PENERAPAN OBJECT ORIENTED DATABASE(db4o) DALAM PERANCANGAN OBJECT ORIENTED SOFTWARE ENGINEERING (STUDI KASUS : APLIKASI PENERJEMAH MULTI BAHASA)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

Disusun oleh :

Nama : Himawan Fajar Yudistiro
No. Mahasiswa : 06 523 136

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2011

LEMBAR PENGESAHAN KEASLIAN TUGAS AKHIR

Saya yang bertanda tangan dibawah ini:

Nama : Himawan Fajar Yudistiro

NIM : 06523136

Menyatakan bahwa, tugas akhir yang saya buat adalah asli, tidak meniru atau mencontoh hasil karya orang lain. Bila ternyata saya meniru, maka saya siap menerima segala sanksi yang ada.

Yogyakarta, 26 Mei 2011

(Himawan Fajar Y)

LEMBAR PENGESAHAN PEMBIMBING

PENERAPAN OBJECT ORIENTED DATABASE(db4o) DALAM PERANCANGAN OBJECT ORIENTED SOFTWARE ENGINEERING (STUDI KASUS : APLIKASI PENERJEMAH MULTI BAHASA)

TUGAS AKHIR

Oleh :

Nama : Himawan Fajar Yudistiro

No. Mahasiswa : 06523136

Yogyakarta, 26 Mei 2011

Menyetujui,

Pembimbing Tugas Akhir

(R. Teduh Dirgahayu, S.T., M.Sc, Ph.D.)

LEMBAR PENGESAHAN PEMBIMBING

PENERAPAN OBJECT ORIENTED DATABASE(db4o) DALAM PERANCANGAN OBJECT ORIENTED SOFTWARE ENGINEERING (STUDI KASUS : APLIKASI PENERJEMAH MULTI BAHASA)

TUGAS AKHIR

Oleh :

Nama : Himawan Fajar Yudistiro

No. Mahasiswa : 06523136

Yogyakarta, 26 Mei 2011

Menyetujui,

Pembimbing Tugas Akhir

(R. Teduh Dirgahayu, S.T., M.Sc, Ph.D.)

LEMBAR PENGESAHAN PENGUJI

PENERAPAN OBJECT ORIENTED DATABASE(db4o) DALAM PERANCANGAN OBJECT ORIENTED SOFTWARE ENGINEERING (STUDI KASUS : APLIKASI PENERJEMAH MULTI BAHASA)

TUGAS AKHIR

Disusun Oleh :

Nama : Himawan Fajar Yudistiro

NIM : 06 523 136

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelara Sarjana Teknik Informatika Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 26 Mei 2011

Tim Penguji

R. Teduh Dirgahayu, S.T., M.Sc, Ph.D. _____

Ketua

Zainudin Zukhri, S.T., MIT. _____

Anggota 1

Hendrik, S.T., M.Eng. _____

Anggota 2

Mengetahui ,

Kepala Jurusan Teknik Informatika

Universitas Islam Indonesia

Yudi Prayudi, S.Si, M.Kom

LEMBAR PENGESAHAN PENGUJI

PENERAPAN OBJECT ORIENTED DATABASE(db4o) DALAM PERANCANGAN OBJECT ORIENTED SOFTWARE ENGINEERING (STUDI KASUS : APLIKASI PENERJEMAH MULTI BAHASA)

TUGAS AKHIR

Disusun Oleh :

Nama : Himawan Fajar Yudistiro

NIM : 06 523 136

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelara Sarjana Teknik Informatika Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 26 Mei 2011

Tim Penguji

R. Teduh Dirgahayu, S.T., M.Sc, Ph.D.

Ketua

Zainudin Zukhri, S.T., MIT.

Anggota 1

Hendrik, S.T., M.Eng.

Anggota 2

Mengetahui ,

Kepala Jurusan Teknik Informatika

Universitas Islam Indonesia

Yudi Prayudi, S.Si, M.Kom

HALAMAN PERSEMBAHAN

- Allah SWT, Terima Kasih sebesar-besarny atas Rahmat dan Hidayah-Nya sehingga Tugas Akhir ini dapat selesei dengan tepat waktu.
- Pak Teduh, selaku Dosen Pembimbing.... Terima kasih telah membimbing, menasehati dan mengarahkan sehingga Tugas Akhir ini dapat terseleseikan.
- Bapak, Ibu dan Adekku... makasih selalu memberikan doa dan semangatnya... selalu mengingatkan buat cepet seleseiin ngerjain TA. nya.....
- Anak Inf'06 yang tidak dapat aq sebutin satu per satu... sukses buat kita semua. Amin

MOTTO

" Orang- orang yang beriman dan hati mereka menjadi tentram dengan mengingat Allah, Ingatlah hanya dengan mengingat Allah hati mereka menjadi tentram. "

(QS : Ar-Ra'd : 28)

"..... sesungguhnya setelah kesulitan tersimpan sebuah kemudahan"

(QS. Al Insyiroh : 6)

"... Allah akan meninggikan orang beriman diantaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat ...."

(QS.Al-Mujaadilah ayat 11)

" Sesungguhnya sesudah kesulitan itu ada kemudahan ; Maka apabila kamu telah selesai (dari suatu urusan), kerjakanlah dengan sungguh-sungguh (urusan) yang lain ".

(Q.S. Alam Nasyrah ayat 6 dan7).

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah penulis panjatkan puji syukur kehadirat Allah SWT karena dengan rahmat, taufik serta hidayah-Nya penulis dapat menyelesaikan laporan Tugas Akhir ini dengan judul " **Penerapan Object Oriented Database(db4o) dalam perancangan Object Oriented Software Engineering(Studi kasus : Aplikasi Penerjemah Multi Bahasa).**" dengan baik. Shalawat dan salam semoga tercurahkan atas junjungan kita Nabi Muhammad SAW, sahabat serta para pengikutnya.

Laporan tugas akhir ini disusun guna melengkapi persyaratan dalam menyelesaikan program studi Strata-1 jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta.

Dengan terselesaikannya laporan tugas akhir ini, penulis mengucapkan terima kasih kepada :

1. Bapak Ir. Gumbolo Hadi Susanto.,M.Sc. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Yudi Prayudi, S.Si., M.Kom. selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Teduh Dirgahayu, S.T., M.Sc, Ph.D. selaku Dosen Pembimbing yang telah membimbing dan memberi motivasi kepada penulis untuk menyelesaikan tugas akhir ini.
4. Kedua Orang Tua dan Saudara, yang telah memberikan doa, dukungan dan motivasi untuk menyelesaikan laporan Tugas Akhir ini..
5. Teman-teman Informatika 2006 dan teman-teman yang tidak dapat disebutkan satu-persatu, terima kasih atas dorongan, bantuan dan motivasinya.

Penulis menyadari bahwa penyusunan laporan tugas akhir ini masih terdapat kekurangan karena keterbatasan ilmu dan pengetahuan yang penulis miliki. Oleh

karena itu, penulis sangat mengharapkan saran dan kritik yang membangun demi kesempurnaan tugas selanjutnya.

Akhir kata, penulis berharap semoga laporan tugas akhir ini dapat mendatangkan manfaat bagi penulis dan dapat menjadi referensi dikemudian hari bagi para pembaca serta mendatangkan ridha dari Allah SWT. Amin.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 26 Mei 2011

Penulis

SARI

Pengembangan basis data berorientasi objek(*OODB*) dilatarbelakangi oleh beralihnya kebutuhan akan pengembangan perangkat lunak dari penggunaan konsep prosedural atau modular ke konsep *object oriented programming*(*OOP*). Dikarenakan *OOP* telah berjasa menyelesaikan banyak persoalan dan mampu menunjukan kehandalan dalam dunia komputasi, *OOP* memberikan inspirasi kepada para pengembang untuk mengimplementasikan *OOP* dalam lingkup ruang pengelolaan basis data. Riset *OODB* dan pengembangan *ODBMS* masih berlanjut sampai sekarang ini guna meningkatkan kehandalan, keunggulan, dan prospek cerah pada dunia komputasi masa depan.

Aplikasi media penerapan konsep *OODB* adalah *Aplikasi Penerjemah Multi Bahasa*. Metodologi perancangan *Aplikasi Penerjemah Multi Bahasa* menggunakan metode analisis sistem dan perancangan sistem. Metode analisis sistem mendefinisikan kebutuhan proses aplikasi dan metode perancangan sistem menjelaskan disain sistem dan disain antarmuka aplikasi. Setelah metodologi perancangan aplikasi selesai, dilakukan pengembangan *Aplikasi Penerjemah Multi Bahasa*. Pengembangan aplikasi memuat implementasi sistem dan pengujian sistem. Implementasi sistem menjelaskan implementasi topologi aplikasi, implementasi *ODBMS db4o* terkait proses aplikasi, dan implementasi antarmuka aplikasi. Selanjutnya adalah pengujian aplikasi yaitu menguji feleksibilitas implementasi odbms *db4o* dan eksekusi proses-proses pada aplikasi.

Kata Kunci : OODB, OOP, ODBMS, db4o, Aplikasi Penerjemah Multi Bahasa

TAKARIR

Activity	Aktivitas
Client	Klien atau cabang
Collections	Koleksi atau Sekumpulan Objek
Computational Linguistic	Bidang Ilmu Komputer Sebagai Penerjemah Bahasa
Embedded	Tertanam
Graphical user interface	Antarmuka Pengguna Grafis
Layer Of Service	Lapisan Komponen Layanan Perangkat Lunak
Nosql	Basis Data Tanpa Menggunakan Teknologi Sql
ODMG	Kelompok Manajemen Data Obyek
Open Source	Sumber Terbuka
ORMS	Sistem Pemetaan Obyek-Relasional
Package	Paket
Persistence	Persistensi
Refactoring	Perbaikan Disain Pada Implementasi
Relationships	Keterhubungan
Sequence	Urutan
Server	Pusat
Small Footprints	Hemat Penggunaan Memori
Software	Perangkat Lunak
Stored Procedure	Prosedur tersimpan
Temporary	Sementara
Transaction	Transaksi
UML	Bahasa Pemodelan Terpadu
Views	Tampilan
Zero Administration	Tanpa Administrasi Basis Data

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Pengembangan ranah basis data tanpa teknologi sql(*not only sql, nosql)* dengan kelompok basis data berorientasi objek *(object oriented database*, *OODB*) dilatar belakangi oleh beralihnya pengembangan perangkat lunak dari penggunaan konsep prosedural atau modular ke konsep pemrograman berorientasi objek (*object oriented programming*, *OOP*). Dikarenakan OOP telah berjasa menyelesaikan banyak persoalan dan mampu menunjukan kehandalan dalam dunia komputasi, OOP memberikan inspirasi kepada para pengembang untuk mengimplementasikan OOP dalam lingkup ruang pengelolaan basis data. Riset OODB dan pengembangan sistem manajemen basis data berorientasi objek*(object oriented database management system, ODBMS)* masih berlanjut sampai sekarang ini guna meningkatkan kehandalan, keunggulan, dan prospek cerah pada dunia komputasi masa depan.

Salah satu vendor pengembang OODB *open source* adalah db4o dari Perusahaan Versant. db4o memiliki beberapa kemampuan dasar yang dimiliki oleh setiap sistem manajemen basis data *(database management system, DBMS)* pada umumnya, seperti mendukung multi bahasa pemrograman, *transaction, stored procedure ,views*, dan lain-lain. Sedangkan kemampuan lebih db4o antara lain tidak membutuhkan file skema sehingga tanpa perlu repotkan dengan konfigurasi, tidak membutuhkan proses pemetaan basis data yang kompleks, arsitektur aplikasi yang sederhana, dan mudah dalam mendefinisikan objek basis data(*table*, views, *procedure*).

Penelitian ini melakukan penerapan konsep *OODB* dalam Aplikasi Penerjemah Multi Bahasa. Alasan pemilihan studi kasus ini dikarenakan

spesifikasi jenis aplikasi tepat dan cocok sebagai penerapan *ODBMS*. Menurut Grehan(2005), terdapat beberapa poin dimana *ODBMS* dapat memberikan solusi terbaik pada aplikasi tersebut, yaitu Aplikasi basis data tertanam, kompleksitas relasi data, struktur objek, terdapat penggantian struktur objek/data, terdapat sekumpulan objek, dan Pemrograman berorientasi objek. Berdasarkan tipe, aplikasi ini tergololong perangkat lunak mesin penerjemah yang seharusnya fleksibel dalam penggunaanya. DBMS db4o dipertimbangkan mampu mewujudkan seutuhnya sifat perangkat lunak yaitu fleksibel pada aplikasi ini.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah bagaimana menerapkan konsep basis data berorientasi objek yang didukung oleh ODBMS db4o pada rekayasa perangkat lunak Aplikasi Penerjemah Multi Bahasa yang bersifat fleksibel dalam pengembangan, konfigurasi, dan penggunaanya.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah.

1. Pengembangan aplikasi ini dikembangkan dengan menggunakan metodologi berorientasi objek murni tanpa menggunakan konversi pemetaan objek relasional antar komponen basis data dan aplikasi.
2. Aplikasi ini menggunakan pustaka antarmuka pemrograman aplikasi pihak ketiga untuk menerjemahkan teks dari bahasa satu ke bahasa yang lain, dengan segala kelebihan dan keterbatasan fitur yang ada.
3. Kebenaran hasil terjemahan bahasa dari aplikasi ini tergantung pada pustaka pihak ketiga yang digunakan pada pengembangan aplikasi ini.
4. Aplikasi ini mengeksploitasi beberapa kemampuan dari dbms db4o yaitu Transaction untuk fitur menyimpan beberapa terjemahan dan Embedded untuk fitur basis data tertanam permanen pada aplikasi.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah menjelaskan suatu alternatif metodologi baru dalam dunia rekayasa perangkat lunak yaitu *Pure Object Oriented Software Engineering* dimana perancangan serta implementasinya menggunakan semua komponen berorientasi objek.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut :

1. Memudahkan perancangan dan implementasi perangkat lunak mesin penerjemah.
2. Mensosialisasikan sebuah paradigma baru dalam dunia basis data *nosql* untuk kelompok berorientasi objek sehingga tidak selalu berorientasi *SQL(Structured Query Language).*
3. Menjadi acuan dan turut memberi andil dalam mengembangkan riset serta mendayagunakan basis data berorientasi objek.

1.6 Metodologi Penelitian

Metodologi yang digunakan dalam penelitian ini meliputi metode pengumpulan data dan pengembangan sistem.

1.6.1 Metode pengumpulan data

Metode ini terdiri dua, yaitu:

1. Studi Pustaka

   Cara ini dilakukan dengan mengumpulkan data dari buku dan literatur-literatur terkait tentang sitem manajemen basis data db4o baik perancangannya pada sitem serta penerapannya dalam sistem.
2. Observasi

   Observasi dilakukan melalui uji coba aplikasi penerjemah bahasa lain yang sudah ada baik fitur-fitur yang ditawarkan, antarmuka aplikasi, dan

kemudahan penggunaan aplikasi. Hal ini bertujuan agar dapat merekayasa sebuah aplikasi penerjemah bahasa dengan kemampuan lebih baik dari pada aplikasi sejenis yang sudah ada.

1.6.2 Metode pengembangan perangkat lunak

Metode ini terdiri dari empat, yaitu:

1. Analisis Sistem

   Pada tahap ini dilakukan sebuah pengamatan terhadap permasalahan pemilihan metodologi yang akan dipakai dalam rekayasa perangkat lunak ini. Hal ini dilakukan dengan harapan perancangan dan implementasi sistem tidak melenceng dari tujuan penelitian.
2. Perancangan Sistem

   Tahap kedua adalah perancangan perangkat lunak yang akan dibangun. Perancangan perangkat lunak ini menggunakan pendekatan penerapan basis data berorientasi objek, analisis kebutuhan sistem (input, proses, dan output), pemodelan UML*(Unified Modeling Languange*), dan disain antarmuka sistem.
3. Implementasi Sistem

   Tahap ketiga adalah implementasi yang dihasilkan pada tahap kedua yaitu menterjemahkan diagram UML yang sudah dirancang dalam bentuk *coding* dan implementasi disain antarmuka sistem dalam bentuk GUI(*graphical user interface).* Selain itu, disertakan juga pemilihan kakas pengembang perangkat lunak yang akan digunakan dalam merekayasa aplikasi penerjemah bahasa dan pustaka pihak ketiga apa saja yang akan dimasukkan sebagai kakas pendukung dalam rekayasa perangkat lunak ini.
4. Pengujian Sistem

   Pada tahap terakhir ini dilakukan pengujian sistem yang dihasilkan. Pengujian ini harus membuktikan bahwa aplikasi diimplementasikan dengan metode dan kakas yang sesuai dan mampu menjawab permasalahan yang dihadapi.

1.7 Sistematika Penulisan

Sistematika penulisan menggambarkan secara singkat isi laporan pada tiap-tiap bagian sehingga laporan menjadi utuh dan jelas.

1. BAB I Pendahuluan, bab ini membahas tentang latar belakang masalah, rumusan masalah, dan batasan masalah yang memuat perkembangan basis data berorientasi objek, pemilihan vendor basis data berorientasi objek, pemilihan studi kasus, dan batasan penerapan basis data berorientasi objek. Pada bab ini juga membahas penelitian terkait penerapan basis data berorientasi objek pada studi kasus mulai dari tujuan penerapan, manfaat penerapan, dan penggunaan metode pada penerapan.
2. BAB II Landasan Teori, bab ini memuat dasar teori untuk memahami permasalahan yang berkaitan dengan basis data berorientasi objek murni yang digunakan pada rekayasa perangkat lunak berorientasi objek studi kasus aplikasi penerjemah multi bahasa.
3. BAB III Metodologi, bab ini memuat uraian tentang analisis kebutuhan perangkat lunak dan perancangan Aplikasi Penerjemah Multi Bahasa, antara lain kebutuhan proses Aplikasi Penerjemah Multi Bahasa, pemodelan UML Aplikasi Penerjemah Multi Bahasa, dan disain antarmuka Aplikasi Penerjemah Multi Bahasa.
4. BAB IV Hasil dan Pembahasan, bab ini memuat uraian implementasi dan pengujian basis data berorientasi objek pada Aplikasi Penerjemah Multi Bahasa. Basis data berorientasi objek diimplementasikan dengan menerjemahkan analisis kebutuhan perangkat lunak yang terdapat pada Bab III dan dilanjutkan dengan uraian pengujian Aplikasi Penerjemah Multi Bahasa dimana hasilnya mampu menjawab permasalahan yang dihadapi.
5. BAB V Kesimpulan dan Saran, bab ini menguraikan kesimpulan berupa rangkuman dari hasil pengujian penerapan basis data berorientasi objek pada Aplikasi Penerjemah Multi Bahasa dan mengemukakan beberapa saran pengembangan penelitian basis data *nosql*.

BAB II

LANDASAN TEORI

2.1 Rekayasa Perangkat Lunak Berorientasi Objek.

Rekayasa perangkat lunak berorientasi objek dibutuhkan banyak pemahaman tentang sebuah disain sistem berorientasi objek yang mempunyai efek pada faktor kualitas. Sifat ini harus mewakili sebuah karakterisitik sistem agar bisa dimengerti, dapat dianalisis, dapat diperpanjang, dan produk perangkat lunak tersebut dapat dipelihara.

Menurut Abreu, Poels, Sahraoui, dan Zuse(2003) beberapa kunci yang berhubungan pada kualitas penilaian sebuah sistem berorientasi objek adalah :

1. Pentingnya menentukan tindakan sebuah karakteristik sistem yang dikembangkan.
2. Adanya pemahaman pada sebuah karakteristik sistem yang dikembangkan.
3. Pentingnya menyediakan sebuah mekanisme mengetahui dan mengukur sebuah karakteristik sistem.
4. Sebuah mekanisme pengukuran sebuah sistem harus objektif.

Walaupun kualitas perangkat lunak tidak mudah untuk dievaluasi, misal efisiensi dan pemeliharaan, beberapa sifat sistem berorientasi objek harus diakomodasi. Matrik perangkat lunak biasa (tidak berorientasi objek) dapat dipakai untuk orientasi objek, tetapi matrik tersebut tidak memadai untuk mengukur semua sifat orientasi objek yang spesifik. Matrik tersebut harus diperbanyak agar dapat meliputi pengukuran perangkat lunak berorientasi objek. Masalah ini karena adanya sebuah tambahan sifat yang melekat pada paradigma orientasi objek seperti abstraksi, penurunan sifat, polymorfisme. Konsep baru ini sangatlah penting dalam sebuah pembangunan perangkat lunak yang fleksibel, dapat beradaptasi, dan dapat digunakan kembali.

