INTEGRASI *MULTI DATABASE* MENGGUNAKAN TEKNOLOGI *WEB SERVICE*

DATABASE MULTI INTEGRATION BY USING WEB SERVICE TECHNOLOGY

SITTI AISA

P2700211457

PROGRAM STUDI S2 TEKNIK ELEKTRO

UNIVERSITAS HASANUDDIN

MAKASSAR

2013

INTEGRASI *MULTI DATABASE* MENGGUNAKAN TEKNOLOGI *WEB SERVICE*

Tesis

Sebagai salah satu syarat untuk mencapai Gelar Magister

Program Studi

Teknik Elektro

Disusun dan Diajukan oleh

SITTI AISA

P2700211457

Kepada

PROGRAM STUDI S2 TEKNIK ELEKTRO

UNIVERSITAS HASANUDDIN

MAKASSAR

2013

TESIS

INTEGRASI MULTI DATABASE MENGGUNAKAN TEKNOLOGI WEB SERVICE

Disusun dan di ajukan oleh

SITTI AISA

Nomor Pokok P2700211457

Telah dipertahankan di depan Panitia Ujian Tesis

Pada Tanggal 18 Juli 2013

Dan dinyatakan telah memenuhi syarat

Menyetujui

Komisi Penasehat,

Amil Ahmad Ilham,ST.,M.IT.,Ph.D	**Dr. Eng. Muh. Niswar, ST.,M.IT**
Ketua	Anggota
Ketua Program Studi Teknik Elektro	Direktur Program Pascasarjana Universitas Hasanuddin,
Prof. Dr. Ir. H. Salama Manjang, MT	**Prof. Dr. Ir. Mursalim**

PERNYATAAN KEASLIAN TESIS

Yang betanda tangan di bawah ini :

Nama : Sitti Aisa

NIM : P2700211457

Program Studi : Teknik Elektro

Konsentrasi : Teknik Informatika

Menyatakan dengan sebenarnya bahwa tesis yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil-alihan tulisan atau pemikiran orang lain. Adapun kutipan atau rujukan sebagai sumber informasi yang saya gunakan dari penulis lain, telah saya sebutkan namanya pada daftar pustaka tesis ini.

Apabila dikemudian hari ada terbukti bahwa tesis ini adalah hasil karya orang lain maka saya bersedia menerima sanksi apapun sesuai peraturan yang berlaku.

Makassar, September 2013

Penulis

(Sitti Aisa)

KATA PENGANTAR

Tiada kata yang pantas diucapkan kecuali puji dan syukur ke hadirat Tuhan Yang Maha Esa karena atas limpahan rahmat dan hidayah-Nya sehingga penyusunan tesis ini dapat terselesaikan dengan baik. Tesis yang berjudul "*Integrasi Multi Database Menggunakan Teknologi Web Service* " ini merupakan hasil penelitian yang ditulis dalam rangka memenuhi syarat untuk melangsungkan seminar hasil penelitian pada program Pascasarjana Program Studi Teknik Elektro Fakultas Teknik Universitas Hasanuddin Makassar, penulis sangat sadar bahwa apa yang dalam proses penulisan dan penyusunan tesis ini, penulis telah mendapatkan bimbingan dan bantuan serta dukungan dari berbagai pihak, dan dengan segala kerandahan hati penulis ucapkan banyak terima kasih kepada yang terhormat :

1. Kedua orang tua yang tercinta beserta saudara-saudaraku yang telah memberikan dorongan semangat dan doa restunya.
2. Amil Ahmad Ilham,ST.,MIT.,Ph.D, selaku Ketua Penasehat dan Muh. Nizwar, ST.,MIT.,Ph.D selaku Anggota Penasehat. Ditengah aktivitas yang padat, beliau berkenan membimbing, mengarahkan penulis dengan kesabaran dan ketelatenan.
3. Prof. Dr. Ir. H.Salama Manjang, MT selaku Ketua Program Studi Pascasarjana Teknik Elektro Universitas Hasanuddin Makassar, beserta seluruh Asisten dan para dosen yang telah mengajar

penulis dan seluruh karyawan yang telah memberikan pelayanan demi kelancaran seluruh proses studi yang penulis tempuh.

4. Para penguji yang berkenan hadir dan telah banyak memberi masukan dan arahan-arahan.
5. Teman-teman mahasiswa pascasarjana Teknik Informatika Universitas Hasanuddin Angkatan 2011 yang telah banyak membantu penulis dalam menyelesaikan hasil penelitian ini.

Serta kepada seluruh pihak yang tidak mungkin penulis sebutkan satu persatu dalam tulisan ini, semoga amal baiknya dibalas oleh Allah SWT, dengan cara dimudahkan segala urusannya dan dilimpahkan rezkinya. Amin.

Makassar, September 2013

Penulis

ABSTRACT

Sitti Aisa. *Multi Using Database Integration Technology Web Service* (supervised by **Amil Ahmad Ilham** and **Muh. Nizwar**)

This study aimed to implement a web service technology that can present the data in the parent table (master) of academic information systems engineering graduate program faculty (Siaka), to be used by other information systems (including information systems lecturer credit score assessment, information system automation correspondence engineering graduate program faculty, and online systems development accreditation forms), so the consistency of master data (master) can be clarified as well as prove that the implementation of web service technology is multi-platform (not limited to the programming language).

Service using data integration methods that exist in the parent table (master) which is in Siaka database, which integrates three (3) existing information systems in the scope of the engineering faculty, in order to facilitate the admin of each information system to retrieve data that has been in upgrade the database web service Siaka from Siaka.

With this allows the existing data in the database can be integrated with Siaka 3 existing information systems so as to maintain the consistency of data between information systems that have been integrated.

Keywords: integration, database, web service.

