PROYEK AKHIR

ANALISA DATA AKADEMIK MAHASISWA BERBASIS DATA WAREHOUSE DENGAN DSS YANG MEREKOMENDASIKAN BEASISWA BAGI MAHASISWA

Aminul Wahib
NRP. 7405 040 021

Dosen Pembimbing :

Wiratmoko Yuwono, ST
NIP. 197911212005011003

Tessy Badriyah, S.Kom, MT
NIP. 197009142001122001

JURUSAN TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA
2009

PROYEK AKHIR

ANALISA DATA AKADEMIK MAHASISWA BERBASIS DATA WAREHOUSE DENGAN DSS YANG MEREKOMENDASIKAN BEASISWA BAGI MAHASISWA

Aminul Wahib
NRP. 7405 040 021

Dosen Pembimbing :

Wiratmoko Yuwono, ST
NIP. 197911212005011003

Tessy Badriyah, S.Kom, MT
NIP. 197009142001122001

JURUSAN TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2009

ANALISA DATA AKADEMIK MAHASISWA BERBASIS DATA WAREHOUSE DENGAN DSS YANG MEREKOMENDASIKAN BEASISWA BAGI MAHASISWA

Oleh :
AMINUL WAHIB
7405 040 021

Proyek Akhir ini digunakan Sebagai Salah Satu Syarat Untuk
Memperoleh Gelar Sarjana Sain Terapan (SST)
di
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember Surabaya

Disetujui Oleh :

Tim Penguji Proyek Akhir	Dosen Pembimbing
1. Afrida Helen ST. Mkom. NIP. 196501281997032001	1. Wiratmoko Yuwono, ST NIP. 197911212005011003
2. Nana Ramadijanti SKom.MKom NIP. 197111091998022001	2. Tessy Badriyah, S.Kom, MT NIP. 197009142001122001
3. Ira Setyoningrum SSi.MT. NIP. 198005292008122005	

Mengetahui :
Ketua Jurusan Teknik Informatika

Arna Fariza, S.Kom, M.Kom
NIP. 197107081999032001

ABSTRAK

Kegiatan evaluasi, perencanaan dan pengambilan keputusan akan dapat dilakukan dengan lebih baik jika sebuah organisasi memiliki informasi yang lengkap,cepat,tepat dan akurat.

Pada tugas akhir ini kami akan mengkaji ekstraksi data operasional ke dalam sebuah data warehouse yang kemudian di lakukan analisis data menggunakan teknik fuzzy database terhadap data warehouse yang sudah di bangun.

Teknik fuzzy digunakan untuk memperoleh Decision Support System (DSS) yaitu mahasiswa yang paling berhak memperoleh beasiswa yang di ukur dari indek prestasi, tingkat ekonomi, dan kedisiplinan mahasiswa. hasil dari tugas akhir ini adalah sebuah datawarehouse lengkap dengan aplikasi pelaporan informasi berbasis web.

Kata kunci – *data warehouse*, *fuzzy database*, *decision support system(DSS).*

ABSTRACT

Taking evaluation, making plan and taking decision support will get better if the organization have full information, fast, correct and accurate system.

In this final project, we take the analysis of the operational data extraction to the data warehouse Then, we make data analysis with fuzzy database technique to data warehouse which is built.

. Technic fuzzy is used to acquiring the Decision Support System (DSS) that university student is the person who can get the scholarship can be measured by score, economic range and the discipline attitude. The result of this final project is the complete data warehouse with information report application in web based

Keywords - data warehouse, fuzzy database, decision support system(DSS).

KATA PENGANTAR

Alhamdulillah, segala puji syukur bagi Allah SWT karena atas berkat rahmat dan hidayah-Nya, penulis berhasil menyelesaikan buku laporan Proyek Akhir ini yang berjudul:

ANALISA DATA AKADEMIK MAHASISWA BERBASIS DATA WAREHOUSE DENGAN DSS YANG MEREKOMENDASIKAN BEASISWA BAGI MAHASISWA

Proyek akhir ini disusun untuk melengkapi persyaratan akademik dalam menyelesaikan kuliah pogram D4 di **Politeknik Elektronka Negeri Surabaya Jurusan Teknik Informatika.**

Saya menyadari bahwa buku ini masih jauh dari kesempurnaan, oleh karena itu kritik dan saran dari pembaca yang bersifat membangun sangat saya harapkan.

Akhirnya saya berharap semoga buku ini dapat bermanfaat bagi mahasiswa Politeknik Elektronika Negeri Surabaya pada khususnya dan para pembaca pada umumnya. Amiin.

Surabaya, Juli 2009

Penyusun

UCAPAN TERIMA KASIH

Alhamdulillah penulis panjatkan kehadirat Allah SWT atas limpahan rahmat, berkah, dan hidayahNya yang begitu besar berupa akal, pikiran, daya cipta, rasa, dan karsa. Tak lupa juga kepada junjungan kita Nabi besar Muhammad SAW, semoga sholawat dan salam tetap melimpah kepada beliau sehingga penulis dapat menyelesaikan Proyek Akhir ini tepat pada waktunya tentu saja dengan segala kelebihan dan kekurangannya. Penulis sadar dalam terwujudnya Proyek Akhir ini tak lepas dari bantuan, bimbingan, dan dukungan dari berbagai pihak.:

Oleh karena itu, dengan segala kerendahan hati penulis mengucapkan terima kasih kepada:

1. Ir. Dadet Pramadihanto, M.Eng, Ph.D. selaku Direktur Politeknik Elektronika Negeri Surabaya Institut Teknologi Sepuluh Nopember.
2. Ibu Arna Fariza,S.Kom,M.Kom selaku Ketua Jurusan Teknik Informatika.
3. Bapak Wiratmoko Yuwono, ST. dan Ibu Tessy Badriyah, S.Kom, MT., selaku pembimbing yang telah dengan sabar menuntun dan membimbing kami sehingga tugas akhir ini dapat diselesaikan.
4. Seluruh staf dosen yang telah memberikan segala ilmunya kepada kami dan semua karyawan di Politeknik Elektronika Negeri Surabaya.
5. Teruntuk Ayah dan Ibu tercinta yang selalu berdo'a tiada henti demi kesuksesan penulis dan perhatian yang begitu berharga dengan segala kasih sayang yang begitu berarti serta pengorbanan yang telah beliau lakukan selama ini.
6. Buat mas Muzammil ,adik Riqqo dan adik Zulfa yang sangat saya sayangi, thanks atas segala bentuk dukungan dan doanya.
7. Teman-teman kost Arfi, Monchu, Dzakki terimakasih atas dukungannya.
8. Teman-teman D4 IT A '05 , terutama Willy, Jirin dan Joko terimaksih atas semua bantuannya.

9. Dan semua pihak yang tidak bisa disebutkan satu persatu – terimakasih banyak atas bantuannya.

Segala ucapan terimakasih dari saya tentunya belum cukup, semoga Allah SWT membalas kebaikan semua pihak. Akhir kata penulis, semoga proyek akhir ini dapat memberikan manfaat dan tambahan pengetahuan bagi pembaca kelak, Amiin.....

Surabaya, Juli 2009

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perguruan tinggi saat ini di tuntut untuk memiliki keunggulan bersaing dengan memanfaatkan semua sumber daya yang dimiliki. Selain sumber daya sarana, prasarana dan manusia, sistem informasi adalah salah satu sumber daya yang dapat di gunakan untuk meningkatkan keunggulan lembaga pendidikan. Sistem informasi dapat di gunakan untuk mendapatkan, mengolah dan menyebar informasi untuk menunjang kegiatan operasional sehari-hari sekaligus menunjang kegiatan pengambilan keputusan yang strategis.

Pembangunan data warehouse merupakan salah satu cara untuk mengekstrak informasi penting dari data yang tersebar di beberapa sistem informasi [1]. Data yang sudah terintegrasi selanjutnya dapat dimanfaatkan untuk kegiaatan penyampaian informasi yang dapat di tinjau dari berbagai dimensi dan dapat di atur tingkatan rinciannya. Pemanfaatan lebih lanjut dari informasi yang ada dalam data warehouse adalah kegiatan analisa data menggunakan teknikdan metode tertentu.

Dalam proyek tugas akhir ini, dibangun sebuah data warehouse berupa analisa data bedasarkan data transaksional dari data akademik mahasiswa Politeknik Elektronika Negeri Surabaya mulai tahun 2002 sampai tahun 2008.

Kami menyadari bahwa untuk memanfaatkan data yang ada di dalam sistem informasi untuk menunjang kegiatan pengambilan keputusan, tidak cukup hanya mengandalkan data operasional saja, diperlukan suatu analisis data untuk menggali potensi-potensi informasi yang ada. Untuk itulah perlu di bangun sebuah data warehouse yang selanjutnya proses pengambilan pengambilan keputusan dilakukan dengan menggunakan teknik fuzzy database.

1.2. Tujuan

Tujuan dari proyek ini adalah membangun sebuah sistem informasi berbasis data warehouse. Data warehouse dapat menjadi salah satu solusi peningkatan kualitas lembaga pendidikan(khususnya PENS) yaitu dengan mengatur dan mengelola data historis menjadi asset yang berperan dalam pengambilan keputusan, pembuatan laporan, hingga analisis ke depan.

.

1.3. Batasan Masalah

Rumusan masalah pada proyek akhir ini antara lain :

1. Perancangan sumber data.
2. Perancangan arsitektur data warehouse.
3. Membangun data warehouse.
4. Penerapan teknik fuzzy database, untuk menentukan "mahasiswa paling berhak menerima beasiswa" di ukur dari indeks prestasi, tingkat ekonomi, dan kedisiplinannya.

Batasan masalah pada proyek akhir ini adalah :

1. Data yang di kelola hanya data AKADEMIK mahasiswa Politeknik Elektronika Negeri Surabaya mulai tahun 2002 – 2008.
2. Teknik fuzzy hanya di gunakan untuk menentukan mahasiswa yang paling berhak menerima beasiswa.

1.4. Metodologi

Proyek akhir ini dilakukan melalui beberapa tahap yaitu sebagai berikut:

1. **Studi Pustaka**
   Pemahaman studi pustaka tentang konsep dari sistem data warehouse, Proses ETL, database ORACLE, PHP dan semua fitur yang berhubungan dengan aplikasi website.

2. **Pengumpulan Data**
   Mengumpulkan data akademik mahasiswa yang akan diproses dalam mesin warehouse dan mengumpulkan materi-materi penunjang pembuatan sistem dan aplikasi datawarehouse.
   .
3. **Perancangan dan Pembuatan Sistem**
   Melakukan perancangan sistem data warehouse mulai dari pemilihan data-data dan skema yang akan di gunakan untuk proses analisa.

4. **Pembuatan Perangkat Lunak (Software)**
   Hasil dari perancangan dan pembuatan sistem data warehouse ini diimplementasikan kedalam aplikasi web berbasis PHP-ORACLE.

5. **Pengujian dan Analisa**
   Mengevaluasi hasil keseluruhan dan sekaligus test running web sistem informasi.

6. **Penyusunan dan Pembuatan Laporan Proyek Akhir**
   Langkah ini adalah langkah yang paling akhir dari perancangan dan penyusunan Proyek Akhir.

1.5. Sistematika Pembahasan

Sistematika pembahasan yang akan diuraikan dalam buku proyek akhir ini terbagi dalam bab-bab yang akan dibahas sebagai berikut :

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang, tujuan, batasan masalah, metodologi, dan sistematika pembahasan yang digunakan dalam pembuatan proyek akhir ini.

BAB II DASAR TEORI

Menjelaskan teori yang berkaitan dengan data warehouse, skema star, database ORACLE, PHP dan semua fitur yang berhubungan dengan aplikasi website.

BAB III PERANCANGAN DAN PEMBUATAN SISTEM

Membahas mengenai perencanaan sumber data warehouse, arsitektur data warehouse, desain skema star, proses ETL dan menampilkan output data, berbasis web PHP-ORACLE.

BAB IV PENGUJIAN DAN ANALISA

Melakukan pengujian dan analisa mengenai cara kerja dari sistem yang telah dibuat.

BAB V KESIMPULAN

Bab ini berisi kesimpulan dari pembahasan pada perancangan awal serta analisa yang diperoleh. Untuk lebih meningkatkan mutu dari sistem yang telah dibuat maka diberikan saran-saran untuk perbaikan dan penyempurnaan sistem.

DAFTAR PUSTAKA

Pada bab ini berisi tentang referensi-referensi yang telah dipakai oleh penulis sebagai acuan dan penunjang yang mendukung penyelesaian proyek akhir ini.

-- Halaman ini sengaja dikosongkan --

BAB II
DASAR TEORI

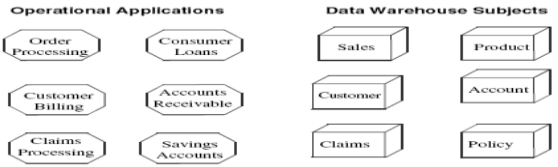
Pada bab ini akan dijelaskan tentang teori-teori yang mendasari permasalahan dan penyelesaian tugas akhir ini. Diantaranya adalah teori tentang data warehouse, Oracle PL/SQL, dan PHP-Oracle. Penjelasan data warehouse meliputi penjelasan tentang schema star, tabel dimensi,tabel fakta dan proses ETL**.**

2.1 Pengertian Data Warehouse

Data warehouse adalah suatu paradigma baru dilingkungan pengambilan keputusan strategik. Data warehouse bukan suatu produk tetapi suatu lingkungan dimana user dapat menemukan informasi strategik [Poniah, 2001, h.14]. Data warehouse adalah kumpulan data-data logik yang terpisah dengan database operasional dan merupakan suatu ringkasan. Adapun karakteristik dari data warehouse [Poniah, 2001,h.20-24] adalah sebagai berikut:

1. Berorientasi subyek

Data warehouse adalah tempat penyimpanan berdasakan subyek bukan berdasakan aplikasi. Subyek merupakan bagian dari suatu perusahaan. Contoh subyek pada perusahaan manufaktur adalah penjualan, konsumen, inventori, daln lain sebagainya. Gambar dibawah ini merupakan perbedaan mengenai data warehouse dan database operasional.



Sumber: Poniah, 2001,h.21

Gambar 2.1 perbedaan data warehouse dan database operasional

Untuk lebih jelasnya mengenai perbedaan antara database operasional dengan data warehouse bisa dilihat pada tabel 1 dibawah ini.