2.1.1 Arsitektur Perangkat Lunak Berorientasi Objek.

Arsitektur perangkat lunak menentukan apa yang dapat dilakukan dan yang tidak dapat dilakukan oleh perangkat lunak tersebut. Arsitektur perangkat lunak juga mendeskripsikan solusi logis secara menyeluruh dari perangkat lunak yang hendak dibangun. Jika ada pertimbangan bahwa sistem terdiri atas perangkat lunak dan hardware, maka arsitektur sistem merupakan bagian integral dari arsitektur perangkat lunak yang tidak terpisahkan.

Biasanya arsitektur sebuah perangkat lunak terdiri dari beberapa *layer of service*. Oleh karena itu, arsitektur tersebut dinamakan *layered architecture.* Setiap layer biasanya berkorespondensi dengan satu atau lebih subsistem. *Layered architecture* terdiri atas dua jenis layer, yaitu :

a. *Closed-layered architecture.*
   Jenis arsitektur layer yang meminimalisasi ketergantungan antara layer-layer dengan cara setiap layer yang berada di atas hanya menggunakan layanan dari sebuah layer yang tepat berada di bawahnya.

b. *Open-layered architecture.*
   Jenis arsitektur layer yang mengijinkan sbuah layer menggunakan secara langsung semua *service* dari semua layer yang berada di bawahnya. Arsitektur Open-layered digunakan dalam rekayasa Aplikasi Penerjemah Multi Bahasa.

Arsitektur sebuah perangkat lunak sebaiknya tidak terdiri dari *layer of service* yang terlalu banyak ataupun terlalu sedikit. Apabila sebuah perangkat lunak memiliki arsitektur *layer of service* yang terlalu banyak, maka arsitektur perangkat lunaknya menjadi tidak efisien. Sementara itu, apabila sebuah perangkat lunak memiliki arsitektur *layer of service* yang terlalu sedikit, maka struktur dari perangkat lunak tersebut menjadi buruk.

Menurut Djon Irwanto(2006) perangkat lunak berorientasi objek sedikitnya terdiri atas tiga *layer of service*, yaitu :

a. *Interface layer* adalah *layer of service* yang memungkinkan actor dapat berinteraksi dengan sistem. Interface layer merupakan komponen sistem yang menyediakan *domain* model bagi actor.
b. *Domain* model adalah layer of service yang mengimplementasikan *problem domain model* dan kebutuhan fungsionalitas sistem.
c. *Object database* access layer adalah *layer of service* untuk mengakses basis data objek.

Tiga layer of service tersebut adalah komponen-komponen yang sekurang-kurangnya harus ada pada sebuah *object oriented software system*.

2.1.2 Pengembangan Perangkat Lunak Berorientasi Objek.

Menurut Djon Irwanto(2005) ketika membangun perangkat lunak atau merancang aristektur perangkat lunak dengan pendekatan berorientasi objek, terkadang kita menghadapi sebuah permasalahan dimana komponen basis data yang merupakan bagian integral masih menggunakan *relational data model*. Membangun *object oriented software system* di atas *RDBMS* sangatlah berbeda dibandingkan dengan *ODBMS*. Hal tersebut disebabkan *RDBMS* selalu menghasilkan sekumpulan record terhadap proses *query* dari sistem yang berada diatasnya, bukan sekumpulan objek(*collection*).

Para pengembang perangkat lunak berorientasi objek sudah biasa dengan kesulitan mengubah pemikiran berorientasi objek ke *relational persistence.* Sejauh ini, mereka terpaksa memilih antara kecepatan dan orientasi objek. Akses *SQL* dasar sangat cepat, tetapi membutuhkan banyak tambahan kode. Pemetaan objek relasional menawarkan kesesuaian, tetapi menurunkan performa secara serius.

Sebuah solusi yang berbeda dapat menyelesaikan keterpaksaan pada kedua perbedaan ini. Contohnya, objek *database* memudahkan pembangunan dengan

mempertimbangkan virtualisasi semua aspek dari persistence dan membebaskan pengembang untuk lebih berkonsentrasi pada aspek pemodelan bisnis.

2.2 Sistem Basis Data Berorientasi Objek.

Menurut Grehan(2005), sebuah sistem manajemen basis data objek (ODBMS) atau juga disebut sebagai sistem manajemen basis data berorientasi objek *(object-oriented database management system, OODBMS)*, adalah sistem manajemen basis data yang mendukung pemodelan dan pembuatan data sebagai objek, termasuk dukungan untuk kelas objek dan pewarisan dari properti kelas dan metode oleh sub kelas dan objek tersebut.

Malcolm Atkinson mendefinisikan pada tahun 1995 bahwa sebuah basis data berorientasi objek harus memenuhi dua kriteria: harus berupa ODBMS, dan harus berupa sistem berorientasi objek, yaitu, sedapat mungkin konsisten dengan bahasa pemrograman berorientasi objek. Kriteria pertama diterjemahkan menjadi lima fitur: persistensi, manajemen penyimpanan sekunder, konkurensi, pemulihan dan fasilitas query. Kriteria kedua diterjemahkan ke dalam delapan fitur: kompkesitas objek, identitas objek, enkapsulasi, tipe atau kelas, warisan, overiding dikombinasikan dengan pengikatan, perpanjangan, dan kelengkapan komputasi.

Saat ini tidak ada kesepakatan standar untuk ODBMS dan bahasa query standar untuk ODBMS sebagaimana SQL untuk RDBMS (DBMS relasional). Inisiatif kelompok industri, Object Data Management Group (ODMG), untuk membuat standar Objek *Query Language* (OQL) telah ditinggalkan pada tahun 2001. Sedangkan penelitian William Cook pada tahun 2005 menyarankan untuk menggunakan bahasa pemrograman itu sendiri, misalnya Java atau NET.

ODBMS pada awalnya dimaksud sebagai solusi untuk mengganti *RDBMS* karena mereka lebih baik dan sesuai dengan bahasa pemrograman berorientasi objek. Namun, biaya perpindahan yang tinggi, masuknya fitur berorientasi objek di RDBMS yang menjadikan ORDBMS, dan munculnya pemetaan objek-

relasional (object-relational mapping, ORM) membuat RDBMS berhasil mempertahankan dominasinya di pusat data.

OODB sekarang ditetapkan sebagai pelengkap, bukan pengganti untuk basis data relasional. OODB digunakan sebagai solusi dalam perangkat lunak embedded. Komunitas open source telah menciptakan sebuah gelombang baru sekarang antusiasme yang memicu pesatnya pertumbuhan ODBMS.

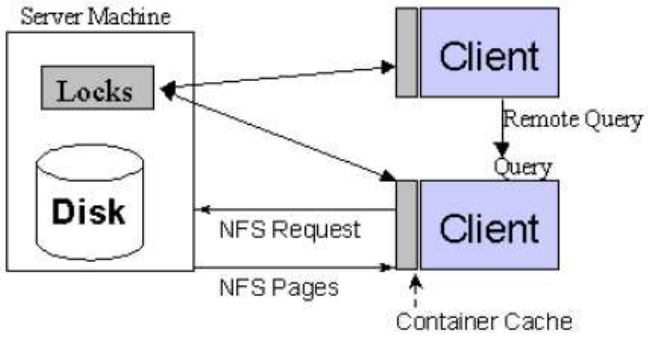
2.2.1 Arsitektur Basis Data Berorientasi Objek.

Menurut Greene(2006), arsitektur OODB dan harapan pengguna telah tercampur. Arsitektur OODB bervariasi dan menunjukkan karakteristik yang sangat berbeda dari arsitektur basis data relasional. Jika OODB akan digunakan dan arsitekturnya tidak cocok untuk kebutuhan aplikasi maka orang akan menyimpulkan bahwa tidak ada OODB yang dapat memenuhi kebutuhan aplikasi. Selain itu, ada kesalahpahaman persepsi tentang OODB seperti OODB terlalu lambat, OODB tidak menangani konkurensi tinggi, dan OODB tidak berskala besar. Pengembang perlu berhati-hati mempertimbangkan karakteristik aplikasi dan memahami arsitektur OODB yang paling cocok untuk memenuhi kebutuhan aplikasi tersebut. Arsitektur OODB yang tepat dapat memberi perbedaan besar dalam karakteristik kinerja dan skalabilitas dibandingkan dalam implementasi menggunakan basis data relasional.

Terdapat beberapa jenis arsitektur OODB yang biasa digunakan, antara lain:

1. **Berbasis Container**

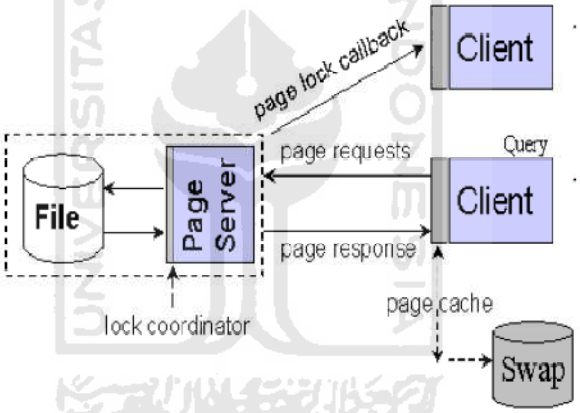
   Arsitektur berbasis container bertumpu pada suatu *client* yang menggunakan standar NFS dalam mengelola penyimpanan pada *server* dan mengirimkannya ke beberapa *client.* Gambar 2.1 menunjukan sebuah arsitektur berbasis container yang disainnya berpusat pada *client.*



Gambar 2. 1 Arsitektur Berbasis Container

2. Berbasis Page

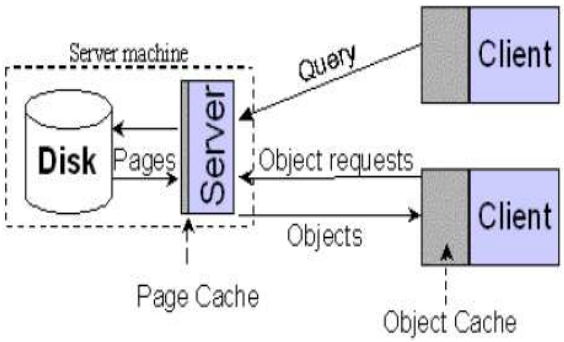
Gambar 2.2 menunjukkan sebuah arsitektur berorientasi pada *client* yang menjalankan proses *server* untuk memenuhi permintaan *page.*



Gambar 2. 2 Arsitektur Berbasis Page

3. Berbasis Objek

Gambar 2.3 menunjukkan sebuah arsitektur menjelaskan sebuah disain seimbang antara fungsi aplikasi *client* dan *server*. *Server* menjalankan fungsi *request page* ke penyimpanan dan *client* menjankan fungsi *object request* ke *server*.

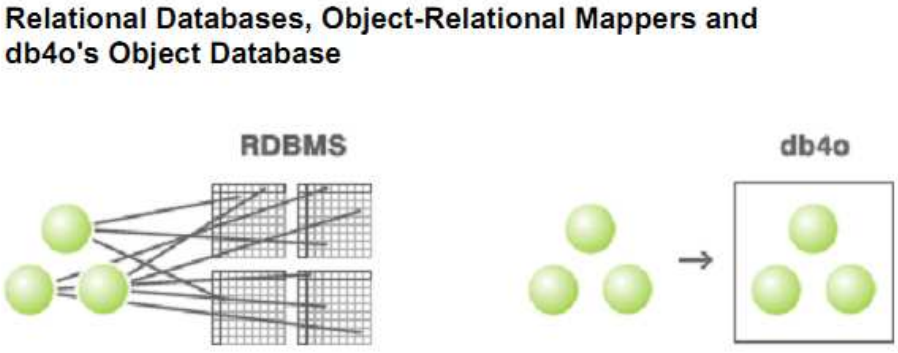


Gambar 2. 3 Arsitektur Berbasis Objek

2.2.2 Penerapan Sistem Basis Data Berorientasi Objek.

Terdapat situasi dimana basis data relasional dapat bekerja dengan baik. Namun, ada kalanya jenis basis data terbaik untuk sebuah aplikasi adalah OODB. Beberapa keterbatasan basis data relasional apabila diterapkan pada sebuah perangkat lunak berorientasi objek, antara lain :

a. Objek harus dipaparkan menjadi table.
b. Objek yang kompleks harus dipilah-pilah dan disimpan ke table yang berbeda.
c. Ketika menerima/mengambil data dari basis data, objek harus dikumpulkan atau disatukan lagi bagian-bagiannya pada table berbeda. Gambar 2.4 menunjukan perbandingan antara basis data relasional dengan teknologi pemetaan objek relasional dan basis data berorientasi objek



Gambar 2. 4 Perbandingan Pemetaan Basis Data Relasional dan Basis Data Objek

Secara obyektif, menurut Sabau(2007) karakteristik RDBMS berbeda dengan ODBMS dalam interaksinya dengan aplikasi, antara lain ada pada Tabel 2.1 :

Tabel 2. 1 Perbandingan Karakteristik ODBMS dan RDBMS

ODBMS	**RDBMS**
ODBMS menyimpan data dan metode	RDBMS hanya menyimpan data
Data bersifat enkasulapsi dapat digunakan hanya melalui akses ke kelas bersangkutan.	Data bebas diakses sesuai dengan kebutuhan user.
Struktur kompleks : dapat merepresentasikan basis data ke banyak tipe seperti struktur graf, pohon, dan jaringan.	Struktur sederhana : hanya merepresentasikan basis data ke kolom, baris, dan tabel.
Data dapat dihubungkan satu-sama lain dengan optimasi metode pada kelas.	Data dapat dihubungkan dengan optimasi operator Join.
Odbms menggunakan konsep inheritence untuk mencegah redudansi data.	Rdbms menggunakan konsep normalisasi untuk mencegah redudansi data.
Performa askes data odbms tergantung optimasi metode pada class dalam mekanisme pengaksesan datanya.	Performa akses data rdbms tergantung optimasi DDL, DML, dan Query.

Menurut Grehan(2005), terdapat beberapa macam aplikasi dimana OODB dapat memberikan solusi terbaik bagi aplikasi, antara lain:

a. *Embedded* database

   Sebuah perangkat lunak yang menginginkan akses data cepat ke penyimpanan dapat menggunakan OODB sebagai solusi basis data tertanam.

b. Hubungan data yang kompleks

Beberapa kelas mendefinisikan referensi atau hubungan silang antar mereka sendiri. Hubungan yang kompleks antar kelas dapat lebih mudah disimpan dalam OODB.

c. Struktur Objek yang dalam

Tidak semua data mudah diatur ke dalam bentuk tabular baris-dan-kolom yang berasosiasi dengan RDBMS, missal data berupa struktur graf atau struktur pohon. Suatu struktur pohon bisa sangat dalam sehingga sulit disimpan dalam basis data relasional

d. Terdapat perubahan struktur data atau objek

Dalam rekayasa perangkat lunak memungkinkan terjadi sebuah perubahan struktur data seperti penambahan atribut baru pada kelas bahkan pembuatan sebuah kelas baru yang beratribut banyak. Perangkat lunak yang sering melakukan perubahan pada struktur datanya lebih tepat menggunakan ODBMS sebagai media penyimpanan.

e. Pengembangan menggunakan teknik agile

Teknik pemrograman agile cepat populer karena mereka menunjukkan keuntungan berupa pengurangan kesalahan pengembangan. Sebuah ODBMS lebih cocok dan lancar diterapkan ke dalam pengembangan berteknik agile dari pada RDBMS.

2.2.3 Sistem Manajemen Basis Data db4o.

db4o adalah ODBMS open source yang memungkinkan pengembang memangkas waktu pengembangan dan biaya dan meraih kinerja yang tinggi. Disain db4o adalah pilihan ideal untuk perangkat lunak *embedded* tertanam di perangkat berjalan.

db4o menyediakan fitur tanpa penggunaan administrasi basis data*(zero administration)*, minimalisir penggunaan memory pada aplikasi(*small footprints)*, kinerja tinggi, kelancaran sinkronisasi, dan kemudahan perbaikan disain aplikasi pada implementasi(*refactoring)*. Dalam Java dan .Net, pustaka db4o mudah

diintegrasikan dalam aplikasi dan meberikan kinerja handal dan penugasan persistence tanpa melihat kompleksitas struktur objek.

Menurut tim db4o(2004), pengembang aplikasi dapat :

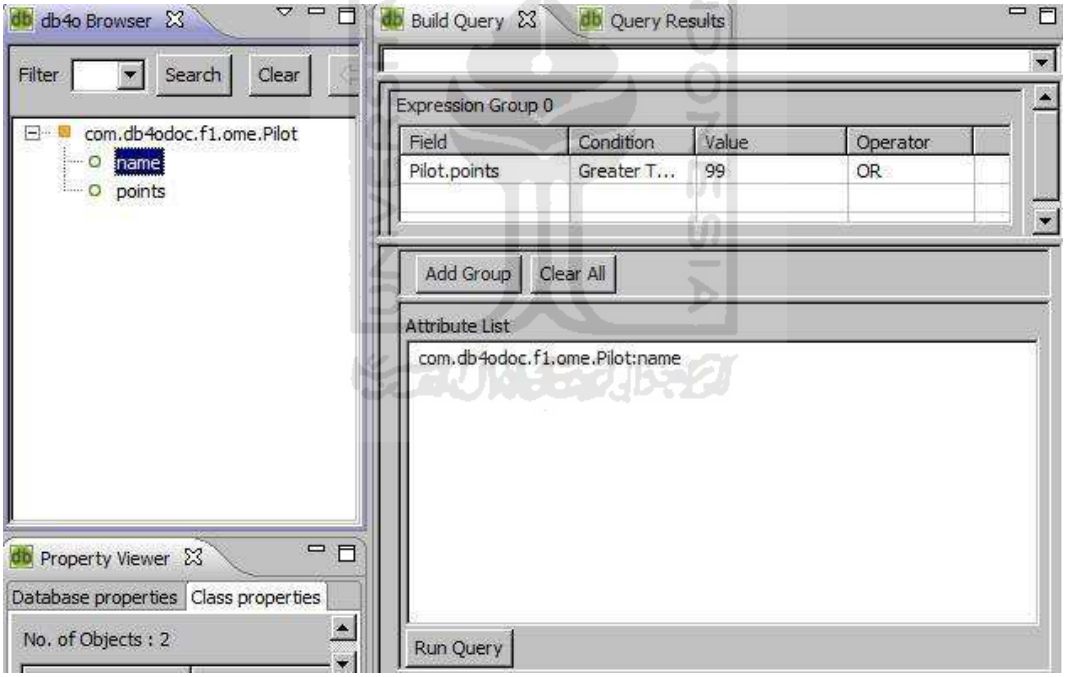
a. Menghilangkan pemetaan objek relasional, yang sangat kompleks, boros sumber daya, serta menghambat kinerja dan *refactoring*. Dengan db4o, pengembang bias menghemat 90% waktu dan biaya pengembangan perangkat lunak.
b. Membangun aplikasi terintegrasi dengan penyimpanan data, yang tidak perlu membutuhkan administrasi, serta handal dan lebih cepat daripada basis data konvensional.
c. Mendapat keuntungan paradigma berorientasi objek tanpa dibatasi oleh kemampuan basis data.
d. Mengganti, melakukan *refactoring*, dan menggunakan kembali komponen perangkat lunak dengan kemampuan untuk menambah fitur baru perangkat lunak tanpa merusak kode sebelumnya.

Menurut Rosenberg (2008), beberapa kemampuan db4o antara lain :

a. terintegrasi pada bahasa pemrograman Java dan teknologi .NET
b. mempunyai banyak tipe pemrosesan *query* yaitu *Query By Example (QBE), Simple Object Data Access(SODA*), dan *Native Query.* QBE adalah tipe yang memiliki karakterisitik cepat, ideal, dan simple untuk *query* yang tidak membutuhkan operator logika. SODA adalah tipe yang menggunakan sebuah *query* graf dengan navigasi rujukan pada kelas dan batasan. *Native Query* adalah tipe yang dapat dimanipulasi sesuai dengan kecocokan objek dan kecepatan akses tergantung pada optimasi *query.*
c. mendukung pengolahan sekumpulan objek berskala besar, replikasi basis data, dan distribusi basis data(*Client*/*Server*)
d. memodelkan data menggunakan kelas yang terdapat di aplikasi.
e. memungkinkan kelas diperlakukan sebagaimana tabel dalam basis data relasional dimana kolom direpresentasikan dengan atribut dalam kelas.
f. bekerja dengan transaksi.

g. menggunakan basis data tunggal sehingga sangat mudah untuk menyiapkan basis data. Jika basis data tidak ada, maka basis data akan dibangun secara otomatis.
h. mendukung beberapa *client* terkoneksi dalam satu basis data tunggal.
i. keuntungan pada sebuah aplikasi dengan kompleksitas objek yang tinggi.