ABSTRAK

Sitti Aisa. Integrasi Multi Database Menggunakan Teknologi *Web Service* (dibimbing oleh **Amil Ahmad Ilham** dan **Muh. Nizwar**)

Penelitian ini bertujuan untuk mengimplementasikan teknologi *web service* yang dapat menyajikan data pada tabel induk (*master*) dari sistem informasi akademik program pascasarjana fakultas teknik (SIAKA), agar dapat digunakan oleh sistem informasi yang lain (diantaranya sistem informasi penilaian angka kredit dosen, sistem informasi otomatisasi persuratan program pasca sarjana fakultas teknik, dan sistem *online* penyusunan borang akreditasi), sehingga konsistensi data induk (*master*) dapat diperjelas serta membuktikan bahwa implementasi teknologi *web service* bersifat *multi platform* (tidak terbatas pada bahasa pemrograman)

Layanan menggunakan metode *REST* dengan integrasi data-data yang ada pada tabel induk (*master*) yang ada pada database SIAKA, dimana untuk mengintegrasikan 3 (tiga) sistem informasi yang ada pada lingkup fakultas teknik, agar memudahkan admin dari masing-masing sistem informasi mengambil data yang telah di *upgrade* pada *web service* siaka dari database SIAKA. Dalam *web service* siaka telah terbentuk file xml dan file json dari masing-masing tabel induk.

Dengan ini menggunakan metode ini memungkinkan data yang ada pada database SIAKA bisa di integrasikan dengan 3 (tiga) sistem informasi yang ada sehingga dapat menjaga konsistensi data antara sistem informasi yang telah di integrasikan. Sehingga admin dari ketiga sistem informasi dapat dengan mudah meng-upgrade data dari web service siaka tanpa perlu menginput ulang data yang sama dari database SIAKA.

Kata Kunci : integrasi, *database, web service*.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Pemanfaatan teknologi informasi di segala bidang telah membawa pengaruh yang signifikan dalam pelaksanaan tugas tidak terkecuali pada perguruan tinggi. Pemanfaatan teknologi informasi di perguruan tinggi antara lain adalah sebagai alat bantu untuk proses administrasi dan pendidikan.

Penyediaan sistem informasi sebagai pendukung proses administrasi pendidikan pada sebuah perguruan tinggi merupakan suatu kepentingan yang mutlak pada era saat ini. Pembuatan sistem informasi pada sebuah perguruan tinggi biasanya dilakukan secara bertahap dan di kembangkan secara terpisah, sehingga beberapa sub sistem tidak terintegrasi. Keadaan tersebut menimbulkan kesulitan pada saat digunakan oleh sistem. Sebagai contoh Fakultas Teknik Unhas saat ini sedang dikembangkan beberapa sistem informasi yaitu:

1. Sistem Informasi Akademik Program Pascasarjana Fakultas Teknik. Sistem ini menyediakan layanan jadwal kuliah, nilai mahasiswa, data dosen, data mahasiswa, data mata kuliah dan sebagainya.
2. Sistem Informasi Penilaian Angka Kredit Dosen. Sistem ini menyediakan layanan bagi dosen yang akan mengurus kenaikan pangkat akademik. Adapun berkas yang diperlukan adalah

pengajaran, penelitian dan pengabdian pada masyarakat.

3. Sistem Otomatisasi Persuratan Program Pascasarjana Fakultas Teknik. Sistem ini menyediakan layanan bagi mahasiswa untuk pengurusan SK pembimbing, pengurusan SK Penguji, pengurusan Surat untuk Seminar proposal, pengurusan Surat untuk Seminar hasil dan pengurusan Surat untuk sidang.
4. Sistem *On-line* Penyusunan Borang Akreditasi. Sistem ini menyediakan layanan untuk penyiapan borang akreditasi seperti data dosen, data kurikulum dan sebagainya.

Masing-masing sistem informasi tersebut dibuat secara terpisah sehingga beberapa data induk yang sama tidak bisa digunakan secara bersama-sama karena struktur database yang berbeda-beda.

Pada Sistem Informasi Akademik Program Pascasarjana Fakultas Teknik terdapat tabel dosen (tbl_dosen) terdiri dari 28 *field*, sedangkan Sistem Otomatisasi Persuratan Program Pascasarjana Fakultas Teknik terdapat Tabel dosen (tbl_dosen) terdiri dari 10 *field*, struktrunya *field* dari kedua tabel tersebut dapat dilihat pada tabel 1.1 dibawah ini :

Tabel 1.1 : struktur *field* pada tabel dosen

Nama *Field* Tabel dosen (SIAKA)	*Type* dan *size field*	Nama *Field* Tabel dosen (persuratan)	Type dan size field
DosenID	Varchar(8)	Nama_dosen	Text
Namadosen	Varchar(30)	Nip	Varchar(30)
Noktp	Varchar(26)	Alamat	Varchar(50)
Tplhr (tempat_lahir)	Varchar(20)	Tempat_lhr	Varchar(30)
Tglhr (tanggal_lahir)	Timestamp	Tgl_lahir	Date

Nama *Field* Tabel dosen (SIAKA)	*Type* dan *size field*	Nama *Field* Tabel dosen (persuratan)	Type dan size field
Kdjek (jenis_kelamin)	Varchar(3)	No_telp_dosen	Varchar(20)
No_telp	Varchar(50)	Pstudi_id	Varchar(30)
Nipns	Varchar(15)	Max	Int (100)

Pada tabel 1.1 telah di jelaskan perbedaan struktur field pada tabel dosen yang ada pada kedua database, dimana dapat terlihat perbedaan nama field dan lebar field yang sudah di deklarasikan. Dimana, keempat sistem yang telah di sebutkan diatas menggunakan data dosen yang sama tetapi karena *item field* dari sistem tersebut berbeda, sehingga pada proses penginputan recordnya harus di lakukan pada setiap sistem informasi.

Oleh karena itu, perlu adanya suatu teknologi yang dapat mengintegrasikan semua sistem informasi yang sudah ada, dengan menggunakan teknologi *web service*, yaitu dengan cara menyediakan layanan yang akan diakses oleh sistem informasi lain yang membutuhkan.