Tabel 2.1 Perbedaan database operasional dan data warehouse

	Database operasional	Data warehouse
Isi data	Bernilai sekarang atau up-to-date	Arsip, *history*, rangkuman
Struktur data	Dioptimasi untuk transaksi, normalisasi	Dioptimasi untuk *query* yang kompleks, Unnormalisasi
Frekuensi akses	Tinggi	Sedang-rendah
Tipe akses	Read, update, delete	Read
Penggunaan	Update secara terus menerus	Update secara periodik
Users	Banyak	Lebih sedikit

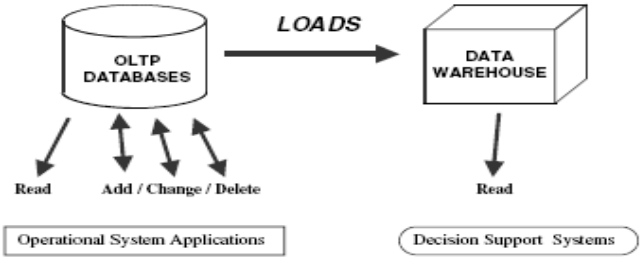
2. Data yang terintegrasi

Sumber data yang ada dalam data warehouse tidak hanya berasal dari database operasional (internal source) tetapi juga berasal dari data diluar sistem (external source). Data pada sumber berbeda dapat di-encode dengan cara yang berbeda. Sebagai contoh, data jenis kelamin dapat di-enkode sebagai 0 dan 1 di satu tempat dan ”m” dan ”f” di tempat lain.

3. Nonvolatile

Data dalam database operasional akan secara berkala atau periodik dipindahkan kedalam data warehouse sesuai dengan jadwal yang sudah ditentukan. Misal perhari, perminggu,

perbulan, dan lain sebagainya. Sekali masuk ke dalam data warehouse, data adalah read-only . Pada gambar 2 dibawah ini bisa dilihat bahwa database OLTP bisa dibaca, diupdate, dan dihapus. Tetapi pada database data warehouse hanya bisa dibaca.



Sumber: Poniah, 2001,h.24

Gambar 2.2 Data warehouse adalah nonvolatile

4. T ime-Variant

Sistem operasional mengandung data yang bernilai sekarang sedangkan data dalam data warehouse mengandung data tidak hanya data terkini tetapi juga data history yang akan digunakan dalam analisis dan pengambilan keputusan. Waktu adalah dimensi penting yang harus didukung oleh semua data warehouse. Data untuk analisis dari berbagai sumber berisi berbagai nilai waktu, misalkan harian, mingguan, dan bulanan.

5. Ringkas

Jika diperlukan, data operasional dikumpulkan ke dalam ringkasan-ringkasan.

6. Granularity

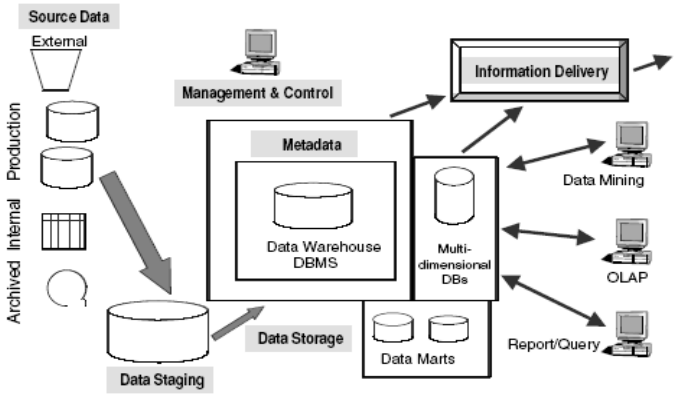
Pada sistem operasional data dibuat secara real-time sehingga untuk mendapatkan informasi langsung dilakukan proses query. Pada data warehouse pada menganalisis harus memperhatikan level-of-detail misalkan perhari, ringkasan perbulan, ringkasan per-tiga-bulan.

7. Tidak ternormalisasi

Data di dalam sebuah data warehouse biasanya tidak ternormalisasi dan sangat redundan.

Dasar dari suatu data warehouse adalah suatu data yang besar yang mengandung informasi bisnis. Data-data yang ada di dalam data warehouse bisa berasal dari banyak sumber, misalkan dari database operasional atau transaksional dan sumber dari luar misalkan dari web, penyedia jasa informasi, dari perusahaan lain, dan lain sebagainya. Data warehouse mengandung beberapa elemen penting antara lain [Mallach, 2000,h.473]:

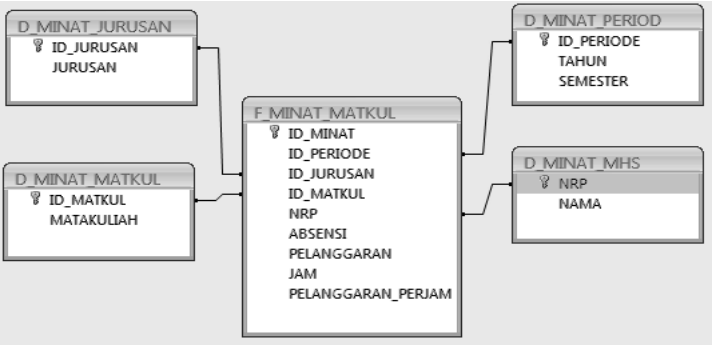
1. Sumber data yang digunakan oleh data warehouse, database transaksional dan sumber data eksternal.
2. Proses ETL (Extraction, Transformation, Loading) dari sumber data ke database data warehouse.
3. Membuat suatu ringkasan atau summary terhadap data warehouse misalkan dengan menggunakan fungsi agregat.
4. Metadata.
   Metadata mengacu data tentang data. Metadata menguraikan struktur dan beberapa arti tentang data, dengan demikian mendukung penggunaan efektif atau tidak efektif dari data.
5. Database data warehouse.
   Database ini berisi data yang detail dan ringkasan data dari data yang ada di dalam data warehouse. Karena data warehouse tidak digunakan dalam proses transaksi individu, maka databasenya tidak perlu diorganisasikan untuk akses transaksi dan untuk pengambilan data, melainkan dioptimisasikan untuk pola akses yang berbeda di dalam analisis.
6. Query Tools yaitu dengan OLAP (Online Analytical Processing). Tool untuk query ini meliputi antarmuka pengguna akhir dalam mengajukan pertanyaan kepada database, dimana proses ini disebut sebagai On-line Analytical Processing (OLAP).
7. User.
   Pengguna yang memanfaatkan data warehouse tersebut.



Gambar 2.3 Alur data arsitektur data warehouse

2.1.1 Star Schema

Pada sistem OLTP (Online Transactional Processing) digunakan suatu teknik pemodelan data yang disebut sebagai E-R (Entity-Relationaship). Pada data warehouse digunakan teknik pemodelan data yang disebut dimensional modelling technique. Pemodelan dimensional adalah suatu model berbasis pemanggilan yang mendukung akses query volume tinggi. Star Schema adalah alat dimana pemodelan dimesional diterapkan dan berisi sebuah tabel fakta pusat. Tabel fakta berisi atribut deskriptif yang digunakan untuk proses query dan foreign key untuk menghubungkan ke tabel dimensi. Atribut analisis keputusan terdiri dari ukuran performa, metrik operasional, ukuran agregat, dan semua metrik yang lain yang diperlukan untuk menganalisis performa organisasi. Tabel fakta menunjukkan apa yang didukung oleh data warehouse untuk analisis keputusan. Tabel dimensi mengelilingi tabel fakta pusat. Tabel dimensi berisi atribut yang menguraikan data yang dimasukkan dalam tabel fakta. Tabel dimensi menunjuk bagaimana data akan dianalisis. Contoh Star Schema ditunjukkan pada gambar 4.



Gambar 2.4 Desain Schema Star

2.1.2 Table Dimensi Waktu

Tabel dimensi waktu adalah suatu tabel dimensi yang harus ada dalam sebuah data warehouse karena setiap data warehouse adalah time series. Waktu merupakan dimensi pertama yang harus diperhatikan dalam proses sort order dalam suatu database karena ketika hal itu pertama kali dilakukan, loading data secara berturut-turut dalam interval waktu tertentu akan masuk kedalam tempat yang sebenarnya dalam sebuah disk.

2.1.3 Proses ETL (Ekstraksi, Transformasi, dan Load data)

Proses ETL merupakan suatu solusi dalam membuat bridge application. ETL merupakan kependekan dari Extract, Transform, Load. Pada dasarnya proses kerja dari application ini adalah pertama yaitu proses Extract. Proses ini adalah proses pengambilan data dari suatu basis data atau database. Data yang diambil dapat berupa bagian tertentu saja, atau tidak keseluruhan data atau hanya data yang diperlukan saja. Kemudian selanjutnya adalah proses Transform. Proses ini memungkinkan data yang di extract diubah menjadi bentuk data baru yang default, atau bisa juga diubah secara custom sesuai dengan keperluan. Biasanya dalam proses tranform ini, data diubah menjadi bentuk atau format yang diperlukan untuk proses load. Untuk proses terakhir adalah proses Load. Dalam proses ini, data yang mengalami

proses transformasi tadi, dikirim ke suatu basis data baru atau basis data tujuan. Dengan demikian telah terjadi suatu proses migrasi data dari basis data dari aplikasi A(database OLTP) ke basis data dari aplikasi B(data warehouse).

Process Extract, Transform dan Loading (ETL) mengambil dan memilih data dari sumber data yang akan diolah ke dalam data warehouse yang disesuaikan dengan kebutuhan Analisa. ETL berkoneksi dengan source system database dan mengambil data dengan query. Setelah data hasil query diambil langkah selanjutnya dilakukan eksekusi proses ETL dan mengirimnya ke database data warehouse. Pada proyek akhir ini proses ETL di bangun menggunakan PL/SQL Query.

2.2 Oracle

Oracle adalah salah satu perusahaan yang membuat database software. Pada mulanya sangat banyak perusahaan-perusahaan yang membuat database software, tetapi karena persaingan yang berat, saat ini hanya tinggal sejumlah kecil perusahaan yang membuat database software.

Oracle merupakan RDBMS (Relational Database Management System) yang paling banyak digunakan oleh perusahaan-perusahaan di dunia ini. Sejarah perkembangannya yang cukup panjang telah membawa Oracle menjadi database yang paling banyak digunakan.

2.2.1 Sejarah Oracle

Database Oracle adalah produksi dari Oracle Corporation, sebuah perusahaan komputer raksasa yang saat ini bermarkas di Redwood City, California. Kisah database Oracle dumulai saat Larry Ellison melihat peluang bagus yang belum dimanfaatkan oleh perusahaan-perusahaan kala itu. Bersama dua orang temannya, Bob Miner dan Ed Qates, dia mendirikan sebuah perusahaan bernama *relational database* menggunakan bahasa C.

Tahun 1979, versi pertama dijual kepada umum. Versi pertama telah menyertakan interface SQL untuk berinteraksi

dengan database. Tahun 1983, mereka mengubah nama perusahaannya menjadi Oracle Corporation. Pada tahun itu juga, Oracle Corporation meluncurkan database versi ketiga mereka. Pada versi keempat yang diluncurkan pada tahun 1984, Oracle telah mendukung beberapa operasi sistem yang ada kala itu. Pengembangan terus dilakukan sesuai dengan kemajuan teknologi komputer. Pada versi yang kedelapan, yang dipasarkan sejak tahun 1998, Oracle mulai mengadopsi konsep orientasi objek (*object oriented*). Konsep orientasi objek pada database sedikit berbeda dengan konsep yang dikenal dengan pemrograman. Pada perkembangan selanjutnya pada versi 8 ini, Oracle memperkenalkan fitur-fitur baru yang dikenal di lingkungan internet sehingga mereka membubuhkan huruf "i" yang merupakan huruf awal "Internet".

Selain memproduksi database, Oracle Corporation juga memproduksi Application Server, Development Tools, E-Business, dan lain-lain.

2.2.2 Kehandalan Database Oracle

Beberapa keunggulan database Oracle yang menempatkannya sebagai produk database yang paling banyak dipakai tertulis dibawah ini.

- *Scalability*, kemampuan menangani banyak user yang melakukan koneksi secara simultan tanpa berkurangnya *performance* secara signifikan. Dalam dokumentasi Oracle disebutkan bahwa database Oracle sanggup melayani puluhan ribu user secara simultan.
- *Reliability* yang bagus, yakni kemampuan untuk melindungi data dari kerusakan jika terjadi kegagalan fungsi pada sistem seperti *disk failure*.
- *Serviceability*, yaitu kemampuan untuk mendeteksi masalah, kecepatan dalam mengoreksi kesalahan, dan kemampuan melakukan konfigurasi ulang struktur data.
- *Stability*, yaitu kemampuan untuk tidak *crash* karena beban kerja yang tinggi. Hal ini berkaitan dengan *scalability.*

- *Availability,* yaitu kemampuan dalam penanganan *crash* atau *failure* agar service dapat tetap berjalan. Misalnya saja dengan tersedianya fasilitas pendistribusian database pada beberapa data *server* dan juga pemulihan database (*recovery*).
- *Multiplatform.* Dapat digunakan pada banyak sistem operasi seperti Windows, Unix, Linux, dan Solaris.
- Mendukung data yang sangat besar. Menurut dokumentasi Oracle, dapat menampung sampai 5112 petabytes (1 petabytes = 1.000.000 gigabytes).
- Sistem sekuriti yang cukup handal.
- Mendukung database berorientasi objek.
- Dapat menampung hampir semua tipe data seperti teks, image, sound, video, dan time series.

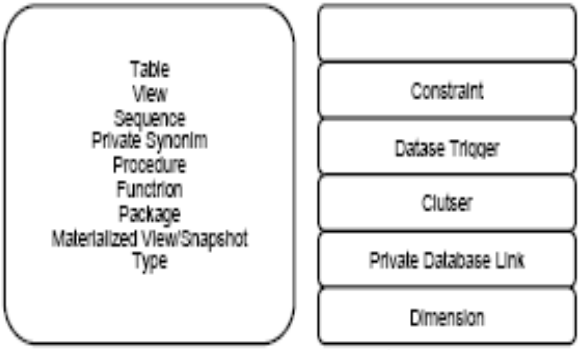
Keunggulan-keunggulan diatas membuat Oracle banyak digunakan pada aplikasi internet maupun aplikasi client-server pada Local Area Network (LAN). Pemilihan database sendiri sangat bergantung pada banyak hal seperti banyaknya data, banyaknya *concurrent user*, kemungkinan pengembangan bisnis, biaya yang tersedia, dan lain-lain. Kebanyakan perusahaan menengah ke atas memilih Oracle sebagai database yang mereka gunakan karena kehandalannya.

2.2.3 Object Schema

Database terdiri dari objek-objek yang disimpan dalam schema dan objek-objek yang tidak dikelola oleh schema. Schema dimiliki oleh user database dan nama schema itu sama dengan nama user-nya. Schema merupakan tempat penyimpanan objek-objek seperti sequence, table, private sinonim, indeks, provate database link, dan stored-procedure; sedangkan objek non-schema merupakan objek yang disimpan di dalam database namun tidak menempati suatu schema seperti context, profile, role dan user, directory, dan tablespace.