Pada awalnya db4o dikembangkan untuk memenuhi kebutuhan aplikasi yang menerapkan konsep *zero administration* dan tertanam pada basis datanya. Versant Corporation selaku pengembang db4o merilis sebuah kakas yang dapat digunakan secara independent untuk memanajemen basis data db4o tanpa harus melalui aplikasi. Kakas tersebut adalah ObjectManager Enterprise(OME). Gambar 2.5 menunjukkan tampilan kakas OME untuk IDE Eclipse.



Gambar 2. 5 ObjectManager Enterprise untuk Eclipse

OME didisain untuk db4o agar user dapat melakukan manajemen dan perawatan data. Terdapat beberapa fitur antara lain:

a. Fitur menampilkan kelas secara hirarkis dan terstuktur.
b. Fitur menampilkan kelas secara detail seperti atribut, sub kelas, dan metode.
c. Fitur menampilkan objek tunggal secara detail dengan representasi pohon.

d. Fitur pembentuk query yang membantu untuk membangun query dari sederhana sampai kompleks. Hasil query ditampilkan dalam bentuk objek ganda dengan representasi list.

2.3 Aplikasi Penerjemah Multi Bahasa.

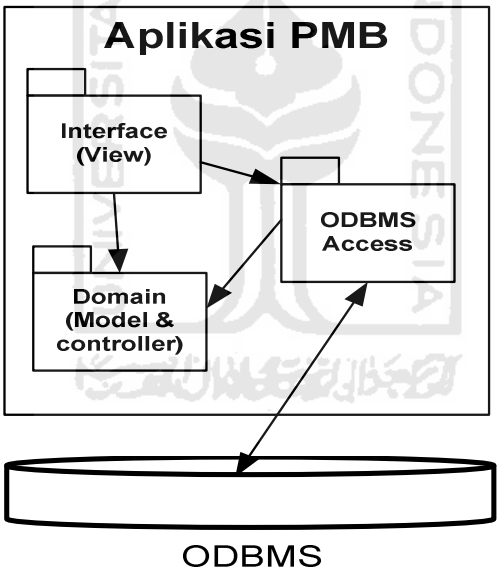
Aplikasi penerjemah bahasa merupakan salah satu jenis aplikasi mesin penerjemah**.** Menurut Hutchins dan Somers(1992), aplikasi *machine translation* atau biasa disingkat dengan MT, merupakan sub bidang dari *computational linguistic* yang membahas penggunaan perangkat lunak komputer dalam menterjemahkan teks atau perkataan dari satu bahasa ke bahasa lain. Pada level dasar, MT melakukan subtitusi kata antara satu bahasa dan lainya. Dengan menggunakan teknik korpus/korpora, penterjemahan yang kompleks dapat dilakukan, memungkinkan untuk penanganan lebih baik bagi perbedaan tipologi linguistik, pengenalan frase, penerjemahan idiom, dan pencegahan anomali.

Perangkat lunak MT terkini sering disesuaikan dengan *domain* atau profesi untuk meningkatkan kualitas keluaran dengan membatasi ruang lingkup subtitusi. Teknik ini sangat efektif dalam profesi dimana bahasa formal digunakan, sehingga penerjemahan MT pada bidang pemerintahan dan hukum lebih mudah dilakukan daripada penerjemahan MT untuk percakapan atau teks biasa.

Peningkatan kualitas keluaran juga dapat dicapai dengan intervensi manusia. Misalnya keluaran lebih akurat jika pengguna telah jelas mengidentifikasi kata-kata dalam teks adalah nama. MT telah terbukti berguna sebagai alat bantu penerjemah bahasa manusia dan dalam jumlah yang terbatas dapat digunakan untuk beberapa kasus sesuai dengan profesi.

2.3.1 Integrasi Aplikasi Penerjemah Multi Bahasa dengan Basis Data Berorientasi Objek.

Tugas pokok yang akan dilakukan pada tugas akhir ini adalah mengintergrasikan basis data berorientasi objek dalam merekayasa perangkat lunak aplikasi penerjemah multi bahasa(PMB). Integrasi antar komponen aplikasi menggunakan representasi layer of services dimana komponen yang terletak pada layer paling atas berhak menggunakan *resource* maupun *method* pada layer dibawahnya. Aplikasi PMB wajib menyediakan interface, *domain*, dan *object database* access layer sebelum mengintegrasikannya dengan ODBMS. Setelah itu, aplikasi cukup memasukkan sebuah pustaka komponen ODBMS(db4o) dalam rekayasanya. Berikut gambar 2.6



Gambar 2. 6 Integrasi Aplikasi Penerjemah multi bahasa dengan *ODBMS*

Dengan adanya integrasi dari dua komponen diatas kedepannya diharapkan mampu mewujudkan sebuah sifat perangkat lunak yang tangguh dan independent.

BAB III

METODOLOGI

3.1 Analisis Kebutuhan Sistem

Subbab ini menjelaskan tentang kebutuhan proses bisnis yang terlibat pada aplikasi penerjemah multi bahasa. Terdapat tiga kebutuhan proses bisnis utama antara lain :

a. Penerjemah Bahasa

Proses ini menjelaskan interaksi pengguna dengan sistem dalam melakukan fungsi utama aplikasi ini yaitu menterjemahkan bahasa.

b. Manajemen Skema

Proses ini menjelaskan cara mendefinisikan sebuah skema untuk menyimpan data terjemahan pengguna.

c. Manajemen Data Skema

Proses ini menjelaskan mekanisme yang mengatur data terjemahan terkait dengan skema yang sudah terdefinisi.

3.1.1 Masukan Sistem

Kebutuhan setiap proses berbeda satu dengan lainya termasuk kebutuhan input.

3.1.1.1 Masukan Penerjemah Bahasa

Kebutuhan masukan dalam proses Penerjemah Bahasa ditujukkan pada tabel 3.1:

Tabel 3. 1 Kebutuhan Masukan Proses Penerjemah Bahasa

No	Kebutuhan Masukan	Penjelasan
1	Penerjemah Bahasa/ Simpan Data Terjemahan *Temporary*	Menerjemahkan teks sumber dan menyimpan data terjemahan.

	- ID *Temporary* - Teks Sumber - Bahasa Sumber - Bahasa Terjemahan	
2	Hapus Terjemahan *Temporary* - ID *Temporary*	Menghapus data terjemahan *temporary*

3.1.1.2 Masukan Manajemen Skema

Kebutuhan masukan dalam proses Manajemen Skema ditunjukkan pada tabel 3.2:

Tabel 3. 2 Kebutuhan Masukan Proses Manajemen Skema

No	Kebutuhan Masukan	Penjelasan
1	Buat Skema - Nama Skema	Membuat sebuah skema dengan nama skema tertentu
2	Hapus Skema - Nama Skema - Tanggal Skema	Menghapus Skema Tertentu dari daftar Skema yang ada
3	Edit Skema - Nama Skema	Mengubah nama sebuah Skema tertentu

3.1.1.3 Masukan Manajemen Data Skema

Kebutuhan masukan dalam proses Manajemen Data Skema ditunjukkan pada tabel 3.3:

Tabel 3. 3 Kebutuhan Masukan Proses Manajemen Data Skema

No	Kebutuhan Masukan	Penjelasan
1	Simpan Data - Data Terjemahan *Temporary*	Menyimpan data terjemahan ke skema tertentu

	- Nama Skema	
2	Hapus Data - Nama Skema - ID Task	Menghapus data tersimpan skema tertentu
3	Edit Data - Nama Skema - ID Task - ID Activity - Teks Sumber - Opsi Bahasa Sumber - Opsi Bahasa Terjemahan	Mengubah data tersimpan skema tertentu
4	Copy Data - Nama Skema Asal - Nama Skema Tujuan - Data Skema Asal	Menyalin data tersimpan sebuah skema ke skema lainya.

3.1.2 Keluaran Sistem

Informasi hasil keluaran setiap proses berbeda satu dengan lainya tergantung keluaran setiap proses.

3.1.2.1 Keluaran Penerjemah Bahasa

Informasi hasil keluaran untuk pengguna dalam proses Penerjemah Bahasa ditunjukkan pada tabel 3.4:

Tabel 3. 4 Informasi Hasil Keluaran Proses Penerjemah Bahasa

No.	Informasi Hasil Keluaran	Penjelasan
1	Teks Terjemahan	Hasil Proses Terjemahan
2	Data Terjemahan Tersimpan. - ID *Temporary* - Teks Sumber	Data tersimpan otomatis secara *Temporary* didefinisikan sebagai data terjemahan *Temporary*.

	- Bahasa Sumber - Bahasa Terjemahan - Teks Terjemahan	
3	Data Terjemahan Terhapus	Data terjemahan tertentu terhapus

3.1.2.2 Keluaran Manajemen Skema

Informasi hasil keluaran untuk pengguna dalam proses Manajemen Skema ditunjukkan pada tabel 3.5:

Tabel 3. 5 Informasi Hasil Keluaran Proses Manajemen Skema

No.	Informasi Hasil Keluaran	Penjelasan
1	Skema terbuat - Nama skema - Tgl skema	Skema tersimpan di daftar skema
2	Nama Skema terhapus - Informasi Terhapus	Skema tertentu terhapus dari daftar skema
3	Nama Skema terubah - Nama skema	Nama Skema tertentu berubah di daftar skema

3.1.2.3 Keluaran Manajemen Data Skema

Informasi hasil keluaran untuk pengguna dalam proses Manajemen Data Skema ditunjukkan pada tabel 3.6:

Tabel 3. 6 Informasi Hasil Keluaran Proses Manajemen Data Skema

No.	Informasi Hasil Keluaran	Penjelasan
1	Data Skema tersimpan - Data Terjemahan *Temporary* - Nama Skema	Data terjemahan *Temporary* tersimpan ke Skema Tertentu.
2	Data Skema terhapus	Data terjemahan Skema tertentu

	- Informasi Terhapus	terhapus
3	Data Skema terubah - Teks Sumber - Opsi Bahasa Sumber - Opsi Bahasa Terjemahan - Teks Terjmahan	Data terjemahan Skema tertentu berubah
4	Data Skema tersalin ke Skema lain. - Data Skema Tujuan	Data terjemahan skema asal tersalin ke skema tujuan.

3.2 Disain Aplikasi

Pada Sub-bab ini akan dibahas disain aplikasi penerjemah multi bahasa. Pada laporan ini disain aplikasi dibagi 2 yaitu disain sistem dan disain antarmuka aplikasi.

3.2.1 Disain Sistem Aplikasi

Disain sistem aplikasi direpresentasikan menggunakan *Use Case* Diagram, *Activity* Diagram, *Class* Diagram, *dan Sequence* Diagram.

3.2.1.1 Use Case Diagram

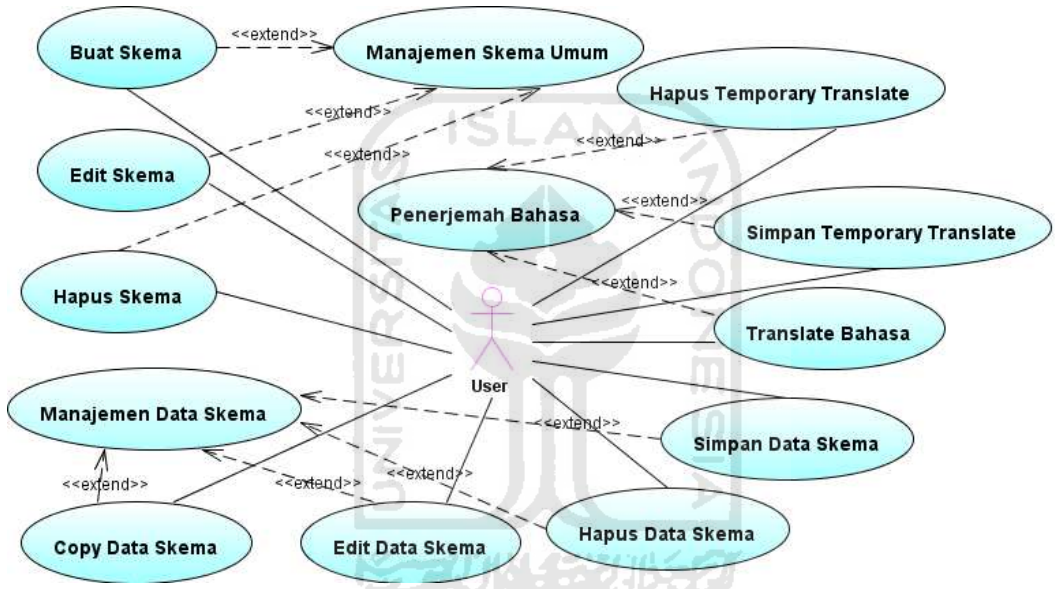
Use case memberikan gambaran bagaimana pengguna berinteraksi dengan proses dalam sistem. Berikut adalah spesifikasi kebutuhan perangkat lunak dan *Use Case* Diagram Aplikasi Penerjemah Multi Bahasa :

Requirements : Pengguna memakai aplikasi penerjemah multi bahasa. Pengguna berinteraksi dengan fitur penerjemah bahasa, manajemen skema, dan manajemen data skema. Pengguna dapat langsung berinteraksidengan fitur penerjemah bahasa dan dan manajemen skema. Untuk berinteraksi dengan fitur manajemen data skema harus menggunakan fitur manajemen skema dan fitur penerjemah bahasa

Aktor : Pengguna

Use Case : *Translate* Bahasa, Simpan *Temporary Translate*, Hapus *Temporary Translate*, Buat Skema, Edit Skema, Hapus Skema, Simpan Data Skema, Hapus Data Skema, Edit Data Skema, Copy Data Skema.

Diagram 3.1 menggambarkan aktor dan proses(usecase) pada sistem. Diagram ini belum mendeskripsikan aktivitas yang terdapat di setiap proses. Oleh karena itu, penjelasan diagram ini perlu ditunjang dengan *Activity* Diagram.



Gambar 3. 1 *Use Case* Diagram Aplikasi Penerjemah Multi Bahasa

3.2.1.2 Activity Diagram

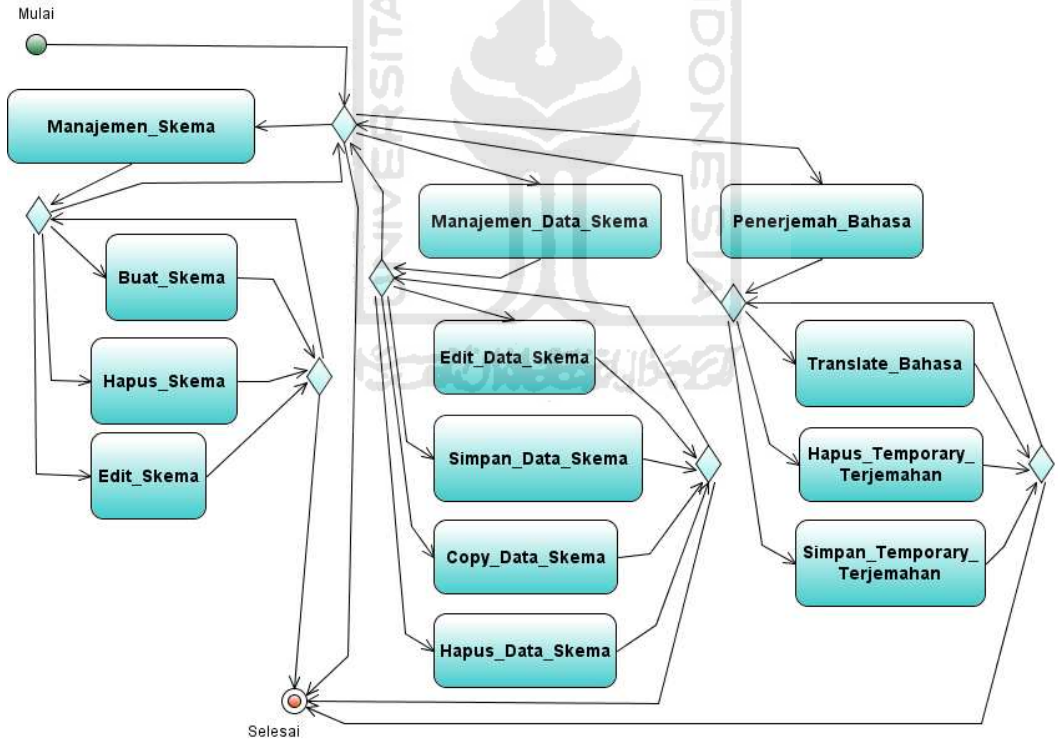
Activity Diagram merepresentasikan alur proses dalam sistem. Proses-proses yang digambarkan pada *Use Case* Diagram diperjelas menggunakan diagram ini beserta aliran-aliran prosesnya. Tabel 3.7 menunjukkan aktivitas aplikasi penerjemah multi bahasa :

Tabel 3. 7 Aktivitas Aplikasi Penerjemah Multi Bahasa

Pengguna		
Proses Penerjemah Bahasa	**Proses Manajemen Skema**	**Proses Manajemen Data Skema**

Translate Bahasa	Buat Skema	Simpan Data Skema
Simpan *Temporary* Terjemahan	Edit Skema	Hapus Data Skema
Hapus *Temporary* Terjemahan	Hapus Skema	Edit Data Skema
		Copy Data Skema

Gambar 3.2 menggambarkan proses-proses pada sistem dari awal sampai selesai. Dalam diagram diatas diperlihatkan pula beberapa turunan proses *Penerjemah_Bahasa*, *Manajemen_Data_Skema*, dan *Manajemen_Skema_Umum* yang sebelumnya terdapat pada *Use Case* Diagram

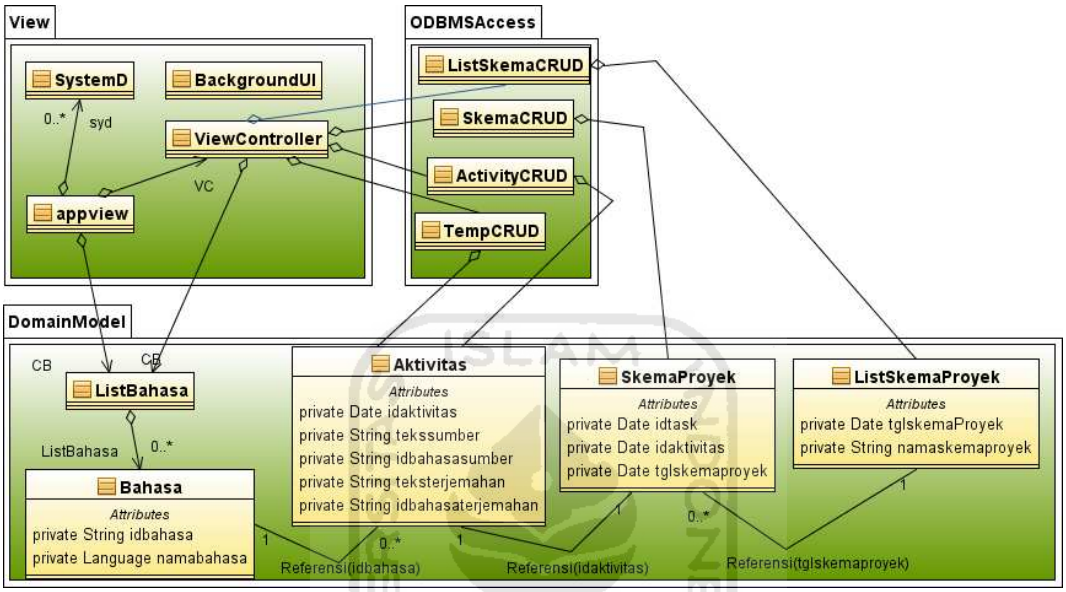


Gambar 3. 2 *Activity* Diagram Aplikasi Penerjemah Multi Bahasa

3.2.1.3 Class Diagram

Class Diagram digunakan untuk menggambarkan *domain* permasalahan sistem yang dibangun. Pada Aplikasi Penerjemah Multi Bahasa terdapat beberapa

package yang berisikan kelas-kelas yang saling terhubung satu sama lainya. Dalam laporan ini *Class* Diagram dikelompokan lebih spesifik pada beberapa *package* yaitu *DomainModel, ODBMSAccess,* dan *View*. Gambar 3.3 menunjukkan *Class* Diagram antar package.