Berdasarkan latar belakang yang telah di jelaskan diatas maka penulis melakukan penelitian dengan judul **"Integrasi Multi *Database* menggunakan Teknologi *Web Service*".**

B. Rumusan Permasalahan

Adapun rumusan permasalahan dalam penelitian ini :

1. Dengan struktur *database* yang berbeda-beda seperti panjang *field* dan penamaan *field* yang berbeda, menimbulkan masalah bagi operator yaitu data yang sama harus dimasukkan secara terpisah pada setiap sistem informasi tersebut. Hal ini tidak efisien karena menambah beban kerja operator sehingga diperlukan suatu sistem yang dapat mengintegrasikan sistem-sistem tersebut dan menjaga konsistensi data yang sama di masing-masing sistem informasi.
2. Bagaimana mengintegrasikan sistem informasi yang sudah ada dalam lingkup Fakultas Teknik Universitas Hasanuddin menggunakan teknologi *Web service.*

C. Tujuan Penelitian

Tujuan penelitian ini adalah untuk mengimplementasikan teknologi *web service* yang dapat menyajikan data induk agar dapat digunakan oleh sistem informasi yang lain, sehingga konsistensi data induk antar sistem informasi dapat diperjelas serta membuktikan bahwa implementasi teknologi *web service* bersifat *multi platform* (tidak terbatas pada bahasa pemrograman tertentu).

D. Manfaat Penelitian

Hasil Penelitian ini diharapkan mempunyai manfaat :

1. Memudahkan operator memasukkan record pada data induk sistem informasi yang telah diintegrasikan.
2. Dapat menjaga konsistensi data antara sistem informasi yang satu dengan yang lainnya.

E. Batasan Masalah

Melihat cakupan dari penelitian luas, maka dalam hal ini di batasi pada Sistem informasi yang sudah ada di lingkup Fakultas Teknik Univesitas Hasanuddin sebagaimana yang telah dijelaskan pada latar belakang yang memuat di dalamnya layanan akademik bagi mahasiswa dan dosen.

BAB II

TINJAUAN PUSTAKA

A. Integrasi

Yang dimaksud dengan integrasi data adalah suatu proses menggabungkan menyatukan data yang berasal dari sumber yang berbeda dan mendukung pengguna untuk melihat kesatuan data.[6]

Integrasi data dibutuhkan seiring dengan perkembangan organisasi dan meningkatnya bisnis proses pada institusi tersebut yang membutuhkan data dan informasi dari divisi atau unit-unit yang berada pada organisasi tersebut.

Proses integrasi bisnis proses antar unit inilah yang menjadi titik kritis dalam proses integrasi data, dimana jika tidak terjadi atau tidak tercapainya kesepakatan antar pihak manajemen terhadap integrasi bisnis proses, mustahil proses integrasi data dapat dilakukan. Tahap pengembangan dari integrasi bisnis proses ini adalah proses integrasi data secara sederhana.[7]

B. Multi Database

Suatu sistem basis data terdistribusi adakalanya dibentuk dari beberapa basis data yang berlainan. Sistem seperti ini disebut *multidatabase*, yang pembentukannya dilakukan dengan integrasi basis data. Dalam melakukan integrasi ini, boleh jadi ada ketidakseragaman

antara basis data-basis data yang membentuknya dan mengakibatkan konflik, baik **konflik skema** (akibat ketidakseragaman skema relasi) maupun **konflik data** (akibat ketidakakuratan isi, misalnya presisi, besaran dan satuan, juga data yang tidak tepat atau sudah tidak berlaku [*obsolete*]). Oleh karena itu, perlu diperhatikan metode penyelesaian konflik yang terjadi.

Penyelesaian konflik saat integrasi basis data harus memerlukan analisis yang mendalam akan komponen basis data dan tidak dapat diotomatisasi. Adapun **sebelum** dilakukan **integrasi**, komponen basis data harus dipersiapkan terlebih dahulu, khususnya untuk penanganan konflik.

Penyelesaian konflik pada restrukturisasi (untuk integrasi) basis data dapat dilakukan dengan menggunakan *view*, yaitu relasi/tabel bentukan virtual yang bukan bagian dari skema relasi itu sendiri, tetapi dapat diperlakukan sebagaimana relasi melalui *query* dan *view* lain. *View* hanya perlu disimpan definisinya di kamus data tanpa perlu relasinya.

Berikut ini adalah konflik yang dapat terjadi pada proses integrasi basis data berikut contoh penanganannya:

1. konflik antartabel
    a) antara dua tabel
        1. nama tabel (homonim/sinonim), dapat diselesaikan dengan definisi *view* sendiri

2. struktur tabel, seperti jumlah atribut berbeda di dua tabel yang informasinya sama, dapat diselesaikan dengan membuang atribut yang keberadaannya tidak di semua relasi atau menambahkan atribut yang kurang pada relasi yang kekurangan atribut (bila perlu dengan nilai *default*-nya)
3. *integrity constraint*, misalnya pada dua situs terdapat tabel yang sama tetapi isi atribut *primary key*-nya berbeda, dapat diselesaikan dengan memberikan *primary key* tambahan yang berisi informasi situs relasi itu disimpan

b) antara banyak tabel, misalnya pada dua komponen basis data jumlah relasinya tidak sama tetapi informasinya sama, dapat diselesaikan dengan penggabungan relasi dengan *view.*

2. konflik antaratribut

a) antara dua atribut

1. nama atribut (homonim/sinonim), dapat diselesaikan dengan penggantian nama (*rename*) atribut di *view*
2. *integrity constraint*, misal tipe data, dapat diselesaikan dengan fungsi-fungsi konversi, seperti to_char(int) atau to_int(char), pada *view*

b) antara banyak atribut, misalnya dalam penyimpanan informasi nama orang dalam tabel yang satu digunakan dua atribut (kolom): nama depan dan nama belakang, sementara pada tabel yang lain digunakan

satu atribut nama lengkap; konflik ini dapat diselesaikan dengan penggabungan string atau pemisahan string dengan fungsi substring

3. konflik tabel-atribut, dapat merupakan kombinasi dari permasalahan di atas.[3]

C. Web Service

1. Pengertian *Web Service*

Teknologi *web services* menawarkan kemudahan dalam menjembatani pulau-pulau informasi tanpa mempermasalahkan perbedaan teknologi yang digunakan masing-masing sumber. Misalkan sebuah situs informasi dibangun dengan menggunakan database *Oracle* sedangkan situs lainya menggunakan *Mysql* sedangkan anda sendiri menggunakan perangkat lunak *Open Source* dalam membangun situs *web services* akan mengatasi perbedaan ini.