Gambar berikut memperlihatkan namespace untuk objek schema dan tampak bahwa tabel, view, sequence serta type

berada dalam namespace sama sehingga tidak boleh ada nama objeknya yang sama dalam satu namespace.



Gambar 2.5 Namespace schema

2.2.4 Tipe Data

Tiap kolom memiliki sebuah tipe data, yang berhubungan dengan format penyimpanan. Berikut adalah tipe data built-in pada Oracle:

- CHAR
- VARCHAR2
- NCHAR and NVARCHAR2
- NUMBER
- DATE
- BLOB
- CLOB and NCLOB
- BFILE
- XMLType

Tipe data CHAR, VARCHAR2, NUMBER, dan DATE disimpan secara langsung pada variable PHP. Tipe data BLOB, CLOB, dan BFILE menggunakan PHP descriptors. Tipe data NVARCHAR2, dan NCLOB tidak support pada OCI8 extension.

2.3 SQL (Struktur Query Language)

Telah dikatakan sebelumnya bahwa Oracle telah menggunakan SQL sebagai bahasa perantara antara user dan database-nya sejak pertama sekali Oracle dibuat. Kemudian, Oracle Corporation juga mengembangkan sebuah bahasa prosedural yang mereka sebut sebagai PL/SQL. Dengan menggunakan kedua bahasa ini seorang user dapat mengoptimalkan penggunaan database Oracle.

2.3.1 Sejarah dan Standardisasi SQL

Bahasa SQL mulanya dikembangkan untuk mendukung model relational yang digagaskan oleh Codd. Setelah gagasan itu diluncurkan, lembaga riset IBM menghabiskan banyak waktu untuk mengembangkan *relational database* berikut bahasa pendukungnya.

Kemudian, IBM berhasil membuat sebuah *relational database* yang bernama System/R pada tahun 1978. System/R didukung oleh sebuah bahasa yang disebut *Structure Query Language* (SEQUEL). Pada perkembangan berikutnya, nama bahasa tersebut disingkat menjadi SQL. Walaupun telah menjadi SQL, saat ini kebanyakan orang masih menyebutnya dengan "sequel" karena dipengaruhi oleh sejarah masa lalunya.

Relational Software Incorporated, cikal bakal Oracle Corporation, juga ikut berperan dalam pengembangan SQL. Sejak relational database versi pertama yang mereka keluarkan tahun 1979, mereka telah mengadopsi SQL untuk mendukung database produk mereka. Relational Software Incorporated menjadi perusahaan pertama mengomersialkan relational database berikut bahasa pendukungnya, SQL. IBM sendiri terus mengembangkan System/R dan SQL. Riset yang dilakukan IBM menghasilkan dua produk yang mulai dipasarkan kepada umum pada tahun 1982, yakni SQL/Data System (SQL/DS) dan DB2. Setelah itu vendor-vendor lain juga mulai mengembangkan *relational database*. Kebanyakan dari mereka menggunakan SQL yang mengacu kepada SQL yang digunakan pada DB2.

Usaha standarisasi dimulai pada tahun 1986 oleh *American National Standards Institute* (ANSI) dan tahun 1987

oleh *International Standards Organization* (OSI). Beberapa tahun kemudian, ISO dan ANSI mempublikasikan SQL standar pada tahun 1989 dan dikenal sebagai SQL-89 (SQL1). Sampai saat ini, ANSI dan ISO telah mempublikasikan tiga standar SQL, dua yang berikutnya dikenal sebagai SQL-92 (SQL2) dan SQL-99 (SQL3). Oracle8i menggunakan standar SQL-92, sedangkan Oracle9i menggunakan standar SQL-99.

Kebanyakan RDBMS yang ada saat ini menyertakan SQL-ANSI yang diperluas. Misalnya saja, T-SQL pada SQL Server produksi Microsoft, Oracle SQL dan PL/SQL produksi Oracle Corporation.

2.3.2 Gambaran Umum SQL

SQL merupakan bahasa yang digunakan oleh banyak aplikasi atau tool untuk berinteraksi dengan server database. SQL adalah bahasa fungsional yang tidak mengenal iterasi dan tidak bersifat prosedural. SQL menggunakan perintah-perintah dengan kata-kata sederhana dan mirip dengan bahasa manusia sehari-hari. Berikut ini contoh sederhana perintah SQL (*SQL Statement*).

```
SELECT deskripsi FROM T_POS;
```

Secara sederhana, perintah diatas diartikan sebagai "pilih data pada kolom deskripsi dari tabel "T_POS". begitu server menerima perintah ini, server akan menginterpretasikan perintah tersebut dan mengirimkan hasilnya ke *requester*.

Sebagai catatan, SQL pada Oracle8i kompatibel dengan SQL92 (SQL2) sedangkan SQL pada Oracle9i kompatibel dengan SQL99 (SQL3). Tentu saja, Oracle SQL mengalami beberapa perbedaaan dengan SQL ANSI. Oracle menyebut SQL yang mereka gunakan pada database-nya sebagai SQL saja tanpa tambahan atribut lainnya

2.3.3 SQL Statement

SQL statement dapat dikelompokkan berdasarkan fungsi dari masing-masing perintah. SQL statement tidak bersifat *case sensitive*. Sebagai contoh, perintah "SELECT" sama saja artinya

dengan “SeLeCt”. Untuk memudahkan pembacaan, sebaiknya, membuat aturan sendiri. Misalnya, perintah-perintah SQL dan kata-kata kunci lainnya ditulis dengan huruf besar. Nama kolom ditulis dengan huruf kecil dan seterusnya.

SQL statement selalu diakhiri dengan sebuah semicolon (;). Pada environment SQL*PLUS, SQL statement dapat juga diakhiri dengan menuliskan slash (/) pada baris setelah baris SQL statement yang terakhir. Setelah memberikan tanda akhir, tekan tombol ENTER untuk mengeksekusi statement tersebut. Perlu diingat, apabila tanda slash ada setelah tanda semicolon pada sebuah SQL statement, statement tersebut akan dieksekusi sebanyak dua kali.

Tabel 2.2 SQL statement

Statement	Deskripsi
SELECT	Mencari dan memperoleh data dari database. Statement ini sering dikelompokkan kedalam Data Manipulation Language.
INSERT UPDATE DELETE	Memasukkan, mengubah, dan menghapus data. Ketiganya disebut dengan istilah Data Manipulation Language (DML).
CREATE ALTER DROP RENAME TRUNCATE	Membuat, mengubah, dan menghapus struktur data dari objek-objek database. Kelompok ini disebut dengan istilah Data Definition Language (DDL)
SET TRANSACTION COMMIT ROLLBACK SAVEPOINT	Mengatur perubahan yang dibuat dengan DML statement, disebut dengan istilah Transaction Control Statement.
GRANT REVOKE	Memberikan dan menghapus hak akses terhadap Oracle database dan struktur didalamnya. Disebut dengan istilah Data Control Language (DCL), tetapi sering juga dikelompokkan kedalam Data Definition Language.

2.3.4 Membuat Object Database

Berikut ini dibahas mengenai cara membuat, mengubah, serta menghapus beberapa objek database.

a) Tabel

Tabel merupakan unit penyimpanan data yang terdiri atas sejumlah baris (rows) dan sejumlah kolom (columns). Kolom-kolom pada suatu tabel didefinisikan ketika tabel tersebut dibuat. Jumlah baris tergantung pada operasi DML yang dikenakan padanya dan cenderung berubah.

- **Create Table**

Perintah CREATE digunakan untuk membuat tabel.

```
CREATE TABLE [Pemakai.] nama_tabel
({nama kolom pertama Tiepdata [default] [constraint kolom] constraint tabel)
[,(nama kolom kedua Tipedata [default] [constraint kolom] constraint tabel } ] ...)
[AS QUERY]
```

- **Alter Table**

Perintah ini digunakan untuk mengubah (modify) tabel yang telah dibuat, seperti:

- Menambah kolom baru.
- Mengubah ukuran kolom.
- Mengubah aturan-aturan yang berlaku untuk suatu kolom.

```
ALTER TABLE [Pemakai] nama_tabel
{[ADD ({nama_kolom | [CONSTRAINT nama_constraint]
Constraint Table }
Constraint Table }] ... )]
[ MODIFY (nama kolom[,nama_kolom]...)]
[DROP nama_constraint]}
[ENABLE nama_constraint | Disable nama_constraint ]
```

- **Truncate**

Membuang tabel disebut juga dengan men-drop, dan mengosongkan dengan men-truncate. Operasi truncate

mengosongkan row dari tabel, namun tetap mempertahankan strukturnya.

```
TRUNCATE TABLE [schema.] tabel
```

- **Drop Tabel**

  Perintah ini digunakan untuk menghapus suatu tabel.

```
DROP TABLE [schema.] tabel
[CASCADE COSNTRAINTS]
```

b) View

User menyimpan data untuk digunakan kembali, namun tidak semua nilai data dari kolom-kolom suatu tabel diperlukan oleh user yang berbeda. Pertimbangannya adalah perbedaan otoritas, perspektif, lokasi kerja, departemen dan sebagainya sehingga data suatu tabel (*base table*) yang diimplementasikan dengan view dapat dipandang secara berbeda menurut perspektif user. Sumber data view dapat berasal dari tabel atau view lain. Mirip dengan tabel, kita dapat melakukan update, delete dan insert pada view sehingga perubahan itu akan direfleksikan pada *base table*-nya. Berbeda dengan tabel, view tidak menyimpan data, view hanya menyimpan definisi query pada data dictionary dan tidak memerlukan ruang penyimpanan data. Penerapan view dapat diaplikasikan pada situasi berikut:

- Membatasi akses sesuai dengan otoritas user.
- Memudahkan pemahaman terhadap kolom penampungan data yang mungkin berbeda definisi kolom pada tabel dasar.
- Menyederhanakan pandangan user terhadap data.
- Menangani data kompleks.
- Memudahkan penggunaan query yang berulangan karena disimpan sebagai stored-query.

- **Membuat View**

  Berikut adalah statement untuk membuat view.

```
CREATE [ OR REPLACE ] VIEW [ schema .]view [ ( { col_alias } ) ]
```

```
AS subquery [ WITH <READ ONLY> |
<CHECK OPTION [CONSTRAINT constraint ] > ];
```

- **Memodifikasi View**

Memodifikasi view yang sudah disimpan di database dapat dilakukan dengan menjalankan CREATE VIEW statement yang menyertakan kata kunci OR REPLACE. Tanpa kata kunci OR REPLACE modifikasi sebuah *existing view* hanya dapat dilakukan dengan cara menghapus view tersebut terlebih dahulu kemudian membuatnya kembali.

- **Alter View Statement**

Statement berikut digunakan untuk meng-*compile* ulang sebuah view yang sudah tersimpan di database.

```
ALTER VIEW [schema. ]view COMPILE;
```

- **Menghapus View**

View dapat dihapus menggunakan DROP VIEW statement. Bentuk umum statementnya seperti yang diperlihatkan di bawah ini.

```
DROP VIEW [schema .]view;
```

c) Indeks

Indeks mempercepat waktu pencarian data di mana operasi full-scan tidak perlu dilakukan. Pencarian itu menggunakan *rowid* untuk menemukan row sehingga mengurangi proses I/O disk. Agar data dalam indeks tetap mutakhir, update indeks selalu dilakukan setiap kali terjadi operasi DML. Jadi indeks pada suatu sisi akan meningkatkan unjuk kerja query dan pada sisi lain melakukan proses tambahan untuk membuat informasi undo dan memperbaharui pohon indeks.

Indeks independent terhadap tabel sehingga indeks bisa dihapus atau diciptakan ulang tanpa mempengaruhi nilai data,

dan bersifat optional. Untuk menyimpan datanya, indeks memerlukan space di segmen indeks.

- **Membuat Indeks**

Bentuk umum statement untuk membuat sebuah index ditunjukkan berikut ini.

```
CREATE INDEX [schema .]index
ON [schema .] table ({ column [ ASC|DESC] });
```

Seperti sudah disebutkan diatas, index dibuat dengan tujuan untuk mempercepat akses data. ini tidak berarti bahwa semakin banyak index, akses data akan semakin cepat. Index dari suatu tabel harus di-*update* setiap kali operasi DML pada tabel tersebut di-*commit*. Dengan demikian, semakin banyak index dari sebuah tabel , semakin banyak pekerjaan yang dilakukan oleh Oracle server untuk mengupdate semua index tersebut setiap kali sebuah DML statement di-*commit*. Oleh karenanya, buatlah index sesuai kebutuhan saja.

- **Mengganti Nama Index**

Index yang sudah dibuat dapat diganti namanya menggunakan statement seperti dibawah ini.

```
ALTER INDEX [schema .]index RENAME TO new_index_name;
```

- **Menghapus Index**

Untuk menghapus sebuah index, gunakan DROP INDEX statement dengan bentuk umum seperti berikut.

```
DROP INDEX [schema .]index;
```

d) Procedure

Stored procedure merupakan schema objek yang berisi sekumpulan SQL statement dan perintah-perintah PL/SQL, yang berjalan di database dan bekerja sebagai sebuah unit yang dapat mengerjakan sekumpulan tugas. Stored procedure sering disebut dengan procedure saja.

- **Membuat Procedure**

Bentuk umum statement untuk membuat sebuah procedure adalah sebagai berikut:

```
CREATE [ OR REPLACE ] PROCEDURE [schema. ]procedure
[ ( { argument [ IN | OUT | IN OUT ] datatype [ DEFAULT value ] } ) ]
[ AUTHID <CURRENT_USER | DEFINER> ]
< IS | AS > pl/sql_subprogram_body;
```

- **Menjalankan Procedure**

Adapun sintak umum untuk mengeksekusi sebuah procedure adalah seperti tertera di bawah ini:

```
EXECUTE nama_procedure(parameter_1, ... );
```

- **Meng-compile Ulang Definisi Procedure**

Procedure kadang perlu di-recompile karena adanya perubahan-perubahan yang telah dilakukan terhadap definisi objek-objek database yang menjadi acuan procedure tersebut. Recompile dimaksudkan untuk memastikan apakah suatu procedure masih valid atau tidak. Statement yang digunakan untuk peng-compile-an ulang tersebut mempunyai bentuk umum seperti berikut ini.

```
ALTER PROCEDURE [schema. ]procedure COMPILE [DEBUG];
```

Pilihan DEBUG dapat diberikan untuk meminta PL/SQL compiler men-generate dan menyimpan kode hasil compile.