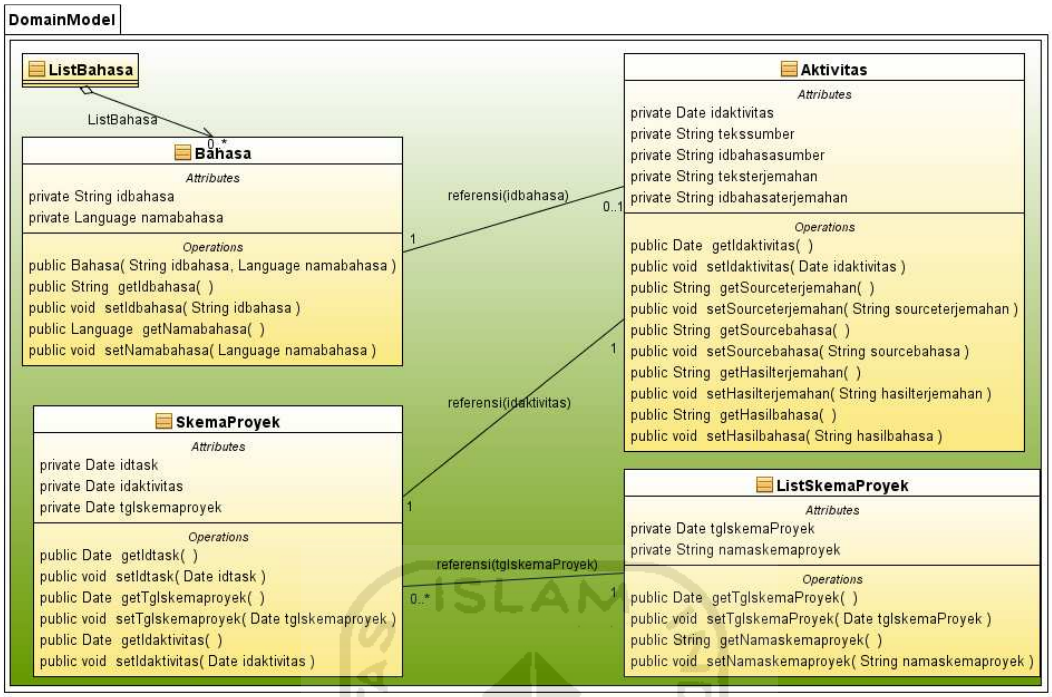


Gambar 3. 3 *Class* Diagram Aplikasi Penerjemah Multi Bahasa

Pada paket *View*, setiap kelas didalamnya digunakan untuk mengolah objek terkait antarmuka aplikasi secara visual ke pengguna. Paket *ODBMSAccess* mengolah objek terkait integrasi dan manajemen data pada ODBMS. Paket *DomainModel* memuat kelas yang berguna sebagai *resource* bagi kelas-kelas yang terdapat di paket-paket lain.

3.2.1.3.1 Class Diagram *Package DomainModel*

Paket *DomainModel* berisi kelas-kelas yang berfungsi sebagai resource dari *package* lain yaitu *ListBahasa, Bahasa, Aktivitas, SkemaProyek,* dan *ListSkemaProyek.* Berikut Gambar 3.4 diagram *class package DomainModel.*

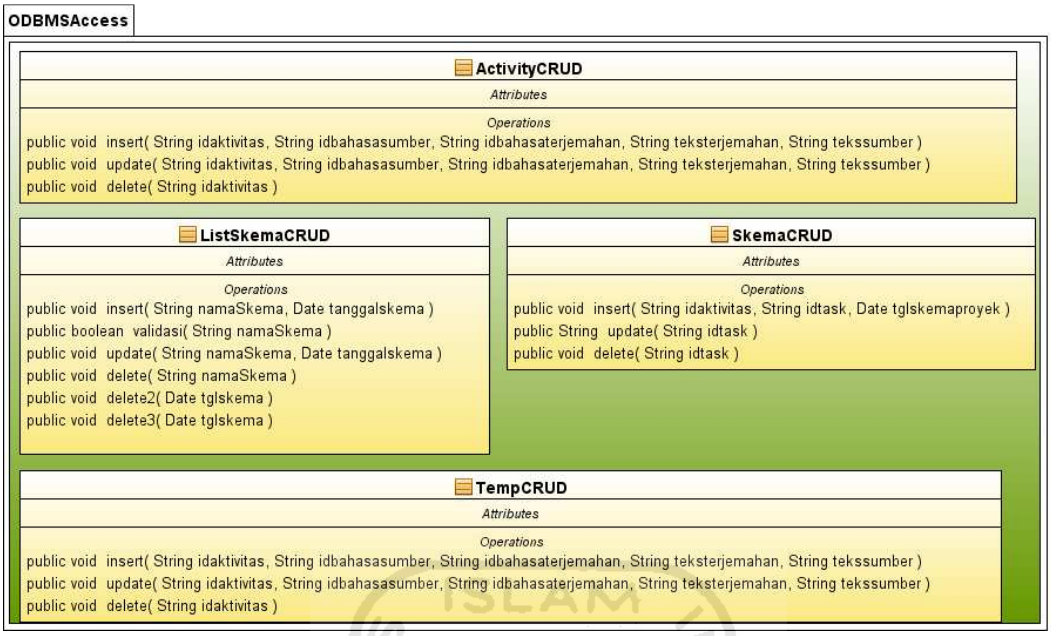


Gambar 3. 4 *Class* Diagram *package DomainModel*

Gambar 3.4 menjelaskan hubungan kelas yang terdapat di paket *DomainModel.* Kelas *ListBahasa* berelasi agregasi dengan *Bahasa* karena *ListBahasa* menggunakan *Bahasa* untuk mencetak daftar bahasa. Relasi yang terdapat antara Kelas *Bahasa, Aktivitas*, *SkemaProyek,* dan *ListSkemaProyek* bersifat asosiasi karena terdapat atribut yang sama pada kelas-kelas tersebut. Deskripsi relasi asosiasi antara *Bahasa, Aktivitas*, *SkemaProyek,* dan *ListSkemaProyek* berfungsi sebagai deskripsi perancangan basis data aplikasi pada laporan tugas akhir ini. Deskripsi dalam bentuk kode sumber bahasa Java tertulis pada lampiran 1 (1) Kelas *Bahasa,* lampiran 1 (2) Kelas *Aktivitas,* lampiran 1 (3) Kelas *SkemaProyek,* dan lampiran 1 (4) Kelas *ListSkemaProyek*

3.2.1.3.2 Class Diagram *Package ODBMSAccess*

Paket *ODBMSaccess* berisi kelas-kelas yang berfungsi sebagai integrasi dan manajemen terhadap mesin basis data berorientasi objek yaitu *ListSkemaCRUD, SkemaCRUD, ActivityCRUD,* dan *TempCRUD.* Berikut Gambar 3.5 diagram class *package ODBMSaccess.*

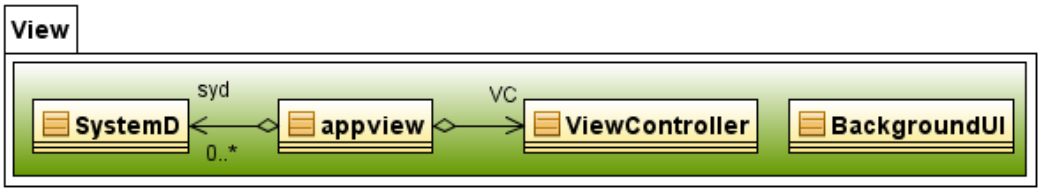


Gambar 3. 5 *Class* Diagram *package ODBMSAccess*

Gambar 3.5 menjelaskan sekumpulan class yang terdapat di *package* ODBMSAccess. Operasi atau method setiap class tersebut isinya hampir sama karena fungsi class tersebut adalah integrasi dengan mesin ODBMS serta manajemen seperti *CRUD(*create, read, update, delete*)* terhadap data.

3.2.1.3.3 Class Diagram *Package View*

Paket *View* berisi kelas-kelas yang berfungsi untuk mengatur tampilan antarmuka aplikasi ke pengguna seperti tampilan masukan aplikasi dan keluaran aplikasi yaitu kelas *appview*, *ViewController*, *SystemD*, dan *BackgorundUI*. Berikut Gambar 3.6 diagraam class *package View*.



Gambar 3. 6 *Class* Diagram *package View*

Class-Class pada *package* diatas digunakan untuk merekayasa antarmuka aplikasi. Terdapat kelas *appview*, yang berfungsi untuk mengeksekusi aplikasi. Selain itu terdapat class pendukung tampilan antarmuka seperti kelas *SystemD* dan

kelas *BackgroundUI.* Pada *package* ini juga terdapat class pengatur *controller* antarmuka yaitu kelas *ViewController.*

3.2.1.4 Sequence Diagram

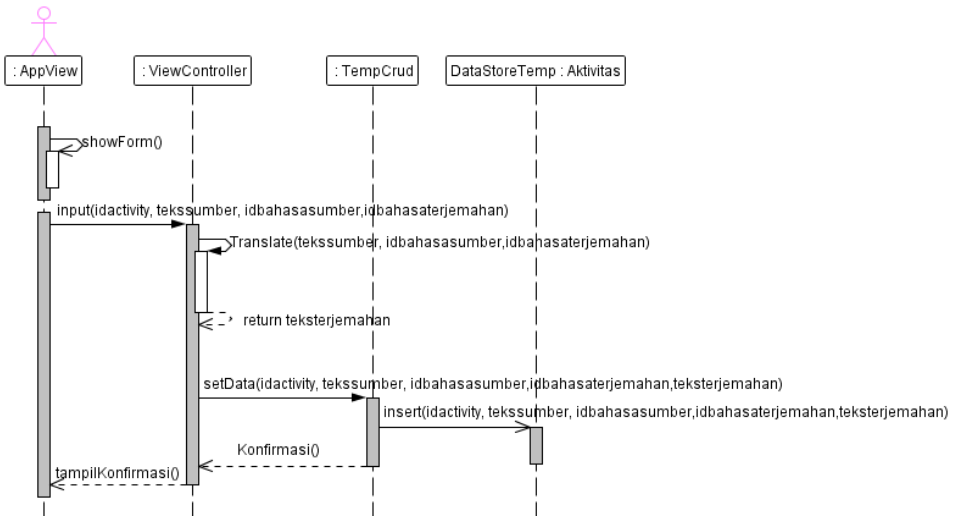
Sequence Diagram menggambarkan interaksi antar objek dalam sistem yang telah dicetak oleh kelas tertentu. Pada diagram ini digambarkan pula skenario dan urutan berinteraksi dalam rangka melengkapi sebuah proses bisnis.

3.2.1.4.1 Sequence Diagram Penerjemah Bahasa

Terdapat beberapa sub-proses pada proses Penerjemah Bahasa, setiap subproses digambarkan dengan Diagram *Sequence*. Berikut Diagram *Sequence* untuk sub-proses Penerjemah Bahasa.

a. *Translate* Bahasa dan Simpan Aktivitas *Translate Temporary*

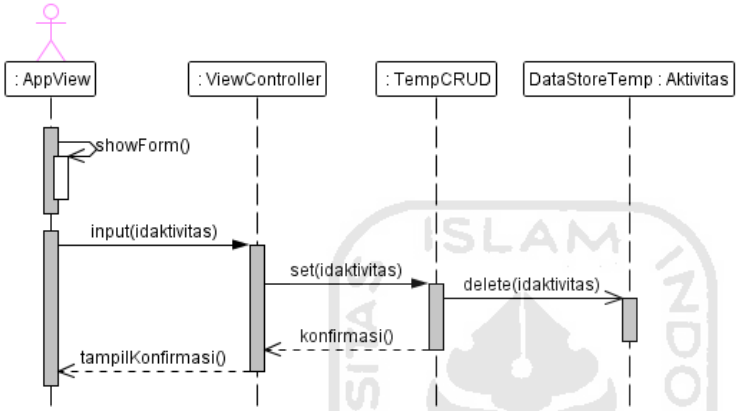
Proses pada Gambar 3.7 menjelaskan urutan interaksi objek untuk menterjemahkan bahasa. Kelas *appview* memanggil class *ViewController* dengan cara mengirim masukan data. Selanjutnya pada *ViewController* melakukan fungsi penerjemah dengan metode *Translate()* dan mengembalikan nilai berupa *teksterjemahan.* Selanjutnya semua atribut data diset ke class *TempCRUD* untuk disimpan ke *Datastore Temporary Aktivitas*



Gambar 3. 7 Sequence Diagram proses *Translate* Bahasa dan Simpan Aktivitas *Temporary*

b. Hapus Aktivitas *Translate Temporary*

Gambar 3.8 menggambarkan proses menghapus Aktivitas *Translate* yang tersimpan secara *Temporary*. Pertama, kelas *appview* mengirimkan data ke kelas *ViewController*. Kedua, kelas *ViewController* mengeset data sebelumnya ke kelas *TempCRUD*. Terakhir, kelas *TempCRUD* menghapus data pada *Datastore Temporary Aktivitas*



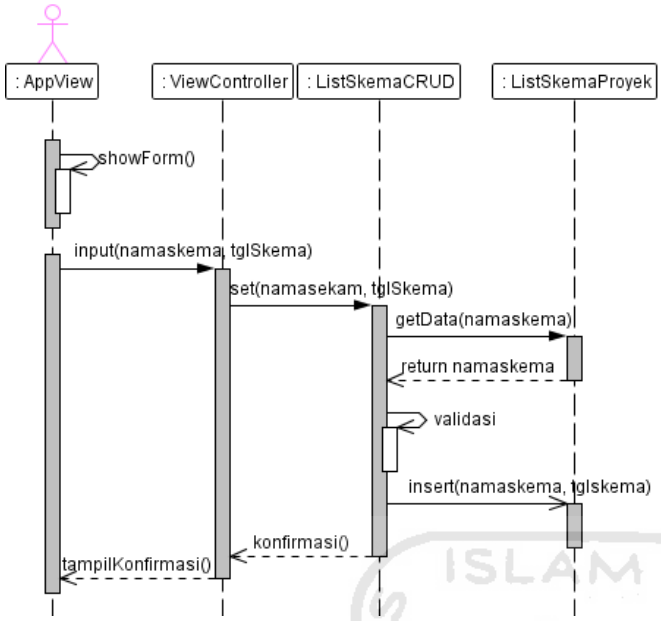
Gambar 3. 8 *Sequence* Diagram proses Hapus Aktivitas *Translate Temporary*

3.2.1.4.2 Sequence Diagram Manajemen Skema

Terdapat beberapa sub-proses pada proses Manajemen Skema, setiap subproses digambarkan dengan Diagram *Sequence*. Berikut Diagram *Sequence* untuk sub-proses Manajemen Skema.

a. Buat Skema

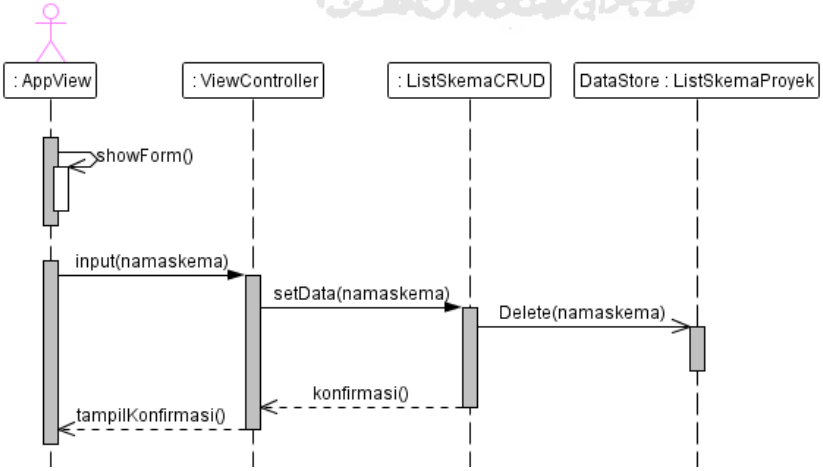
Gambar 3.9 menggambarkan proses Buat Skema. Kelas *appview* mengirimkan data berupa *namaskema* dan *tglskema*, kelas *ViewController* mengeset data ke kelas *ListSkemaCRUD*. Kelas *ListSkemaCRUD* memproses validasi ketersediaan namaskema dan menyimpan *namaskema*.



Gambar 3. 9 *Sequence* Diagram proses Buat Skema

b. Hapus Skema

Gambar 3.10 menggambarkan proses Hapus Skema. Kelas *appview* mengirimkan data berupa *namaskema*, kelas *ViewController* mengeset data ke kelas *ListSkemaCRUD*. Pada kelas *ListSkemaCRUD* melakukan proses penghapusan data pada *Datastore ListSkemaProyek*.

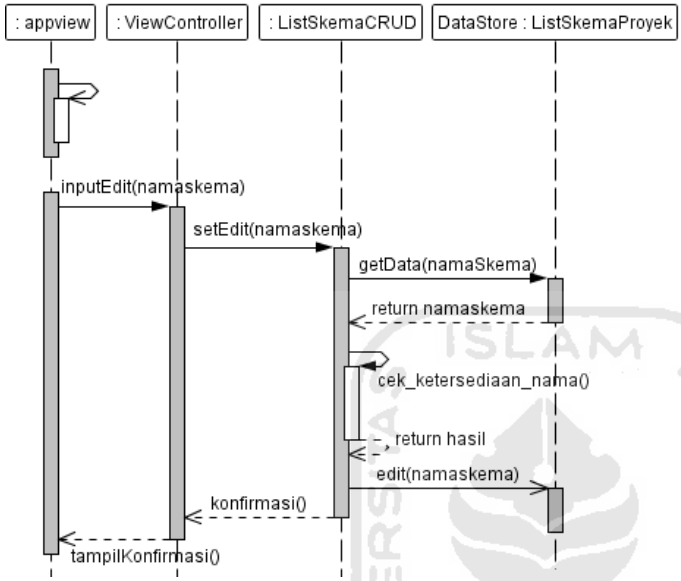


Gambar 3. 10 *Sequence* Diagram proses Hapus Skema

c. Edit Nama Skema

Gambar 3.11 menggambarkan proses Edit Nama Skema. Kelas *appview* mengirimkan data berupa *namaskema*, kelas *ViewController* mengeset data

ke kelas *ListSkemaCRUD.* C*lass ListSkemaCRUD* melakukan proses pengecekan data pada method *cek_ketersediaan_nama()* dan dilanjutkan dengan pengubahan data *namaskema* pada *Datastore ListSkemaProyek*.



