_
                    .Code, .name, tTableP.Code, tTableP.ID,_
                    tTableC.Code, tTableC.ID, IIf(.NAOMandatory,_
                    1, 0), "n", IIf(.NADMandatory, True, False),_
                    .Description, .Annotation, 1, .ID
                End If
            End If
        End With
    Next
    For Each iInherit In iInheritances
        Set tTableP = GetPDMTable_ID(iInherit.ParentID)
        If eEntities.Item(iInherit.ParentID).IsGenerate Then
            For Each iInheritC In iInherit.InhChilds
                Set tTableC = GetPDMTable_ID(iInheritC.ChildID)
                If eEntities.Item(iInheritC.ChildID).IsGenerate_
                    Then
                    '== Add Reference ==
                    InhChildCount = InhChildCount + 1
                End If
            Next
        End If
    Next
End Sub

'======= Prosedur untuk menkonversi obyek diagram yang
'=======berhubungan dengan obyek yang tidak digenerate
Sub CreateObjectPDMModified()
    For Each vColTab In vTableListFor(i).vCols
        ColCode = vColTab.Code
        For Each vColTabTemp In tTable.Columns
            If vColTab.Code = vColTabTemp.Code Then
                vColID = vColID + 1
                vColTab.name = Left(tTable2.Code, 3) & vColID &_
                                "_" & vColTab.name
                vColTab.Code = Left(tTable2.Code, 3) & vColID &_
                                "_" & vColTab.Code
```

```
            End If
        Next

        With vColTab
            .ID = tTable.Columns.Count + 1
            .TableRef = tTable2.Code
            .ColumnRef = ColCode
            tTable.Columns.Add , , , , , , , , , vColTab
        End With
        AddColumnList_Code tTable2, ColCode, vColTab.Code,_
                    tTable.Code
        pRel.RelColumns.Add ColCode, vColTab.Code
    Next
End Sub

'====== Fungsi untuk mencari foreign key yang primary dari tabel =
Function GetForeignKeyPrimary () As Boolean
    For Each vColTab In vTableListFor(i).vCols
        ColCode = vColTab.Code
        For Each vColTabTemp In tTable.Columns
            If vColTab.Code = vColTabTemp.Code Then
                vColID = vColID + 1
                vColTab.name = Left(tTable2.Code, 3) & vColID &_
                                "_" & vColTab.name
                vColTab.Code = Left(tTable2.Code, 3) & vColID &_
                                "_" & vColTab.Code
            End If
        Next

        With vColTab
            .ID = tTable.Columns.Count + 1
            .TableRef = tTable2.Code
            .ColumnRef = ColCode
            tTable.Columns.Add , , , , , , , , , vColTab
        End With
        AddColumnList_Code tTable2, ColCode, vColTab.Code, _
                    tTable.Code
        pRel.RelColumns.Add ColCode, vColTab.Code
    Next
End Function

'======= Fungsi untuk mencari foreign key tabel =====
Function GetForeignKey () As Boolean
    For Each cRel In rRelations
        If eEntities.Item(cRel.NADBaseID).IsGenerate And _
            eEntities.Item(cRel.NAOBaseID).IsGenerate Then
            Set tTable1 = GetPDMTable_ID(cRel.NAOBaseID)
            Set tTable2 = GetPDMTable_ID(cRel.NADBaseID)

            SetForeignKey cRel.name, tTable1, tTable2, _
                    cRel.NAOMandatory, False
            Set tTable = GetPDMTable(cRel.name)

            '=== add Reference '
    Next
End Function

'======= Prosedur untuk menghapus obyek model fisik yang tidak di
'======= generate =====
Sub RemoveUnGenerated()
    For Each vEnt In eEntities
        If Not vEnt.IsGenerate Then
            For Each vTab In tPDMTables
```

```
                If vTab.name = vEnt.name Then
                    tPDMTables.Remove CStr(vTab.ID)
                    Exit For
                End If
            Next
        End If
    Next
End Function
```

5.1.7. Fungsi Dan Prosedur Yang Berhubungan Reverse Engineering

```
Public Function GenerateCDMObject() As Boolean
    Dim vTable As PDMTable, vTableTemp As PDMTable
    Dim vRel As PDMRelation
    Dim vCol As PDMColumn
    Dim vColList As PDMColList
    Dim vTableID_MM() As String
    Dim Refered As Boolean,Flagged As Boolean,InhNotAdd As Boolean
    Dim i As Integer, vCountInhChild As Integer

    Dim vEntity As CDMEntity
    Dim vRelation As CDMRelation
    Dim vInheritance As CDMInheritance, vInhTemp As CDMInheritance
    Dim vInhChild As CDMInhChild

    If Not eEntities Is Nothing Then Set eEntities = Nothing
    If Not rRelations Is Nothing Then Set rRelations = Nothing
    If Not iInheritances Is Nothing Then Set iInheritances=Nothing
    If Not iInhChilds Is Nothing Then Set iInhChilds = Nothing

    If gTables.Count = 0 Then
        MsgBox "No Table in worksheet!", vbExclamation
        Exit Function
    End If

    ReDim vTableID_MM(0)
    For Each vTable In gTables
        If IsDoubleRelation(vTable.ID) Then
            For Each vCol In vTable.Columns
                If vCol.PrimaryKey And vCol.ForeignKey Then
                    If vCol.ColLists.Count > 0 Then
                        Refered = True
                        Exit For
                    End If
                Else
                    Refered = True
                    Exit For
                End If
            Next

            If Not Refered Then
                If vTableID_MM(0) <> "" Then
                    ReDim vTableID_MM(UBound(vTableID_MM) + 1)
                End If
                vTableID_MM(UBound(vTableID_MM)) = vTable.ID
            End If
            Refered = False
        End If
```

```
    Next

    '=== generate entity dari tabel yang berasal dari entity
    For Each vTable In gTables
        If vTable.Cdm_Flag = 1 Then
            Flagged = False
            For i = 0 To UBound(vTableID_MM)
                If vTableID_MM(i) = CStr(vTable.ID) Then
                    Flagged = True
                    Exit For
                End If
            Next i

            If Not Flagged Then
                Set vEntity = CopyTableToEntity(vTable)
                eEntities.Add , , , , , , , , vEntity
            End If
        End If
    Next

    '=== generate entity dari tabel baru
    For Each vTable In gTables
        If vTable.Cdm_Flag = 0 Then
            Flagged = False
            For i = 0 To UBound(vTableID_MM)
                If vTableID_MM(i) = CStr(vTable.ID) Then
                    Flagged = True
                    Exit For
                End If
            Next i

            If Not Flagged Then
                Set vEntity = CopyTableToEntity(vTable)
                vEntity.ID = LastIDEnt + 1
                vTable.Cdm_Entity_ID = vEntity.ID
                eEntities.Add , , , , , , , , vEntity
            End If
        End If
    Next

    '=== generate relation dari reference dari relation
    '=== generate inheritance dari reference
    For Each vRel In gRelations
        Flagged = False
        If vRel.Cdm_Flag = 1 Then
            For i = 0 To UBound(vTableID_MM)
                If vTableID_MM(i) = CStr(vRel.ChildID) Then
                    Flagged = True
                    Exit For
                End If
            Next i

            If Not Flagged Then
                Set vRelation = CopyRefToRelation(vRel)
                rRelations.Add ,,,,,, , , , , , , , , , vRelation
            End If
        ElseIf vRel.Cdm_Flag = 2 Then
            InhNotAdd = False
            For Each vInhTemp In iInheritances
              If vInhTemp.ParentID=
eEntities.Item(CStr(vRel.ParentID)).ID Then
                    InhNotAdd = True
                    Exit For
```

```
                End If
            Next

            If Not InhNotAdd Then
                Set vInheritance = CopyRefToInh(vRel)
                Set vInhChild = New CDMInhChild
                vInhChild.ID = vRel.Cdm_InhChild_ID
                vInhChild.ChildID =
eEntities.Item(CStr(vRel.ChildID)).ID
                vInhChild.ParentID =
eEntities.Item(CStr(vRel.ParentID)).ID

                vInheritance.InhChilds.Add , , , vInhChild

                iInheritances.Add , , , , , vInheritance
            Else
                Set vInhChild = New CDMInhChild
                vInhChild.ID = vRel.Cdm_InhChild_ID
                vInhChild.ChildID =
eEntities.Item(CStr(vRel.ChildID)).ID
                vInhChild.ParentID =
eEntities.Item(CStr(vRel.ParentID)).ID

                vInhTemp.InhChilds.Add , , , vInhChild
            End If
        End If
    Next

    '=== generate relation dari reference baru
    For Each vRel In gRelations
        Flagged = False
        If vRel.Cdm_Flag = 0 Then
            For i = 0 To UBound(vTableID_MM)
                If vTableID_MM(i) = CStr(vRel.ChildID) Then
                    Flagged = True
                    Exit For
                End If
            Next i

            If Not Flagged Then
                Set vRelation = CopyRefToRelation(vRel)
                vRelation.ID = LastIDRel + 1
                vRelation.NAOID = vRelation.ID
                vRelation.NADID = vRelation.ID
                vRelation.NPOID = vRelation.ID
                vRelation.NPDID = vRelation.ID
                rRelations.Add ,,,,,, , , , , , , , , , vRelation
            End If
        End If
    Next

    '=== generate relation dari table
    For Each vTable In gTables
        Flagged = False
        For i = 0 To UBound(vTableID_MM)
            If vTableID_MM(i) = CStr(vTable.ID) Then
                Flagged = True
                Exit For
            End If
        Next i

        If Flagged Then
            Set vRelation = CreateRelMToM(vTable)
```

```
            If vTable.Cdm_Flag <> 2 Then
                vRelation.ID = LastIDRel + 1
                vRelation.NAOID = vRelation.ID
                vRelation.NADID = vRelation.ID
                vRelation.NPOID = vRelation.ID
                vRelation.NPDID = vRelation.ID
            End If
            rRelations.Add , , , , , , , , , , , , , , , vRelation
        End If
    Next
End Function
```

5.2. Pembuatan Aplikasi Server

Aplikasi server dalam tugas akhir ini diimplementasikan dengan ISAPI Extension yang dibuat dengan menggunakan bahasa pemrograman Visual C++ 6.0. Aplikasi ini terdiri dari beberapa *class* yang telah disebutkan dan dijelaskan pada bab sebelumnya. *Class* utama dalam aplikasi server ini adalah **CCDMServerExtention** yang berisi fungsi-fungsi yang akan diproseskan oleh server dijalankan.

Sebagaimana aplikasi *client*, penjelasan mengenai fungsi-fungsi penting yang terdapat dalam aplikasi *server* akan dituliskan secara singkat. Berikut ini adalah penjelasan mengenai variabel dan fungsi-fungsi yang ada pada *class* utama.

5.2.1. Varibel Global

1. **cWorksheetList**, merupakan obyek yang berisi daftar atau koleksi worksheet yang sedang aktif.
2. **cUserList**, merupakan obyek yang berisi daftar atau koleksi pengguna yang sedang menggunakan worksheet yang sedang aktif.

5.2.2. Fungsi-Fungsi Dalam Class Utama