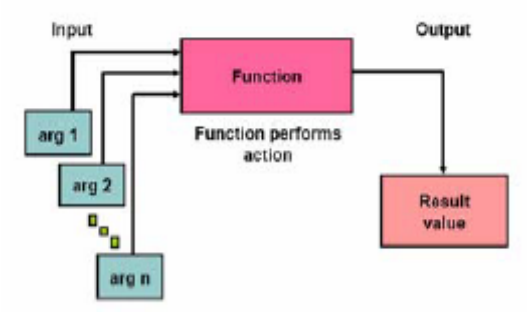
- **Menghapus Procedure**

Menghapus procedure dapat dilakukan dengan menggunakan DROP PROCEDURE statement dengan bentuk umum seperti dibawah ini.

```
DROP PROCEDURE [schema. ]procedure;
```

e) Function

Stored function tidak berbeda jauh dengan stored procedure. Perbedaan satu-satunya adalah stored function selalu mempunyai nilai balikan.



Gambar 2.6 Diagram blok struktur function

➢ **Membuat Function**

Statement yang digunakan untuk membuat function mempunyai bentuk umum seperti berikut.

```
CREATE [ OR REPLACE ] FUNCTION [ schema . ]function
[ ( { argument [ IN | OUT | IN OUT ] ] datatype [ DEFAULT value ] } ) ]
RETURN datatype
[ AUTHID <CURRENT_USER | DEFINER> ]
< IS | AS > pl/sql_subprogram_body;
```

Perbedaaan CREATE FUNCTION dengan CREATE PROCEDURE hanya terletak pada kata kunci RETURN saja. Kata kunci tersebut bertugas untuk menentukan tipe data dari nilai balikan, dinyatakan oleh datatype setelah kata kunci RETURN. Perbedaan lainnya adalah function dapat dipanggil dari SELECT dan DML statement.

➢ **Mencompile Ulang Function**

Berikut ini bentuk umum statement untuk meng-compile ulang sebuah function.

```
ALTER FUNCTION [schema. ]function COMPILE [DEBUG];
```

➢ **Menghapus Function**

Statement di bawah ini digunakan untuk menghapus function.

```
DROP FUNCTION [schema. ]function;
```

f) Trigger

Oracle menyediakan trigger yang berjalan sebagai reaksi atas adanya aksi yang ditimbulkan oleh suatu kejadian (*event*). Aksi ini dapat berupa proses pendefinisian, manipulasi data maupun *event* database. Trigger tidak sekedar menerapkan constraint bisnis tetapi bisa diterapkan untuk berbagai situasi berikut:

- Menerapkan referential integrity pada sistem database terdistribusi.
- Menerapkan aturan bisnis yang tidak dapat dinyatakan pada deklaratif *constraint check, not null, promary key, foreign key, delete cascade* dan *delete set null.*
- Memvalidasi transaksi data.
- Memutahirkan data tabel ketika terjadi manipulasi view.
- Membuat log audit dan log suatu event terhadap transaksi yang dikerjakan.

Eksekusi trigger berlangsung secara otomatis jika terjadi event pemicu. Trigger dapat diapandang sebagai stored-procedure karena disimpan dalam database dan mendukung penggunaan statement SQL dan PL/SQL. Perbedaannya terletak pada cara eksekusi dimana procedure harus dipanggil secara eksplicit oleh aplikasi sedangkan eksekusi trigger dikerjakan secara implicit oleh Oracle ketika suatu event atau statement SQL terjadi.

➢ **Mendapatkan Nilai Kolom dengan menggunakan Alias**

Alias atau referensi merupakan alat yang digunakan untuk mendapatkan nilai-nilai kolom yang terdapat pada tabel. Oracle menyediakan dua buah alias, yaitu **:new** dan **:old.** Alias :new digunakan untuk mengambil nilai kolom dari baris yang akan dimasukkan ke dalam sebuah tabel, sedangkan alias :old

digunakan untuk mengambil nilai kolom dari baris yang tersimpan di dalam tabel. Sintak umum untuk mengambil nilai kolom dari suatu tabel dengan menggunakan alias adalah sebagai berikut.

- :new.JUMLAH_STOK
- :new.NAMA_BARANG
- dan sebagainya

Setiap proses manipulasi data (INSERT, UPDATE, dan DELETE) mempunyai alias yang berbeda-beda.

- Proses INSERT
  Pada proses INSERT, di dalamnya hanya terdapat alias :new yang berfungsi untuk mengambil nilai kolom dari baris yang akan dimasukkan. Misalnya, terdapat sintak SQL berikut.
  :new.KODE akan bernilai '0007',
  :new.NAMA_BARANG akan bernilai 'Komputer' dan
  :new.JUMLAH_STOK akan bernilai 5.
- Proses UPDATE
  Pada proses UPDATE, memiliki dua buah alias yaitu :new dan :old. Alias :new digunakan untuk mengambil nilai baru *(nilai yang digunakan untuk mengubah data)* dan alias :old digunakan untuk mengambil nilai lama *(nilai yang akan diubah)*.
- Proses DELETE
  Dalam proses DELETE, hanya dikenal sebuah alias :old yang digunakan untuk mengambil nilai kolom dari baris yang dihapus.

➢ **Membuat Trigger**

Bentuk umum CREATE TRIGGER statement adalah seperti berikut ini.

```
CREATE [ OR REPLACE ] TRIGGER [schema .]trigger
< BEFORE | AFTER | INSTEAD OF>
dml_event _clause ON [schema .] table [ FOR EACH ROW ]
[ WHEN (condition) ] pl/sql_blok;
```

Bentuk umum di atas dapat dibagi ke dalam tiga bagian dasar, yakni:

- Event atau statement pemicu: dinyatakan pada bagian statement yang dimulai dari CREATE sampai sebelum klausa WHEN.
- Batasan Trigger: kondisi yang dinyatakan pada klausa WHEN.
- Aksi Trigger: dinyatakan oleh *pl/sql_block* yang isinya berupa perintah-perintah PL/SQL yang harus dilakukan ketika event pemicu muncul.

➢ **Alter Trigger**

Statement ini digunakan untuk membuat sebuah existing trigger menjadi aktif atau tidak aktif atau untuk meng-compile ulang trigger. Bentuk umum perintahnya adalah sebagai berikut.

```
ALTER TRIGGER [schema .]trigger
<ENABLE | DISABLE | <COMPILE [DEBUG] >>;
```

➢ **Menghapus Trigger**

Statement ini digunakan untuk menghapus trigger. Bentuk umumnya seperti terlihat berikut ini.

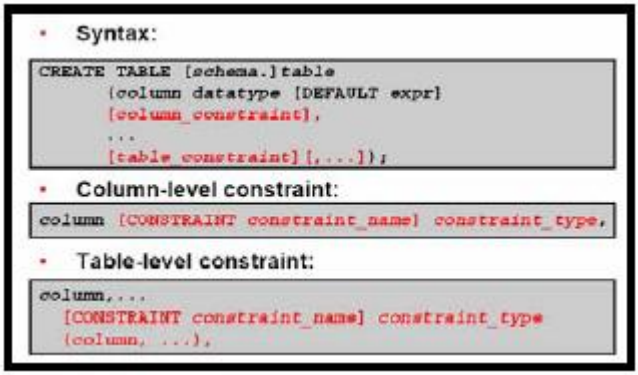
```
DROP TRIGGER [schema .]trigger
```

2.3.5 Constraint

Sebuah constraint atau lebih dapat diberikan ke suatu tabel. Kegunaan constraint tersebut adalah untuk membuat suatu aturan atau batasan yang berlaku pada tabel tersebut sehingga dapat mencegah kesalahan operasi DML yang dikenakan kepadanya. Pembuatan constraint dapat dilakukan pada waktu yang bersamaan dengan pembuatan tabel (menggunakan CREATE TABLE) atau setelah tabel tersebut dibuat (menggunakan perintah ALTER TABLE).

2.3.5.1 Tipe-tipe Constraint

Oracle menyediakan lima constraint yang dapat digunakan untuk memberikan batasan-batasan operasi DML baik pada level kolom maupun level tabel. Constraint level kolom adalah constraint yang hanya melibatkan satu kolom saja dan didefinisikan langsung pada definisi kolom tersebut. Sedangkan constraint level tabel melibatkan lebih dari satu kolom (*composite constraint*) atau didefinisikan secara terpisah, seperti yang ditunjukkan gambar dibawah ini.



Gambar 2.7 Constraint level kolom dan level tabel

Kelima constraint adalah sebagai berikut:

a) NOT NULL

Berikut umum klausa constraint untuk tipe ini adalah sebagai berikut.

[CONSTRAINT constraint_name] NOT NULL

Constraint NOT NULL bertujuan untuk membuat suatu kolom tidak boleh berisi nilai NULL. Untuk tiap baris yang dimasukkan, kolom tersebut harus diberi nilai aktual. Apabila constraint NOT NULL tidak diberikan, secara default kolom tersebut dapat berisi NULL.

b) UNIQUE

Constraint tipe ini berlaku pada level kolom maupun level tabel. Bentuk umum klausa constraint tipe ini ada dua. Untuk level kolom adalah sebagai berikut.

[CONSTRAINT *constraint_name*] UNIQUE

Sedangkan bentuk umum untuk level tabel adalah sebagai berikut.

[CONSTRAINT *constraint_name*] UNIQUE (

col_1 [, col_2[, ...[, col_n]]])

Constraint UNIQUE ini bertujuan untuk membuat sebuah kolom atau kombinasi dari beberapa kolom pada sebuah tabel menjadi bersifat unik. Dengan demikian didalam tabel tersebut tidak diperbolehkan ada dua baris data yang nilai unique key-nya sama.

c. PRIMARY KEY

Constraint tipe ini juga berlaku pada level kolom maupun level tabel. Bentuk umum untuk level kolom adalah sebagai berikut.

[CONSTRAINT *constraint_name*] PRIMARY KEY

Sedangkan bentuk umum untuk level tabel adalah sebagai berikut.

[CONSTRAINT *constraint_name*] PRIMARY KEY (
col_1 [, col_2 [, ...[, col_n]]])

Constraint ini bertujuan untuk menentukan suatu kolom atau kombinasi dari beberapa kolom sebagai primary key dari sebuah tabel. Primary key merupakan sebuah identitas dari baris-baris data pada suatu tabel. Sebagai identitas, primary key hanya boleh ada satu pada sebuah tabel, tentu saja dapat berupa kombinasi kolom-kolom. Selain itu kolom atau kombinasi kolom yang menjadi primary key tersebut tidak boleh diberi nilai NULL. Dengan kata lain, primary key adalah unique key yang diberi constraint NOT NULL. Sebagai catatan, primary key tidak boleh diberi constraint UNIQUE tetapi diperbolehkan diberi constraint NOT NULL.

d. FOREIGN KEY

Constraint ini bertujuan untuk menetapkan suatu kolom atau kombinasi dari beberapa kolom menjadi foreign key dari sebuah tabel. Foreign key sering disebut sebagai *referential integrity constraint.* Kegunaan foreign key adalah untuk membentuk sebuah relasi antara suatu tabel dengan tabel lainnya. Kolom atau kombinasi kolom yang menjadi foreign key pada tabel anak (*dependent/child table*) mengacu ke kolom atau kombinasi kolom yang menjadi primary key atau unique key pada tabel induk (*referenced/parent table*). Tabel induk dapat berupa tabel itu sendiri atau tabel lainnya. Berikut umum klausa constraint untuk membuat sebuah foreign key pada level kolom.

[CONSTRAINT *constraint_name*]
REFERENCE [schema.]table [(column)]
[ON DELETE <CASCADE | SET NULL>]

Sedangkan bentuk umum untuk level tabel adalah sebagai berikut.

[CONSTRAINT *constraint_name*] FOREIGN KEY (col_list)
REFERENCE [schema.]table [(col_list)
[ON DELETE <CASCADE | SET NULL>]

Nilai *col_list* adalah col_1 [, col_2 [, … [, col_n]]]

e. CHECK

Constraint CHECK dapat berlaku untuk level kolom maupun level tabel. Bentuk umum klausa CHECK untuk kedua level tersebut sama saja, yakni seperti yang ditunjukkan berikut ini.

[CONSTRAINT *constraint_name*] CHECK (condition)

Constraint CHECK berguna untuk membuat suatu kondisi yang harus dipenuhi oleh setiap baris data di dalam suatu tabel. Kondisi tersebut dapat berupa salah satu dari kondisi-kondisi berikut ini:

- Kondisi perbandingan.
- Kondisi range.
- Kondisi NULL.
- Kondisi LIKE.
- Kondisi EXISTS.

Kondisi yang memenuhi adalah yang menghasilkan nilai TRUE atau unknown (untuk nilai NULL). Kondisi pada constraint CHECK di suatu tabel dapat mengacu ke sembarang kolom pada tabel tersebut tetapi tidak boleh mengacu ke kolom-kolom pada tabel lainnya. Ketika Oracle mengevaluasi suatu kondisi pada constraint CHECK untuk sebuah baris, nama nama kolom yang diberikan pada kondisi tersebut mengacu ke nilai-nilai kolom pada baris tersebut.

2.3.5.2 Status Constraint

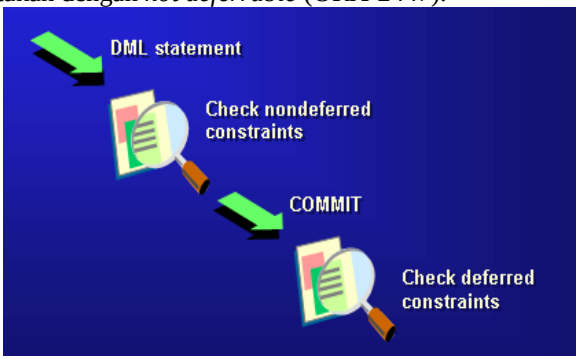
Constraint bisa diterapkan ketika transaksi di-commit (*deferred*) atau segera dijalanakan (*immediate*) pada akhir setiap perintah manipulasi data. Secara default validasi constraint dilakukan pada akhir statement insert dan update. Jika validasi gagal, kesalahan dilaporkan dan perbaikan nilai data harus dilakukan untuk memenuhi rule constraint. Apabila validasi berhasil dan commit dijalankan, maka data diperbarui dan transaksi berakhir. *Deferred* dan *immediate* itu dapat ditetapkan melalui statement DDL seperti *create* dan *alter table* serta perintah *set constraint.*

Status enable memastikan modifikasi data memenuhi kondisi constraint yang saat ini aktif (*enforcement*). Status *validate* memvalidasi isi semua row tabel agar tidak melanggar kondisi constraint. Validasi dan enforcement data dapat dijalankan sendiri-sendiri (independent) sehingga mungkin saja constraint telah di-enforce namun tidak divalidasi atau sebaliknya. Secara default constraint didefinisikan dengan enable dan validate.