Gambar 3. 11 *Sequence* Diagram proses Edit Nama Skema

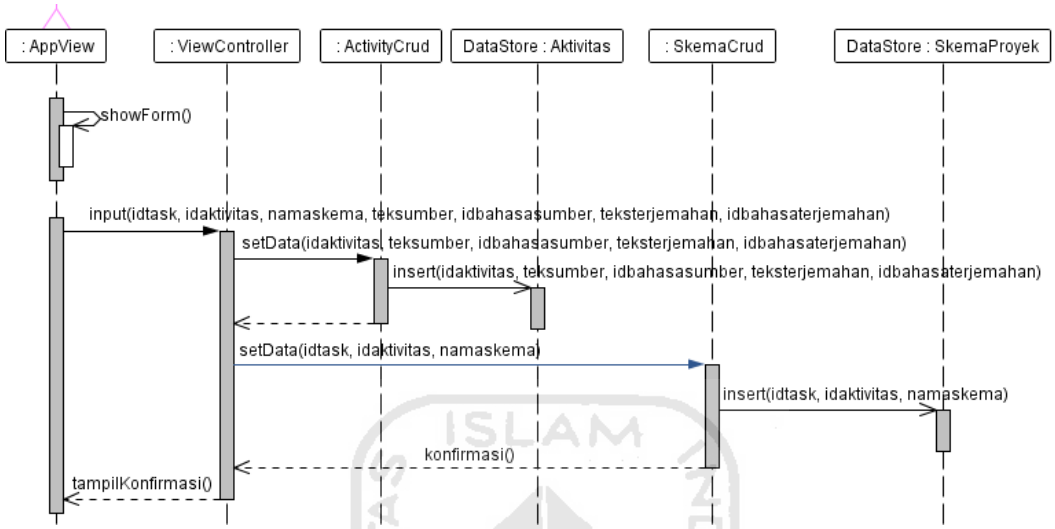
3.2.1.4.3 Sequence Diagram Manajemen Data Skema

Terdapat beberapa sub-proses pada proses Manajemen Data Skema, setiap subproses dapat digambarkan dengan Diagram *Sequence*. Berikut Diagram *Sequence* untuk sub-proses Manajemen Data Skema.

a. Insert Data Skema

Gambar 3.12 menjelaskan proses Insert Data Skema, Data terjemahan yang sudah tersedia akan disimpan berdasarkan atribut *namaskema* teretentu. Kelas *apapview* mengirim data terlebih dahulu ke kelas *ViewController.* Setelah itu kelas *ViewController* melakukan dua subproses yaitu menyimpan data terkait sesuai atribut kelas dan sesuai urutannya, yaitu pertama pada *Datastore Aktivitas* dan kedua pada *Datastore SkemaProyek.* Pada subproses yang pertama kelas *ViewController* mengeset data *idaktivitas*, *tekssumber, idbahasasumber*, *teksterjemahan,* dan *idbahasaterjemahan* pada kelas *ActivityCRUD* yang akan disimpan pada *Datastore Aktivitas.* Pada subproses

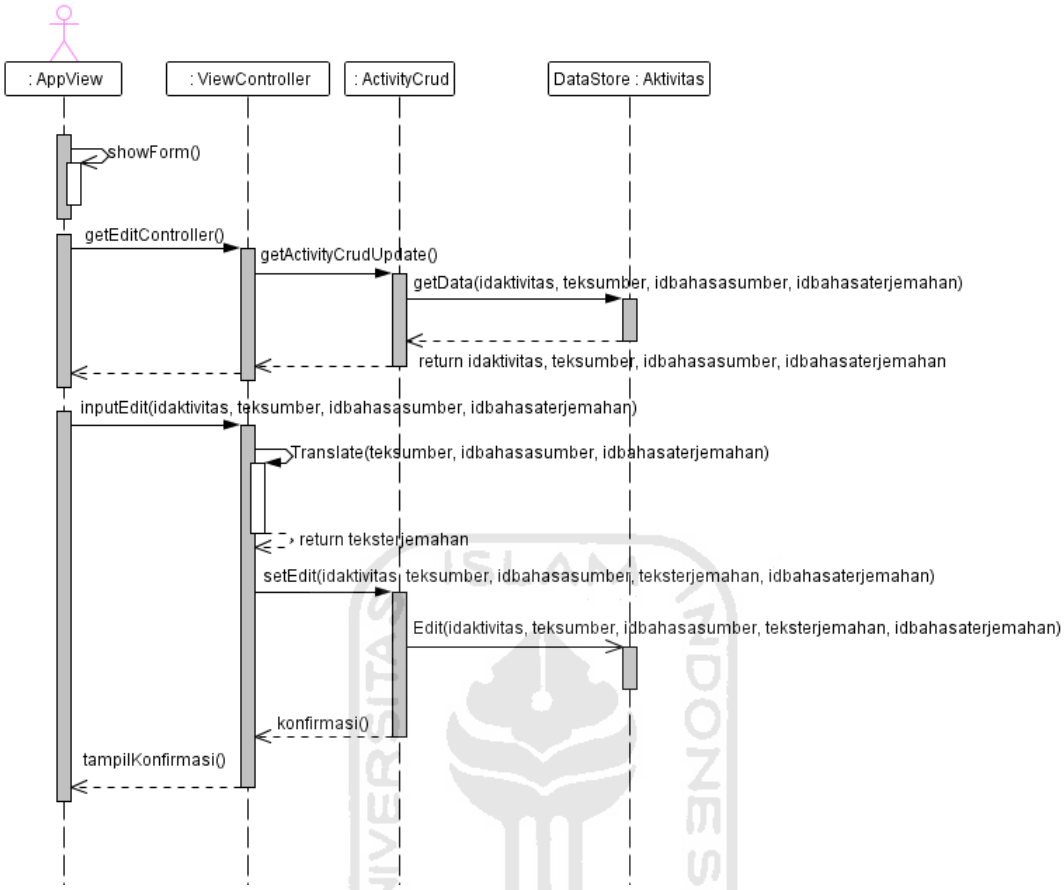
yang kedua class View Controller mengeset data *idtask, idaktivitas,* dan *namaskema* pada kelas *SkemaCRUD* yang akan disimpan pada *Datastore SkemaProyek.*



Gambar 3. 12 *Sequence* Diagram proses Insert Data Skema

b. Edit Data Skema

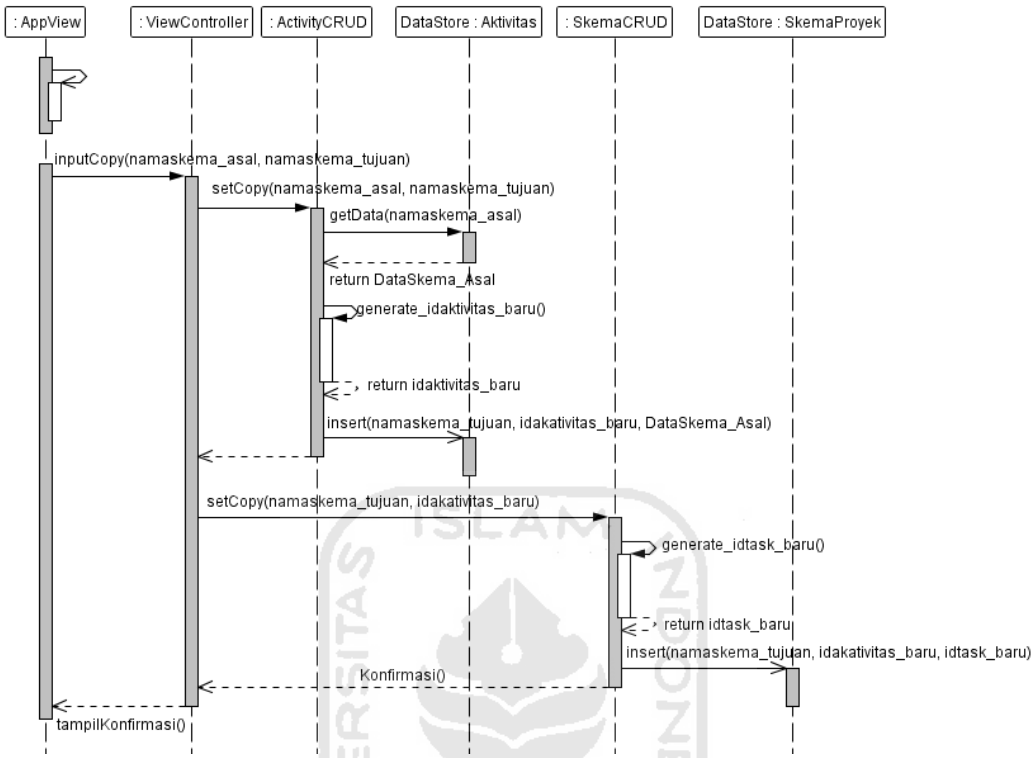
Gambar 3.13 menjelaskan bahwa proses edit Data Skema dilakukan dengan dua subproses, yaitu proses pengambilan data dan proses pengubahan data pada *Datastore Aktivitas.* Pada subproses pertama, data yang diambil adalah *idaktivitas, tekssumber, idbahasasumber*, dan *idbahasa terjemahan.* Setelah data diambil, dilakukan subproses pengubahan data. Pertama kelas *appview* mengirim data terkait ke class *ViewController.* Kelas *ViewController* melakukan fungsi terjemahan ulang dengan method *Translate()* dengan data *tekssumber* dan mengembalikan nilai *teksterjemahan* yang baru. Selanjutnya kelas *ViewController* mengeset data *teksterjemahan* pada kelas *ActivityCRUD* dimana nantinya dilakukan pengubahan data ke *Datastore Aktivitas.*



Gambar 3. 13 *Sequence* Diagram proses Edit Data Skema

c. Copy Data Skema

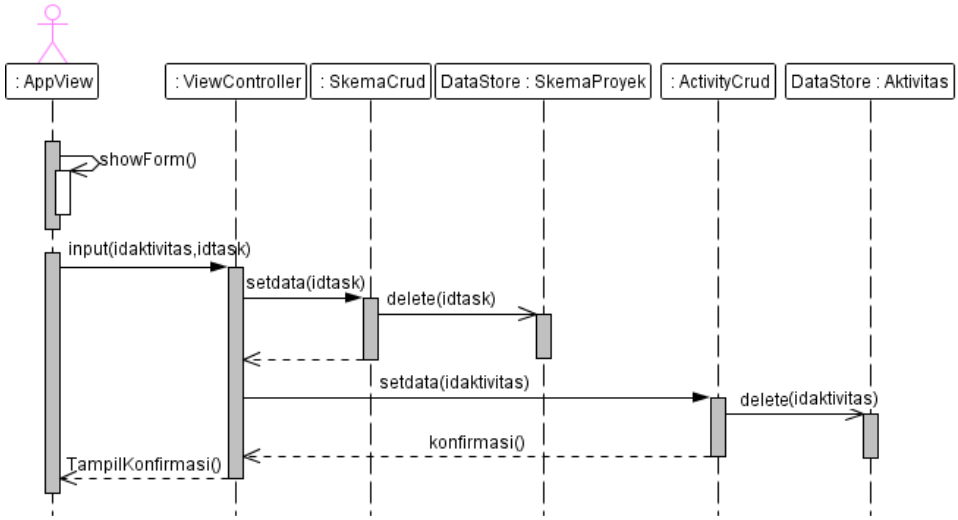
Gambar 3.14 menjelaskan proses penyalinan data antar skema. Proses penyalinan data diperlukan dua subproses. Yang pertama adalah menyalin atribut data *dataSkema_Asal* lalu mengeset data tersebut dengan menggunakan *idaktivitas_baru* dan disimpan pada *Datastore Aktivitas.* Yang kedua adalah mengeset atribut data *namaskema_tujuan, idaktivitas_baru* dengan menggunakan *idtask_baru* dan disimpan pada *Datastore SkemaProyek.* Atribut data untuk *dataSkema_Asal* adalah semacam pengelompokan dari beberapa atribut data yaitu *tekssumber, idbahasasumber, teksterjemahan,* dan *idbahasaterjemahan.*



Gambar 3. 14 *Sequence* Diagram proses Copy Data Skema

d. Hapus Data Skema

Gambar 3.15 menjelaskan cara menghapus data skema tertentu. Kelas *AppView* cukup mengirim atribut data *idaktivitas* dan *idtask* saja. Selanjutnya dua atribut data akan diset secara terpisah, *idtask* diset ke kelas *SkemaCRUD* dan *idaktivitas* akan diset pada kelas *ActivityCRUD.* Yang terakhir Kelas *SkemaCRUD* akan menghapus *idtask* pada *Datastore SkemaProyek* juga kelas *ActivityCRUD* akan menghapus *idaktivitas* pada *Datastore Aktivitas.*



Gambar 3. 15 *Sequence* Diagram proses Hapus Data Skema

3.2.2 Disain Antarmuka Aplikasi

Antarmuka merupakan media interaksi antara pengguna dengan aplikasi. Melalui tampilan antar muka pengguna dapat dengan mudah menjalankan aplikasi dan mendapatkan hasil pemrosesan aplikasi. Pada laporan ini, disain utama antarmuka dikelompokan dan dijelaskan menjadi tiga sesuai dengan kebutuhan bisnis proses utama yaitu penerjemah bahasa, manajemen skema, dan manajemen data skema.

3.2.2.1 Disain Antarmuka proses Penerjemah Bahasa

Pada proses Penerjemah Bahasa terdapat dua buah proses yaitu Terjemah Bahasa/Simpan Aktivitas *Temporary* dan Hapus Aktivitas *Temporary*. Berikut disain Antarmuka berdasarkan proses.

a. Terjemah Bahasa/Simpan Aktivitas *Temporary*

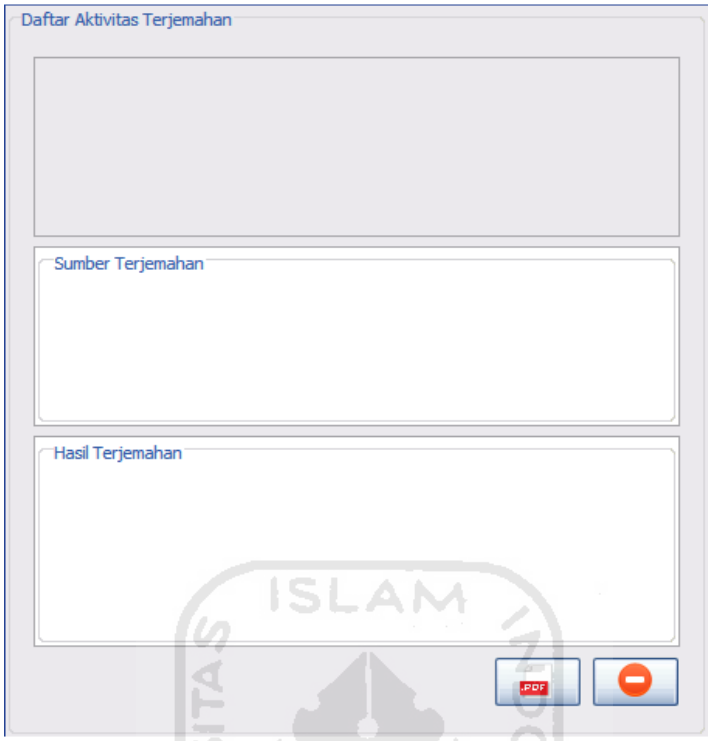
Proses Terjemah Bahasa/Simpan Aktivitas *Temporary* membutuhkan sebuah antarmuka dimana pengguna dapat berinteraksi dengan cara memasukan teks sumber, memilih opsi terjemahan, dan manampilkan output teks terjemahan. Gambar 3.16 menjelaskan antarmuka untuk memasukan teks sumber dengan komponen *Text Area*, untuk memilih opsi penerjemah bahasa dengan komponen *Radio*, *Button*, *Combo Box*, untuk menampilkan

output teks terjemahan dengan *Text Area*, dan untuk memproses terjemahan dengan *button* berikon bintang.

Gambar 3. 16 Disain Antarmuka Hapus Aktivitas *Temporary*

b. Hapus Aktivitas *Temporary*

Proses Hapus Aktivitas *Temporary* hanya membutuhkan dua interaksi. Interaksi pertama adalah memilih *Temporary* yang akan dihapus dan interaksi kedua adalah mengirim penghapusan *Temporary* tersebut. Gambar 3.17 menjelaskan antarmuka untuk memilih *Temporary* terjemahan dengan menggunakan komponen *Table* dan untuk mengirim penghapusan *temporary* dengan button berikon tanda minus(-).



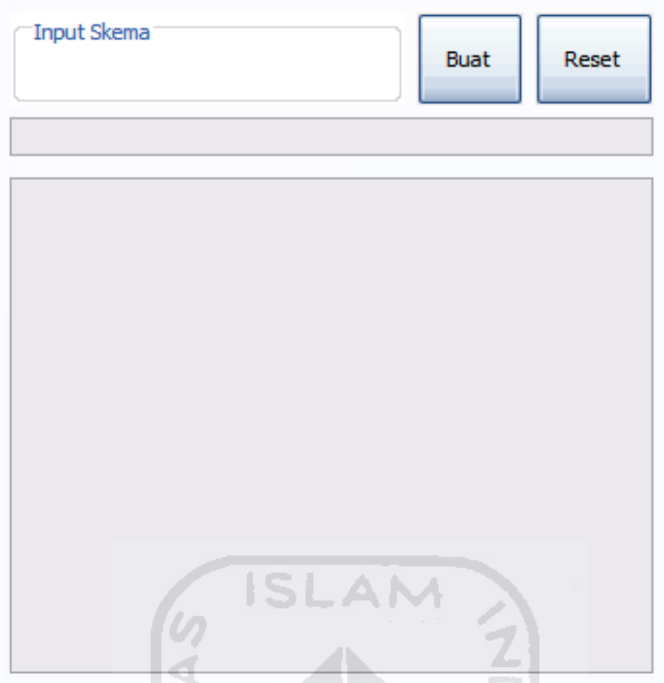
Gambar 3. 17 Disain Antarmuka Hapus Aktivitas *Temporary*

3.2.2.2 Disain Antarmuka proses Manajemen Skema

Pada proses Manajemen Skema terdapat tiga buah subproses yaitu Buat Skema, Edit Skema, dan Hapus Skema. Berikut disain Antarmuka berdasarkan proses.

a. Buat Skema

Tahapan interaksi pengguna dalam melakukan proses Buat Skema hanyalah mengisi data berupa nama skema dan mengirim data tersebut atau mereset jika data tersebut salah. Pada gambar 3.18 komponen *Text Field* untuk mengisi data nama skema dan komponen *button* untuk mengirim data nama skema atau mereset jika data nama skema salah. Terdapat dua tambahan komponen yaitu *Text Field* untuk menampilkan pesan terkait nama skema dan *Table* untuk menampilkan data nama skema yang tersimpan.



Gambar 3. 18 Disain Antarmuka Proses Buat Skema

b. Edit Skema

Pada proses Edit Skema, interaksi yang dilakukan adalah mengisi data nama skema dan mengirim edit data nama skema. Pada gambar 3.19 komponen *Text Field* digunakan untuk pengisian data nama skema dan *button* berikon pensil digunakan untuk mengirim edit data nama skema.

Gambar 3. 19 Disain Antarmuka Proses Edit Skema

c. Hapus Skema

Proses Hapus Skema dilakukan dengan dua interaksi yaitu memilih nama skema dan mengirim penghapusan nama skema yang terpilih. Pada gambar 3.20 komponen *table* digunakan untuk memilih nama skema dan komponen *button* berikon tanda minus(-) digunakan untuk menghapus data nama skema terpilih.

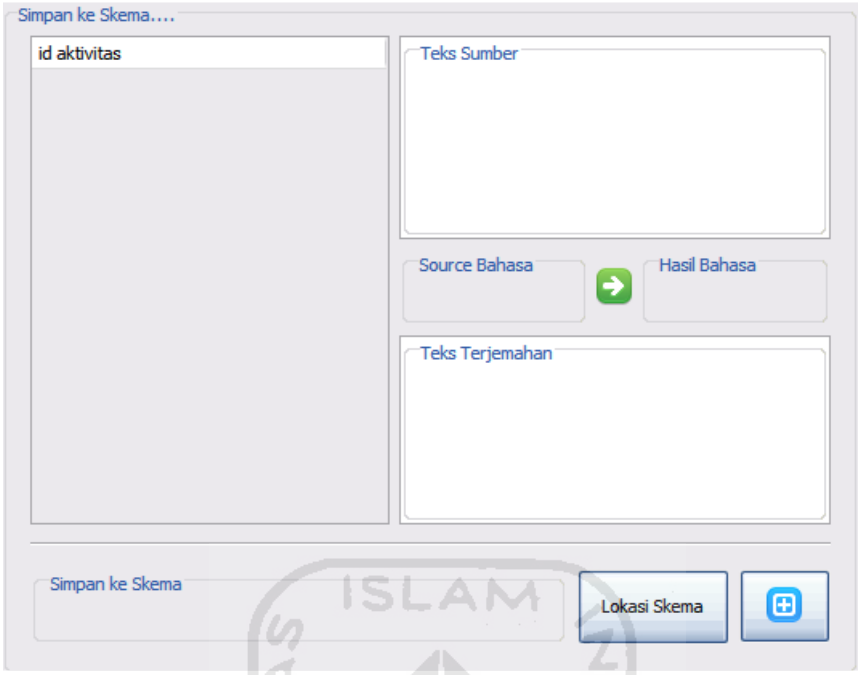
Gambar 3. 20 Disain Antarmuka proses Hapus Skema

3.2.2.3 Disain Antarmuka proses Manajemen Data Skema

Pada proses Manajemen Data Skema terdapat empat buah proses yaitu Simpan Data Skema, Hapus Data Skema, Edit Data Skema, dan Copy Data Skema. Berikut disain antarmuka berdasarkan proses.

a. Simpan Data Skema

Proses Simpan Data Skema membutuhkan interaksi sebelumnya yakni proses Terjemah Bahasa/Simpan Aktivitas *Temporary* karena membutuhkan data *Temporary* untuk dimasukkan ke data skema. Pengguna harus memilih data *Temporary* dan selanjutnya memilih nama skema untuk menyimpan data tersebut. Gambar 3.12 menjelaskan antarmuka untuk pemilihan data *temporary* menggunakan komponen *table* dengan output spesifikasi pada komponen *Text Area* dan *Text Field*. Untuk pemilihan nama skema menggunakan *button* “Lokasi Skema” dan untuk mengirim penambahan data menggunakan *button* belogo tanda plus(+).