```
///// Fungsi untuk membuat worksheet /////
void CCDMServerExtension::CreateWorksheet(CHttpServerContext
*pCtxt, LPCTSTR work, LPCTSTR pwd, LPCTSTR nick)
{
	CCWorksheet WS, WS_;
	CCDMUser Us;
	CString mWork,mNick, temp, WSname, temp2,mPwd;

	WS_.WorksheetName = mWork;
	WS_.TimeCreate = time(&tNow);
	WS_.intUser = 0;
	WS_.UserCount = 1;
	WS_.Pwd = mPwd;

	cWorkSheetList->Add(WS_);

	Us.nickname = mNick;
	Us.status = "ADMIN";
	Us.worksheet = mWork;
	Us.TimeCreate = time(&tNow);

	cUserList->Add(Us);

	*pCtxt << _T("OK" );
}


///// Fungsi untuk membuat User /////
void CCDMServerExtension::CreateUser(CHttpServerContext *pCtxt,
LPCTSTR work, LPCTSTR pwd, LPCTSTR nick)
{
	CCWorksheet WS;
	CCDMUser Us, UsC;
	CString mWork, mNick,UsName;
	Us.nickname = mNick;
	Us.worksheet = mWork;
	Us.status = "JOIN";
	Us.TimeCreate = time(&tNow);
	cUserList->Add(Us);
	*pCtxt << _T("OK");
}



///// Fungsi untuk mendapatkan daftar User /////
void CCDMServerExtension::GetUserList(CHttpServerContext *pCtxt,
LPCTSTR work)
{
	intList = cUserList->GetSize();
	i=0;
	while (i < intList )
	{
		vUser = cUserList->GetAt(i);
		WorkName.Format(vUser.worksheet);
		WorkName.MakeUpper();

		if (WorkName==mWork)
		{
			UserName.Format(vUser.nickname);
			UserStatus.Format(vUser.status);
```

```
                    UserNode = docPtr->createNode(varTyp,
                                _T("user"), "");
                    DOMElementPtr->appendChild(UserNode);
                    propNode = docPtr->createNode(varTyp,
                                _T("name"), "");
                    propNode->text = _bstr_t(UserName);
                    UserNode->appendChild(propNode);
                    propNode = docPtr->createNode(varTyp,
                                _T("status"), "");
                    propNode->text = _bstr_t(UserStatus);
                    UserNode->appendChild(propNode);
            }
            i++;
        }
        strOut.Format(docPtr->xml);
        *pCtxt << strOut;
}


/////  Fungsi untuk mendapatkan daftar Worksheet /////
void CCDMServerExtension::GetWorkList(CHttpServerContext *pCtxt)
{

        intList = cWorkSheetList->GetSize();
        i=0;
        while (i < intList )
        {
                WS = cWorkSheetList->GetAt(i);
                WSname.Format(WS.WorksheetName);
                WorkNode = docPtr->createNode(varTyp, _T("work"),
                            "");
                WorkNode->text = _bstr_t(WSname);
                DOMElementPtr->appendChild(WorkNode);
                i++;
        }
        strOut.Format(docPtr->xml);
        *pCtxt << strOut;
}

/////  Fungsi untuk mengambil Obyek model /////
void CCDMServerExtension::GetDiagram(CHttpServerContext* pCtxt,
        LPCTSTR work, LPCTSTR user, LPCTSTR status)
{
        // strOutput merupakan hasil konversi data kedalam bentuk
        // XML
        strOutput.Format(DOMDocPtr->xml);
        *pCtxt << (CString) strOutput;
}


/////  Fungsi untuk melakukan action pada diagram /////
void CCDMServerExtension::SetDiagram(CHttpServerContext* pCtxt,
LPCTSTR work, LPCTSTR user, LPCTSTR chgObj)
{
        // proses penambahan, penghapusan dan update data dilakukan
        // dalam fungsi ini

        // strAct merupakan flag apakah action yang dilakukan oleh
        // pengguna berhasil dilakukan atau tidak

        *pCtxt << strAct;
}
```

```
///// Fungsi untuk menyimpan ke dalam Repository /////
void CCDMServerExtension::SaveToRepository(CHttpServerContext*
pCtxt, LPCTSTR service, LPCTSTR work,LPCTSTR nick,LPCTSTR strXML)
{
      // Proses penyimpanan kedalam repository

      // strErr berisi informasi apakah penyimpanan berhasil atau
      // tidak

      *pCtxt << _T(strErr);
}


///// Fungsi untuk mengambil data dari repository /////
void CCDMServerExtension::OpenFromRepository(CHttpServerContext*
pCtxt, LPCTSTR service, LPCTSTR work,LPCTSTR nick)
{
      // Proses pengambilan data dari repository dan di konversi
      // dalam bentuk XML

      // data dalam bentuk XML disimpan dalam variabel strXML;
      *pCtxt << strXML;
}

///// Fungsi untuk menghapus User /////
void CCDMServerExtension::RemoveUser(CHttpServerContext *pCtxt,
LPCTSTR work, LPCTSTR nick)
{
      CCWorksheet myWorksheet;
      CCDMUser Us;
      i=0;
      intList = cUserList->GetSize();
      while (i<intList)
      {
            Us = cUserList->GetAt(i);
            UsName = Us.nickname;
            UsName.MakeUpper();
            UsWork = Us.worksheet;
            UsWork.MakeUpper();
            if ((UsName == mNick) && (UsWork == mWork))
            {
                  cUserList->RemoveAt(i);
                  break;
            }
            i++;
      }
      *pCtxt << _T("OK");
}


///// Fungsi untuk Login /////
void CCDMServerExtension::Login(CHttpServerContext *pCtxt, LPCTSTR
user, LPCTSTR pwd, LPCTSTR service, LPCTSTR nick, LPCTSTR nickpwd)
{
      // Proses login pengguna