- **DEFERRABLE dan NOT DEFERRABLE**

Constraint yang didefinisikan dengan klausa *initially immediate* menetapkan agar pengecekan constraint dilakukan pada saat berakhirnya statement manipulasi data, sedangkan *initially deferred* memvalidasi constraint ketika transaksi di-commit. Perubahan status dari *deffered* ke *immediate* atau sebaliknya dapat dilakukan apabila constraint didefinisikan

dengan clausa *deferrable*, *initially deferred* atau *deferrable initially immediate*. Status constraint itu tidak dapat diubah jika dinyatakan dengan *not deferrable* (ORA-2447).



Gambar 2.8 Deferred constraints

- **RELY dan NORELY**

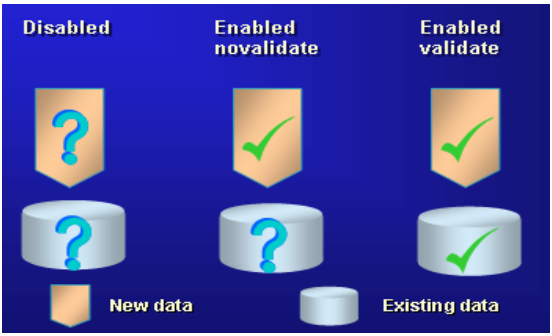
Rely mengaktifkan constraint yang berstatus *novalidate* untuk digunakan oleh query rewrite pada *unenforced* query rewrite integrity. Rely tidak melakukan enforcement dan sering digunakan pada materialized view dan query rewrite (default *norely*). Pada model optimasi CBO, Oracle menggunakan query rewrite untuk mengakses materialized view. Query rewrite menggunakan constraint berstatus *validate* dan *rely novalidate* bergantung pada mode yang ditentukan oleh parameter inisialisasi QUERY_REWRITE_INTEGRITY. Rely dan norely juga digunakan jika ada modifikasi terhadap constraint yang sudah ada.

- **ENABLE dan DISABLE**

Secara otomatis constraint aktif pada saat diciptakan sehingga sistem selalu memeriksa nilai data untuk rule yang ditetapkan. Constraint ini dapat dinonaktifkan misalnya ketika mengimpor data atau mengubah key tabel dengan klausa *disable.* Jika kemudian constraint tadi diaktifkan dengan *enable*, efeknya

akan diterapkan pada nilai data yang baru. Kedua klausa ini menentukan bilamana *enforcement* constraint harus diterapkan.

Untuk memastikan semua data lama dan data baru dalam tabel mengikuti rule maka digunakan *enabled validate*. Adapun *disabled validate* akan menonaktifkan dan menghapus indeks constraint sementara memelihara validitas data. *Enable novalidate* memastikan semua nilai data baru memenuhi constraint namun tidak memeriksa nilai data lama.



Gambar 2.9 Constraint states

Suatu constraint dapat dibuat menjadi aktif atau tidak aktif. Terkadang tindakan tersebut dibutuhkan untuk pengaturan atau pemeliharaan tabnel-tabel yang ada. Bentuk umum perintah untuk membuat constriant menjadi aktif atau tidak aktif sebagai berikut:

```
ALTER TABLE [schema.] table
        MODIFY CONSTRAINT costraint_name <ENABLE|DISABLE>
```

Kata kunci ENABLE bertujuan untuk mengaktifkan sedangkan DISABLE untuk menonaktifkan constraint. Kata kunci DISABLE atau ENABLE dapat juga digunakan pada saat pembuatan suatu constraint, yaitu dengan menambahkan kata kunci tersebut pada bagian akhir perintah pembuatan constraint (dengan CREATE TABLE maupun ALTER TABLE). Tanpa menambahkan salah satu kata kunci tersebut, status constraint tersebut secara default adalah aktif (ENABLED).

2.4 PHP *(Hypertext Preprocessor)*

PHP merupakan bahasa pemrograman aplikasi web dinamis yang bekerja pada sisi server dan memungkinkan interaksi dengan berbagai tipe RDBMS (Relational Database Management System) seperti Oracle, MySQL, IBM, DB2, Microsoft SQL Server, PostgreSQL, InnoDB, SQLite, dan lain sebagainya.

Dengan dukungan sistem terbuka yang sangat baik menjadikan PHP sebagai salah satu bahasa pemrograman web yang paling populer dan mendukung kemampuan cross-platform, di antaranya adalah mampu beroperasi pada lingkungan sistem Unix, Linux, Windows, dan Mac OSX.

2.4.1 Sejarah Singkat PHP

PHP dibuat pertama kali pada musim gugur tahun 1994 oleh Rasmus Lerdoff (rasmus@php.net), awalnya digunakan pada websitenya untuk mencatat siapa saja yang berkunjung dan melihat biodatanya. Versi pertama yang di-release tersedia pada awal tahun 1995, dikenal sebagai tool Personal Home Page, yang terdiri atas engine parser yang sangat sederhana yang hanya mengerti beberapa makro khusus dan sejumlah utilitas yang sering digunakan pada halaman-halaman web, seperti buku tamu, counter pengunjung dan yang lainnya. Parser diprogram ulang pada pertengahan 1995 dan diberi nama PHP/F1 versi 2.0. FI berasal dari paket Rasmus lainnya yang ditulis untuk mengintepretasikan data pada form, yang kemudian dikombinasikan dengan tool Personal Home Page dan ditambahkan dengan dukungan database mSQL (mini SQL).

Tahun 1995 ini dianggap sebagai tahun kelahiran dari PHP/F1 yang kemudian membuat pertumbuhan aplikasi web yang pesat, dan banyak orang yang kemudian berkontribusi mengembangkan PHP/F1. Sulit untuk mendapatkan statistik yang tepat untuk memperkirkan penggunaan PHP/F1, tetapi diperkirakan pada akhir 1996 telah digunakan oleh sedikitnya 15000 web site diseluruh dunia. Dan pertengahan 1997 mencapai 50000 situs.

Pada pertengahan 1997 ini juga terjadi perubahan pengembangan PHP. Pengembangan dilakukan oleh tim yan

terorganisasi bukan oleh Rasmus sendiri saja lagi. Parser dikembangkan oleh Zeev Suraski dan Andi Gutmans yang kemudian menjadi dasar untuk versi 3, dan banyak utilitas tambahan yang diprogram untuk menambah kemampuan dari versi 2. Versi PHP 4 menggunakan engine script Zend untuk lebih meningkatkan kinerja (performance) dan mempunyai dukungan yang banyak berupa ekstensi dan fungsi dari berbagai library pihak ketiga (third party), dan berjalan seolah modul asli (native) dari berbagai server web yang populer.

Sejak Januari 2001 PHP3 dan PHP4 disertakan pada sejumlah produk server web komersial seperti server web StrongHold RedHat. Perkiraan konservatif yang didapat dari angka yang diberikan oleh Netcraft (http://www.netcraft.com) yang diekstrapolasi, pengguna PHP sekitar 5.100.000 sedikit lebih banyak dari server web yang menggunakan Microsoft IIS (5.03 juta) di Internet. PHP versi 4.2.0 di *release* pada tanggal 22 April 2002. Perbaikan pada bug (kesalahan-kesalahan) terutama pada upload file melalui browser telah dibetulkan, dan banyak penambahan fungsi yang lebih memudahkan lagi pengembang aplikasi untuk membuat program yang lebih baik. Sampai dengan versi 4.3.7 tercatat ada 125 kelompok fungsi yang dimiliki oleh PHP. Saat ini pengembangan php telah memasuki versi 5.

2.4.2 Prinsip Kerja PHP

Pengaksesan berkas PHP dimulai dengan perintaan (*request*) pengguna terhadap berkas PHP melalui protokol HTTP. Permintaan dikirimkan ke Web Server. Setelah menerima permintaan dari pengguna, web server akan mengambil berkas PHP dan akan diproses (interpret) oleh PHP interpretor. PHP interpretor akan melakukan pemrosesan kode-kode PHP, dan kemudian membentuk blok-blok kode HTML. Kode HTML yang terbentuk selanjutnya dikirim kembali ke web browser sehingga dapat diproses. Hasil proses berupa halaman web yang diminta oleh pengguna.

2.4.3 PHP dan Database

Salah satu keunggulan dari PHP sebagai bahasa pemrograman script adalah banyak fasilitas (librari fungsi) yang memungkinkan untuk mengakses database. Kecepatan akses dengan menggunakan engine/driver yang khusus untuk setiap database merupakan satu kelebihan dan kekurangan. Kelebihannya adalah dari sisi kecepatan tidak dapat disangkal, karena dibuat khusus fungsinya. Kekurangannya adalah karena ketidakseragaman nama fungsi (perintah), sehingga sulit bagi aplikasi yang dihasilkan yang dikatakan independen terhadap database yang digunakan. Prosedur standar untuk melakukan operasi akses database adalah:

- Open database.
- Eksekusi SQL.
- Proses record set yang dihasilkan.
- Close database.

Proses inti dari manipulasi database adalah pada pembangunan perintah SQL yang digunakan untuk melakukan query, insert, update, atau pun delete data untuk database.

2.4.4 Kemampuan PHP

Kemampuan (Feature) PHP yang paling diandalkan dan signifikan adalah dukungan kepada banyak database. Membuat halaman web dengan menggunakan data dari database dengan sangat mudah dapat dilakukan. Berikut adalah daftar database yang didukung oleh PHP:

- Adabas D
- dBase
- Empress
- FilePro (read only)
- FrontBase
- Hyperwave
- IBM DB2
- Oracle
- MySQL
- ODBC
- dll

PHP juga mendukung untuk berkomunikasi dengan layanan lain menggunakan protokol IMAP, SNMP, NNTP, POP3, HTTP, dan lainnya yang tidak terhitung. Pemrograman juga dapat membuka soket jaringan secara mentah dan berinteraksi dengan menggunakan protokol lainnya.

2.4.5 Script PHP

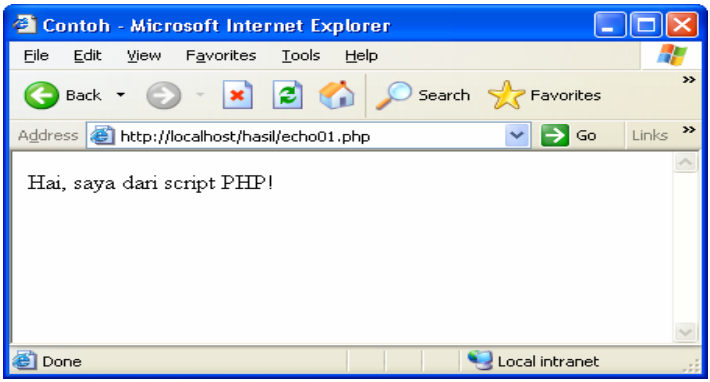
Setiap program PHP disebut dengan script. Script berupa file teks, yang dapat dibuat dengan menggunakan program editor file teks biasa seperti notepad, edit, vi (dalam lingkungan Unix/Linux), atau pun lainnya.

Script PHP diawali dengan tag <? Kemudian diakhiri dengan tag ?>. Setiap baris perintah/statement harus diakhiri dengan menggunakan tanda titik koma (;). Umumnya setiap statement dituliskan dalam satu baris. Script PHP merupakan script yang digunakan untuk menghasilkan halaman-halam web. Cara penulisan dibedakan menjadi embedded dan non embedded script.

```
<html>
<head>
<title>Contoh</title>
<head>
<body>
<?php
echo "Hai, saya dari script PHP!";
?>
</body>
</html>
```

Gambar 2.10 Contoh embedded script

Gambar diatas berisi contoh script PHP sederhana yang disebut dengan embedded script. Script PHP digunakan apabila isi dari suatu dokumen HTML diinginkan dari hasil eksekusi suatu script PHP.



Gambar 2.11 Hasil tampilan embedded script pada browser

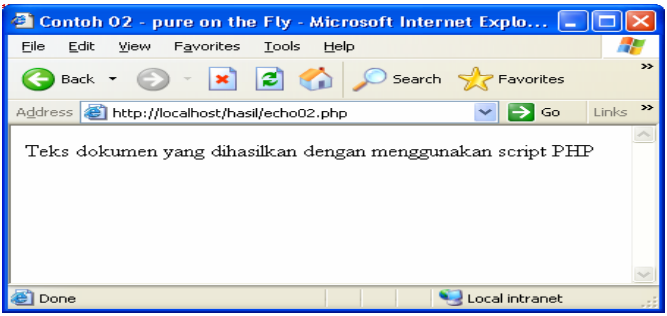
Source dokumen HTML yang ditampilkan berupa dokumen HTML murni, tidak ada lagi tag dan script PHP (diawali dengan tanda <? Dan diakhiri dengan ?>) yang berisi script PHP, karena semuanya telah menjadi tag HTML, karena pada saat dieksekusi maka bukan scriptnya yang dikirim tetapi hasil eksekusi dari script tersebut yang dikirim. Hasilnya berupa dokumen HTML, mekanismenya inilah disebut dengan *'on the fly HTML creation'*. Proses *'on the fly HTML creation'* akan terlihat lebih jelas lagi dengan menggunakan script non embedded script.

Script PHP dengan cara Non Embedded Script adalah murni dengan PHP, tag HTML yang dihasilkan untuk membuat dokumen merupakan bagian dari script PHP. Gambar berikut dibawah ini adalah contoh Non Embedded Script.

```
<?php
echo "<html>";
echo "<head>";
echo "  <title>";
echo "     Contoh 02 – pure on the Fly";
echo "  </title>";
echo "</head>";
echo "<body>";
echo "<p>Teks dokumen yang dihasilkan dengan menggunakan
script PHP</p>";
echo "</body>";
echo "</html>";
?>
```

Gambar 2.12 Contoh non embedded script

Berikut adalah hasil tampilan dari script diatas diakses dari browser web.



Gambar 2.13 Hasil tampilan non embedded script.

Script PHP menerapkan aturan case sensitive yakni adanya perbedaan penulisan huruf besar dengan huruf kecil, case sensitive dikenakan terutama untuk nama-nama variable.

2.4.6 Variable

Variable merupakan tempat penyimpanan data, di dalam PHP diawali dengan karakter $ diikuti dengan huruf sebagai karakter pertama setelah $, kemudian kombinasi karakter dan angka. Tidak boleh ada spasi dan tanda baca dalam penamaannya, kecuali karakter _ (garis bawah, *underscore*). Gambar dibawah ini merupakan contoh penamaan variable yang benar.

```
$namauser
$password
$kota2
$tmpat_lahir
```

Gambar 2.14 Contoh nama variabel yang benar

Gambar berikut dibawah ini adalah contoh penamaan variable yang salah.

```
$nama user
$pass/word
$kota 2
```

Gambar 2.15 Contoh nama variabel yang salah

2.4.7 PHP untuk ORACLE

Oracle, sebagai sistem manajemen database relasional, saat ini dapat dikatakan paling populer diantara sistem database yang ada. Apalagi saat ini, Oracle memiliki strategi membebaskan software Oracle untuk dicoba oleh orang-orang yang berminat, dengan mempersilakan untuk men-*download* sendiri Oracle; tidak ada batas waktu pemakaian software tersebut untuk uji coba.