Gambar 3. 21 Disain Antarmuka proses Simpan Data Skema

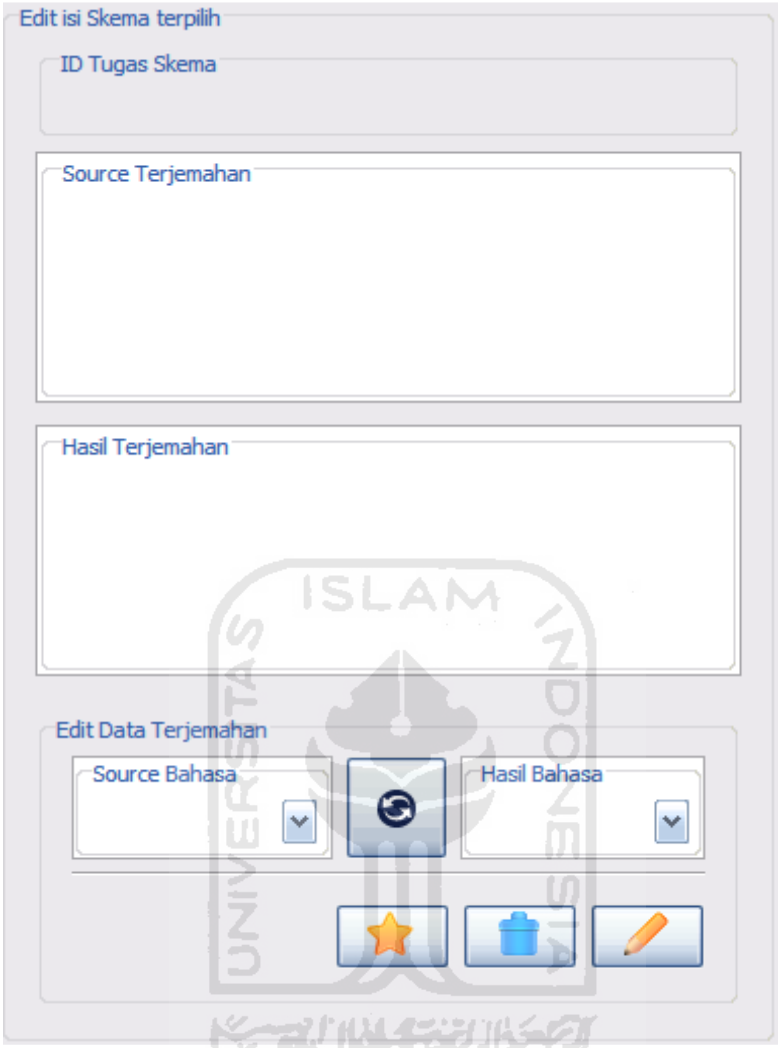
b. Hapus Data Skema

Proses Hapus Data Skema membutuhkan interaksi pengguna dalam memilih nama skema terkait untuk menampilkan data skema dan setelah itu memilih data yang akan dihapus. Pada gambar 3.22, pemilihan nama skema dilakukan dengan *button* "pilih nama skema", pemilihan spesifikasi data nama skema dengan menggunakan *table*, dan untuk mengirim pemrosesan penghapusan data menggunakan *button* berikon tanda minus (-).

Gambar 3. 22 Disain Antarmuka proses Hapus Data Skema

c. Edit Data Skema

Sebelum berinteraksi dengan proses Edit Data Skema, pengguna harus terlebih dahulu memilih nama skema dan data skema. Pada proses ini, pengguna diperbolehkan untuk mengubah data seperti teks sumber, bahasa sumber, dan bahasa terjemahan untuk kemudian dilakukan penerjemahan lagi. Jika hasil sudah tepat, pengguna dapat mengirim perubahan data tersebut. Pada gambar 3.23 pengubahan data teks sumber dan teks terjemahan menggunakan komponen *Text Area.* Pengubahan bahasa sumber dan bahasa terjemahan menggunakan komponen *button* dan *combo box.* Interaksi penterjemahan ubahan data menggunakan *button* berikon bintang dan interaksi mengirim ubahan data menggunakan *button* berikon pensil.



Gambar 3. 23 Disain Antarmuka proses Edit Data Skema

d. Copy Data Skema

Pada proses Copy Data Skema, interaksi yang harus dilakukan adalah memilih nama skema asal dan setelah itu memilih nama skema tujuan atau data skema asal secara bergantian. Jika pemilihan sudah lengkap maka pengguna dapat mengirim penyalinan data dari skema asal ke skema tujuan. Pada gambar 3.24 pemilihan skema asal dan skema tujuan menggunakan komponen *button*. Pemilihan data skema asal menggunakan komponen *table*. Interaksi yang terakhir yaitu mengirim penyalinan data skema asal ke tujuan menggunakan *button* berikon berkas.

Gambar 3. 24 Disain Antarmuka proses Copy Data Skema

BAB IV

HASIL DAN PEMBAHASAN

4.1 Batasan Implementasi

Batasan Implementasi meliputi perangkat lunak dan perangkat lunak yang diperlukan agar Aplikasi Penerjemah Multi Bahasa dapat digunakan dan berfungsi sebagaimana mestinya.

4.1.1 Perangkat Lunak yang digunakan

Penggunaan perangkat lunak dibagi menjadi dua yaitu untuk kebutuhan pengembangan sistem(kakas) dan kebutuhan pengujian sistem. Perangkat lunak yang dibutuhkan untuk pengembangan aplikasi Penerjemah Bahasa adalah :

a. Integrated Development Environment(IDE) Netbeans 6.5

IDE Netbeans 6.5 mempunyai beberapa fitur untuk mendisain, menulis, dan mengeksekusi aplikasi penerjemah multi bahasa. Dalam hal rekayasa antarmuka aplikasi secara visual, Netbeans mempunyai fitur GUI Builder untuk framework Swing pada Java. Sedangkan dalam perancangan UML untuk kebutuhan laporan digunakan Netbeans Plugin UML version 2.0 yang dapat diunduh dan diinstall secara gratis melalui IDE tersebut.

b. Basis Data Objek db4o 7.4

db4o merupakan mesin basis data objek yang bersifat *Open Source* atau terbuka. db4o memberikan akses manajemen data yang cepat, ringan, dan *embedeed*. Dalam rekayasa aplikasi penerjemah bahasa pustaka db4o diintegrasikan ke aplikasi.

c. Java Development Kit(J2SE) 6 Update 20

*Java Development Kit (*J2SE*) 6 Update 20* adalah perangkat lunak yang digunakan untuk merekayasa aplikasi Java. Karena aplikasi yang akan direkayasa adalah aplikasi desktop, maka digunakan paket J2SE.

d. Pustaka Google *Translate* 0.92

Mesin penerjemah bahasa yang digunakan adalah Google *Translate* versi 0.92. Pustaka yang berupa konsep *Web Service* ini digunakan untuk melakukan *request source text* dan mendapatkan *response* berupa *translate* Text.

Perangkat lunak yang dibutuhkan untuk pengujian sistem adalah :

1. Sistem Operasi mendukung Java Runtime Environment versi 1.6 ke atas.
2. Java Runtime Environment versi 1.6 ke atas.
3. Koneksi Internet.

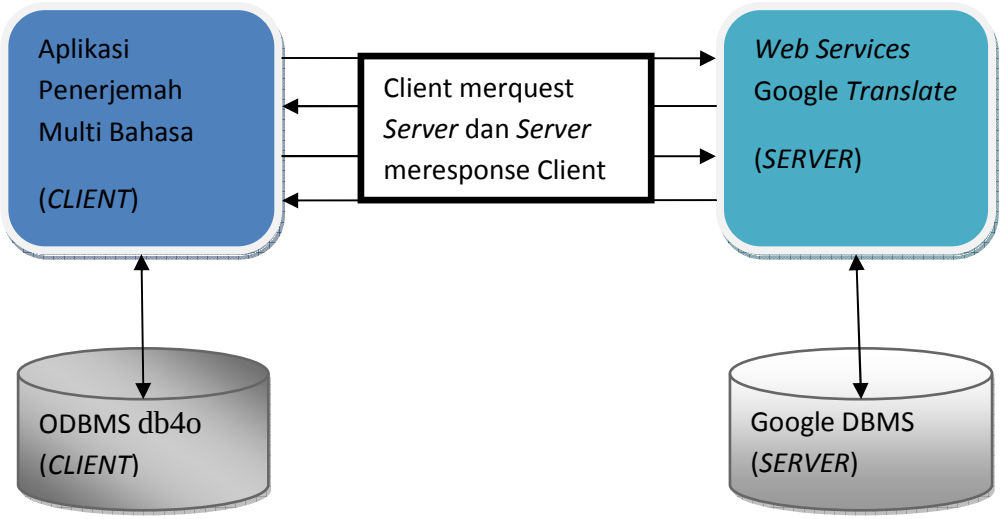
4.1.2 Perangkat Keras yang digunakan

Perangkat keras yang digunakan untuk pengembangan dan pengujian Aplikasi Penerjemah Multi Bahasa adalah :

1. *Monitor* minimal VGA atau SVGA
2. Prosessor Pentium IV 1.7 GHz atau lebih tinggi
3. *RAM* minimal 512 MB
4. *Hardisk* minimal 20 GB
5. *Mouse dan keyboard*

4.2 Implementasi Sistem

Aplikasi Penerjemah Multi Bahasa dikembangkan dengan menggunakan teknologi Google *Web Services* yaitu Google *Translate*. Aplikasi ini bertindak sebagai *client* yang meminta layanan fungsi penerjemahan ke *server* Google. *Server* Google akan menjawab dengan memberikan hasil penerjemah yang dapat dibaca pada aplikasi. Gambar 4.1 menunjukkan topologi implementasi Aplikasi Penerjemah Multi Bahasa.



Gambar 4. 1 Topologi Impelementasi Sistem Aplikasi Penerjemah Multi Bahasa

Pada gambar 4.1 dapat dijelaskan secara kronologis yaitu :

1. *Client mengirimkan Request ke Server.*
2. *Server menerima request dan melakukan query data sesuai request*
3. *Server memberikan response ke Client*
4. *Client menerima response dan melakukan parsing*
5. *Client menyimpan hasil parsing ke dalam ODBMS db4o.*

Implementasi ODBMS db4o pada Aplikasi Penerjemah Multi Bahasa diterapkan di 9 dari 10 proses yang terdapat di aplikasi. Proses aplikasi yang dijadikan media penerapan db4o yaitu :

a. Simpan Temporary Terjemahan

Proses ini menyimpan data terjemahan sementara ke dalam database. Dalam Implementasi, proses ini terdapat di kelas *TempCRUD* khususnya metode *insert* dan juga mendayagunakan kelas *Aktivitas* sebagai model. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (2) kelas *Aktivitas* dan lampiran 2 (2) kelas *TempCRUD*

b. Hapus Temporary Terjemahan

Proses ini menghapus data terjemahan sementara pada database. Dalam Implementasi, proses ini terdapat di kelas *TempCRUD* khususnya metode *delete* dan juga mendayagunakan kelas *Aktivitas* sebagai model. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (2) kelas *Aktivitas* dan lampiran 2 (2) kelas *TempCRUD.*

c. Buat Skema

Proses ini menyimpan skema ke dalam database. Dalam Implementasi, proses ini terdapat di kelas *ListSkemaCRUD* khususnya metode *insert* dan juga mendayagunakan kelas *ListSkemaProyek* sebagai model. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (4) kelas *ListSkemaProyek* dan lampiran 2 (4) kelas *ListSkemaCRUD.*

d. Edit Skema

Proses ini mengubah skema ke dalam database. Dalam Implementasi, proses ini terdapat di kelas *ListSkemaCRUD* khususnya metode *update* dan juga mendayagunakan kelas *ListSkemaProyek* sebagai model. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (4) kelas *ListSkemaProyek* dan lampiran 2 (4) kelas *ListSkemaCRUD.*

e. Hapus Skema

Proses ini menghapus skema ke dalam database. Dalam Implementasi, proses ini terdapat di kelas *ListSkemaCRUD* khususnya sekumpulan metode delete(*delete ,delete2, dan delete3)* dan juga mendayagunakan kelas *ListSkemaProyek* sebagai model. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (4) kelas *ListSkemaProyek* dan lampiran 2 (4) kelas *ListSkemaCRUD.*

f. Simpan Data Skema / Copy Data Skema

Proses ini menyimpan data skema ke dalam database. Dalam Implementasi, proses ini terdapat di dua kelas yaitu *ActivityCRUD*(metode *insert)* dan *SkemaCRUD*(metode *insert*). Untuk kelas modelnya menggunakan kelas *Aktivitas* dan *SkemaProyek*. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (2) kelas *Aktivitas,* lampiran 1 (3) kelas

SkemaProyek, lampiran 2 (1) kelas *ActivityCRUD*, dan lampiran 2 (3) kelas *SkemaCRUD.*

g. Hapus Data Skema

Proses ini menghapus data skema pada database. Dalam Implementasi, proses ini terdapat di dua kelas yaitu *ActivityCRUD*(metode *delete)* dan *SkemaCRUD*(metode *delete*). Untuk kelas modelnya menggunakan kelas *Aktivitas* dan *SkemaProyek*. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (2) kelas *Aktivitas*, lampiran 1 (3) kelas *SkemaProyek*, lampiran 2 (1) kelas *ActivityCRUD*, dan lampiran 2 (3) kelas *SkemaCRUD.*

h. Edit Data Skema

Proses ini mengubah data skema pada database. Dalam Implementasi, proses ini terdapat di kelas *ActivityCRUD*(metode *update)* dan menggunakan kelas *Aktivitas* sebagai model. Implementasinya dalam kode sumber bahasa Java tertulis pada lampiran 1 (2) kelas *Aktivitas*, lampiran 1 (3) kelas *SkemaProyek*, lampiran 2 (1) kelas *ActivityCRUD*, dan lampiran 2 (3) kelas *SkemaCRUD.*

Aplikasi Penerjemah Multi Bahasa menawarkan dukungan terjemahan berjumlah sedikitnya 35 bahasa. Berikut tabel Daftar Bahasa

Tabel 4. 1 Daftar bahasa yang didukung Aplikasi Penerjemah Multi bahasa

Afrikaans	English	Icelandic	Norwegian	Swedish
Albanian	Estonian	Indonesian	Polish	Turkish
Catalan	Filipino	Irish	Portuguese	Vietnamese
Croatian	Finnish	Italian	Romanian	Welsh
Czech	French	Latvian	Slovak	Swahili
Danish	Galician	Lithuanian	Slovenian	Maltese
Dutch	German	Malay	Spanish	Hungarian

4.2.1 Implementasi Antarmuka Penerjemah Bahasa

Implementasi Antarmuka Penerjemah bahasa, dibagi menjadi 2 bagian sesuai dengan proses terkait. Antarmuka yang terkait proses Penerjemah Bahasa adalah Antarmuka Proses Penerjemah Bahasa/Simpan *Temporary* dan Antarmuka Proses Hapus *Temporary*.

4.2.1.1 Implementasi Antarmuka Proses Penerjemah Bahasa

Pada antarmuka gambar 4.2 pengguna dapat berinteraksi untuk menggunakan fungsi penerjemah bahasa dengan cara mengisikan teks sumber pada Text Area yang tersedia, memilih opsi penerjemah pada pada panel “Option Penerjemah Bahasa”, dan melakukan proses terjemahan dengan mengklik tombol berikon bintang.

Gambar 4. 2 Antarmuka Proses Penerjemah Bahasa

4.2.1.2 Impelementasi Antarmuka Hapus *Temporary*

Pada antarmuka gambar 4.3 terdapat *dialog* berisi data terjemahan *Temporary* yang dapat diakses pengguna dan menampilkan datanya pada *Text Area* di bawahnya. Setelah pengguna mengakses data terjemahan *Temporary* tersedia fitur mencetak data terjemahan *Temporary* ke file *PDF* dengan mengklik *button* berikon dokumen *PDF* atau menghapus spesifikasi data terjemahan *Temporary* dengan mengklik button berikon tanda minus(-).

Gambar 4. 3 Antarmuka Proses Hapus *Temporary*

4.2.2 Implementasi Antarmuka Manajemen Skema

Implementasi Antarmuka Manajemen Skema, dibagi menjadi 3 bagian sesuai dengan proses terkait. Antarmuka yang terkait proses Manajemen Skema adalah Antarmuka Buat Skema, Antarmuka Hapus Skema, dan Antarmuka Edit Skema.

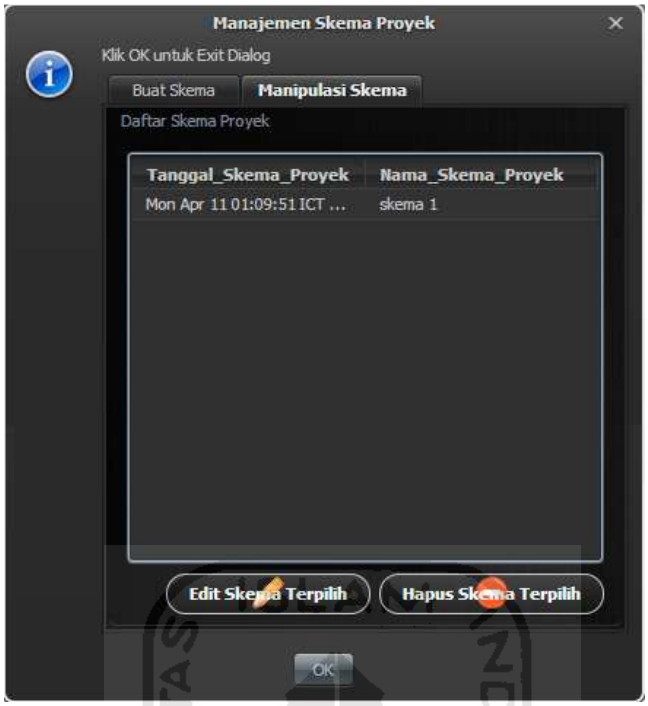
4.2.2.1 Implementasi Antarmuka Buat Skema

Pada antarmuka gambar 4.4 pengguna dapat membuat sebuah skema untuk menyimpan data terjemahan secara kelompok. Pengguna dapat mengisikan nama skema pada *Text Field* yang tersedia dan jika terdapat redudansi nama skema maka pengguna akan diberitahu terlebih dahulu dengan fitur pengecekan per kata dan hasilnya ada pada *Text Field* di bawahnya. Untuk melakukan pembuatan skema, pengguna dapat mengklik *button* bertuliskan “buat”

Gambar 4. 4 Antarmuka Proses Buat Skema

4.2.2.2 Implementasi Antarmuka Hapus Skema

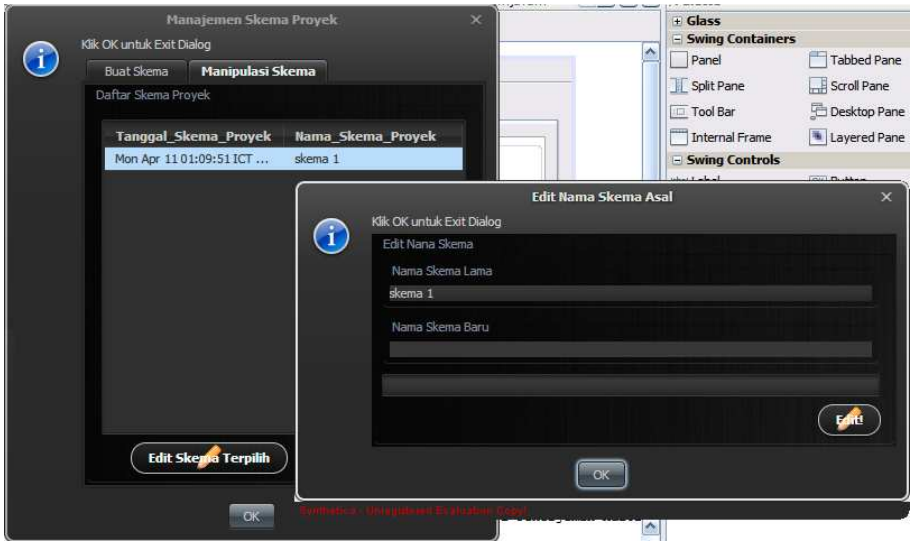
Pada antarmuka gambar 4.5 pengguna dapat menghapus atau mengubah nama skema yang tersimpan pada basis data. Pengguna dapat mengakses *table* data nama skema untuk memilih skema yang akan di ubah atau dihapus. Setelah skema terpilih, pengguna dapat mengubah nama skema dengan mengklik *button* berikon pinsil atau menghapus data dengan mengklik *button* berikon tanda minus (-).



Gambar 4. 5 Antamuka Proses Hapus Skema

4.2.2.3 Implementasi Antarmuka Edit Skema

Pada antarmuka gambar 4.6 sebelumnya pengguna mengklik *button* berikon pinsil pada antarmuka gambar 4.5 sebelumnya. Setelah muncul *dialog* edit nama skema, pengguna dapat mengubah nama skema dan jika terdapat redudansi nama skema maka pengguna akan diberitahu terlebih dahulu dengan fitur pengecekan per kata dan hasilnya ada pada *Text Field* di bawahnya. Pengubahan data nama skema dilakukan dengan mengklik *button* berikon pinsil.