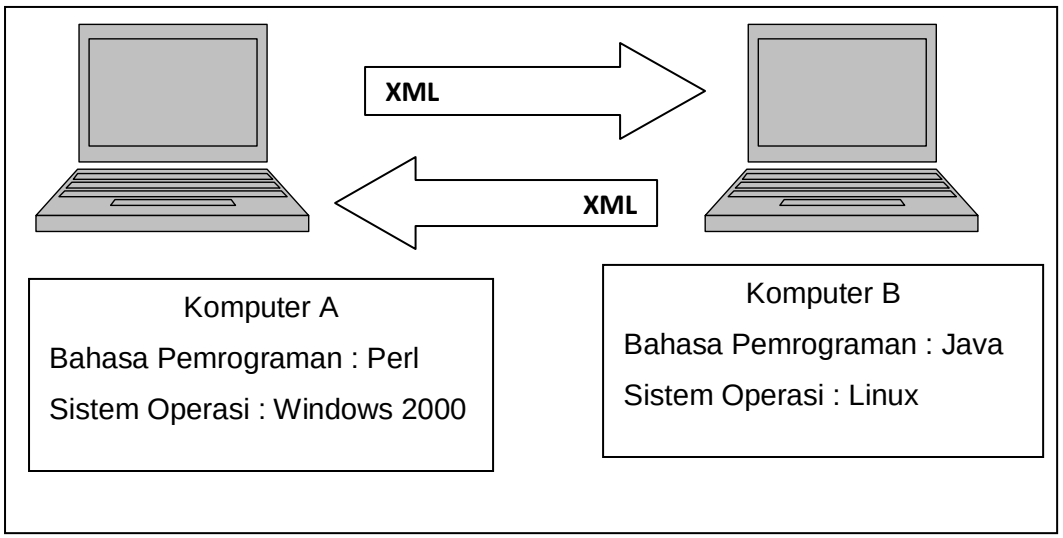
Web-service diartikan sebagai sebuah antar muka (*interface*) yang menggambarkan sekumpulan operasi-operasi yang dapat diakses melalui jaringan, misalnya internet dalam bentuk pesan *XML*.

Web-service diartikan juga sebagai sepotong atau sebagian informasi atau proses yang dapat diakses oleh siapa saja, kapan saja dengan menggunakan piranti apa saja, tidak terikat dengan sistem operasi atau bahasa pemograman yang digunakan.[1]

Web Services sebenarnya adalah kumpulan dari fungsi dan *method* yang terdapat pada sebuah server yang dapat dipanggil oleh klien dari

jarak jauh, kemudian untuk memanggil *method-method* tersebut kita bebasa menggunakan aplikasi yang akan dibuat dengan bahasa pemrograman apa saja yang dijalankan pada platform apa saja.

Web Services diperlukan karena pada masa sekarang ini perangkat keras, sistem operasi, aplikasi hingga bahasa pemrograman semakin beraneka ragam jenisnya. Keadaan tersebut dapat menimbulkan masalah dalam proses pertukaran data antar perangkat yang menggunakan aplikasi dan platform yang berbeda.[9]



Gambar 2.1 : Dasar *Web Service*

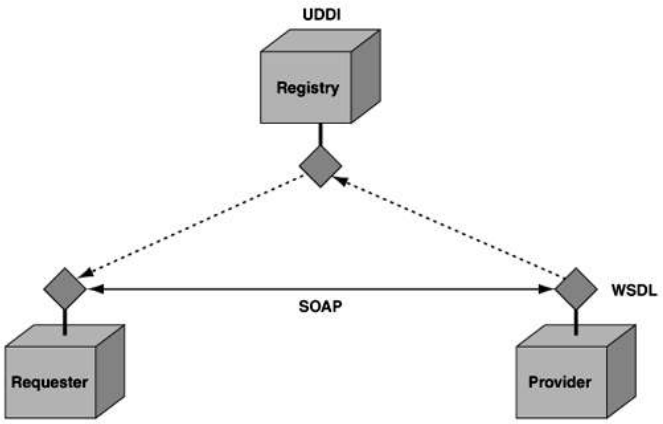
Pada gambar 2.1 dijelaskan tentang dasar web service, dimana Web services adalah layanan yang tersedia di Internet yang menggunakan bahasa standar XML untuk pengiriman pesannya, tidak terikat kepada bahasa pemrograman atau sistem operasi tertentu.

2. Arsitektur Web Service

Konsep arsitektur yang mendasari teknologi *Web service* adalah *Service Oriented Architecure* (SOA). Dalam aritektur ini, suatu aplikasi dimodelkan sebagai komposisi dari sekumpulan *service* yang disediakan oleh suatu komponen.

Lokasi keberadaan komponen tersebut dapat ditemukan oleh client secara dinamis, dalam arti tidak dinyatakan secara statis tetapi menggunakan mekanisme *discovery* untuk mencari keberadaan komponen tersebut. Demikian pula, *client* dapat meminta (*invoke*) *service* tersebut secara dinamis.

W3C mengembangkan draft arsitektur *web service* yang terdiri dari beberapa teknologi yang saling berhubungan, seperti ditunjukkan dalam gambar 2.2. Berdasarkan gambar tersebut, *web service* tersusun dari beberapa komponen yang semuanya berbasis XML (*eXtensible Markup Language*) yaitu *SOAP*, *WSDL* dan *UDDI*.



Gambar 2.2 : Arsitektur *Web Service*

Pada gambar 2.2 diatas dapat di jelaskan bahwa dalam arsitektur web service , SOA mendefinisikan 3 peran berbeda yang menunjukkan peran dari masing-masing komponen dalam sistem, yaitu :

1. *Service provider*, yaitu suatu entitas yang menyediakan *interface* terhadap sistem yang menjalankan suatu sekumpulan tugas tertentu. *Service provider* dapat merepresentasikan statu entitas bisnis ataupun suatu komponen software yang *reusable*.
2. *Service requestor* , yaitu suatu entitas yang meminta/memperoleh (dan menemukan) *software service* dalam rangka meyelesai kan suatu tugas tertentu atau menyediakan solusi bisnis tertentu.
3. *Service registry* , yaitu entitas yang bertindak sebagai penyimpan (*repository*) suatu *software service* yang dipublikasikan oleh *service provider.*

3. Komponen Web Service

Komponen web service yaitu :

a. *Extensible Markup Language* (XML)

XML merupakan dasar yang penting atas terbentuknya *Web Services*. *Web Services* dapat berkomunikasi dengan aplikasi-aplikasi yang memanggilnya dengan menggunakan XML, karena XML berbentuk teks sehingga mudah untuk ditransportasikan menggunakan protokol HTTP.