Pada gambar dibawah ditunjukkan mengenai hubungan PHP dengan Oracle client Libraries.

Gambar 2.16 PHP di-link dengan Oracle Client Libraries

2.4.8 Fungsi-fungsi PHP Oracle

Berikut ini beberapa fungsi PHP oracle yang sering digunakan.

- **Koneksi ke database – ocilogon() atau oci_connect()**

ocilogon() merupakan fungsi digunakan untuk menentukan koneksi kepada server Oracle. Jika koneksi berhasil

maka akan dihasilkan suatu variable handle, yang berisi nomor id (identification) yang menunjukkan bahwa koneksi berhasil atau tidak. Jika koneksi berhasil maka nilai dari variabel handle lebih besar dari 0. dengan menggunakan variabel handle inilah, maka proses manipulasi database Oracle melalui PHP dilakukan. Fungsi ocilogon() membutuhkan parameter berupa nama pemakai, password, dan nama database yang akan diakses.

Sintak:

```
$varHandle=ocilogon("namapemakai","password","namaDB
");
```

namaDB dapat tidak dimasukkan, jika kita yakin bahwa bahwa konfigurasi pada software client Oracle yang kita milki hanya untuk satu koneksi database.

- **Parsing – ociparse() atau oci_parse()**

Digunakan untuk melakukan parsing setiap perintah SQL yang akan dieksekusi, proses pemeriksaan apakah perintah SQL yang akan dieksekusi sudah benar atau belum. Perintah ini digunakan setelah koneksi kepada server database Oracle berhasil.

Sintak:

```
$ociparsehandle=ociparse(handleKoneksi,"perintahSQL"
);
```

Hasil dari perintah ini adalah benar atau salah. Benar, jika perintah SQL yang akan dieksekusi sudah benar, dan salah apabila perintah SQL yang dieksekusi masih salah perintahnya. Berikut ini merupakan perintah SQL yang dapat langsung diberikan sebagai parameter pada fungsi ociparse.

```
$conn=ocilogon("system","manager","BKTAMU");
$sqlhandle=ociparse($conn,"select * from bukutamu");
```

Atau dapat juga ditampung dahulu pada sebuah variable, kemudian dimasukkan sebagai parameter pada fungsi ociparse, seperti ditunjukkan seperti gambar berikut dibawah.

```
$conn=ocilogon("system","manager","BKTAMU");
$sqlstr="select * from bukutamu";
$sqlhandle=ociparse($conn, $sqlstr);
```

- **Eksekusi – ociexecute() atau oci_execute()**

Perintah ociexecute digunakan untuk mengeksekusi perintah SQL yang telah benar di-parse.

Sintak:

```
ociexecute("ociparsehandle"[,ocioption]);
```

ocioption adalah pilihan mode eksekusi. Berikut adalah beberapa mode yang dapat digunakan:

- OCIDEFAULT, mode eksekusi perintah, perintah tidak secara otomatis di commit
- OCI_COMMIT_ON_SUCCESS, mode eksekusi perintah, jika berhasil maka proses commit langsung dilakukan, merupakan **pilihan DEFAULT** jika kita tidak menggunakannya.
- OCI_ASSOC, mode eksekusi perintah, agar hasil eksekusi dapat digunakan oleh perintah oci_fetch_all() dan ocifetcharray()

```
$conn=ocilogon("system","manager","BKTAMU");
$sqlstr="select * from bukutamu";
$sqlhandle=ociparse($conn, $sqlstr);
ociexecute($sqlhandle);
```

Script di atas adalah contoh script untuk penggunaan ociexecute. Koneksi menggunakan user system, dengan password

manager, serta menggunakan database BKTAMU. Variable $sqlstr digunakan untuk menampung perintah SQL, kemudian variable tersebut dimasukkan sebagai parameter untuk fungsi ociparse. Kemudian jalankan perintah ociexecute Perintah ini berfungsi untuk mengeksekusi perintah SQL yang telah benar di-parse.

- **Memasukkan hasil eksekusi ke dalam array-ocifetchstatement() atau oci_fetch_all()**

Hasil eksekusi dari suatu query perintah SQL dapat dimasukkan kedalam array assosiatif, agar kita mudah untuk menampilkannya. Hasil perintah ini adalah array hasil query dan jumlah elemen (baris) array. Perintah ini diberikan setelah berhasil mengeksekusi perintah ociexecute().

Sintak:

```
$jmlbaris=ocifetchstatement($sqlhandle,$arrayhasil);
```

- **Membaca hasil query-ocifetch() atau oci_fetch()**

Perintah ocifetch() digunakan untuk langsung melakukan pemrosesan hasil query baris per baris.

Sintak:

```
ocifetch($sqlhandle);
```

Perintah ocifetch() digunakan tanpa perlu mengetahui berapa jumlah baris terlebih dahulu, karena proses yang dilakukan akan diberikan kepada seluruh hasil query dari yang pertama sampai dengan terakhir.

BAB III
PERANCANGAN DAN PEMBUATAN SISTEM

Pada bab ini akan di jelaskan mengenai perencanaan sumber data, desain data warehouse, spesifikasi sistem dan pembuatan aplikasi untuk menampilkan analisa pada data warehouse.

3.1. Perencanaan Sumber Data

Pertama kali yang harus disiapkan dalam membangun data warehouse adalah mengumpulkan data dan merencanakan analisa data dari sumber data yang ada, pada tugas akhir ini sumber data dan analisa yang akan di bangun dapat di lihat pada tabel 3.1 berikut :

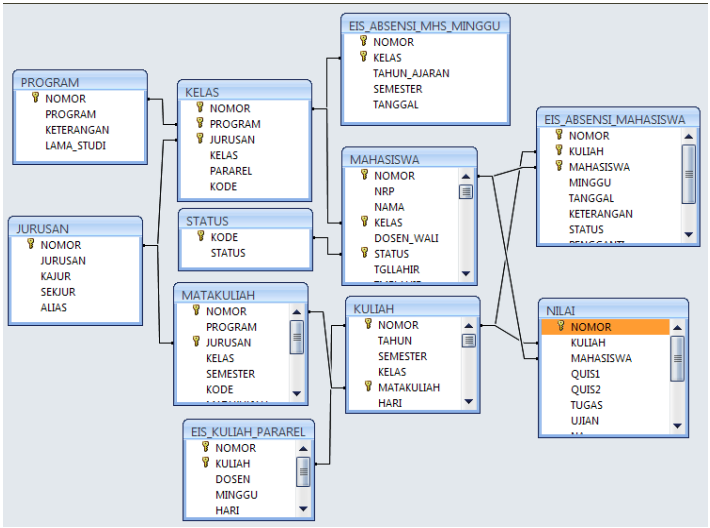
Tabel 3.1 Perencanaan Analisa dan Sumber data

No	Analisa	Database	Tabel
1	Analisa Indeks Prestasi ➢ Rata-Rata IP Persemester Pertahun ➢ Rangking IP tiap semester	-EIS	➢ Mahasiswa ➢ Kuliah ➢ Jurusan ➢ Nilai
2	Analisa Kedisiplinan ➢ Rata-Rata Kedisiplinan Mahasiswa Persemester	-EIS	➢ Eis_absensi_mhs ➢ Mahasiswa ➢ Kuliah ➢ Matakuliah ➢ Jurusan

	Pertahun ➢ Analisa Pelanggaran tiap semester		
3	Analisa Minat Mhs terhadap Matakuliah tertentu	- EIS	➢ Eis_absensi_mhs ➢ Kuliah ➢ Matakuliah ➢ Jurusan
4	Analisa Kurikulum	- EIS	➢ Nilai ➢ Kuliah ➢ Matakuliah ➢ Jurusan
5	Recommended Beasiswa, berdasarkan IP dan Kedisiplinann	- EIS - WAREHOUSE	➢ Eis_absensi_mhs ➢ Mahasiswa ➢ Kuliah ➢ TempDSS1 ➢ TempDSS2 ➢ F_Prestasi ➢ F_DSS
6	Recommended Beasiswa, berdasarkan IP, Kedisiplinann dan Ekonomi	- EIS - WAREHOUSE	➢ Eis_absensi_mhs ➢ Mahasiswa ➢ Kuliah ➢ TempDSS1 ➢ TempDSS2 ➢ F_Prestasi

			➢ F_DSS

Adapun ER diagram pada data akademik adalah sebagai berikut :



Gambar 3.1 ER diagram data akademik PENS-ITS.

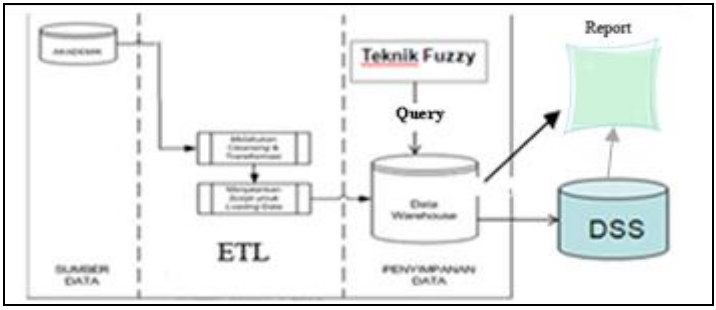
3.2. Desain Data Warehouse

Pada fase ini aktifitas yang dilakukan adalah merancang arsitektur data warehouse, membuat model data dimensional yang berupa star Schema dan menganalisis metadata yang digunakan dalam data warehouse.

3.2.1 Perancangan Arsitektur data warehouse

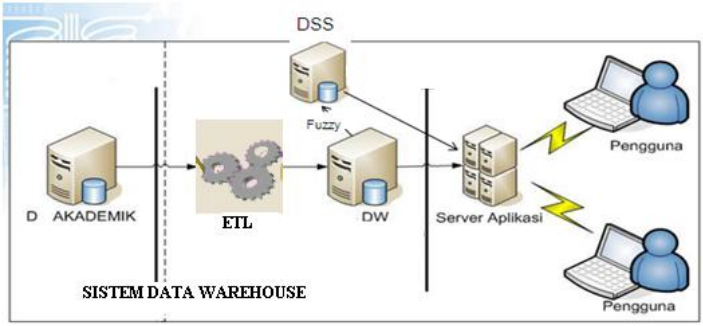
Sumber data operasional yang di gunakan adalah database akademik yang berisi semua data akademik PENS-ITS. Dari sumber data tersebut secara periodik dilakukan pemilihan data setelah itu dilakukan proses pembersihan dan transformasi, hasil dari proses pembersihan dan transformasi inilah yang kemudian di simpan ke dalam data warehouse. Gambar 3.2

memperlihatkan rancangan arsitektur logical dari data warehouse, yang sekaligus menggambarkan proses pengisian data ke dalam data warehouse.



Gambar 3.2. Arsitektur *logical* data warehouse

Rancangan arsitektur fisik dari data warehouse dapat di lihat pada gambar 3.3.

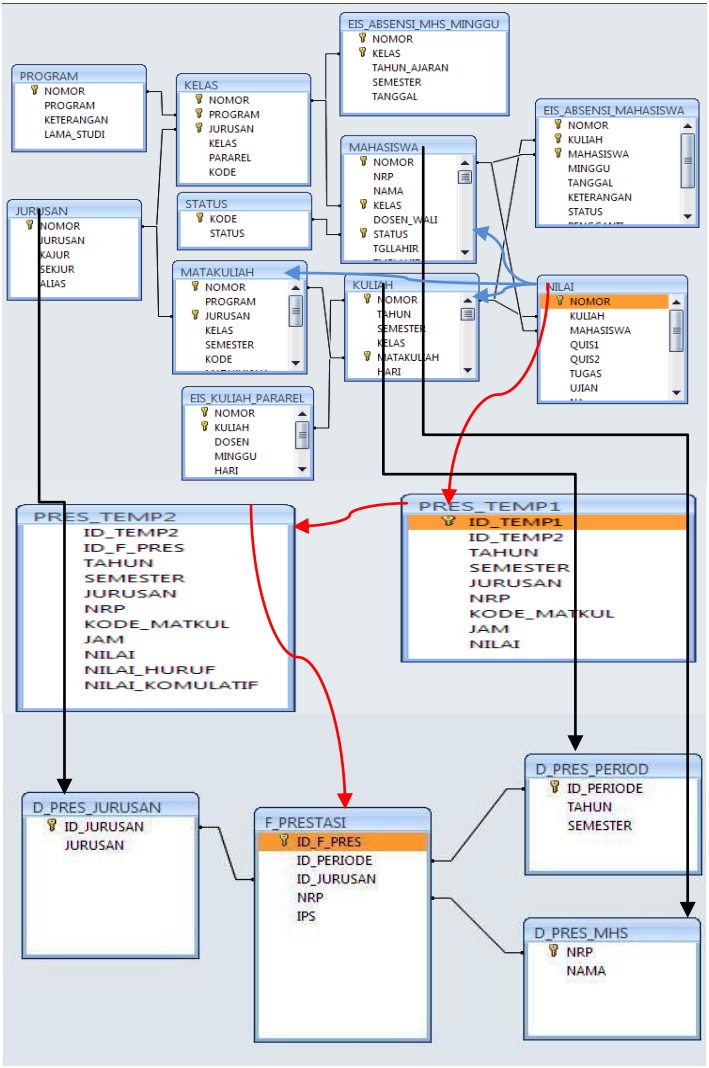


Gambar 3.3 Arsitektur fisik data warehouse

3.2.2 Model Data Dimensional

Schema yang kita gunakan untuk pemodelan data dimensional adalah star schema dimana terdapat satu table fakta dan beberapa table dimensi,penggunaan star schema memungkinkan proses query yang lebih ringan dan memudahkan penjelajahan terhadap data dimensinya[2].

Desain Skema Star untuk Analisa Prestasi



Gambar 3.4 Aliran data, dari data sumber ke *star schema* f_prestasi.