      // sts berisi informasi apakah pengguna berhasil login atau
      // tidak

      *pCtxt << sts;
}
```

```
///// Fungsi untuk Logout /////
void CCDMServerExtension::Logout(CHttpServerContext *pCtxt,
LPCTSTR work, LPCTSTR nick)
{
      // Proses logout pengguna
      // Pengguna dihapus dari daftar dalam cUserList
}
```

5.3. Pembuatan Repository

Repository database oracle di server dibuat dengan menjalankan script database yang dapat dilihat dalam lampiran B.

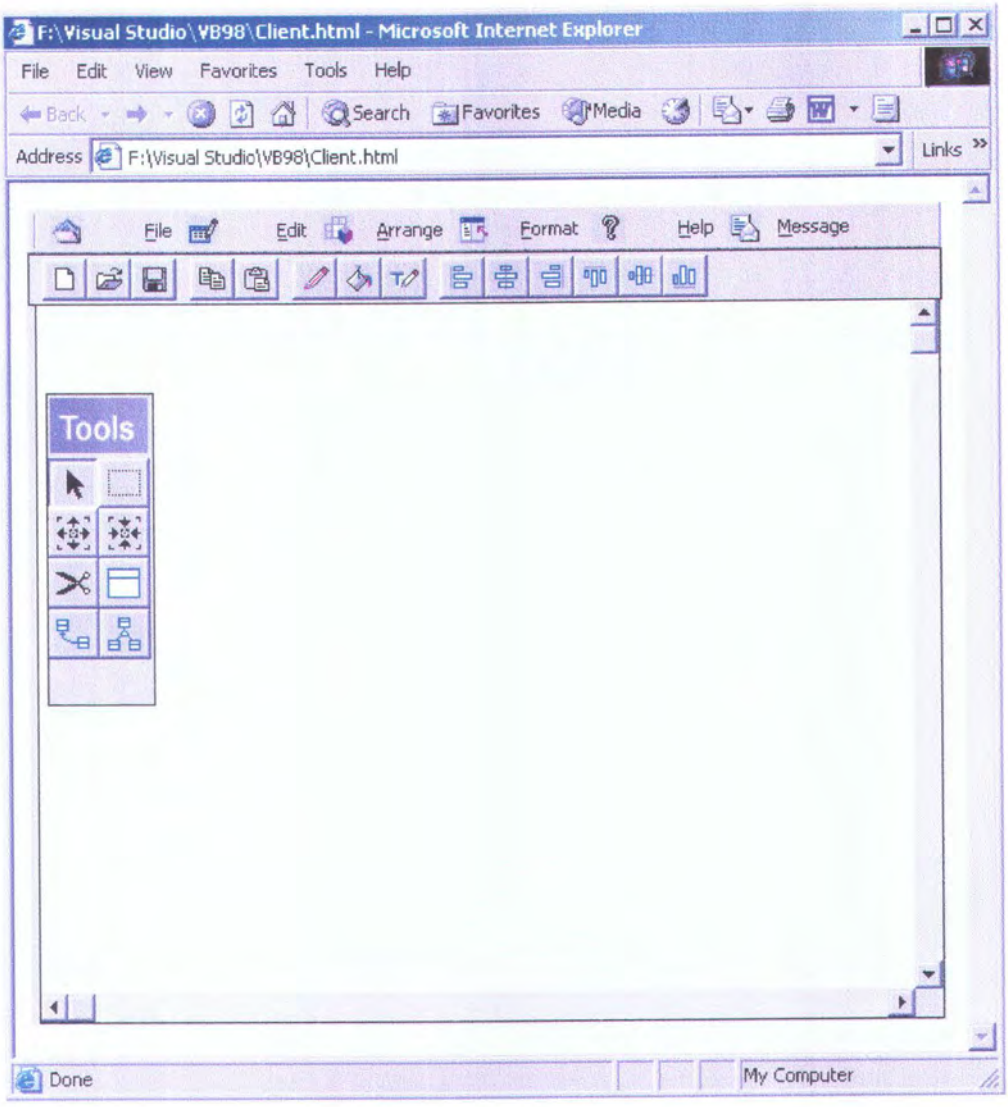
Pengguna yang akan menggunakan aplikasi tugas akhir ini harus terdaftar dalam tabel USERS dimana pengaturan pendaftaran pengguna dilakukan oleh siapapun yang mempunyai kewenangan dalam sistem ini.

5.4. Kebutuhan Sistem

Penggunaan aplikasi *case tool* ini memerlukan sistem yang disesuaikan dengan sistem yang digunakan pada saat pengembangan. Sistem ini dikembangkan dengan menggunakan komputer dengan spesifikasi : prosesor Intel Petium II 300 MHz, RAM 128 dan harddisk 20 Gbyte. Spesifikasi ini merupakan komposisi sistem hardware minimum. Penggunaan aplikasi ini dengan menggunakan spesifikasi yang lebih tinggi tentunya akan dapat menambah kinerja aplikasi dan diharapkan dapat bekerja secara optimal.

Aplikasi ini berjalan di lingkungan Microsoft Windows 2000 Server sebagai sistem operasi aplikasi servernya. Untuk aplikasi client penggunaan Microsoft Windows 98 sudah mencukupi. Aplikasi *case tool* ini menggunakan Internet Information System (IIS) sebagai web servernya dan menggunakan database Oracle 8i. Ada beberapa ActiveX Control yang harus diinstall di

komputer *client* yaitu Addflow dan Sheridan Sofware. Aplikasi client yang dibuat dalam tugas akhir ini berupa komponen ActiveX Control yang berjalan dalam sebuah halaman html dengan menggunakan Internet Explorer 5.0.



Gambar 5.7 Aplikasi Client pada sebuah browser

Aplikasi yang digunakan dalam pengembangan perangkat lunak ini adalah:

1. Visual Basic 6.0 untuk pembuatan aplikasi *client* yaitu komponen ActiveX Control.

2. Visual C++ 6.0 untuk pembuatan aplikasi *server* yaitu ISAPI Extension.
3. Adobe Photoshop 5.5 untuk pembuatan gambar dan icon.

BAB VI

UJI COBA DAN EVALUASI

BAB VI

UJI COBA DAN EVALUASI

Di bagian sebelumnya telah dijelaskan mengenai perancangan dan implematasi perangkat lunak. Dalam bab ini akan dibahas mengenai ujicoba terhadap aplikasi yang telah dibuat. Aplikasi yang telah selesai dibuat akan diuji dengan berbagai macam kondisi.

6.1.Lingkungan Uji Coba

Uji coba terhadap sistem yang telah dibuat dilakukan di Laboratorium Rekayasa Perangkat Lunak Teknik Informatika ITS dengan menggunakan dua buah komputer yang terhubung melalui intranet. Satu komputer sebagai komputer client untuk menjalankan aplikasi client dan satu komputer sebagai kmputer server yang digunakan untuk menjalankan aplikasi client dan server. Masing-masing kompouter berada pada lingkungan dengan spesifikasi sebagai berikut:

Spesifikasi sistem:

- Microsoft Windows 2000
- Oracle 9i, untuk komputer server
- Visual Basic 6
- Visual C++ 6
- Komponen AddFlow 2 dan SSDataWidget 3

Spesifikasi hardware:

- Komputer I (aplikasi server)
  - Pentium 4 2,4 GHz

- 512 MB RAM

- KomputerII (aplikasi client)
  - Pentium 4 2,4 GHz
  - 480 MB RAM

6.2.Skenario Uji Coba

Pada uji coba ini diberikan beberapa skenario yang ditujukan untuk mengetahui fungsionalitas dari perangkat lunak yang dibuat. Uji coba dilaksanakan untuk menguji kesesuaian perangkat lunak dengan desain fitur yang ingin dibuat. Pada uji coba ini akan dipilih fitur-fitur yang utama yang antara lain memodelkan basis data konseptual, manajemen model, sifat online-real time, komunikasi antar pengguna, konkurensi data, forward dan reverse engineering.

Pemodelan basis data konseptual dan manajemen model diuji dengan skenario perancangan model basis data konseptual secara single user. Komunikasi antar pengguna, sifat oline-realtime dan konkurensi data diuji dengan skenario perancangan model basis data konseptual secara multi user. Forward dan reverse engineering diuji dengan skenario forward dan reverse enginering pada pemodelan secara multi user. Berikut 3 skenario yang diujikan, yaitu:

1. Perancangan basis data konseptual secara single user.
2. Perancangan basis data konseptual secara multi user.
3. Forward dan reverse engineering pada perancangan basis data konseptual secara multi user.

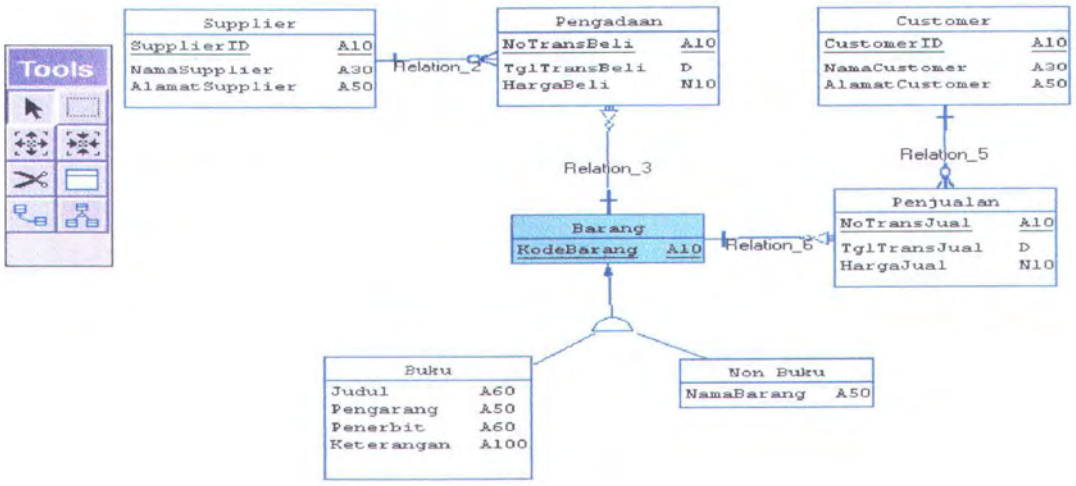
6.2.1. Uji Coba Skenario 1

Pada hari Rabu tanggal 19 Mei 2004, sebuah toko memesan sebuah perangkat lunak retail untuk mengolah data transaksi yang sudah dilakukan. Untuk itu seorang analis merancang sebuah sistem informasi berupa diagram basis data konseptual. Setelah melalui beberapa proses interview dengan pihak toko, didapatkan bahwa ada beberapa entitas yang diperlukan dalam merancang diagram yaitu Supplier, Customer, Pengadaan, Penjualan, dan Barang. Entitas Barang memiliki subclass yaitu Buku dan Non Buku.

Pada skenario ini ujicoba dilakukan untuk mengetahui, apakah benar bahwa perangkat lunak yang telah dibuat dapat digunakan untuk merancang basis data konseptual dan apakah manajemen model yang dilakukan berjalan sesuai dengan perancangan.

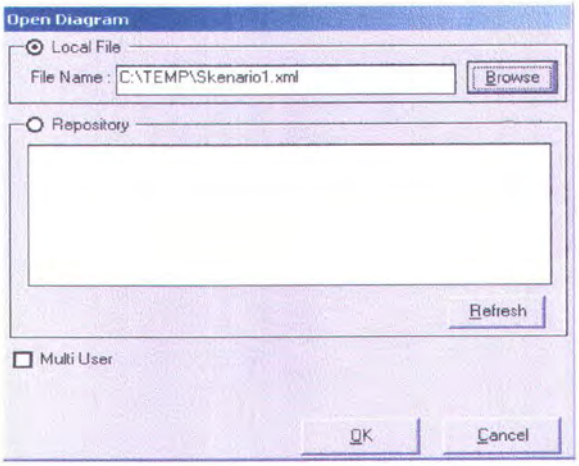
Pertama-tama yang dilakukan adalah menggambar semua obyek diagram yang diperlukan baik itu entitas, relasi maupun turunan. Atribut dalam entitas, jenis relasi atau *cardinality*, maupun property turunan yang ada. Perubahan warna pada entitas dapat dilakukan untuk mempermudah pengenalan entitas tertentu.

Setelah proses perancangan atau penggambaran diagram basis data konseptual dilakukan maka diagram yang ada pada aplikasi client tampak seperti gambar berikut ini dimana gambar ini dapat disesuaikan dengan keinginan pengguna.



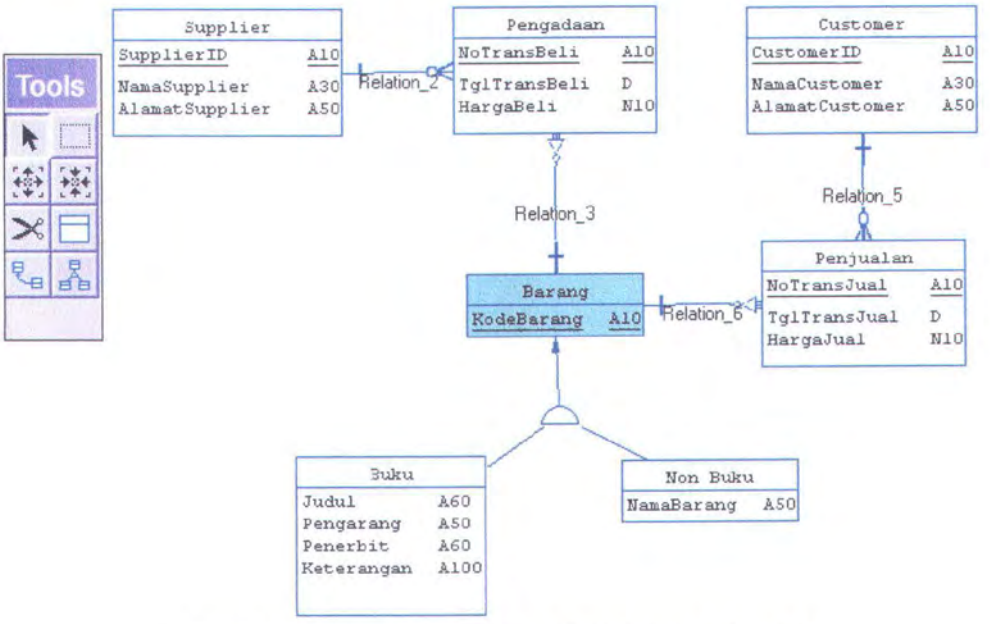
Gambar 6.1 Diagram Konseptual Skenario 1

Diagram yang telah dibuat kemudian disimpan dalam file lokal dengan menekan toolbar dengan icon disket atau melalui menu File -> Save -> File dengan lokasi file di "C:\TEMP\Skenario.xml". Untuk membuktikan apakah file yang telah disimpan dapat dibuka dan tampil dengan bentuk diagram seperti semula, maka diagram tersebut dihapus dulu dengan menekan toolbar dengan icon kertas. Melalui menu File -> Open File akan dimunculkan dialog untuk membuka file yang telah disimpan sebelumnya. Dialog tersebut akan tampak seperti berikut ini:



Gambar 6.2 Dialog Open File

Diagram yang dibuka melalui dialog diatas tampil dengan susunan diagram seperti gambar dibawah ini.



Gambar 6.3 Diagram Konseptual yang dibuka melalui dialog Open

Dari skenario pertama dapat dilihat bahwa perangkat lunak dapat digunakan untuk merancang sebuah diagram basis data secara konseptual. Diagram yang dibuka setelah disimpan tampak sama seperti diagram yang dibuat pertama kali. Hal ini menunjukkan bahwa manajemen model perangkat lunak dalam hal ini penyimpanan dan pengambilan kembali data diagram berjalan dengan baik.

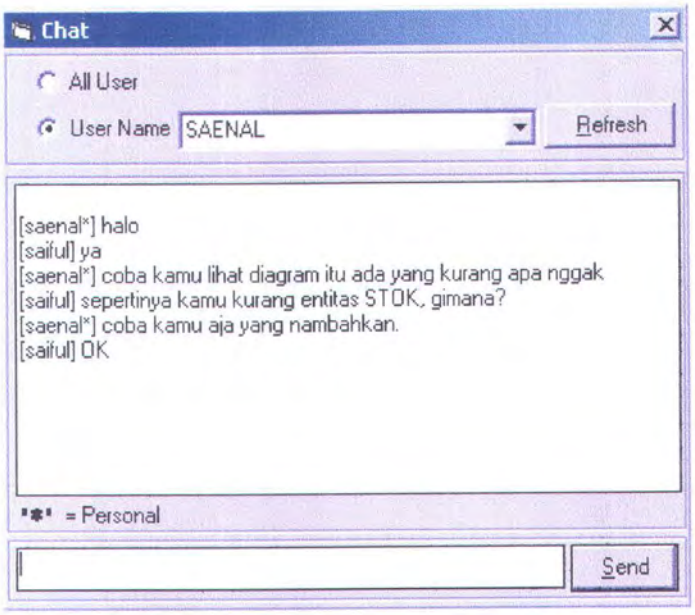
6.2.2. Uji Coba Skenario 2

Pada hari Kamis tanggal 20 Mei 2004, analis pada skenario 1 (pengguna 1) meminta masukan dari salah seorang rekannya (pengguna 2) atas desain

diagram yang telah dibuatnya. Pengguna 1 membuka diagramnya secara multi user melalui dialog 'Open File' agar dapat diakses oleh pengguna 2 melalui intranet. Pengguna 1 memberi nama worksheetnya 'SKENARIO2' dan menyimpannya dalam repository.

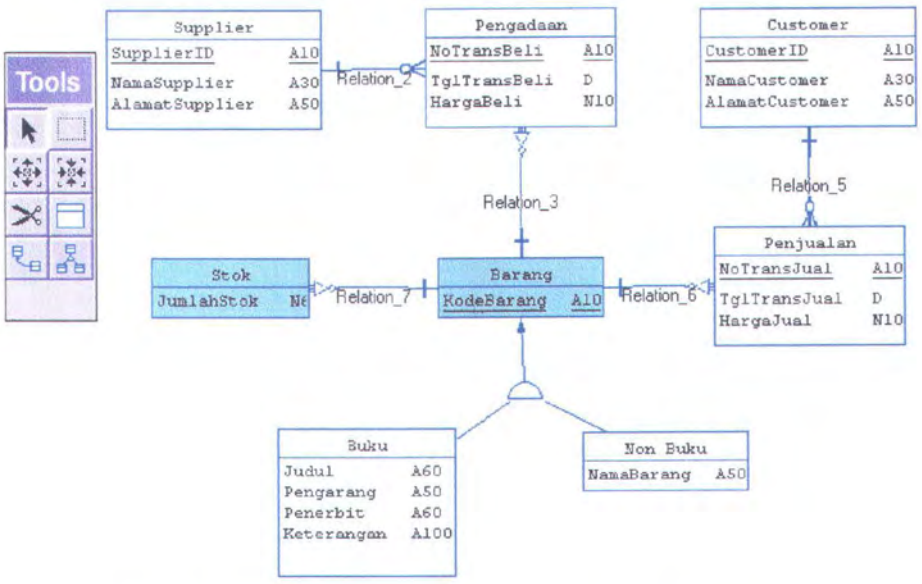