Selain itu, XML juga bersifat *platform independen* sehingga informasi di dalamnya bisa baca oleh aplikasi apapun pada platform apapun selama aplikasi tersebut menerjemahkan tag-tag *XML*.

Kesimpulannya adalah apabila *Web Services* dan aplikasi dianggap sebagai manusia yang berbeda ras dan bahasa, maka *XML* adalah sebuah bahasa universal yang dapat mempersatukan mereka digunakan untuk saling berkomunikasi dan bertukar informasi.

Secara singkat, berikut ini adalah *feature – feature* yang ditawarkan *XML* :

a. *XML* dapat menyimpan dan mengorganisir semua jenis informasi dalam bentuk yang kita suka (dapat disesuaikan dengan kebutuhan).
b. Sebuah open standard, *XML* tidak terikat dengan perusahaan atau perangkat lunak manapun.
c. Dengan *Unicode* sebagai karakter set standar, XML mendukung berbagai macam sistem penulisan (*scripts*) dan simbol. Dari karakter Skandinavia sampai ideograf bangsa China Han.
d. *XML* menawarkan berbagai cara untuk memeriksa kualitas sebuah dokumen dengan aturan syntax, internal *link checking*, pembandingan dengan modul dokumen, dan *datatyping*.

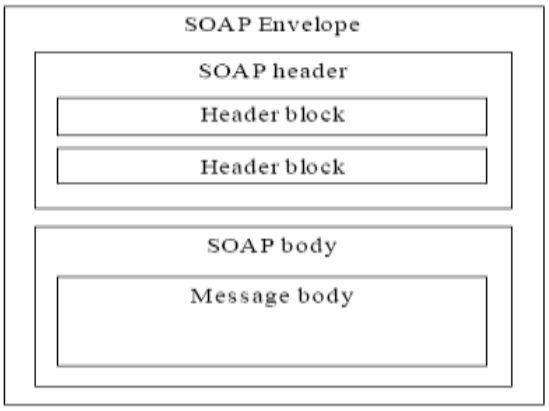
e. Sintaks *XML* sederhana dan tidak mempunyai strukur yang ambigu, sehingga mudah dibaca oleh manusia maupun program.

f. *XML* mudah untuk dikombinasikan dengan *stylesheet* untuk membuat format dokumen sesuai dengan *style* yang kita inginkan.

b. *Simple Object Access Protocol* (SOAP)

XML saja tidak cukup agar *Web Services* dapat berkomunikasi dengan aplikasi yang lainya. *XML* yang digunakan untuk saling bertukar informasi antara *web services* dengan aplikasi yang lainya harus menggunakan sebuah format standar yang dapat dimengerti oleh keduaya. Format itulah yang dikenal dengan nama *SOAP*.

SOAP (*Simple Object Access Protocol)* merupakan suatu format standard dokumen berbentuk XML yang digunakan untuk melakukan proses *request* dan *responses* antara *web services* dengan aplikasi yang memanggilnya. Dokumen SOAP digunakan untuk melakukan *request* disebut dengan SOAP *request* sedangkan dokumen SOAP yang diperoleh dari Web Services disebut dengan SOAP *responses*.



Gambar 2.3 : Struktrur pesan *Soap*

Pada gambar 2.3 diatas ditunjukkan struktur pesan *soap* dimana, pesan *soap* berbentuk seperti sebuah *envelope* yang berisi *header* (optional) dan *body* (*required*). *Header* berisi blok informasi yang berhubungan dengan bagaimana pesan tersebut diproses. Hal ini meliputi pe-routingan dan *delivery setting*, *authentication* atau *authorization assertions*, and *transaction contexts*. *Body* berisi pesan sebenarnya yang dikirim dan diproses. Semua yang dapat ditampilkan dengan sintaks *XML* dapat dimasukkan dalam pesan body.

c. *Web Service Definition Language* (WSDL)

Sebelum mengakses sebuah *Web Services* pastinya perlu mengetahui *method-method* apa saja yang disediakan oleh *Web Services* tersebut, untuk mengetahuinya memerlukan *Description Language*) adalah sebuah dokumen dalam format XML yang isinya menjelaskan informasi detail sebuah *Web Services*. Di dalam WSDL dijelaskan method-method apa saja yang tersedia dalam

Web Services, parameter apa saja yang diperlukan untuk memanggil sebuah *method*, dan apa hasil atau tipe data yang dikembalikan oleh method yang dipanggil tersebut.

WSDL mendeskripsikan service dengan menggunakan elemen sebagai berikut :

- ✓ *Type* : tipe data yang digunakan sebagai argumen dan return type
- ✓ *Message* : merepresentasikan definisi data yang ditransmisikan
- ✓ *Port type* : sekumpulan operasi yang didukung oleh satu atu lebih *endpoint*
- ✓ *Binding* : mendefinisikan protokol dan format pertakaran data untuk operasi yang didefinisikan oleh Port type
- ✓ *Port* : menspesifikasikan end-point yang digunakan untuk binding
- ✓ *Service* : koleksi endpoint yang berkaitan yang disediakan oleh Web service
- ✓ *Operation* : mendefinisikan kemampuan yang didukung oleh servis tertentu.

d. *Universal Description, Discovery & Integration* (UDDI)

UDDI merupakan sekumpulan spesifikasi yang menunjukkan registry informasi mengenai *web service*. UDDI menyediakan mekanisme untuk mempublikasikan informasi mengenai bisnis dan *service* pada satu lokasi (*repository*) yang dikelola secara terpusat dan melakukan *query* mengenai informasi tersebut secara dinamis dan programatis.