Keterangan :

: Proses ETL dari tabel sumber ke tabel fakta
: Proses ETL dari tabel sumber ke tabel dimensi
: Relasi tabel sumber yang di butuhkan oleh tabel fakta

Aliran data pada skema star fakta prestasi (Gambar 3.4), dapat di jelaskan sebagai berikut :

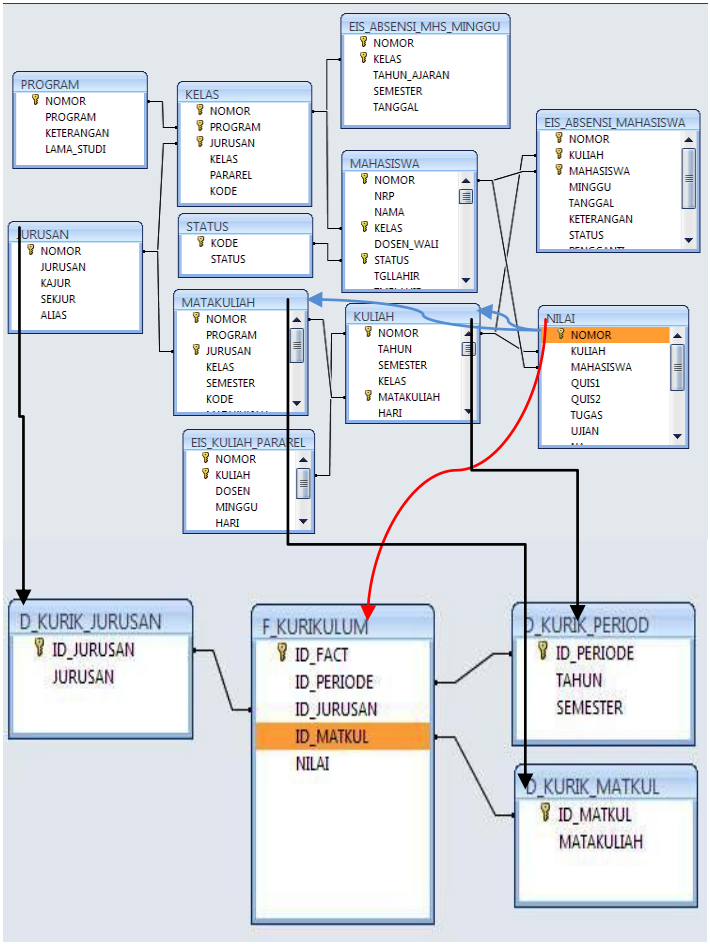
- Tabel d_pres_jurusan,d_pres_period dan d_pres_mhs merupakan tabel dimensi prestasi, sumber data dari ke tiga tabel tersebut secara berurutan adalah tabel jurusan, tabel kuliah dan tabel mahasiswa pada data akademik PENS-ITS.

- Tabel f_prestasi berisi data yang berhubungan dengan nilai indeks prestasi mahasiswa tiap semester, sumber data tabel f_prestasi berasal dari tabel Nilai yang sebelumnya data dari tabel Nilai sudah diproses oleh tabel pres_temp1dan tabel pres_temp2. Tabel pres_temp1 di gunakan untuk menampung nilai sesudah di lakukan pengecekan ada tidaknya mahasiswa yang ujian ulang (HER), tabel pres_temp2 di gunakan untuk menghitung nilai dari matakuliah group kemudian menentukan nilai komulatif (jam_matakuliah x nilai_huruf) untuk tiap-tiap matakuliah.

Desain skema star fakta kurikulum (Gambar 3.5), dapat di jelaskan sebagai berikut :

- Tabel d_kurik_jurusan,d_kurik_period dan d_kurik_matkul merupakan tabel dimensi kurikulum, tabel sumber dari ke tiga tabel tersebut secara berurutan adalah tabel jurusan,tabel kuliah dan tabel matakuliah pada data akademik.
- Tabel f_kurikulum berisi data nilai mahasiswa terhadap tiap-tiap matakuliah yang diambil mahasiswa,atribut nilai pada tabel f_kurikulum adalah nilai angka yang di ambil dari tabel Nilai pada data akademik PENS-ITS.

Desain Skema Star untuk Analisa Kurikulum

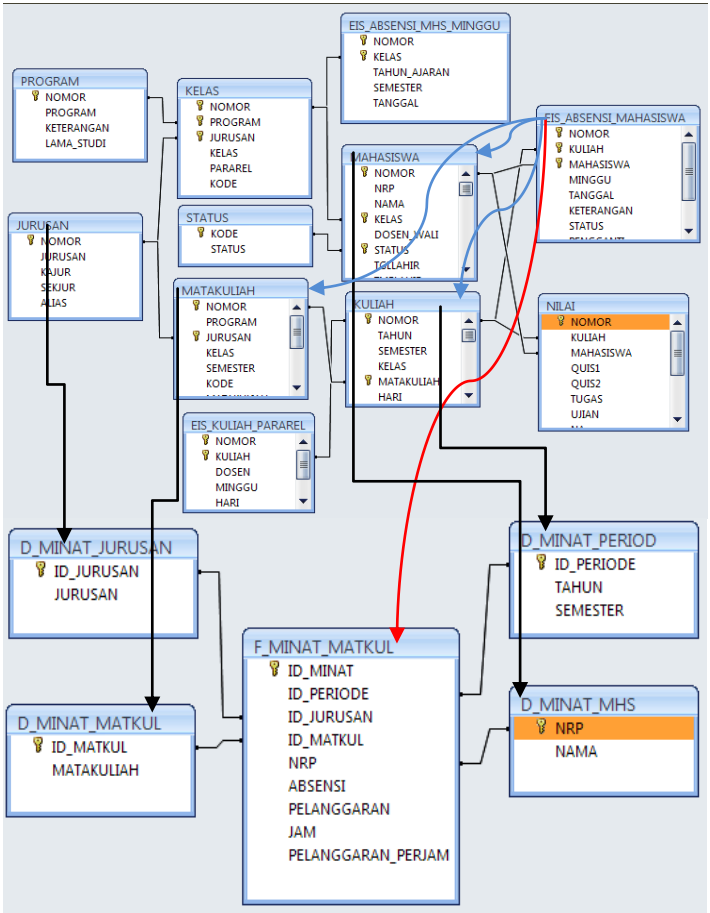


Gambar 3.5 Aliran data, dari data sumber ke *star schema* f_kurikulum

Keterangan :

: Proses ETL dari tabel sumber ke tabel fakta
: Proses ETL dari tabel sumber ke tabel dimensi
: Relasi tabel sumber yang di butuhkan oleh tabel fakta

Desain Skema Star untuk Analisa Disiplin dan Matakuliah



Gambar 3.6 Aliran data, dari sumber ke *star schema* f_minat_matkul.

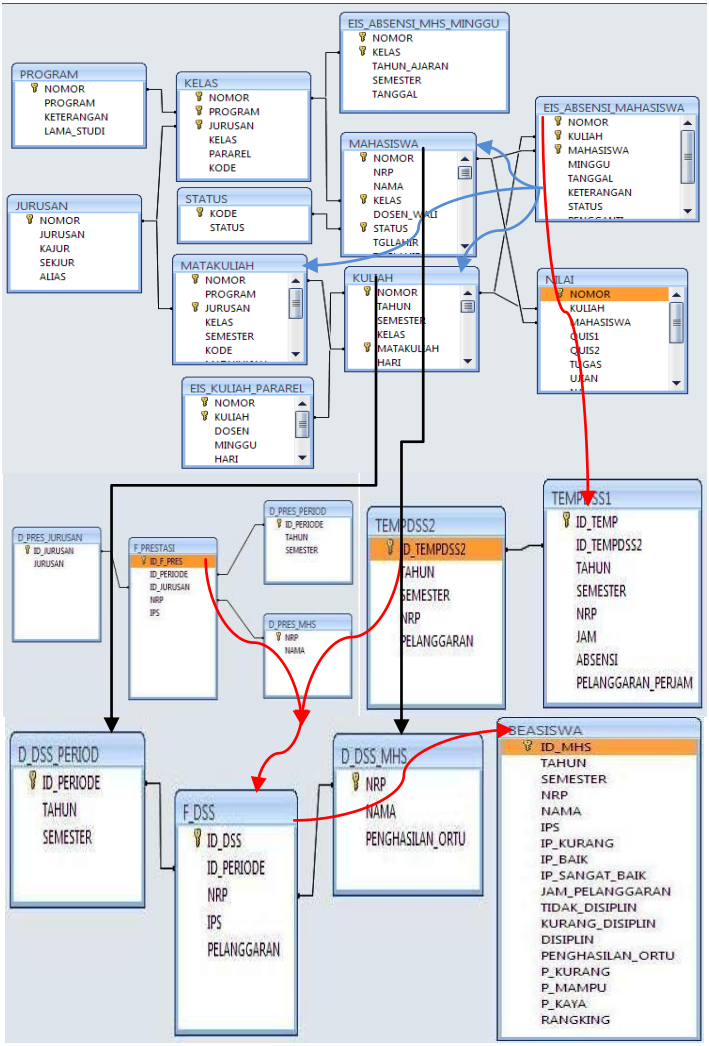
Keterangan :

: Proses ETL dari tabel sumber ke tabel fakta
: Proses ETL dari tabel sumber ke tabel dimensi
: Relasi tabel sumber yang di butuhkan oleh tabel fakta

Aliran data pada skema star fakta minat_matkul (Gambar 3.6), dapat di jelaskan sebagai berikut :

- Tabel d_minat_period,d_minat_matkul dan d_minat_jurusan merupakan tabel dimensi yang di butuhkan untuk analisa minat mahasiswa terhadap matakuliah tertentu berdasarkan absensi mahasiswa terhadap matakuliah tersebut.
- Tabel d_minat_period dan d_minat_jurusan merupakan tabel dimensi yang di butuhkan untuk analisa rata-rata disiplin mahasiswa pada tiap semester dan tiap tahun.
- Tabel d_minat_period,d_minat_mhs dan d_minat_juruan merupakan tabel dimensi yang di butuhkan untuk analisa rangking pelanggaran mahasiswa berdasarkan absensi mahasiswa.
- Tabel d_minat_period_d_minat_matkul,d_minat_jurusan dan d_minat_mh, tabel sumber dari ke empat tabel tersebut secara berurutan adalah tabel kuliah, tabel matakuliah tabel jurusan dan dan tabel mahasiswa pada data akademik PENS-ITS.
- Tabel f_minat_matkul merupakan tabel fakta yang berisi data absensi mahasiswa tiap hari selama masa perkuliahan aktif, tabel ini bersumber pada tabel eis_absensi_mahasiswa pada data akademik PENS-ITS.

Desain Skema Star untuk Merekomendasikan Beasiswa



Gambar 3.7 Aliran data, dari sumber ke *star schema* f_dss.

Keterangan :

: Proses ETL dari tabel sumber ke tabel fakta
: Proses ETL dari tabel sumber ke tabel dimensi
: Relasi tabel sumber yang di butuhkan oleh tabel fakta

Aliran data skema star fakta dss (Gambar 3.7), dapat di jelaskan sebagai berikut:

- Karena tabel sumber tidak menyediakan data *banyaknya jam pelanggaran mahasiswa tiap tiap semester*, maka sebelum mengalirkan data tabel eis_absensi_mahasiswa ke tabel dimensional, data perlu di transformasi terlebih dahulu pada saat proses ETL, yang kemudian hasil dari proses tersebut kita tampung ke tabel tempdss1.
- Karena pada tabel tempdss1 informasi pelanggaran masih bersifat pelanggaran mahasiswa pada tiap jam matakuliah dan data yang di butuhkan adalah *total jam pelanggaran mahasiswa tiap semester* maka pada proses ETL perlu dilakukan agregasi data pada temp1 yang hasilnya adalah total pelanggaran (total absen mahasiswa) tiap semester, kemudian hasilnya di tampung pada tabel tempdss2.
- Untuk analisa recommended beasiswa, variabel yang di butuhkan adalah jumlah total jam pelanggaran mahasiswa tiap semester, indeks prestasi mahasiswa tiap semester dan penghasilan orang tua, untuk variabel penghasilan orang tua mahasiswa, kami mengambil data yang telah kami siapkan pada tabel dimensi d_dss_mhs di mana disitu ada atribut penghasilan_ortu yang berisi penghasilan ayah + penghasilan ibu, dan untuk variabel indeks prestasi kami mengambil data dari tabel f_prestasi yang telah kami bangun sebelumya.
- Untuk mengalirkan variabel data indeks prestasi dan variabel pelanggaran mahasiswa pada saat proses ETL perlu di adakan pengecekan apakah masing-masing ID variabel ini sama,jika sama maka data dapat di inputkan ke tabel f_dss dan jika tidak sama maka data kita singkirkan, hal ini bertujuan untuk menghindari adanya data null yang masuk ke f_dss.

- Setelah data pada tabel f_dss telah tersedia selanjutnya adalah melakukan perhitungan terhadap atribut IPS, Pelanggaran dan penghasilan_ortu dimana ketiga atribut tersebut merupakan variabel dari teknik fuzzy database dalam menentukan mahasiswa yang paling berhak menerima beasiswa.
- Setelah teknik fuzzy selesai dilakukan pada saat proses ETL, maka hasilnya di tampung pada tabel beasiswa, tabel inilah yang nanti akan menjadi sumber analisa yaitu merekomendasikan beasiswa bagi mahasiswa.

Dalam membangun data warehouse ini, hanya data-data yang di perlukan untuk analisa saja yang di alirkan ke tabel tujuan(tabel dimensional.

3.3. Spesifikasi Sistem

Berikut ini merupakan spesifikasi sistem yang terdiri dari segi hardware serta software yang penulis gunakan dalam membangun aplikasi ini.

3.3.1 Spesifikasi Hardware

Untuk spesifikasi hardware yang penulis gunakan dalam membangun aplikasi ini adalah :

Tabel 3.2 Spesifikasi Hardware

Hardware	***Spesifikasi***
Physical memory (RAM)	2 GB
Ruang Disk	80 GB
Video Adapater	234 MB
Processor	Core 2 Duo 2.0 GHz

3.3.2 Spesifikasi Software

Berikut spesifikasi perangkat lunak yang penulis gunakan:

1. Operating Sistem
   Operating Sistem yang digunakan adalah XP Professional.
2. Protokol
   Oracle database mendukung penggunaan protokol jaringan yang umum dipakai oleh standar industri yang berlaku, pada proyek akhir ini digunakan protokol TCP/IP.
3. Browser
   Mozilla Firefox 3.0.1
4. Oracle 10g Enterprise.
   Versi Oracle yang penulis gunakan adalah Oracle 10g Enterprise versi 10.2.0.1
5. Zend for oracle
   Zend Core for Oracle adalah Engine PHP yang dibuat untuk koneksi antara PHP dan Oracle seperti versi Oracle 8i, 9i dan 10g secara default oci juga sudah enable. Zend Core for Oracle versi terbaru dibundle dengan Apache 2 dan PHP 5 yang didesain untuk mencapai kinerja yang maksimal, dioptimisasi untuk Oracle versi 10g, aman, cepat, dan berita gembiranya adalah aplikasi ini free. Catatan pengalaman masalah yg sering timbul kalau koneksi antara PHP-ORACLE pakai selain zend for oracle[8] adalah :
   - listener berkali-kali mati.
   - catatan log ekstra besar hingga menghabiskan space karena request invalid.
   - datafile system dan sysaux membengkak.
6. PL/SQL Developer
   PL/ SQL developer yang di gunakan adalah versi 7.1.5.1398 dimana software ini untuk mengenerate coding ETL data warehouse.