Pada ujicoba yang kedua ini untuk mengetahui apakah perangkat lunak yang dibuat mampu berkerja secara online realtime, dapat menjaga konkurensi data, dan dapat berkomunikasi dengan pengguna lain. Pada ujicoba yang kedua digunakan diagram yang sama dengan skenario 1.

Hal pertama yang dilakukan oleh pengguna 2 adalah melakukan *join* dalam worksheet yang telah dibuka tersebut. Pengguna 1 membuka form dialog komunikasi untuk berkomunikasi dengan pengguna 2.



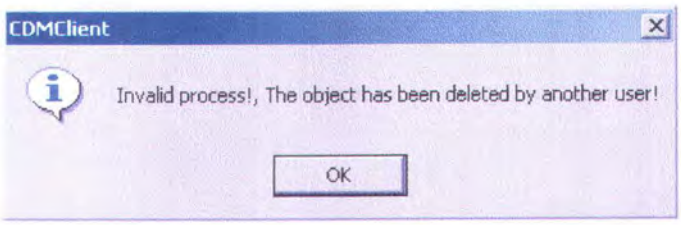
Gambar 6.4 Media Komunikasi dengan pengguna lain

Dari hasil komunikasi dengan pengguna 1, pengguna 2 menambahkan sebuah entitas yaitu STOK beserta atributnya. Diagram yang dihasilkan tampak sebagai berikut.



Gambar 6.5 *Diagram Konseptual setelah ditambahkan sebuah obyek entitas*

Perubahan ini langsung diketahui oleh pengguna 1 yang ditunjukkan dengan diagram yang ada pada pengguna 1 sama dengan diagram diatas. Tetapi entitas STOK dihapus oleh pengguna 1 dan hal ini dilakukan juga oleh pengguna 2 dalam waktu yang bersamaan. Pengguna 1 dapat menghapus entitas stok, sedangkan pengguna dua tidak dapat menghapusnya karena sudah dihapus oleh pengguna 1. Hal ini ditunjukkan dengan tampilnya pesan pada editor pengguna2.



Gambar 6.6 Pesan pemberitahuan

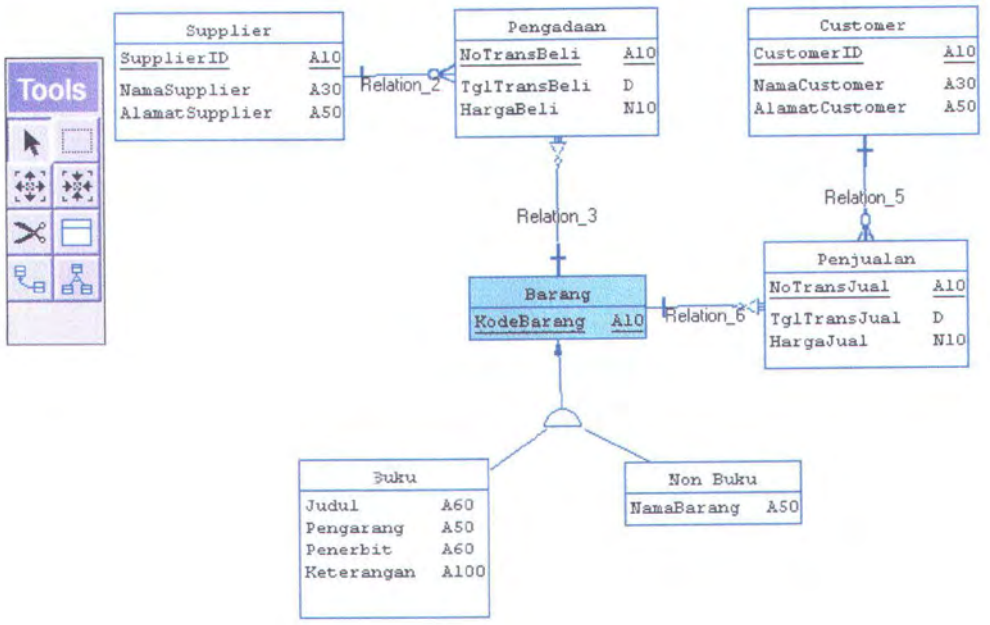
Dari hasil studi kasus di atas telah terbukti seorang pengguna yang berada pada mode multi user melakukan perubahan pada diagram akan mengakibatkan perubahan yang sama pada diagram dengan worksheet yang sama dalam interval waktu yang singkat (kurang lebih 3 detik). Dari sini bisa disimpulkan bahwa perangkat lunak yang dibuat telah mampu bekerja secara online dan realtime (mendekati realtime). Munculnya informasi penghapusan obyek yang dilakukan oleh pengguna lain ditujukan untuk menjaga konkurensi data sehingga tidak ada penghapusan obyek yang dilakukan berulang kali. Hal ini menunjukkan bahwa perangkat lunak ini memperhatikan konkurensi data pada mode multi user. Fasilitas Chatting yang disediakan oleh perangkat lunak bekerja dengan baik sehingga dapat disimpulkan bahwa perangkat lunak ini dapat digunakan untuk berkomunikasi dengan pengguna lain.

6.2.3. Uji Coba Skenario 3

Pada hari Jumat tanggal 21 Mei 2004 worksheet yang dibuat oleh analis yang bertindak sebagai administrator dalam hal ini adalah pengguna 1 dibuka secara multi user. Pengguna 1 akan melakukan proses forward engineering dari model konseptual menjadi model fisik.

Pada ujicoba yang ketiga ini bertujuan untuk mengetahui apakah proses forward dan reverse engineering berjalan dengan baik. Hal ini dilakukan dengan melakukan forward engineering kemudian dilakukan perubahan pada model fisik. Model fisik ini kemudian di kembalikan menjdai model konseptual melalui proses reverse engineering dengan menggunakan perangkat lunak PDM Editor.

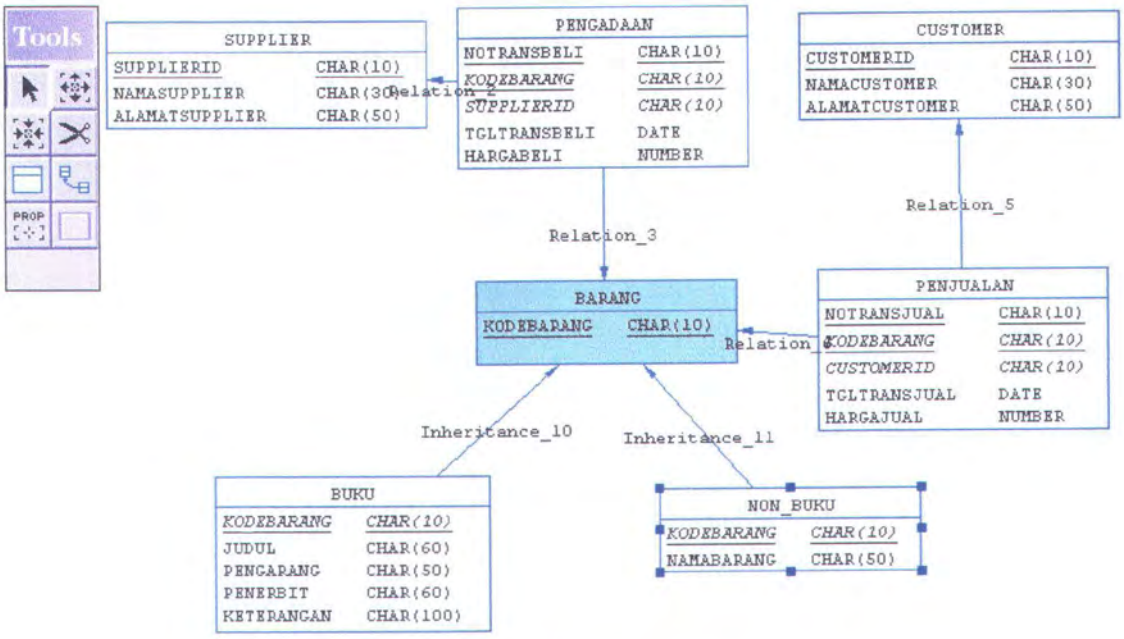
Berikut adalah diagram yang dibukan oleh pengguna 1.



Gambar 6.7 Diagram Konseptual pada Skenario 3

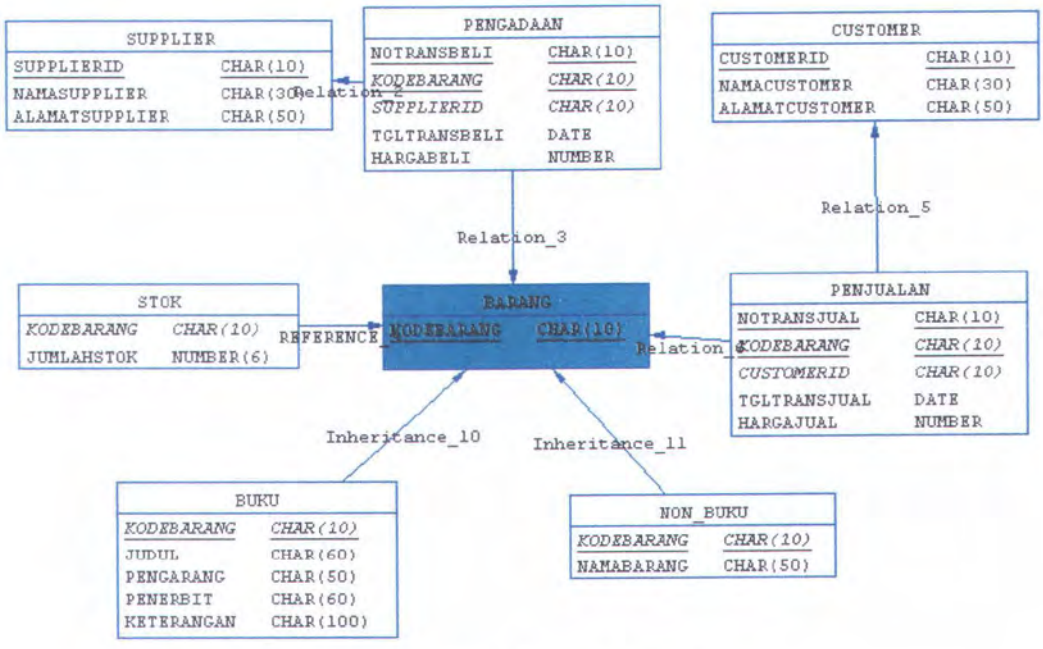
Diagram pada worksheet dengan nama SKENARIO2 ini digenerate menjadi model fisik melalui menu File -> Generate Physical Model. Worksheet ini berada pada mode multi user sehingga hasil proses forward engineering disimpan dalam database repository.

Pengguna 1 kemudian meminta pengguna yang lain (pengguna 2) untuk melihat hasil forward engineering melalui PDM Editor. Diagram yang dibuka oleh pengguna 2 tampak pada PDM Editor seperti gambar berikut.



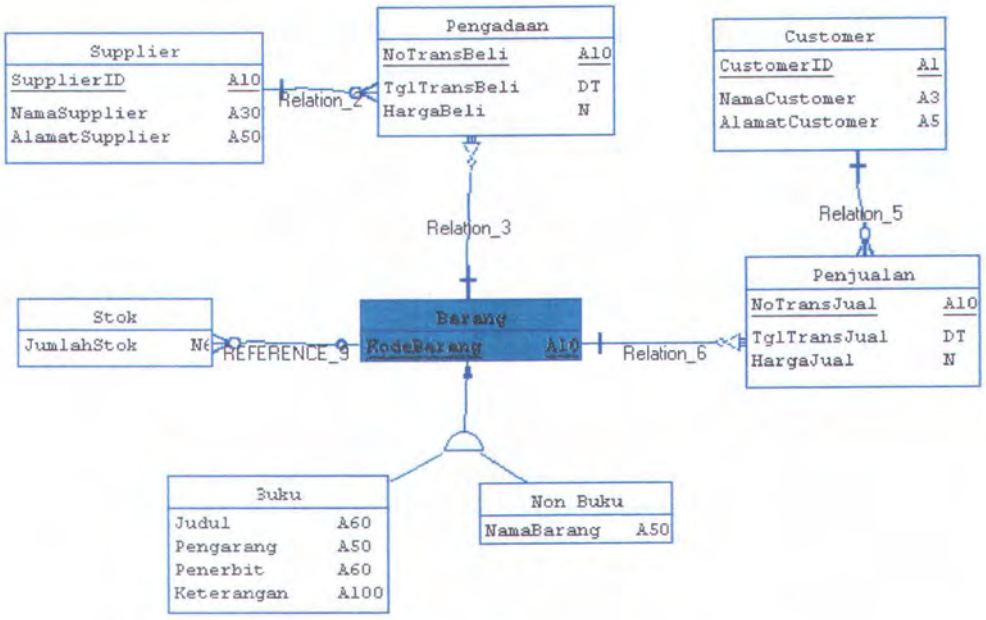
Gambar 6.8 Diagram Fisik hasil forward engineering

Pengguna 2 menambahkan tabel STOK pada model fisik diatas dan tampak pada diagram sebagai berikut.



Gambar 6.9 Diagram fisik setelah dimodifikasi

Dari diagram fisik yang telah berubah ini, pengguna 2 melakukan reverse engineering untuk menghasilkan model konseptual. Pengguna 2 kemudian meminta pengguna 1 untuk membuka model konseptual pada worksheet SKENARIO2. Model konseptual yang dibuka oleh pengguna 1 melalui perangkat lunak ini tampak sebagai berikut.



Gambar 6.10 Diagram Konseptual hasil reverse engineering

Dalam diagram diatas tampak bahwa model konseptual ini hampir sama dengan model konseptual pada waktu pertama kali dibuka dan hanya mendapat tambahan entitas Stok sesuai dengan perubahan yang dilakukan oleh pengguna 2 pada model fisik menggunakan PDM editor.

Dari ujicoba yang ketiga telah terbukti bahwa ketika hasil forward engineering yang mengalami perubahan berupa penambahan obyek tidak akan mempengaruhi obyek yang lain. Hal ini tampak pada obyek turunan yang pada proses forward engineering berubah menjadi referensi dan pada saat dilakukan

reverse engineering refersensi tersebut kembali berubah menjadi obyek turunan. Hal ini dikarenakan adanya informasi turunan yang dibawa oleh obyek referensi model fisik pada saat proses forward engineering. Sehingga pada saat proses reverse engineering dilakukan maka obyek refersensi yang mempunyai informasi turunan akan dikembalikan menjadi obyek turunan.

Pada ujicoba yang ketiga ini terbukti bahwa perangkat lunak yang dibuat mampu melakukan proses forward engineering. Begitu pula dengan fasilitas reverse engineering yang ditambahkan pada PDM Editor dapat bekerja dengan baik.

6.3.Analisa Hasil Ujicoba

Ujicoba dilakukan untuk menguji perangkat lunak yang sudah dibuat, apakah sudah sesuai dengan fitur yang telah dirancang atau tidak. Studi kasus pada ujicoba disesuaikan agar bisa menjawab pertanyaan-pertanyaan tersebut.

Pada ujicoba yang pertama yaitu menggambarkan diagram model basis data konseptual berjalan dengan baik. Diagram yang terbentuk dapat disimpan dalam file lokal dan dapat ditampilkan kembali kedalam bentuk diagram.

Pada ujicoba yang kedua, dimana diagram dibuka dengan mode multi user sehingga pengguna lain dapat mengakses atau *join* kedalam worksheet yang dibuat. Antar pengguna dapat melakukan komunikasi *chatting* dengan menggunakan fasilitas yang telah disediakan. Perubahan pada diagram yang dilakukan oleh satu pengguna dalam satu worksheet dapat diketahui atau dapat mengakibatkan perubahan yang serupa pada diagram pengguna yang lain dalam worksheet yang sama. Penghapusan obyek secara bersamaan akan menampilkan

pesan pemberitahuan akan adanya penghapusan obyek oleh pengguna yang lain. Penghapusan yang diproses oleh server adalah penghapusan yang dilakukan pertama kali. Sedangkan penghapusan oleh pengguna yang lain pada obyek yang sama sesudah penghapusan pertama akan diabaikan dan diberikan pesan sebagaimana diatas.

Pada ujicoba yang ketiga, dimana fitur yang diuji adalah forward dan reverse engineering. Hasil proses forward engineering dapat dibuka melalui PDM Editor yang merupakan perangkat lunak untuk memodelkan basis data fisik. Model fisik ini dapat diubah dan dilakukan proses reverse engineering dari model fisik menjadi model konseptual. Hasil yang didapatkan adalah model konseptual yang mirip dengan model konseptual sebelum dilakukan proses forward engineering, kecuali jika perubahan terjadi pada model fisik akan mengakibatkan model konseptual pada saat setelah dilakukan reverse engineering mengikuti perubahan yang terjadi.

Mengacu pada hasil ketiga uji coba tersebut, terbukti bahwa fitur-fitur yang telah dibuat memang yang diperlukan. Kombinasi faktor pemodelan basis data konseptual, pemodelan basis data fisik, sifat online realtime, konkurensi data, komunikasi dengan pengguna lain serta manajemen model merupakan salah satu solusi untuk membantu meningkatkan kinerja analis dalam merancang sebuah sistem informasi.