Direktory pada UDDI bertindak seperti '*Yellow Pages*' dimana *service* dikategorikan sesuai tujuan utamanya. *Direktory* UDDI terdiri dari 3 bagian, yaitu :

- *White pages* : menyediakan informasi rinci mengenai organisasi yang menawarkan *service*.
- *Yellow pages* : mencakup pengakatagorian jenis industri berdasarkan standar taxonomi industri.
- *Green pages* : mendeskripsikan *interface* dan kebutuhan untuk memperoleh service , seperti *return type.*

UDDI merupakan file *XML Schema* yang mendefinisikan struktur data mengenai karakteristik bisnis dan *service*. Deskrisi *service* didefinisikan menggunakan dokumen *Type* Model (tModel). Secara umum UDDI berisi informasi mengenai siapa yang menyediakan *service* (*businessEntity*), *Service* apa yang disediakan (*businessService*), dimana lokasi *service* tersedia (*bindingTemplate*), referensi mengenai informasi bagaimana *servic*e tersebut diperoleh (tModel).[9]

4. *Restful Web Service*

REST adalah sebuah metode dalam menyampaikan *resource* melalui media *web*. Sedangkan *resource* sendiri didefinisikan sebagai segala sesuatu yang dapat disimpan di dalam sebuah komputer dan ditampilkan sebagai urutan bit, misalnya sebuah dokumen, tabel dalam sistem basis data, atau hasil dari sebuah perhitungan.

Didalam *RESTful Web service*, sesuatu dapat dikatakan sebagai "*Resource*" jika mempunyai minimal satu buah URI (*Universal Resource Identifier*). URI akan menjadi identitas yang akan menunjukkan nama dan alamat dari *resource* di *Web server* tidak dapat mengirimkan *resource* secara langsung, melainkan melalui kumpulan bit dalam format tertentu, dan dalam bahasa tertentu. Inilah yang dinamakan representasi dari *resource*.

Mengingat bahwa sebuah *resource* bersifat dinamis dan bisa berubah kapan saja, maka representasi yang dikirimkan/ditransfer adalah kondisi (*state*) *resource* pada saat diakses oleh *client*.

Perbedaan mendasar layanan *Big Web service* dan layanan REST adalah bahwa *Big Web Service* bersifat *activity-oriented* yaitu berdasarkan pada "apa yang bisa dilakukan", sedangkan *REST* lebih cenderung kepada "apa yang bisa digunakan".[2]

a. Komponen *Rest*

Kombinasi dari *noun, verb* dan tipe konten ini sering disebut sebagai segitiga *REST*. Ketiganya, membentuk tiga sudut dari segitiga yang mendefinisikan arsitektur. Sebuah desain yang berorientasi *REST* sering bisa didekomposisikan dengan menentukan *noun* (pengidentifikasian dan penamaan sesuatu), memilih seperangkat *verb* yang seragam (ini mudah dilakukan jika menggunakan HTTP), dan memilih tipe konten.

1. *Verb*

Verb berhubungan dengan aksi terhadap *resource*. Sebuah *verb* akan mengirimkan suatu representasi dari sebuah *resource* dari server ke klien atau memperbaharui *resource* di server dengan informasi dari klien. Di dalam REST, *verb* merupakan daerah yang penuh dengan konstrain. Sementara seperangkat tipe konten terbuka untuk revisi dan ekspansi, dan nama-nama *resource* dapat diekspansi sampai tak berhingga, sedangkan seperangkat *verb* sudah *fix* (tetap). Namun, setiap konstrain yang diletakkan pada ruang lingkup *verb* mengijinkannya untuk bersifat universal, *verb* apapun dapat diterapkan kepada setiap *noun*.

HTTP mendefinisikan serangkaian metoda yang bisa diperluas oleh protokol lain seperti *WebDAV*, tetapi seperangkat dasar sudah cukup untuk *REST*. Empat metoda yang paling umum adalah *GET, PUT, DELETE dan POST*.

Untuk menjelaskannya dapat dibuat beberapa analogi linguistik sebagai simplifikasi dari empat metode umum. “Ini” akan menunjuk pada *requestbody*, dan “di sana” menunjuk kepada *URI* dimana ia beraksi.

a. *GET*: “Berikan aku yang ada di sana”
b. *PUT*: “Simpan ini di sana”
c. *DELETE*: “ Hapus yang ada di sana”
d. *POST*: “Hey kamu yang ada di sana, proses ini”

a) *GET*.

Metoda *GET* mengirim representasi *resource* dari *server* ke klien. Ini digunakan hanya untuk mengakses *resource* secara *read-only*. Sejauh ini, *GET* adalah *verb* paling umum di Web dimana website statik sering hanya mempergunakan metode ini. Kesalahan yang umum dilakukan adalah mempergunakan *GET* untuk aksi yang memperbaharui *resource* (*update*). *GET* didefinisikan sebagai metode *safe* yang seharusnya digunakan untuk pengambilan (*fetching*), bukan *update*. Pengunaan GET untuk *update* dapat menyebabkan banyak problem karena ia telah merusak asumsi klien dan proxy-proxy yang dilewati terhadap sifat asli *request GET*.

b) PUT

Metoda *PUT* meng-*update resource* dengan representasi yang inyatakan di dalam *body request PUT*. Jika *resource* tidak ditemukan, *request* akan membuat *resource* yang baru dengan representasi yang diberikan. Sebuah hal umum yang membingungkan adalah bagaimana menentukan nama-nama *resource* (URI) apakah diterapkan pada *request PUT* atau *POST*. *Request* PUT digunakan jika klien mengetahui URI dari *resource*, yang apabila tidak ada maka

akan dibuat r*esource* yang baru sebagai mana yang sudah ditetapkan pada URI. Jika klien tidak mengetahui URI dari *resource* (contohnya, jika ia diambil dari ID yang dibangkitkan secara otomatis oleh *server*), maka *request POST* yang seharusnya digunakan.

c) DELETE

Sebagaimana diimplikasikan dari namanya, metoda *DELETE* menghapus *resource* yang diidentifikasikan oleh url-nya. Jika penghapusan ditolak oleh *server* yang tidak mengijinkan penghapusan, *subsequent query* GET ke URI yang sama harus mengembalikan kode status 410 (*Gone*) atau 404 (*Not Found*).