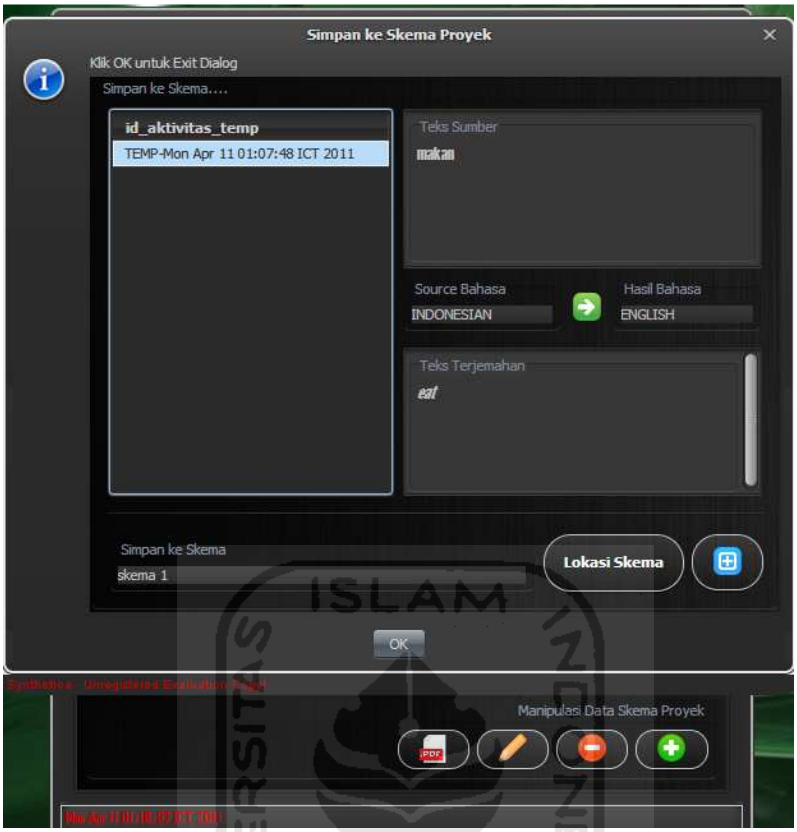
Gambar 4. 6 Antarmuka Proses Edit Skema

4.2.3 Implementasi Antarmuka Manajemen Data Skema

Implementasi Antarmuka Manajemen Data Skema, dibagi menjadi 4 bagian sesuai dengan proses terkait. Antarmuka yang terkait proses Manajemen Data Skema adalah Antarmuka Simpan Data Skema, Antarmuka Hapus Data Skema, Antarmuka Edit Data Skema, dan Antarmuka Copy Data Skema.

4.2.3.1 Implementasi Antarmuka Simpan Data Skema

Pada antamuka gambar 4.7 sebelumnya pengguna mengakses *tab* manajemen data skema dan mengklik *button* berikon tanda plus(+). Pengguna dapat memilih *table* data terjemahan *Temporary* yang akan dimasukkan ke basis data sesuai nama skema yang dipilih. Selanjutnya pengguna dapat memilih nama skema dengan cara mengklik *button* “Lokasi Skema” dan melakukan penambahan data dengan mengklik *button* tanda plus(+).



Gambar 4. 7 Antarmuka Proses Simpan Data Skema

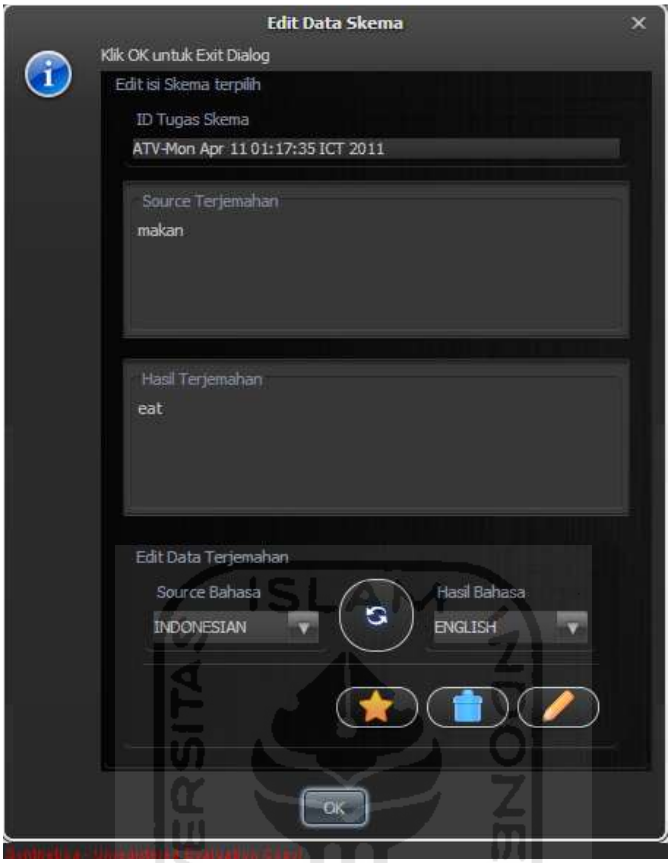
4.2.3.2 Impelementasi Antarmuka Hapus Data Skema

Pada antarmuka gambar 4.8 pengguna dapat memilih nama skema dengan mengklik *button* "Pilih Nama Skema". Jika nama skema yang dipilih berelasi dengan data terjemahan tersimpan maka data terjemahan akan tampil pada *table* dan pengguna dapat mengubah, menghapus, dan mencetak file *PDF* data terjemahan tersebut. Untuk menghapus data atau mencetak file *PDF* pengguna memilih data terjemahan pada *table* lalu mengklik *button* berikon tanda minus(-) untuk menghapus spesifikasi data terkait atau mengklik *button* berikon dokumen *PDF* untuk mencetak spesifikasi data terkait ke file PDF.

Gambar 4. 8 Antarmuka Proses Hapus Data Skema

4.2.3.3 Impelementasi Antarmuka Edit Data Skema

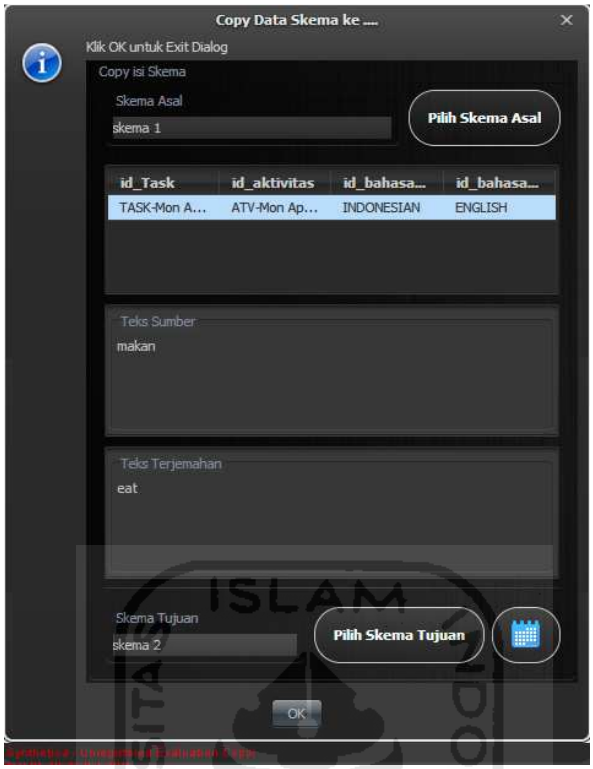
Pada antarmuka gambar 4.9 sebelumnya pengguna mengakses *tab* manajemen data skema, memilih spesifikasi data skema pada *table*, dan mengklik *button* berikon pinsil. Selanjutnya pengguna dapat mengubah data terjemahan pada skema tersebut dengan mengisi teks sumber pada *Text Area*, memilih bahasa sumber serta bahasa terjemahan pada *combo box* yang disediakan, dan melakukan penerjemahan bahasa kembali dengan mengklik *button* berikon bintang. Untuk mengubah data terjemahan, pengguna dapat mengklik *button* berikon pinsil.



Gambar 4. 9 Antarmuka Proses Edit Data Skema

4.2.3.4 Implementasi Antarmuka Copy Data Skema

Pada antarmuka gambar 4.10 penguna dapat menyalin data terjemahan. Pengguna memilih nama skema asal dengan mengklik *button* bertuliskan "Pilih Skema Asal" dan memilih nama skema tujuan dengan mengklik button bertuliskan "Pilih Skema Tujuan". Untuk menyalin data terjemahan skema asal ke skema tujuan, pengguna dapa mengakses *table* data terjemahan skema asal dan melakukan penyalinan data terjemahan dengan mengklik *button* berikon berkas.



Gambar 4. 10 Antarmuka Proses Copy Data Skema

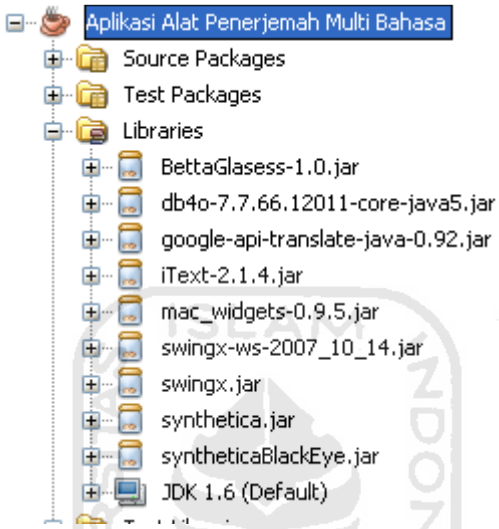
4.3 Pengujian Aplikasi Penerjemah Multi Bahasa

Pada tahap ini akan dilakukan pengujian perangkat lunak aplikasi penerjemah multi bahasa. Pengujian dilakukan untuk memastikan perangkat lunak berjalan sesuai dengan yang diharapkan. Dalam melakukan pengujian diharapkan semua kesalahan dapat ditemukan untuk diperbaiki shingga kesalahan dari *system* dapat diminimalisasi atau bahkan dihilangkan dari *system*.

Pengujian pertama adalah menguji penerapan fleksibilitas mesin basis data berorientasi objek pada Aplikasi Penerjemah Multi Bahasa. Berikut adalah beberapa gambar pengujian fleksibilitas basis data berorientasi objek.

Pada **Gambar 4.11** dijelaskan bahwa rekayasa Aplikasi Penerjemah Multi Bahasa pada IDE Netbeans 6.9.1 menggunakan pustaka basis data berorientasi objek yaitu db4o tanpa menggunakan pustaka basis data lain seperti MySql. Selanjutnya pada **Gambar 4.12** menjelaskan ekstensi file basis data db4o yang digunakan pada aplikasi ini adalah *.ODB*. File *.ODB* ini tertanam pada aplikasi

karena berada dalam satu folder dengan file eksekutor aplikasi. Jika File *.ODB* hilang atau terhapus pada folder aplikasi maka aplikasi akan membuat kembali file tersebut.



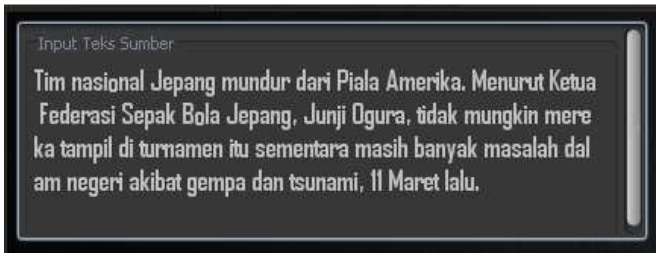
Gambar 4. 11 Spesifikasi Pustaka Rekayasa Aplikasi Penerjemah Bahasa

Gambar 4. 12 Spesifikasi Folder Aplikasi Penerjemah Multi Bahasa

4.3.1 Pengujian Proses Penerjemah Bahasa

Pengujian Penerjemah Bahasa meliputi sub-proses Terjemah Bahasa, Simpan *Temporary* Terjemahan dan Hapus *Temporary* Terjemahan. Pengujian dimulai dengan menerjemahkan Teks Sumber berbahasa Indonesia :

"Tim nasional Jepang mundur dari Piala Amerika. Menurut Ketua Federasi Sepak Bola Jepang, Junji Ogura, tidak mungkin mereka tampil di turnamen itu sementara masih banyak masalah dalam negeri akibat gempa dan tsunami, 11 Maret lalu.".



Gambar 4. 13 Pengujian Input Teks Sumber Bahasa Indonesia

Teks Sumber berbahasa Indonesia pada gambar 4.13 akan diterjemahkan ke beberapa bahasa, yakni bahasa Inggris, Perancis dan Belanda.

Gambar 4. 14 Hasil Pengujian Data Terjemahan

Setelah Teks Terjemahan dhasilkan seperti pada gambar 4.14 maka data tersebut otomatis tersimpan dalam basis data *TempStore.ODB*. Selanjutnya data-data tersebut dapat dikelola sesuai kebutuhan pengguna.

4.3.2 Pengujian Proses Manajemen Skema

Pengujian Manajemen Skema meliputi sub-proses Buat Skema, Edit Skema, dan Hapus Skema. Dialog Manajemen Skema dapat diakses melalui komponen *Menu File* atau *button* berikon orang pada *tab* Manajemen Data Skema. Pengujian dimulai dengan membuat skema dengan nama "*Terjemahan 1*". Berikut pengujian buat skema

Gambar 4. 15 Pengujian Buat Skema

Setelah membuat nama skema dalam basis data seperti gambar 4.15, skema dapat dilakukan manipulasi seperti edit skema dan hapus skema. Berikut gambar 4.16 pengujian edit skema.

Gambar 4. 16 Pengujian Edit Skema

4.3.3 Pengujian Proses Manajemen Data Skema

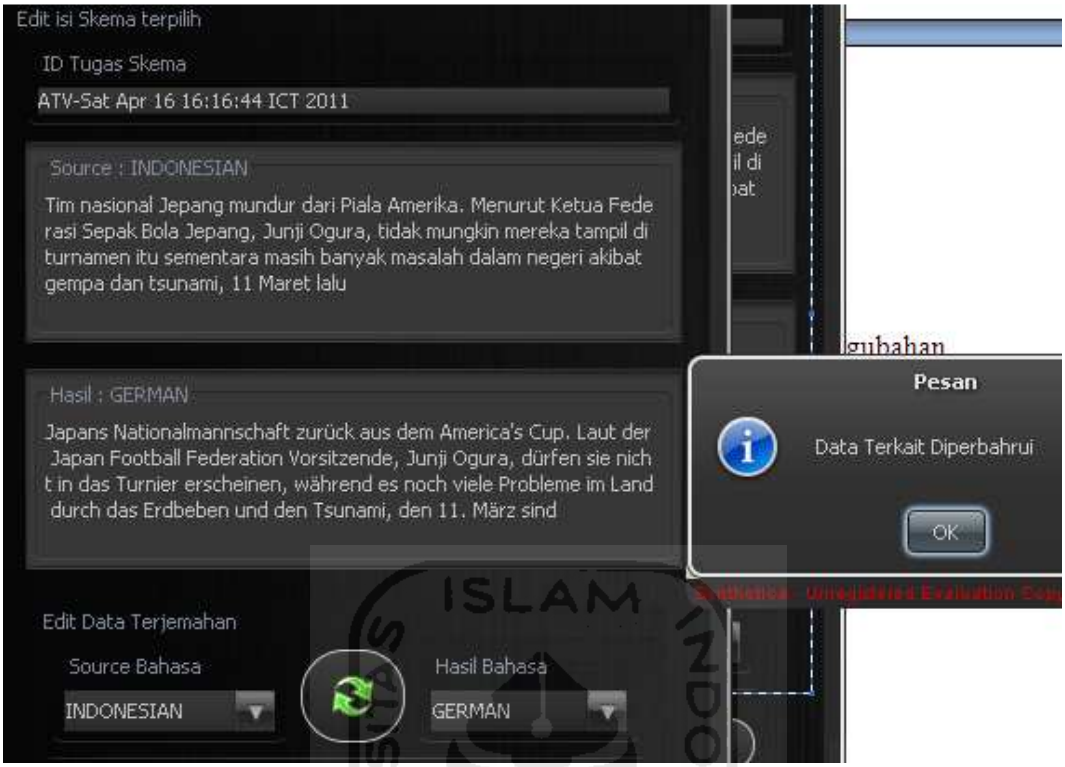
Pengujian Manajemen Data Skema meliputi sub-proses Simpan Data Skema, Edit Data Skema, Hapus Data Skema, dan Copy Data Skema. Pengujian dimulai dengan menyimpan data terjemahan yang bersumber dari *Temporary* data terjemahan ke skema "*Terjemahan 1*".

Gambar 4. 17 Pengujian Simpan Data Skema

Setelah data tersimpan pada skema seperti gambar 4.17, data dapat dilihat pada *tab* manajemen data skema. Berikut gambar 4.18 hasil pengujian simpan data skema.

Gambar 4. 18 Hasil Pengujian Simpan Data Skema

Setelah data tersimpan pada skema, dapat diubah dan dihapus. Pengujian selanjutnya adalah edit data skema. Data terjemahan yang diubah adalah Bahasa Terjemahan dari Bahasa Belanda ke Bahasa Jerman.



Gambar 4. 19 Pengujian Edit Data Skema

Setelah data skema terkait diubah seperti gambar 4.19, data dapat dilihat pada *tab* manajemen data skema. Berikut gambar 4.20 hasil pengujian edit data skema.

Gambar 4. 20 Hasil Pengujian Edit Data Skema

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah :

1. Basis data berorientasi objek merupakan salah satu solusi media penyimpanan data yang tepat untuk ditanam pada aplikasi *client* untuk melengkapi aplikasi *server* yang media penyimpanan datanya berupa basis data dengan berbagai macam tipe.
2. Aplikasi Penerjemah Multi Bahasa yang dibangun bersifat multi *platform* dan sudah tepat menggunakan media penyimpanan data tertanam yang tanpa perlu mengkonfigurasi ulang basis datanya apabila berpindah *platform.*

5.2 Saran

Saran pengembangan penelitian ini di masa yang akan datang adalah sebagai berikut.

1. Penelitian ini hanya membahas teknologi *nosql* untuk kelompok basis data berorientasi objek. Penulis berharap pengembangan ke depannya dapat membahas teknologi *nosql* lainya seperti basis data berorientasi dokumen, graph, dan XML.
2. Aplikasi Penerjemah Multi Bahasa yang dibangun masih memerlukan pengembangan ke depannya, yakni meliputi pemutakhiran versi pustaka yang digunakan, penambahan jumlah dukungan bahasa, dan rekayasa Aplikasi Penerjemah Multi Bahasa untuk versi *mobile*.

DAFTAR PUSTAKA

Abreu, Poels, Sahraoui, Zuse. 2003. *Quantitive Approaches in Object-Oriented Software Engineering*. Kogan Page Ltd.

Grehan. 2005. *Introduction to ODBMS*. Available at http://www.odbms.org

Greene. 2006. *OODBMS Architectures September*. Available at http://www.odbms.org.

Hutchins, Somers. 1992. *An introduction to machine translation*. Available at http://www.hutchinsweb.me.uk/IntroMT-TOC.htm.

Irwanto, Djon. 2005. *Perancangan Object Oriented Software dengan UML*. Andi, Yogyakarta.

Irwanto, Djon. 2007. *Membangun Object Oriented Software dengan Java dan Object Database*. Andi, Yogyakarta.

Irwanto, Djon. 2010. *Refactoring pada Object Oriented Software dan Object Database*. Andi, Yogyakarta.

Paterson, Edlich, Hörning. 2006. *The Definitive Guide to db4o*. Appress.

Rosenberger. 2008. *Tech View Product Report: db4o*. Available at http://www.odbms.org.

Sabau. 2007. *Comparison of RDBMS, OODBMS, and ORDBMS*. Informatica economica, 4(44):83.

Tim db4o. 2004. *Db4o product information*. Available at http://www.db4o.com

Tim Penyusun. 2009. *Modul Praktikum Rekayasa Perangkat Lunak 2008/2009*. Laboratorium Sistem Informasi dan Rekayasa Perangkat Lunak, Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta.