Kesimpulan akhir, bahwa dengan adanya perangkat lunak pemodelan basis data konseptual untuk RDBMS Oracle berbasis web ini dapat membantu proses desain basis data konseptual untuk nantinya digenerate manjadi model fisik

dengan RDBMS Oracle yang pada umumya dilakukan oleh perangkat lunak secara single user yang tidak dapat digunakan untuk mendesain jarak jauh.

BAB VII

PENUTUP

BAB VII

PENUTUP

7.1. Kesimpulan

Berdasarkan analisa hasil uji coba perangkat lunak maka dibuat kesimpulan sebagai berikut :

1. Perangkat lunak ini dapat digunakan untuk merancang sebuah diagram entity relationship secara konseptual berbasis web.
2. Arsitektur yang paling sesuai untuk perancangan dan pembuatan aplikasi pemodelan ini adalah arsitektur 3-*tier*. Untuk *User Interface Layer* digunakan ActiveX Control yang diimplementasikan dalam web browser. Di lingkungan server, komponen-komponen yang tercakup dalam *Business Logic Layer* adalah IIS dan aplikasi ISAPI DLL. Untuk *Database Layer* digunakan RDBMS Oracle dengan memakai koneksi ADO.
3. Forward engineering dari model konseptual ke model fisik dan reverse engineering dari model fisik ke model konseptual dilakukan dengan mengkonversi obyek diagram menjadi data XML serta menyimpan informasi obyek diagram tersebut dalam XML. Proses konversi ini dilakukan dengan menggunakan algoritma yang telah dijelaskan pada bab IV
4. Masalah konkurensi antar pengguna dapat diatasi dengan pengecekan terhadap setiap perubahan data yang dikirim. Pengecekan dilakukan pada aplikasi server dengan memeriksa apakah ada *update* data yang datang bersamaan. Jika

ada, maka *update* data yang belakangan masuk disesuaikan dengan *update* data yang sekarang valid. Aplikasi tidak memperbolehkan terjadi *update* data pada data yang sedang diproses.

7.2. Saran

Berdasarkan hasil evaluasi yang dilakukan terhadap sistem, ada beberapa saran yang perlu dipertimbangkan dalam pengembangan aplikasi ini, yaitu :

1. Proses forward engineering tidak hanya digunakan untuk *database* Oracle, namun lebih ke arah multi *database*.
2. Mengimplementasikan komponen-komponen skema *database* yang lebih detail yang belum tercakup dalam aplikasi ini.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. Microsoft Corporation, Microsoft Developer Network, www.msdn.microsoft.com, July 2001
2. Ramez Elmasri, Shamkant B. Navathe, Fundamentass Of Database Systems Third Edition, Addison-Wesley, 2000
3. Rezin Kristanto, Perancangan dan Pembuatan Perangkat Lunak Case Tool Untuk Melakukan Pemodelan Basis Data Fisik Untuk RDBMS Oracle Berbasis Web, FTIf, T. Informatika, Institut Teknologi Sepuluh Nopember Surabaya, 2003
4. David J. Kruglinski, Programming Microsoft Visual C++ Fifth Edition, Microsoft Press, 1998
5. Microsoft Corporation, Distributed Applications with Microsoft Visual Basic 6.0, Microsoft Press, 1999
6. PowerDesigner DataArchitect Help Topics, Sybase Inc. 1997
7. Oracle Corp, Oracle Documentation Library, Release 8.1.7, 2000
8. David Hunter, Beginning XML, Wrox
9. Terry Quatrani, Visual Modeling With Rational Rose And UML, Booch Jacobson Rumbaugh.

LAMPIRAN

CDM_INHID = CDM_INHID
CDM_INH_P_WORKID = CDM_INH_P_WORKID
CDM_INH_PARENT_ENTITYID = CDM_INH_PARENT_ENTITYID
CDM_INH_C_WORKID = CDM_INH_C_WORKID
CDM_INH_CHILD_ENTITYID = CDM_INH_CHILD_ENTITYID
CDM_INHCHILDID = CDM_INHCHILDID
USERS
NAME VARCHAR2(20) not null
PASSWORD VARCHAR2(20) null
NAME = NAME
WORKSHEETUSER
NAME VARCHAR2(20) not null
WORKSHEETID VARCHAR2(10) not null
PDM_REL_COLUMN
PDM_COL_C_WORKID VARCHAR2(10) not null
PDM_COL_C_TABLE_ID VARCHAR2(10) not null
PDM_COL_C_CODE VARCHAR2(50) not null
PDM_COL_P_WORKID VARCHAR2(10) not null
PDM_COL_P_TABLE_ID VARCHAR2(10) not null
PDM_COL_P_CODE VARCHAR2(50) not null
PDM_CHILD_WORKID VARCHAR2(10) not null
PDM_CHILD_TABLE_ID VARCHAR2(10) not null
PDM_PARENT_WORKID VARCHAR2(10) not null
PDM_PARENT_TABLE_ID VARCHAR2(10) not null
PDM_RELATION_ID VARCHAR2(10) not null
PDM_REL_COL_ID VARCHAR2(10) not null
PDM_COLUMN
WORKSHEETID VARCHAR2(10) not null
PDM_TABLE_ID VARCHAR2(10) not null
PDM_COLUMN_CODE VARCHAR2(50) not null
PDM_COLUMN_NAME VARCHAR2(50) null
PDM_COLUMN_TYPE VARCHAR2(30) null
PDM_COLUMN_PK NUMBER(1) null
PDM_COLUMN_FK NUMBER(1) null
PDM_COLUMN_ALLOWNULL NUMBER(1) null
PDM_COL_DESC VARCHAR2(1000) null
PDM_COL_ANNOTATION VARCHAR2(1000) null
CDM_INHERITANCE
CDM_INHID VARCHAR2(10) not null
WORKSHEETID VARCHAR2(10) not null
CDM_ENTITYID VARCHAR2(10) not null
INH_NAME VARCHAR2(50) null
INH_CODE VARCHAR2(50) null
INH_DESC VARCHAR2(1000) null
INH_ANNOTATION VARCHAR2(1000) null
INH_MUTUALLY NUMBER(1) null
VALUE_INH VARCHAR2(1000) null
CDM_INHID = CDM_INHID
RKSHEETID = CDM_INH_P_WORKID
NTITYID = CDM_INH_PARENT_ENTITYID
CDM_ID_LIST
WORKSHEETID VARCHAR2(10) not null
CDMIDLIST VARCHAR2(500) not null
ID = CDM_ENTITYID
WORKSHEETID = WORKSHEETID
CDM_INH_CHILD
CDM_INHID VARCHAR2(10) not null
CDM_INH_P_WORKID VARCHAR2(10) not null
CDM_INH_PARENT_ENTITYID VARCHAR2(10) not null
CDM_INH_C_WORKID VARCHAR2(10) not null
CDM_INH_CHILD_ENTITYID VARCHAR2(10) not null
CDM_INHCHILDID CHAR(10) not null
PDM_RELATION_ID = PDM_RELATION_ID
PDM_CHILD_WORKID = PDM_CHILD_WORKID
PDM_CHILD_TABLE_ID = PDM_CHILD_TABLE_ID
PDM_PARENT_WORKID = PDM_PARENT_WORKID
PDM_PARENT_TABLE_ID = PDM_PARENT_TABLE_ID
WORKSHEET
WORKSHEETID VARCHAR2(10) not null
PASSWORDW VARCHAR2(20) null
SHEETID = WORKSHEETID
WORKSHEETID = WORKSHEETID
WORKSHEETID = WORKSHEETID
WORKSHEETID = WORKSHEETID
PDM_RELATION
PDM_RELATION_ID VARCHAR2(10) not null
PDM_CHILD_WORKID VARCHAR2(10) not null
PDM_CHILD_TABLE_ID VARCHAR2(10) not null
PDM_PARENT_WORKID VARCHAR2(10) not null
PDM_PARENT_TABLE_ID VARCHAR2(10) not null
PDM_RELATION_NAME VARCHAR2(50) null
PDM_RELATION_CODE VARCHAR2(50) null
PDM_REL_PARENT_CODE VARCHAR2(50) null
PDM_REL_CHILD_CODE VARCHAR2(50) null
PDM_REL_CHILD_MIN NUMBER(3) null
PDM_REL_CHILD_MAX VARCHAR2(3) null
PDM_REL_PARENT_MAND NUMBER(1) null
PDM_REL_DESC VARCHAR2(1000) null
PDM_REL_ANNOTATION VARCHAR2(1000) null
CDM_REL_FLAG NUMBER(1) null
CDM_RELATION_ID VARCHAR2(10) null
CDM_CHILD_WORKID VARCHAR2(10) null
CDM_CHILD_ENTITYID VARCHAR2(10) null
CDM_PARENT_WORKID VARCHAR2(10) null
PDM_TABLE
PDM_TABLE_ID VARCHAR2(10) not null
WORKSHEETID VARCHAR2(10) not null
PDM_TABLE_NAME VARCHAR2(50) null
PDM_TABLE_CODE VARCHAR2(50) null
PDM_TABLE_PKCONSTRAINT VARCHAR2(50) null
PDM_TABLE_DESC VARCHAR2(1000) null
PDM_TABLE_ANNOTATION VARCHAR2(1000) null
PDM_TABLE_VALUE VARCHAR2(1000) null
CDM_TAB_FLAG NUMBER(1) null
CDM_WORKSHEETID VARCHAR2(10) null
CDM_ENTITYID VARCHAR2(10) null
CDM_REL_C_WORKID VARCHAR2(10) null
CDM_REL_C_ENTITYID VARCHAR2(10) null
CDM_ENTITY
CDM_ENTITYID VARCHAR2(10) not null
WORKSHEETID VARCHAR2(10) not null
ENT_NAME VARCHAR2(50) null
ENT_CODE VARCHAR2(50) null
ENT_DESC VARCHAR2(1000) null
ENT_ANNOTATION VARCHAR2(1000) null
VALUE_ENT VARCHAR2(1000) null
YID = CDM_INH_CHILD_ENTITYID
ETID = CDM_ENTITYID = CDM_ENTITYID
WORKSHEETID = WORKSHEETID
CDM_ENTITYID = CDM_PARENT_ENTITYID CDM_CHILD_ENTITYID
WORKSHEETID = CDM_PARENT_WORKID= CDM_CHILD_WORKID
CDM_RELATION
CDM_RELATION_ID VARCHAR2(10) not null
CDM_CHILD_WORKID VARCHAR2(10) not null
CDM_CHILD_ENTITYID VARCHAR2(10) not null
CDM_COLUMN
COLID VARCHAR2(10) not null
WORKSHEETID VARCHAR2(10) not null

LAMPIRAN B : Script Pembuatan Repository