d) POST

POST disebutkan belakangan karena merupakan perhentian terakhir. Metoda ini tidak *safe*, tidak juga *idempotent*, sehingga ada sedikit pembatasan teknis pada kekuatannya. Sehingga, sebaiknya tidak menggunakannya untuk operasi-operasi yang dapat lebih baik direpresentasikan dengan *verb* lainnya. Secara teoretis, *POST* bisa digunakan untuk setiap aksi terhadap Web tanpa merusak RPC. Walaupun *POST powerful*, itu tidak seharusnya digunakan dimana *GET, PUT*, atau *DELETE* sudah mencukupi. Semantik dari tiga metoda tersebut lebih sederhana, dan konstrain-

konstrain yang diletakkan padanya mengijinkan *caching* yang memudahkan dan skalabilitas. *POST* bisa, secara teori, di-*cache* menggunakan *header Cache-Control* dan *Expires*, tetapi pada praktiknya ini jarang diimplementasikan. *POST* utamanya digunakan dalam salah satu dari dua cara berikut, penciptaan objek baru dan pemberian anotasi objek yang sudah ada. Pada kedua kasus tersebut, URI dari *POST* adalah kontainer objek tersebut atau *parent*-nya. *RFC* menggambarkannya dengan sebuah analogi dari struktur direktori dimana untuk membuat atau meng- *update* sebuah objek, harus mem-*POST* pada direktori penampungnya.

2. *Resource*

Konsep paling mendasar dari REST adalah *resource*. Definisi paling umum dari *resource* adalah sesuatu dengan identitas. Dalam pemakaian yang populer digunakan, istilah "*resource*" biasanya berarti sesuatu yang dapat dialamati jaringan pada internet. Tetapi sebuah *resource* dapat berupa apa saja, baik itu nyata ataupun yang tidak dapat diraba asalkan dapat dinamai.

Sebuah *resource* dapat berarti apa saja yang memiliki identitas. Contoh yang familiar termasuk sebuah dokumen elektronik, sebuah gambar, *service* (contohnya, "laporan cuaca hari ini untuk Indonesia), dan koleksi dari *resource* lain. Tidak semua *resource*

bisa diambil via jaringan; contohnya, manusia, perusahaan, dan setumpuk buku di dalam perpustakaan dapat juga dikatakan *resource*. Tersembunyi di dalam definisi *resource* ini adalah bahwa *resource* mempunyai *state* (*resource* bisa saja mempunyai *state* kosong pada kasus degenerasi, tetapi ini jarang dijumpai).

Salah satu dari konstrain yang menempatkan *REST* pada interaksi dengan *resource* adalah bahwa setiap *RESTful resource* memiliki antarmuka yang seragam. Tidak ada klien yang memiliki akses *ad-hoc* (baca atau tulis) pada sebuah *resource* menyangkut *state resource*, hal tersebut hanya diketahui oleh internal *resource*. Setiap akses didapat dengan melalui transfer representasi dari *state resource* via seperangkat metoda yang seragam (dalam hal ini, HTTP).[9]

5. JSON (*JavaScript Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. *JSON* merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, *Java, JavaScript, Perl, Python* dll. Oleh

karena sifat-sifat tersebut, menjadikan *JSON* ideal sebagai bahasa pertukaran-data.

Beberapa orang lebih suka JSON, karena paling mudah untuk mem-parse-nya, hanya menempatkan sebuah eval dan selesai sudah. JSON terbuat dari dua struktur :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (struct), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan.

Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut :

1. Object adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Array adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).
3. Nilai (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.
4. String adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.
5. Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.

Sekarang mari kita lihat contoh-contoh bentuk JSON yang saya ambil dari json.org. Berikut bentuk XML, yang telah familiar dengan kita.[8]

```
<menu id="file" value="File">
<popup>
      <menuitem value="New" onclick="CreateNewDoc()" />
      <menuitem value="Open" onclick="OpenDoc()" />
      <menuitem value="Close" onclick="CloseDoc()" />
</popup>
```

Jika kita ubah ke bentuk JSON adalah :

```
{"menu": {
      "id": "file",
      "value": "File",
      "popup": {
      "menuitem": [
            {"value": "New", "onclick": "CreateNewDoc()"},
            {"value": "Open", "onclick": "OpenDoc()"},
```

D. Roadmap Penelitian

Adapun penelitian yang telah ada sebelumnya adalah :

1. “Penyajian Data Induk Mahasiswa Menggunakan Teknologi *Web Service”* yang ditulis oleh Aris Wibowo, Maman Somantri, R. Rizal Isnanto.[2]

   menjelaskan tentang implementasi teknologi web service untuk menyajikan data induk mahasiswa, dimana ada 3 (tiga) Sistem yang menggunakannya yaitu Situs web Informasi Akademik, Situs web Informasi Beasiswa dan Pendaftaran Beasiswa serta Situs web Sistem Informasi Pendaftaran Mahasiswa Baru, yang digunakan sebagai sumber data pada waktu pembuatan data induk mahasiswa. Dimana dalam pembuatan dan pengaksesan menggunakan bahasa pemrograman yang berbeda yaitu php dan java.

   Dalam penelitin ini, belum memperhitungkan keamanan sistem serta *Client* yang mengakses data induk masih menggunakan satu jenis *account*, sehingga mempunyai hak akses yang sama. Akan lebih baik jika dapat membedakan hak akses masing-masing *client.*

2. “ Web Service sebagai Solusi Integrasi Data Pada Sistem Informasi Akademik Universitas Bina Darma “ yang di tulis oleh Susan Dian Purnama.[10]

   Menjelaskan tentang perubahan sistem informasi akademik yang masih berbasis web menjadi sistem informasi akademik dengan menggunakan teknologi web service dengan menambahkan kode

XML. Dengan memanfaatkan XML, maka integrasi data dari Universitas Bina Darma yang memiliki basis data berbeda dapat dilakukan. Pendistribusian secara *online* juga tidak membantu pendistribusian data tersebut menjadi mudah karena harus berpindah-pindah dari satu situs ke situs yang lain untuk mengambil data.