LAMPIRAN 1

PACKAGE DomainModel

(1) Class Bahasa

```
package DomainModel;

import com.google.api.translate.Language;

public class Bahasa {

    private String idbahasa;
    private Language namabahasa;

    public Bahasa(String idbahasa, Language namabahasa) {
        this.idbahasa = idbahasa;
        this.namabahasa = namabahasa;
    }

    /**
     * @return the idbahasa
     */
    public String getIdbahasa() {
        return idbahasa;
    }

    /**
     * @param idbahasa the idbahasa to set
     */
    public void setIdbahasa(String idbahasa) {
        this.idbahasa = idbahasa;
    }

    /**
     * @return the namabahasa
     */
    public Language getNamabahasa() {
        return namabahasa;
    }

    /**
     * @param namabahasa the namabahasa to set
     */
    public void setNamabahasa(Language namabahasa) {
        this.namabahasa = namabahasa;
    }
}
```

(2) Class Aktivitas

```
package DomainModel;

import java.util.Date;
```

```
public class Aktivitas {

    private String idaktivitas;
    private String idbahasasumber;
    private String idbahasaterjemahan;
    private String teksterjemahan;
    private String tekssumber;

    /**
     * @return the idaktivitas
     */
    public String getIdaktivitas() {
        return idaktivitas;
    }

    /**
     * @param idaktivitas the idaktivitas to set
     */
    public void setIdaktivitas(String idaktivitas) {
        this.idaktivitas = idaktivitas;
    }

    /**
     * @return the idbahasasumber
     */
    public String getIdbahasasumber() {
        return idbahasasumber;
    }

    /**
     * @param idbahasasumber the idbahasasumber to set
     */
    public void setIdbahasasumber(String idbahasasumber) {
        this.idbahasasumber = idbahasasumber;
    }

    /**
     * @return the idbahasaterjemahan
     */
    public String getIdbahasaterjemahan() {
        return idbahasaterjemahan;
    }

    /**
     * @param idbahasaterjemahan the idbahasaterjemahan to set
     */
    public void setIdbahasaterjemahan(String idbahasaterjemahan) {
        this.idbahasaterjemahan = idbahasaterjemahan;
    }

    /**
     * @return the teksterjemahan
     */
    public String getTeksterjemahan() {
        return teksterjemahan;
    }
```

```
    /**
     * @param teksterjemahan the teksterjemahan to set
     */
    public void setTeksterjemahan(String teksterjemahan) {
        this.teksterjemahan = teksterjemahan;
    }

    /**
     * @return the tekssumber
     */
    public String getTekssumber() {
        return tekssumber;
    }

    /**
     * @param tekssumber the tekssumber to set
     */
    public void setTekssumber(String tekssumber) {
        this.tekssumber = tekssumber;
    }
}
```

(3) Class SkemaProyek

```
package DomainModel;

import java.util.Date;

public class SkemaProyek {

    private String idtask;
    private String idaktivitas;
    private Date tglskemaproyek;

    /**
     * @return the idtask
     */
    public String getIdtask() {
        return idtask;
    }

    /**
     * @param idtask the idtask to set
     */
    public void setIdtask(String idtask) {
        this.idtask = idtask;
    }

    /**
     * @return the tglskemaproyek
     */
    public Date getTglskemaproyek() {
        return tglskemaproyek;
    }
```

```
    /**
     * @param tglskemaproyek the tglskemaproyek to set
     */
    public void setTglskemaproyek(Date tglskemaproyek) {
        this.tglskemaproyek = tglskemaproyek;
    }

    /**
     * @return the idaktivitas
     */
    public String getIdaktivitas() {
        return idaktivitas;
    }

    /**
     * @param idaktivitas the idaktivitas to set
     */
    public void setIdaktivitas(String idaktivitas) {
        this.idaktivitas = idaktivitas;
    }
}
```

(4) Class ListSkemaProyek

```
package DomainModel;

import java.util.Date;

public class ListSkemaProyek {

    private Date tglskemaProyek;
    private String namaskemaproyek;

    public Date getTglskemaProyek() {
        return tglskemaProyek;
    }

    /**
     * @param tglskemaProyek the tglskemaProyek to set
     */
    public void setTglskemaProyek(Date tglskemaProyek) {
        this.tglskemaProyek = tglskemaProyek;
    }

    /**
     * @return the namaskemaproyek
     */
    public String getNamaskemaproyek() {
        return namaskemaproyek;
    }

    public void setNamaskemaproyek(String namaskemaproyek) {
        this.namaskemaproyek = namaskemaproyek;
    }
}
```

LAMPIRAN 2

PACKAGE ODBMSAccess

(1) Class ActivityCRUD

```
package ODBMSAccess;

import DomainModel.Aktivitas;
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import com.db4o.query.Query;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;

public class ActivityCRUD {

    public void insert(String idaktivitas, String idbahasasumber,
String   idbahasaterjemahan,   String   teksterjemahan,   String
tekssumber) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Aktivitas ATV = new Aktivitas();
            ATV.setIdaktivitas(idaktivitas);
            ATV.setIdbahasasumber(idbahasasumber);
            ATV.setIdbahasaterjemahan(idbahasaterjemahan);
            ATV.setTekssumber(tekssumber);
            ATV.setTeksterjemahan(teksterjemahan);
            db.store(ATV);
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,   "error   ="   +
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void update(String idaktivitas, String idbahasasumber,
String   idbahasaterjemahan,   String   teksterjemahan,   String
tekssumber) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(idaktivitas);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                //db.delete(hasil.next());
                Aktivitas ATV = (Aktivitas) hasil.next();
                ATV.setIdbahasasumber(idbahasasumber);
```

```
                ATV.setIdbahasaterjemahan(idbahasaterjemahan);
                ATV.setTekssumber(tekssumber);
                ATV.setTeksterjemahan(teksterjemahan);
                db.store(ATV);
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void delete(String idaktivitas) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(idaktivitas);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                db.delete(hasil.next());
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil_Cari1(String idbahasa, String idterjemahan,
JTable TempTable, JScrollPane jscrollPane) {
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);

qry.descend("idbahasasumber").constrain(idbahasa).like().and(qry.d
escend("idbahasaterjemahan").constrain(idterjemahan).like());
            ObjectSet hasil = qry.execute();
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][3],
                    new String[]{
                        "id",                   "id_bahasa_sumber",
"id_bahasa_terjemahan"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false, false
                };
```

```
                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            //return TempTable.getModel();
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                //Aktivitas data1 = (Aktivitas) hasil1.next();
                Aktivitas data = (Aktivitas) hasil.next();
                TempTable.setValueAt(data.getIdaktivitas(), i, 0);
                TempTable.setValueAt(data.getIdbahasasumber(),  i,
1);
                TempTable.setValueAt(data.getIdbahasaterjemahan(),
i, 2);
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil_Cari2(String idbahasa, String idterjemahan,
JTable TempTable, JScrollPane jscrollPane) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);

qry.descend("idbahasasumber").constrain(idbahasa).like().and(qry.d
escend("idbahasaterjemahan").constrain(idterjemahan).like());
            ObjectSet hasil = qry.execute();
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][3],
                    new String[]{
                        "id",                "id_bahasa_sumber",
"id_bahasa_terjemahan"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false, false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            //return TempTable.getModel();
```

```
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                //Aktivitas data1 = (Aktivitas) hasil1.next();
                Aktivitas data = (Aktivitas) hasil.next();
                TempTable.setValueAt(data.getIdaktivitas(), i, 0);
                TempTable.setValueAt(data.getIdbahasasumber(),  i,
1);
                TempTable.setValueAt(data.getIdbahasaterjemahan(),
i, 2);
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil_Cari3(String idbahasa, String idterjemahan,
JTable TempTable, JScrollPane jscrollPane) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        ObjectContainer db2 = Db4o.openFile("TempDataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);

qry.descend("idbahasasumber").constrain(idbahasa).like().and(qry.d
escend("idbahasaterjemahan").constrain(idterjemahan).like());
            ObjectSet hasil = qry.execute();
            //----------------------------
            Query qry2 = db2.query();
            qry2.constrain(Aktivitas.class);

qry2.descend("idbahasasumber").constrain(idbahasa).like().and(qry2
.descend("idbahasaterjemahan").constrain(idterjemahan).like());
            ObjectSet hasil2 = qry2.execute();
            int SIZE = hasil.size() + hasil2.size();
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[SIZE][3],
                    new String[]{
                        "id",                   "id_bahasa_sumber",
"id_bahasa_terjemahan"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false, false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
```

```
            //return TempTable.getModel();
            jscrollPane.setViewportView(TempTable);
            int i = 0;
            while (hasil.hasNext()) {
                Aktivitas data = (Aktivitas) hasil.next();
                TempTable.setValueAt(data.getIdaktivitas(), i, 0);
                TempTable.setValueAt(data.getIdbahasasumber(),  i,
1);
                TempTable.setValueAt(data.getIdbahasaterjemahan(),
i, 2);
                i++;
            }
            int j = i;
            while (hasil2.hasNext()) {
                Aktivitas data = (Aktivitas) hasil2.next();
                TempTable.setValueAt(data.getIdaktivitas(), j, 0);
                TempTable.setValueAt(data.getIdbahasasumber(),  j,
1);
                TempTable.setValueAt(data.getIdbahasaterjemahan(),
j, 2);
                j++;
            }
        } catch (Exception e) {
            db.rollback();
            db2.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
            db2.close();
        }
    }

    public void klik(JTable TempTable, JTextArea A, JTextArea B) {
        String                          del                          =
String.valueOf(TempTable.getValueAt(TempTable.getSelectedRow(),
1));
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {

//db.ext().configure().queries().evaluationMode(QueryEvaluationMod
e.SNAPSHOT);
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(del);
            ObjectSet hasil = qry.execute();
            //int i = 0;
            while (hasil.hasNext()) {
                Aktivitas data = (Aktivitas) hasil.next();
                A.setText(data.getTekssumber());
                B.setText(data.getTeksterjemahan());
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
```

```
        } finally {
            db.close();
        }
    }

    public void klik_table_cari(JTable TempTable, JTextArea A,
JTextArea B) {
        String                          del                          =
String.valueOf(TempTable.getValueAt(TempTable.getSelectedRow(),
0));
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        ObjectContainer db2 = Db4o.openFile("TempDataStore.ODB");
        try {

//db.ext().configure().queries().evaluationMode(QueryEvaluationMod
e.SNAPSHOT);
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(del);
            ObjectSet hasil = qry.execute();
            //int i = 0;
            Query qry2 = db2.query();
            qry2.constrain(Aktivitas.class);
            qry2.descend("idaktivitas").constrain(del);
            ObjectSet hasil2 = qry2.execute();
            while (hasil.hasNext()) {
                Aktivitas data = (Aktivitas) hasil.next();
                A.setText(data.getTekssumber());
                B.setText(data.getTeksterjemahan());
            }
            while (hasil2.hasNext()) {
                Aktivitas data2 = (Aktivitas) hasil2.next();
                A.setText(data2.getTekssumber());
                B.setText(data2.getTeksterjemahan());
            }
        } catch (Exception e) {
            db.rollback();
            db2.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
            db2.close();
        }
    }
}
```

(2) Class TempCRUD

```
package ODBMSAccess;

import DomainModel.Aktivitas;
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import com.db4o.query.Query;
```

```
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class TempCRUD {

    public void insert(String idaktivitas, String idbahasasumber,
String   idbahasaterjemahan,   String   teksterjemahan,   String
tekssumber) {
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {
            Aktivitas ATV = new Aktivitas();
            ATV.setIdaktivitas(idaktivitas);
            ATV.setIdbahasasumber(idbahasasumber);
            ATV.setIdbahasaterjemahan(idbahasaterjemahan);
            ATV.setTekssumber(tekssumber);
            ATV.setTeksterjemahan(teksterjemahan);
            db.store(ATV);
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void update(String idaktivitas, String idbahasasumber,
String   idbahasaterjemahan,   String   teksterjemahan,   String
tekssumber) {
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(idaktivitas);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                //db.delete(hasil.next());
                Aktivitas ATV = (Aktivitas) hasil.next();
                ATV.setIdbahasasumber(idbahasasumber);
                ATV.setIdbahasaterjemahan(idbahasaterjemahan);
                ATV.setTekssumber(tekssumber);
                ATV.setTeksterjemahan(teksterjemahan);
                db.store(ATV);
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
```

```
    }

    public void delete(String idaktivitas) {
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(idaktivitas);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                db.delete(hasil.next());
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil(JTable TempTable, JScrollPane jscrollPane)
{
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {

//db.ext().configure().queries().evaluationMode(QueryEvaluationMod
e.SNAPSHOT);
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").orderAscending();
            ObjectSet hasil = qry.execute();
            //int i = 0;
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][3],
                    new String[]{
                        "id_aktivitas_temp",   "id_bahasa_sumber",
"id_bahasa_terjemahan"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false, false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            //return TempTable.getModel();
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                Aktivitas data = (Aktivitas) hasil.next();
```

```
                TempTable.setValueAt(data.getIdaktivitas(), i, 0);
                TempTable.setValueAt(data.getIdbahasasumber(),  i,
1);
                TempTable.setValueAt(data.getIdbahasaterjemahan(),
i, 2);
                //TempTable.setValueAt(data.getTekssumber(),    i,
3);
                //  TempTable.setValueAt(data.getTeksterjemahan(),
i, 4);
                //i++;
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil2(JTable TempTable, JScrollPane jscrollPane)
{
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {

//db.ext().configure().queries().evaluationMode(QueryEvaluationMod
e.SNAPSHOT);
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").orderAscending();
            ObjectSet hasil = qry.execute();
            //int i = 0;
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][3],
                    new String[]{
                        "id_aktivitas_temp"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            //return TempTable.getModel();
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                Aktivitas data = (Aktivitas) hasil.next();
                TempTable.setValueAt(data.getIdaktivitas(), i, 0);
                //TempTable.setValueAt(data.getTekssumber(),    i,
3);
```

```
                //  TempTable.setValueAt(data.getTeksterjemahan(),
i, 4);
                //i++;
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void klik(JTable TempTable, JTextArea A, JTextArea B) {
        String                          del                          =
String.valueOf(TempTable.getValueAt(TempTable.getSelectedRow(),
0));
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {

//db.ext().configure().queries().evaluationMode(QueryEvaluationMod
e.SNAPSHOT);
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(del);
            ObjectSet hasil = qry.execute();
            //int i = 0;
            while (hasil.hasNext()) {
                Aktivitas data = (Aktivitas) hasil.next();
                A.setText(data.getTekssumber());
                B.setText(data.getTeksterjemahan());
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void klik2(JTable TempTable, JTextArea A, JTextArea B,
JTextField C, JTextField D) {
        String                          del                          =
String.valueOf(TempTable.getValueAt(TempTable.getSelectedRow(),
0));
        ObjectContainer db = Db4o.openFile("TempDataStore.ODB");
        try {

//db.ext().configure().queries().evaluationMode(QueryEvaluationMod
e.SNAPSHOT);
            Query qry = db.query();
            qry.constrain(Aktivitas.class);
            qry.descend("idaktivitas").constrain(del);
            ObjectSet hasil = qry.execute();
            //int i = 0;
```

```
            while (hasil.hasNext()) {
                Aktivitas data = (Aktivitas) hasil.next();
                A.setText(data.getTekssumber());
                B.setText(data.getTeksterjemahan());
                C.setText(data.getIdbahasasumber());
                D.setText(data.getIdbahasaterjemahan());
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }
}
```

(3) Class SkemaCRUD

```
package ODBMSAccess;

import DomainModel.Aktivitas;
import DomainModel.SkemaProyek;
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import com.db4o.query.Query;
import java.util.Date;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class SkemaCRUD {

    public void insert(String idaktivitas, String idtask, Date
tglskemaproyek) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            SkemaProyek SCH = new SkemaProyek();
            SCH.setIdaktivitas(idaktivitas);
            SCH.setIdtask(idtask);
            SCH.setTglskemaproyek(tglskemaproyek);
            db.store(SCH);
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,  "error  =  "  +
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public String update(String idtask) {
        String idaktivitas = null;
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
```

```
        try {
            Query qry = db.query();
            qry.constrain(SkemaProyek.class);
            qry.descend("idtask").constrain(idtask);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                SkemaProyek SCH = (SkemaProyek) hasil.next();
                idaktivitas = SCH.getIdaktivitas();
            }
            //return idtask;
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
            return idaktivitas;
        }
    }

    public void delete(String idtask) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(SkemaProyek.class);
            qry.descend("idtask").constrain(idtask);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                db.delete(hasil.next());
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil(JTable TempTable, JScrollPane jscrollPane,
Date Tanggal) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(SkemaProyek.class);
            qry.descend("tglskemaproyek").constrain(Tanggal);
            qry.descend("idtask").orderAscending();
            ObjectSet hasil = qry.execute();
            //-------------------------

            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][4],
                    new String[]{
```

```
                        "id_Task",                 "id_aktivitas",
"id_bahasa_sumber", "id_bahasa_terjemahan"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false, false, false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                SkemaProyek data = (SkemaProyek) hasil.next();
                TempTable.setValueAt(data.getIdtask(), i, 0);
                Query qry1 = db.query();
                qry1.constrain(Aktivitas.class);

qry1.descend("idaktivitas").constrain(data.getIdaktivitas());
                ObjectSet hasil1 = qry1.execute();
                while (hasil1.hasNext()) {
                    Aktivitas data1 = (Aktivitas) hasil1.next();
                    TempTable.setValueAt(data1.getIdaktivitas(),
i, 1);

TempTable.setValueAt(data1.getIdbahasasumber(), i, 2);

TempTable.setValueAt(data1.getIdbahasaterjemahan(), i, 3);
                }
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }
}
```

(4) Class ListSkemaCRUD

```
package ODBMSAccess;

import DomainModel.Aktivitas;
import DomainModel.ListSkemaProyek;
import DomainModel.SkemaProyek;
import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
import com.db4o.query.Query;
import java.util.Date;
import javax.swing.JOptionPane;
```

```
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class ListSkemaCRUD {
    public void insert(String namaSkema, Date tanggalskema) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            ListSkemaProyek LSP = new ListSkemaProyek();
            LSP.setNamaskemaproyek(namaSkema);
            LSP.setTglskemaProyek(tanggalskema);
            db.store(LSP);
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public boolean validasi(String namaSkema) {
        boolean tidakada = true;
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(ListSkemaProyek.class);
            qry.descend("namaskemaproyek").constrain(namaSkema);
            ObjectSet hasil = qry.execute();
            if (!hasil.isEmpty()) {
                tidakada = false;
            }
            db.commit();
            return tidakada;
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
            return tidakada;
        }
    }

    public void update(String namaSkema, Date tanggalskema) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(ListSkemaProyek.class);
            qry.descend("tglskemaProyek").constrain(tanggalskema);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                //db.delete(hasil.next());
                ListSkemaProyek    ATV    =    (ListSkemaProyek)
hasil.next();
                ATV.setNamaskemaproyek(namaSkema);
```

```
                db.store(ATV);
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void delete(String namaSkema) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(ListSkemaProyek.class);
            qry.descend("namaskemaproyek").constrain(namaSkema);
            ObjectSet hasil = qry.execute();
            while (hasil.hasNext()) {
                db.delete(hasil.next());
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,  "Error  :  "  +
e.getMessage());
        } finally {
            db.close();
        }
    }

    public void delete2(Date tglskema) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry1 = db.query();
            qry1.constrain(SkemaProyek.class);
            qry1.descend("tglskemaproyek").constrain(tglskema);
            ObjectSet hasil1 = qry1.execute();
            while (hasil1.hasNext()) {
                SkemaProyek SP = (SkemaProyek) hasil1.next();
                Query qry2 = db.query();
                qry2.constrain(Aktivitas.class);

qry2.descend("idaktivitas").constrain(SP.getIdaktivitas());
                ObjectSet hasil2 = qry2.execute();
                while (hasil2.hasNext()) {
                    db.delete(hasil2.next());
                }
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,  "Error  :  "  +
e.getMessage());
        } finally {
```

```
            db.close();
        }
    }

    public void delete3(Date tglskema) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry1 = db.query();
            qry1.constrain(SkemaProyek.class);
            qry1.descend("tglskemaproyek").constrain(tglskema);
            ObjectSet hasil1 = qry1.execute();
            while (hasil1.hasNext()) {
                db.delete(hasil1.next());
            }
            db.commit();
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,  "Error  :  "  +
e.getMessage());
        } finally {
            db.close();
        }
    }

    public void tampil(JTable TempTable, JScrollPane jscrollPane)
{
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(ListSkemaProyek.class);
            qry.descend("tglskemaProyek").orderAscending();
            ObjectSet hasil = qry.execute();
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][2],
                    new String[]{
                        "Tanggal_Skema_Proyek",
"Nama_Skema_Proyek"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                ListSkemaProyek    data    =    (ListSkemaProyek)
hasil.next();
                TempTable.setValueAt(data.getTglskemaProyek(),  i,
0);
```

```
                TempTable.setValueAt(data.getNamaskemaproyek(), i,
1);
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }

    public void tampil2(JTable TempTable, JScrollPane jscrollPane,
String Data) {
        ObjectContainer db = Db4o.openFile("DataStore.ODB");
        try {
            Query qry = db.query();
            qry.constrain(ListSkemaProyek.class);
            qry.descend("namaskemaproyek").constrain(Data).not();
            qry.descend("tglskemaProyek").orderAscending();
            ObjectSet hasil = qry.execute();
            TempTable.setModel(new
javax.swing.table.DefaultTableModel(
                    new Object[hasil.size()][2],
                    new String[]{
                        "Tanggal_Skema_Proyek",
"Nama_Skema_Proyek"
                    }) {

                boolean[] canEdit = new boolean[]{
                    false, false
                };

                @Override
                public  boolean  isCellEditable(int  rowIndex,  int
columnIndex) {
                    return canEdit[columnIndex];
                }
            });
            jscrollPane.setViewportView(TempTable);
            for (int i = 0; i < hasil.size(); i++) {
                ListSkemaProyek    data    =    (ListSkemaProyek)
hasil.next();
                TempTable.setValueAt(data.getTglskemaProyek(),  i,
0);
                TempTable.setValueAt(data.getNamaskemaproyek(), i,
1);
            }
        } catch (Exception e) {
            db.rollback();
            JOptionPane.showMessageDialog(null,
e.getLocalizedMessage());
        } finally {
            db.close();
        }
    }}
```