```
-- ==============================================================
--   Database name:  REPOSITORY
--   DBMS name:      ORACLE Version 8
--   Created on:     5/9/2004  3:57 PM
-- ==============================================================

-- ==============================================================
--   Table: USERS
-- ==============================================================
create table USERS
(
    NAME                    VARCHAR2(20)          not null,
    PASSWORD                VARCHAR2(20)          null    ,
    constraint PK_USERS primary key (NAME)
)
/

-- ==============================================================
--   Table: WORKSHEET
-- ==============================================================
create table WORKSHEET
(
    WORKSHEETID             VARCHAR2(10)          not null,
    PASSWORDW               VARCHAR2(20)          null    ,
    constraint PK_WORKSHEET primary key (WORKSHEETID)
)
/

-- ==============================================================
--   Table: CDM_ENTITY
-- ==============================================================
create table CDM_ENTITY
(
    CDM_ENTITYID            VARCHAR2(10)          not null,
    WORKSHEETID             VARCHAR2(10)          not null,
    ENT_NAME                VARCHAR2(50)          null    ,
    ENT_CODE                VARCHAR2(50)          null    ,
    ENT_DESC                VARCHAR2(1000)        null    ,
    ENT_ANNOTATION          VARCHAR2(1000)        null    ,
    VALUE_ENT               VARCHAR2(1000)        null    ,
    constraint PK_CDM_ENTITY primary key (CDM_ENTITYID, WORKSHEETID)
)
/

-- ==============================================================
--   Index: CDM_ENTITY_FK
-- ==============================================================
create index CDM_ENTITY_FK on CDM_ENTITY (WORKSHEETID asc)
/

-- ==============================================================
--   Table: CDM_RELATION
-- ==============================================================
create table CDM_RELATION
(
    CDM_RELATION_ID         VARCHAR2(10)          not null,
    CDM_CHILD_WORKID        VARCHAR2(10)          not null,
```

```
    CDM_CHILD_ENTITYID       VARCHAR2(10)          not null,
    CDM_PARENT_WORKID        VARCHAR2(10)          not null,
    CDM_PARENT_ENTITYID      VARCHAR2(10)          not null,
    REL_NAME                 VARCHAR2(50)          null    ,
    REL_CODE                 VARCHAR2(50)          null    ,
    ORG_MANDATORY            NUMBER(1)             not null,
    ORG_DEPENDENT            NUMBER(1)             not null,
    ORG_DOMINANT             NUMBER(1)             null    ,
    DEST_MANDATORY           NUMBER(1)             null    ,
    DEST_DEPENDENT           NUMBER(1)             null    ,
    DEST_DOMINANT            NUMBER(1)             null    ,
    CARDINALITY              NUMBER(1)             not null
        constraint CKC_CARDINALITY_CDM_RELA check (
            CARDINALITY in (1,2,3,4)),
    REL_DESC                 VARCHAR2(1000)        null    ,
    REL_ANNOTATION           VARCHAR2(1000)        null    ,
    VALUE_REL                VARCHAR2(1000)        null    ,
    constraint   PK_CDM_RELATION   primary   key   (CDM_RELATION_ID,
CDM_CHILD_WORKID,       CDM_CHILD_ENTITYID,       CDM_PARENT_WORKID,
CDM_PARENT_ENTITYID)
)
/