Dalam penelitian ini hanya berfokus pada layanan akademik, sehingga belum mengintegrasikan data induk pada sistem informasi yang ada, misalnya untuk sistem informasi perpustakaan, sistem informasi keuangan.

3. " Penggunaan Teknologi *Web Service* pada Sistem Registrasi PPJK disusun oleh Khilmi mubarok.[5]

   Menjelaskan tentang pengembangan dari sistem registrasi PPJK dengan menggunakan teknologi *web service*, memberikan kemudahan dalam hal pengaksesan data, mudah digunakan tetapi kerahasiaanya tetap terjaga.

   Dalam penelitian ini, aplikasi yang dikembangkan oleh DJBC selama ini lebih bersifat *distribute application,* dimana di setiap Kantor Pelayanan DJBC terdapat *server* aplikasi dan *database*. Meskipun untuk beberapa segi aplikasi ini mempunyai keunggulan seperti tidak bergantung pada ketersediaan jaringan intranet (aplikasi yang lebih *independent*), aplikasi yang dikembangkan dengan model seperti ini menyulitkan untuk keperluan *maintenance* (*updating*) dan *collecting data*.

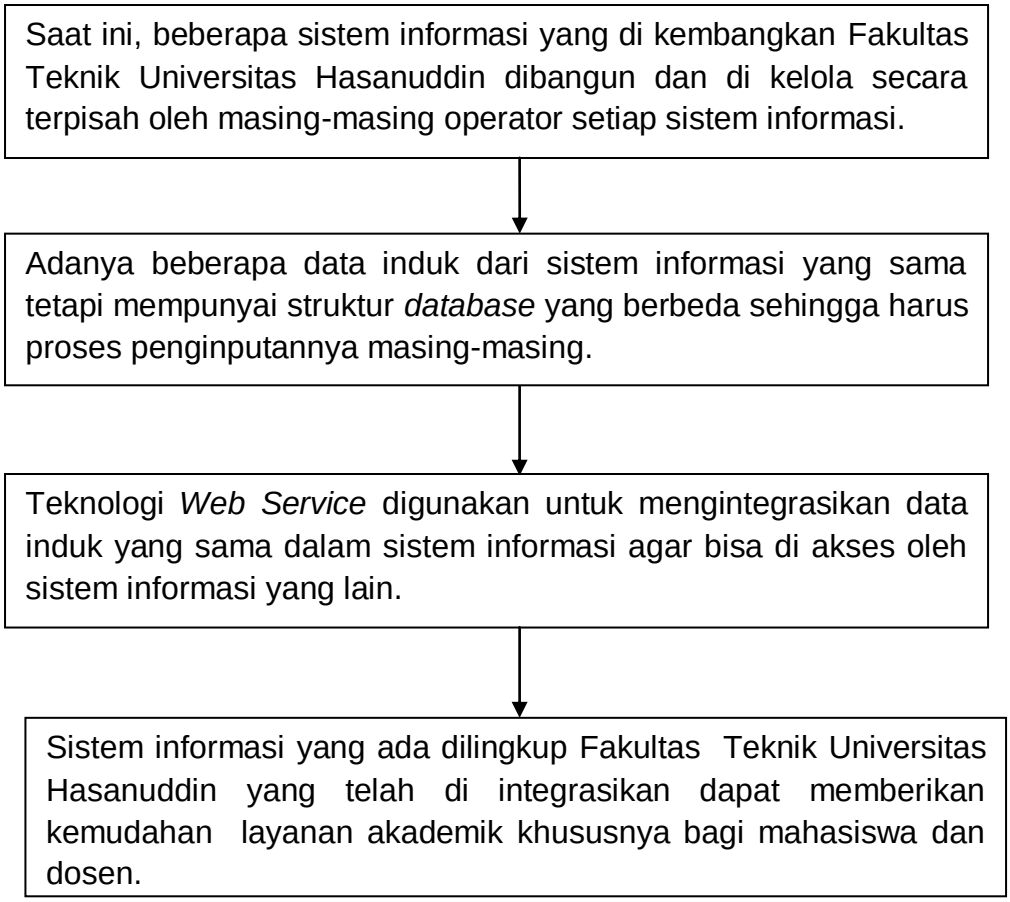
4. Sinkronisasi Database Fakultas Teknik dan Rektorat Universitas Hasanuddin Makassar disusun oleh yuyun.[11]

   Dalam penelitian ini, melakukan sinkronisasi pada tabel dosen dan mahasiswa antara database fakultas teknik dan database rektorat universitas hasanuddin dengan mencari persamaan field antar keduanya.

Dari keempat penelitian yang telah ada, maka untuk penelitian ini merancang sebuah sistem informasi yang di integrasikan dengan teknologi web service yang melibatkan 4 sistem informasi, dimana sistem informasi akademik pasca sarjana fakultas teknik universitas hasanuddin yang akan menginput record untuk masing-masing data induk dan ketiga sistem informasi yang lain dapat meng-upgrade data yang ada pada sistem informasi akademik pasca sarjana fakultas teknik, serta memberikan kemudahan dalam *maintenance (updating)* data.

E. Kerangka Pikir Penelitian

Kerangka pikir pada penelitian ini dapat di tunjukkan pada gambar 2.4 di bawah ini :



Gambar 2.4 : Kerangka Pikir

Sistem informasi yang ada pada Fakultas Teknik Universitas Hasanuddin adalah Sistem Informasi Akademik Program Pasca Sarjana Universitas Hasanuddin, Sistem Informasi Penilaian Angka Kredit Dosen, Sistem Otomatisasi Persuratan Program Pascasarjana Fakultas Teknik, Sistem *On-line* Penyusunan Borang Akreditasi. Dimana, setiap sistem informasi tersebut di kembangkan secara terpisah sehingga adanya perbedaan struktur tabel induk yang sama pada setiap database.

Oleh sebab itu, perlu adanya suatu metode yang dapat mengintegrasikan semua sistem informasi yang sudah ada dengan dengan menggunakan teknologi *web service*, yaitu layanan yang dapat digunakan untuk mengakses data pada tabel induk pada web service siaka. Sehingga, dapat dengan mudah dilakukan pendistribusian data ke sistem informasi yang sudah ada pada Fakultas Teknik Universitas Hasanuddin.