-- ============================================================
--   Index: RELATION_ENTITY1_FK
-- ============================================================
create index RELATION_ENTITY1_FK on CDM_RELATION (CDM_PARENT_WORKID
asc, CDM_PARENT_ENTITYID asc)
/

-- ============================================================
--   Index: RELATION_ENTITY2_FK
-- ============================================================
create index RELATION_ENTITY2_FK on CDM_RELATION (CDM_CHILD_WORKID
asc, CDM_CHILD_ENTITYID asc)
/

-- ============================================================
--   Table: CDM_INH_CHILD
-- ============================================================
create table CDM_INH_CHILD
(
    CDM_INHID                VARCHAR2(10)          not null,
    CDM_INH_P_WORKID         VARCHAR2(10)          not null,
    CDM_INH_PARENT_ENTITYID  VARCHAR2(10)          not null,
    CDM_INH_C_WORKID         VARCHAR2(10)          not null,
    CDM_INH_CHILD_ENTITYID   VARCHAR2(10)          not null,
    CDM_INHCHILDID           CHAR(10)              not null,
    constraint    PK_CDM_INH_CHILD    primary    key    (CDM_INHID,
CDM_INH_P_WORKID,      CDM_INH_PARENT_ENTITYID,      CDM_INH_C_WORKID,
CDM_INH_CHILD_ENTITYID, CDM_INHCHILDID)
)
/

-- ============================================================
--   Index: INHCHILD_ENTITY_FK
-- ============================================================
create index INHCHILD_ENTITY_FK on CDM_INH_CHILD (CDM_INH_C_WORKID
asc, CDM_INH_CHILD_ENTITYID asc)
/

-- ============================================================
--   Index: INH_INHCHILD_FK
```

```
-- ============================================================
create index INH_INHCHILD_FK on CDM_INH_CHILD (CDM_INH_P_WORKID asc,
CDM_INH_PARENT_ENTITYID asc, CDM_INHID asc)
/

-- ============================================================
--   Table: CDM_INHERITANCE
-- ============================================================
create table CDM_INHERITANCE
(
    CDM_INHID               VARCHAR2(10)          not null,
    WORKSHEETID             VARCHAR2(10)          not null,
    CDM_ENTITYID            VARCHAR2(10)          not null,
    INH_NAME                VARCHAR2(50)          null     ,
    INH_CODE                VARCHAR2(50)          null     ,
    INH_DESC                VARCHAR2(1000)        null     ,
    INH_ANNOTATION          VARCHAR2(1000)        null     ,
    INH_MUTUALLY            NUMBER(1)             null     ,
    VALUE_INH               VARCHAR2(1000)        null     ,
    constraint PK_CDM_INHERITANCE primary key (CDM_INHID, WORKSHEETID,
CDM_ENTITYID)
)
/

-- ============================================================
--   Index: INHERITANCE_ENTITY_FK
-- ============================================================
create index INHERITANCE_ENTITY_FK on CDM_INHERITANCE (WORKSHEETID
asc, CDM_ENTITYID asc)
/

-- ============================================================
--   Table: PDM_TABLE
-- ============================================================
create table PDM_TABLE
(
    PDM_TABLE_ID            VARCHAR2(10)          not null,
    WORKSHEETID             VARCHAR2(10)          not null,
    PDM_TABLE_NAME          VARCHAR2(50)          null     ,
    PDM_TABLE_CODE          VARCHAR2(50)          null     ,
    PDM_TABLE_PKCONSTRAINT  VARCHAR2(50)          null     ,
    PDM_TABLE_DESC          VARCHAR2(1000)        null     ,
    PDM_TABLE_ANNOTATION    VARCHAR2(1000)        null     ,
    PDM_TABLE_VALUE         VARCHAR2(1000)        null     ,
    CDM_TAB_FLAG            NUMBER(1)             null     ,
    CDM_WORKSHEETID         VARCHAR2(10)          null     ,
    CDM_ENTITYID            VARCHAR2(10)          null     ,
    CDM_REL_C_WORKID        VARCHAR2(10)          null     ,
    CDM_REL_C_ENTITYID      VARCHAR2(10)          null     ,
    CDM_REL_P_WORKID        VARCHAR2(10)          null     ,
    CDM_REL_P_ENTITYID      VARCHAR2(10)          null     ,
    CDM_RELATION_ID         VARCHAR2(10)          null     ,
    constraint PK_PDM_TABLE primary key (PDM_TABLE_ID, WORKSHEETID)
)
/

-- ============================================================
--   Index: PDM_TABLE_WORKSHEET_FK
-- ============================================================
create index PDM_TABLE_WORKSHEET_FK on PDM_TABLE (WORKSHEETID asc)
/

-- ============================================================
```

```
--   Index: CDM_ENT_PDM_TAB_FK
-- ============================================================
create index CDM_ENT_PDM_TAB_FK on PDM_TABLE (CDM_WORKSHEETID asc,
CDM_ENTITYID asc)
/

-- ============================================================
--   Index: CDM_REL_PDM_TAB_FK
-- ============================================================
create index CDM_REL_PDM_TAB_FK on PDM_TABLE (CDM_REL_C_WORKID asc,
CDM_REL_C_ENTITYID asc, CDM_REL_P_WORKID asc, CDM_REL_P_ENTITYID asc,
CDM_RELATION_ID asc)
/

-- ============================================================
--   Table: PDM_COLUMN
-- ============================================================
create table PDM_COLUMN
(
    WORKSHEETID             VARCHAR2(10)          not null,
    PDM_TABLE_ID            VARCHAR2(10)          not null,
    PDM_COLUMN_CODE         VARCHAR2(50)          not null,
    PDM_COLUMN_NAME         VARCHAR2(50)          null    ,
    PDM_COLUMN_TYPE         VARCHAR2(30)          null    ,
    PDM_COLUMN_PK           NUMBER(1)             null    ,
    PDM_COLUMN_FK           NUMBER(1)             null    ,
    PDM_COLUMN_ALLOWNULL    NUMBER(1)             null    ,
    PDM_COL_DESC            VARCHAR2(1000)        null    ,
    PDM_COL_ANNOTATION      VARCHAR2(1000)        null    ,
    constraint PK_PDM_COLUMN primary key (WORKSHEETID, PDM_TABLE_ID,
PDM_COLUMN_CODE)
)
/

-- ============================================================
--   Index: PDM_TABLE_COL_FK
-- ============================================================
create index PDM_TABLE_COL_FK on PDM_COLUMN (WORKSHEETID asc,
PDM_TABLE_ID asc)
/

-- ============================================================
--   Table: PDM_RELATION
-- ============================================================
create table PDM_RELATION
(
    PDM_RELATION_ID         VARCHAR2(10)          not null,
    PDM_CHILD_WORKID        VARCHAR2(10)          not null,
    PDM_CHILD_TABLE_ID      VARCHAR2(10)          not null,
    PDM_PARENT_WORKID       VARCHAR2(10)          not null,
    PDM_PARENT_TABLE_ID     VARCHAR2(10)          not null,
    PDM_RELATION_NAME       VARCHAR2(50)          null    ,
    PDM_RELATION_CODE       VARCHAR2(50)          null    ,
    PDM_REL_PARENT_CODE     VARCHAR2(50)          null    ,
    PDM_REL_CHILD_CODE      VARCHAR2(50)          null    ,
    PDM_REL_CHILD_MIN       NUMBER(3)             null    ,
    PDM_REL_CHILD_MAX       VARCHAR2(3)           null    ,
    PDM_REL_PARENT_MAND     NUMBER(1)             null    ,
    PDM_REL_DESC            VARCHAR2(1000)        null    ,
    PDM_REL_ANNOTATION      VARCHAR2(1000)        null    ,
    CDM_REL_FLAG            NUMBER(1)             null
        constraint CKC_CDM_REL_FLAG_PDM_RELA check (
            CDM_REL_FLAG is null or (CDM_REL_FLAG in (1,2))),
```

```
    CDM_RELATION_ID          VARCHAR2(10)          null    ,
    CDM_CHILD_WORKID         VARCHAR2(10)          null    ,
    CDM_CHILD_ENTITYID       VARCHAR2(10)          null    ,
    CDM_PARENT_WORKID        VARCHAR2(10)          null    ,
    CDM_PARENT_ENTITYID      VARCHAR2(10)          null    ,
    CDM_INHID                VARCHAR2(10)          null    ,
    CDM_INH_P_WORKID         VARCHAR2(10)          null    ,
    CDM_INH_PARENT_ENTITYID  VARCHAR2(10)          null    ,
    CDM_INHCHILDID           CHAR(10)              null    ,
    CDM_INH_C_WORKID         VARCHAR2(10)          null    ,
    CDM_INH_CHILD_ENTITYID   VARCHAR2(10)          null    ,
    constraint   PK_PDM_RELATION   primary   key   (PDM_RELATION_ID,
PDM_CHILD_WORKID,        PDM_CHILD_TABLE_ID,        PDM_PARENT_WORKID,
PDM_PARENT_TABLE_ID)
)
/

-- ============================================================
--   Index: PDM_TABLE_RELATION2_FK
-- ============================================================
create index PDM_TABLE_RELATION2_FK on PDM_RELATION (PDM_PARENT_WORKID
asc, PDM_PARENT_TABLE_ID asc)
/

-- ============================================================
--   Index: PDM_TABLE_RELATION1_FK
-- ============================================================
create index PDM_TABLE_RELATION1_FK on PDM_RELATION (PDM_CHILD_WORKID
asc, PDM_CHILD_TABLE_ID asc)
/

-- ============================================================
--   Index: CDM_PDM_REL_FK
-- ============================================================
create index CDM_PDM_REL_FK on PDM_RELATION (CDM_CHILD_WORKID asc,
CDM_CHILD_ENTITYID asc, CDM_PARENT_WORKID asc, CDM_PARENT_ENTITYID
asc, CDM_RELATION_ID asc)
/

-- ============================================================
--   Index: INH_PDM_REL_FK
-- ============================================================
create index INH_PDM_REL_FK on PDM_RELATION (CDM_INH_P_WORKID asc,
CDM_INH_PARENT_ENTITYID asc, CDM_INHID asc, CDM_INH_C_WORKID asc,
CDM_INH_CHILD_ENTITYID asc, CDM_INHCHILDID asc)
/

-- ============================================================
--   Table: CDM_COLUMN
-- ============================================================
create table CDM_COLUMN
(
    COLID                    VARCHAR2(10)          not null,
    WORKSHEETID              VARCHAR2(10)          not null,
    CDM_ENTITYID             VARCHAR2(10)          not null,
    COL_NAME                 VARCHAR2(50)          null    ,
    COL_CODE                 VARCHAR2(50)          null    ,
    COL_TYPE                 VARCHAR2(30)          not null,
    PRIMARYKEY               NUMBER(1)             not null,
    ALLOWNULL                NUMBER(1)             not null,
    COL_DESC                 VARCHAR2(1000)        null    ,
    COL_ANNOTATION           VARCHAR2(1000)        null    ,
```

```
    constraint   PK_CDM_COLUMN   primary   key   (COLID,   WORKSHEETID,
CDM_ENTITYID)
)
/

-- ============================================================
--   Index: COL_ENTITY_FK
-- ============================================================
create   index   COL_ENTITY_FK   on   CDM_COLUMN   (WORKSHEETID   asc,
CDM_ENTITYID asc)
/

-- ============================================================
--   Table: PDM_REL_COLUMN
-- ============================================================
create table PDM_REL_COLUMN
(
    PDM_COL_C_WORKID          VARCHAR2(10)            not null,
    PDM_COL_C_TABLE_ID        VARCHAR2(10)            not null,
    PDM_COL_C_CODE            VARCHAR2(50)            not null,
    PDM_COL_P_WORKID          VARCHAR2(10)            not null,
    PDM_COL_P_TABLE_ID        VARCHAR2(10)            not null,
    PDM_COL_P_CODE            VARCHAR2(50)            not null,
    PDM_CHILD_WORKID          VARCHAR2(10)            not null,
    PDM_CHILD_TABLE_ID        VARCHAR2(10)            not null,
    PDM_PARENT_WORKID         VARCHAR2(10)            not null,
    PDM_PARENT_TABLE_ID       VARCHAR2(10)            not null,
    PDM_RELATION_ID           VARCHAR2(10)            not null,
    PDM_REL_COL_ID            VARCHAR2(10)            not null,
    constraint   PK_PDM_REL_COLUMN   primary   key   (PDM_COL_C_WORKID,
PDM_COL_C_TABLE_ID,          PDM_COL_C_CODE,          PDM_COL_P_WORKID,
PDM_COL_P_TABLE_ID,          PDM_COL_P_CODE,          PDM_CHILD_WORKID,
PDM_CHILD_TABLE_ID,       PDM_PARENT_WORKID,       PDM_PARENT_TABLE_ID,
PDM_RELATION_ID, PDM_REL_COL_ID)
)
/

-- ============================================================
--   Index: PDM_REL_REL_COLUMN_FK
-- ============================================================
create index PDM_REL_REL_COLUMN_FK on PDM_REL_COLUMN (PDM_CHILD_WORKID
asc,      PDM_CHILD_TABLE_ID      asc,      PDM_PARENT_WORKID      asc,
PDM_PARENT_TABLE_ID asc, PDM_RELATION_ID asc)
/

-- ============================================================
--   Index: PDM_REL_COL_COLUMN1_FK
-- ============================================================
create      index      PDM_REL_COL_COLUMN1_FK      on      PDM_REL_COLUMN
(PDM_COL_P_WORKID asc, PDM_COL_P_TABLE_ID asc, PDM_COL_P_CODE asc)
/

-- ============================================================
--   Index: PDM_REL_COL_COLUMN2_FK
-- ============================================================
create      index      PDM_REL_COL_COLUMN2_FK      on      PDM_REL_COLUMN
(PDM_COL_C_WORKID asc, PDM_COL_C_TABLE_ID asc, PDM_COL_C_CODE asc)
/

-- ============================================================
--   Table: CDM_ID_LIST
-- ============================================================
create table CDM_ID_LIST
```

```
(
    WORKSHEETID             VARCHAR2(10)          not null,
    CDMIDLIST               VARCHAR2(500)         not null,
    constraint PK_CDM_ID_LIST primary key (WORKSHEETID, CDMIDLIST)
)
/

-- ============================================================
--   Index: CDMIDLIST_WORKSHEET_FK
-- ============================================================
create index CDMIDLIST_WORKSHEET_FK on CDM_ID_LIST (WORKSHEETID asc)
/

-- ============================================================
--   Table: WORKSHEETUSER
-- ============================================================
create table WORKSHEETUSER
(
    NAME                    VARCHAR2(20)          not null,
    WORKSHEETID             VARCHAR2(10)          not null,
    constraint PK_WORKSHEETUSER primary key (NAME, WORKSHEETID)
)
/

-- ============================================================
--   Index: WORKSHEETUSER_FK2
-- ============================================================
create index WORKSHEETUSER_FK2 on WORKSHEETUSER (NAME asc)
/

-- ============================================================
--   Index: WORKSHEETUSER_FK
-- ============================================================
create index WORKSHEETUSER_FK on WORKSHEETUSER (WORKSHEETID asc)
/

alter table CDM_ENTITY
    add   constraint   FK_CDM_ENTI_CDM_ENTIT_WORKSHEE   foreign   key
(WORKSHEETID)
       references WORKSHEET (WORKSHEETID)
/

alter table CDM_RELATION
    add   constraint   FK_CDM_RELA_RELATION_CDM_ENTI1   foreign   key
(CDM_PARENT_ENTITYID, CDM_PARENT_WORKID)
       references CDM_ENTITY (CDM_ENTITYID, WORKSHEETID)
/

alter table CDM_RELATION
    add   constraint   FK_CDM_RELA_RELATION_CDM_ENTI2   foreign   key
(CDM_CHILD_ENTITYID, CDM_CHILD_WORKID)
       references CDM_ENTITY (CDM_ENTITYID, WORKSHEETID)
/

alter table CDM_INH_CHILD
    add   constraint   FK_CDM_INH__INHCHILD__CDM_ENTI   foreign   key
(CDM_INH_CHILD_ENTITYID, CDM_INH_C_WORKID)
       references CDM_ENTITY (CDM_ENTITYID, WORKSHEETID)
/

alter table CDM_INH_CHILD
    add   constraint   FK_CDM_INH__INH_INHCH_CDM_INHE   foreign   key
(CDM_INHID, CDM_INH_P_WORKID, CDM_INH_PARENT_ENTITYID)
```

```
        references      CDM_INHERITANCE      (CDM_INHID,      WORKSHEETID,
CDM_ENTITYID)
/

alter table CDM_INHERITANCE
    add    constraint    FK_CDM_INHE_INHERITAN_CDM_ENTI    foreign    key
(CDM_ENTITYID, WORKSHEETID)
       references CDM_ENTITY (CDM_ENTITYID, WORKSHEETID)
/

alter table PDM_TABLE
    add    constraint    FK_PDM_TABL_PDM_TABLE_WORKSHEE    foreign    key
(WORKSHEETID)
       references WORKSHEET (WORKSHEETID)
/

alter table PDM_TABLE
    add    constraint    FK_PDM_TABL_CDM_ENT_P_CDM_ENTI    foreign    key
(CDM_ENTITYID, CDM_WORKSHEETID)
       references CDM_ENTITY (CDM_ENTITYID, WORKSHEETID)
/

alter table PDM_TABLE
    add    constraint    FK_PDM_TABL_CDM_REL_P_CDM_RELA    foreign    key
(CDM_RELATION_ID,          CDM_REL_C_WORKID,          CDM_REL_C_ENTITYID,
CDM_REL_P_WORKID, CDM_REL_P_ENTITYID)
       references   CDM_RELATION   (CDM_RELATION_ID,   CDM_CHILD_WORKID,
CDM_CHILD_ENTITYID, CDM_PARENT_WORKID, CDM_PARENT_ENTITYID)
/

alter table PDM_COLUMN
    add    constraint    FK_PDM_COLU_PDM_TABLE_PDM_TABL    foreign    key
(PDM_TABLE_ID, WORKSHEETID)
       references PDM_TABLE (PDM_TABLE_ID, WORKSHEETID)
/

alter table PDM_RELATION
    add    constraint    FK_PDM_RELA_PDM_TABLE_PDM_TAB2    foreign    key
(PDM_PARENT_TABLE_ID, PDM_PARENT_WORKID)
       references PDM_TABLE (PDM_TABLE_ID, WORKSHEETID)
/

alter table PDM_RELATION
    add    constraint    FK_PDM_RELA_PDM_TABLE_PDM_TAB1    foreign    key
(PDM_CHILD_TABLE_ID, PDM_CHILD_WORKID)
       references PDM_TABLE (PDM_TABLE_ID, WORKSHEETID)
/

alter table PDM_RELATION
    add    constraint    FK_PDM_RELA_CDM_PDM_R_CDM_RELA    foreign    key
(CDM_RELATION_ID,          CDM_CHILD_WORKID,          CDM_CHILD_ENTITYID,
CDM_PARENT_WORKID, CDM_PARENT_ENTITYID)
       references   CDM_RELATION   (CDM_RELATION_ID,   CDM_CHILD_WORKID,
CDM_CHILD_ENTITYID, CDM_PARENT_WORKID, CDM_PARENT_ENTITYID)
/

alter table PDM_RELATION
    add    constraint    FK_PDM_RELA_INH_PDM_R_CDM_INH_    foreign    key
(CDM_INHID,          CDM_INH_P_WORKID,          CDM_INH_PARENT_ENTITYID,
CDM_INH_C_WORKID, CDM_INH_CHILD_ENTITYID, CDM_INHCHILDID)
       references    CDM_INH_CHILD    (CDM_INHID,    CDM_INH_P_WORKID,
CDM_INH_PARENT_ENTITYID,   CDM_INH_C_WORKID,   CDM_INH_CHILD_ENTITYID,
CDM_INHCHILDID)
```

```
/

alter table CDM_COLUMN
    add    constraint    FK_CDM_COLU_COL_ENTIT_CDM_ENTI    foreign    key
(CDM_ENTITYID, WORKSHEETID)
       references CDM_ENTITY (CDM_ENTITYID, WORKSHEETID)
/

alter table PDM_REL_COLUMN
    add    constraint    FK_PDM_REL__PDM_REL_R_PDM_RELA    foreign    key
(PDM_RELATION_ID,          PDM_CHILD_WORKID,          PDM_CHILD_TABLE_ID,
PDM_PARENT_WORKID, PDM_PARENT_TABLE_ID)
       references   PDM_RELATION   (PDM_RELATION_ID,   PDM_CHILD_WORKID,
PDM_CHILD_TABLE_ID, PDM_PARENT_WORKID, PDM_PARENT_TABLE_ID)
/

alter table PDM_REL_COLUMN
    add    constraint    FK_PDM_REL_PDM_REL_C_PDM_COLU1    foreign    key
(PDM_COL_P_WORKID, PDM_COL_P_TABLE_ID, PDM_COL_P_CODE)
       references      PDM_COLUMN      (WORKSHEETID,      PDM_TABLE_ID,
PDM_COLUMN_CODE)
/

alter table PDM_REL_COLUMN
    add    constraint    FK_PDM_REL_PDM_REL_C_PDM_COLU2    foreign    key
(PDM_COL_C_WORKID, PDM_COL_C_TABLE_ID, PDM_COL_C_CODE)
       references      PDM_COLUMN      (WORKSHEETID,      PDM_TABLE_ID,
PDM_COLUMN_CODE)
/

alter table CDM_ID_LIST
    add    constraint    FK_CDM_ID_L_CDMIDLIST_WORKSHEE    foreign    key
(WORKSHEETID)
       references WORKSHEET (WORKSHEETID)
/

alter table WORKSHEETUSER
    add constraint FK_WORKSHEE_WORKSHEET_USERS foreign key  (NAME)
       references USERS (NAME)
/

alter table WORKSHEETUSER
    add    constraint    FK_WORKSHEE_WORKSHEET_WORKSHEE    foreign    key
(WORKSHEETID)
       references WORKSHEET (WORKSHEETID)
/
```