Dalam tugas akhir ini di asumsikan semua software yang di butuhkan telah terinstal dengan baik.

3.4. Pembuatan Sistem dan Aplikasi

3.4.1 Pembuatan data warehouse

Berikut ini adalah daftar tabel – tabel data warehouse yang harus di buat dalam tugas akhir ini:

Tabel 3.3 Daftar table analisa indeks Prestasi

Tabel	Keterangan
D_PRES_JURUSAN	Tabel dimensi jurusan
D_PRES_PERIOD	Tabel dimensi waktu
D_PRES_MHS	Tabel dimensi mahasiswa
PRES_TEMP1	Tabel yang menampung nilai sesudah di periksa ada tidaknya ujian perbaikan (HER)
PRES_TEMP2	Tabel untuk menghitung nilai komulatif IP (nilai_huruf x jam_matakuliah)
F_PRESTASI	Tabel Fakta Prestasi (IP)

Tabel 3.4 Daftar tabel analisa kurikulum

Tabel	Keterangan
D_KURIK_JURUSAN	Tabel dimensi jurusan
D_KURIK_PERIOD	Tabel dimensi waktu
D_KURIK_MATKUL	Tabel dimensi matakuliah
F_KURIKULUM	Tabel Fakta Kurikulum

Tabel 3.5 Daftar tabel analisa disiplin dan minat mahasiswa terhadap matakuliah tertentu

Tabel	Keterangan
D_MINAT_JURUSAN	Tabel dimensi jurusan
D_MINAT_MATKUL	Tabel dimesi matakuliah
D_MINAT_PERIOD	Tabel dimensi waktu
D_MINAT_MHS	Tabel dimesi mahasiswa
F_MINAT_MATKUL	Tabel fakta matakuliah

Tabel 3.6 Daftar tabel analisa untuk recommeded beasiswa

Tabel	Keterangan
TEMPDSS1	Tabel ini gunakan untuk menampung hasil banyaknya jam pelanggaran mahasiswa tiap tiap semester
TEMPDSS2	Tabel ini di gunakan untuk menampung data total jam pelanggaran mahasiswa tiap semester
D_DSS_PERIOD	Tabel dimensi waktu
D_DSS_MHS	Tabel dimensi mahasiswa
F_DSS	Tabel fakta dss di gunakan untuk menampung data variabel IP, Penghasilan dan Absensi Mhs
BEASISWA	Tabel untuk menampung hasil perhitungan teknik fuzzy.

Setelah tabel tabel diatas selesai di buat langkah selan jutnya adalah membuat ekstraksi, transformasi dan meload data dari tabel sumber ke tabel target (data warehouse).

3.4.2 Proses ekstraksi, transformasi dan Loading data

Sebelum kita meload data dari tabel sumber ke tabel target tentu kita akan memilih(mengektrak) data yang di butuhkan oleh sistem, melalui proses transformasi memungkinkan data akan di ubah bentuknya misal mengubah nilai angka menjadi nilai huruf, setelah proses ektraksi dan transformasi selesai di lakukan, selanjutnya adalah meload data dari tabel sumber ke tabel tujuan.

3.4.2.1 Proses ETL Pada Analisa Indeks Prestasi

Pada proses ETL analisa indeks prestasi , proses dimulai dengan load data ke tabel dimensi jurusan, dimensi mahasiswa dan dimensi periode, kemudian dilanjutkan load data ke tabel pres_temp1 setelah melalui proses transformasi yaitu memberikan nilai angka 56 (nilai huruf C) kepada mahasiswa

yang terkena HER (ujian ulang) melalui fungsi NILAI_FINAL. Kemudian di lanjutkan proses load data ke tabel prest_temp2 yang juga terdapat proses transformasi yaitu merubah nilai angka menjadi nilai huruf melalui fungsi NA_KE_NH. Setelah proses ini, proses yang paling akhir adalah load data ke tabel fakta prestasi. Berikut Listing program proses ETL analisa indeks prestasi:

```
declare
begin
 --insert update dimensi prestasi_jurusan
 MERGE INTO warehouse.d pres_jurusan j
 USING (SELECT nomor,jurusan
          FROM wahib.jurusan
        )wj
 ON (j.id_jurusan=wj.nomor)

 WHEN MATCHED THEN
   UPDATE
      SET j.jurusan=wj.jurusan
      DELETE where wj.jurusan is null
 WHEN NOT MATCHED THEN
      INSERT (j.id jurusan,j.jurusan)
      VALUES (wj.nomor,wj.jurusan);


 --insert update dimensi prestasi_mhs

 MERGE INTO warehouse.d_pres_mhs m
 USING (SELECT nrp,nama
          FROM wahib.mahasiswa
        )wm
 ON (m.nrp=wm.nrp)

 WHEN MATCHED THEN
   UPDATE
      SET m.nama=wm.nama
 WHEN NOT MATCHED THEN
      INSERT (m.nrp,m.nama)
      VALUES (wm.nrp,wm.nama);


  --insert update dimensi prestasi_periode
 MERGE INTO warehouse.d pres period p
 USING (SELECT DISTINCT tahun,semester
          FROM wahib.kuliah)wp
```

```
  ON (p.id_periode=wp.tahun||wp.semester)

  WHEN MATCHED THEN
    UPDATE
       SET p.tahun=wp.tahun,
      p.semester=wp.semester

  WHEN NOT MATCHED THEN
       INSERT (p.id_periode,p.tahun,p.semester)
       VALUES
(wp.tahun||wp.semester,wp.tahun,wp.semester);

  --insert update Prestasi_temporary1

  MERGE INTO warehouse.pres temp1 t
  USING (SELECT n.nomor as nomor,
                k.tahun as tahun,
                k.semester as semester,
                mt.jurusan as jurusan,
                m.nrp as nrp,

                mt.kode as kode,
                mt.jam as jam,
                n.na as na,
                n.her as her
         FROM wahib.kuliah k,
              wahib.matakuliah mt,
              wahib.nilai n,
              wahib.mahasiswa m
         WHERE n.kuliah=k.nomor
           and k.matakuliah=mt.nomor
           and n.mahasiswa=m.nomor
           and n.na is not null
           )wt
  ON (t.id_temp1=wt.nomor)

  WHEN MATCHED THEN
    UPDATE
       SET
t.id temp2=wt.tahun||wt.semester||wt.nrp||wt.kode,
           t.tahun=wt.tahun,
           t.semester=wt.semester,
           t.jurusan=wt.jurusan,
           t.nrp=wt.nrp,
           t.kode matkul=wt.kode,
           t.jam=wt.jam,
           t.nilai=NILAI_FINAL(wt.na,wt.her) --fungsi
menentukan nilai angka jika kena HER
```

```
  WHEN NOT MATCHED THEN
       INSERT (
               t.id_temp1,t.id_temp2,
               t.tahun,t.semester,
               t.jurusan,t.nrp,


t.kode_matkul,t.jam,t.nilai
               )
       VALUES (

wt.nomor,wt.tahun||wt.semester||wt.nrp||wt.kode,
wt.tahun,wt.semester,wt.jurusan,wt.nrp,wt.kode,
wt.jam,NILAI_FINAL(wt.na,wt.her)
               );


 --insert update  Prestasi_temporary2

  MERGE INTO warehouse.pres_temp2 t2
  USING (SELECT id temp2,tahun,
                semester,jurusan,
                nrp,kode matkul,
                avg(jam)as jam,
                avg(nilai)as nilai
           FROM warehouse.pres_temp1
         group by
id temp2,tahun,semester,jurusan,nrp,kode_matkul
         )wt2
  ON (t2.id_temp2=wt2.id_temp2)

  WHEN MATCHED THEN
    UPDATE
       SET
t2.id_f_pres=wt2.tahun||wt2.semester||wt2.nrp,
           t2.tahun=wt2.tahun,
           t2.semester=wt2.semester,
           t2.jurusan=wt2.jurusan,
           t2.nrp=wt2.nrp,
           t2.kode matkul=wt2.kode_matkul,
           t2.jam=wt2.jam,
           t2.nilai=wt2.nilai,
           t2.nilai_huruf=NA_KE_NH(wt2.nilai),--
merubah nilai angka menjadi huruf

t2.nilai_komulatif=NH_KE_NILAI(NA_KE_NH(wt2.nilai))*wt
2.jam
  WHEN NOT MATCHED THEN
```

```
       INSERT (t2.id temp2,
               t2.id_f_pres,
               t2.tahun,
               t2.semester,
               t2.jurusan,


       t2.nrp,
       t2.kode_matkul,
      t2.jam,t2.nilai,
               t2.nilai_huruf,t2.nilai_komulatif
               )
       VALUES (

wt2.id temp2,wt2.tahun||wt2.semester||wt2.nrp,
               wt2.tahun,wt2.semester,wt2.jurusan,
               wt2.nrp,wt2.kode matkul,
               wt2.jam,wt2.nilai,
               NA_KE_NH(wt2.nilai),

NH KE NILAI(NA KE_NH(wt2.nilai))*wt2.jam
               );

            --insert update  fakta Prestasi
  MERGE INTO warehouse.f_prestasi fp
  USING (SELECT id_f_pres,tahun,
                semester,jurusan,nrp,
                sum(nilai komulatif)/sum(jam) as IPS
           FROM warehouse.pres_temp2
         GROUP BY
id_f_pres,tahun,semester,jurusan,nrp
         )wfp
  ON (fp.id_f_pres=wfp.id_f_pres)

  WHEN MATCHED THEN
    UPDATE
       SET fp.id_periode=wfp.tahun||wfp.semester,
           fp.id jurusan=wfp.jurusan,
           fp.nrp=wfp.nrp,
           fp.ips=wfp.IPS
  WHEN NOT MATCHED THEN
       INSERT
(fp.id_f_pres,fp.id_periode,fp.id_jurusan,fp.nrp,fp.ip
s)
       VALUES
(wfp.id f pres,wfp.tahun||wfp.semester,wfp.jurusan,wfp
.nrp,wfp.IPS);
```

```
  commit;

 end;
```

Dari sirkulasi proses ini, yang perlu di perhatikan adalah hubungan antara tabel satu dengan tabel yang lain harus mempunyai id_tabel yang berhubungan(trigger), karena pada saat proses refresh, id_tabel inilah yang akan mempengaruhi perubahan antar tabel.

Adapun fungsi yang di panggil dalam proses ETL analisa indeks prestasi adalah :

Fungsi NA_KE_NH(merubah nilai angka ke nilai huruf)
Berikut listing program fungsi NA_KE_NH :

```
create or replace function NA_KE_NH(NH NUMBER) return
varchar2 is
  Result varchar2(4);
begin
     if NH >80 then
        Result:='A';
     elsif (NH>70 and NH<81) then
        Result:='AB';
     elsif (NH>65 and NH<71) then
        Result:='B';
     elsif (NH>60 and NH<66) then
        Result:='BC';
     elsif (NH>55 and NH<61) then
        Result:='C';
     elsif (NH>40 and NH<56) then
        Result:='D';
     elsif (NH>=0 and NH<41) then
        Result:='E';
     end if;
return(Result);
end NA_KE_NH;
```

Fungsi NH_KE_NILAI(merubah nilai huruf ke nilai angka)
Berikut listing program fungsi NH_KE_NILAI :

```
create or replace function NH_KE_NILAI(NH CHAR) return
number is
  Result number;
```

```
begin

  if NH='A' then
    Result:=4;
  elsif NH='AB' then
    Result:=3.5;
  elsif NH='B' then
    Result:=3;
   elsif NH='BC' then
    Result:=2.5;
  elsif NH='C' then
    Result:=2;
  elsif NH='D' then
    Result:=1;
  elsif NH='E' then
    Result:=0;
  end if;
return(Result);
end NH_KE_NILAI;
```

Fungsi NILAI_FINAL (Set Nilai Ujian Perbaikan (HER))
Berikut listing program fungsi NILAI_FINAL :

```
create or replace function NILAI_FINAL(NA NUMBER,HER
NUMBER) return number is
  Result number;
begin
  if HER is null then
    Result:=NA;
  elsif HER is not null then
    Result:=56;
  end if;
  return(Result);
end NILAI_FINAL;
```

Fungsi NILAI_FINAL bekerja mengenali tabel Nilai, apabila pada tabel Nilai field HER bernilai null (kosong) berarti mahasiswa tersebut tidak mendapatkan ujian ulangan,sehingga nilai yang di ambil fungsi adalah nilai yang berada pada field NA(nilai angka). Jika pada field HER terdapat nilai, dengan nilai angka berapapun maka nilai yang dikembalikan fungsi adalah 56 (nilai minimum dari nilai huruf C).

3.4.2.2 Proses ETL Pada Analisa Kurikulum

Pada proses ETL analisa Kurikulum, proses dimulai dengan load data ke tabel dimensi jurusan,dimensi matakuliah dan dimensi periode, kemudian proses load di lanjutkan pada tabel fakta kurikulum, Pada proses ini, pemindahan data dari tabel sumber ke tabel target proses yang terjadi hanyalah ekstraksi dan load data, sedangkan untuk proses transformasinya sama sekali tidak di perlukan. Berikut listing program proses ETL analisa kurikulum

```
declare

begin

  --insert update dimensi kurikulum jurusan
  MERGE INTO warehouse.d_kurik_jurusan j
  USING (SELECT nomor,jurusan
           FROM wahib.jurusan
         )wj
  ON (j.id_jurusan=wj.nomor)
  WHEN MATCHED THEN
    UPDATE
       SET j.jurusan=wj.jurusan
  WHEN NOT MATCHED THEN
       INSERT (j.id_jurusan,j.jurusan)
       VALUES (wj.nomor,wj.jurusan);


     --insert update dimensi kurikulum matakuliah
  MERGE INTO warehouse.d_kurik_matkul mt
  USING (SELECT nomor,matakuliah
           FROM wahib.matakuliah
         )wmt
  ON (mt.id_matkul=wmt.nomor)

  WHEN MATCHED THEN
    UPDATE
       SET mt.matakuliah=wmt.matakuliah
  WHEN NOT MATCHED THEN
       INSERT (mt.id matkul,mt.matakuliah)
       VALUES (wmt.nomor,wmt.matakuliah);

--insert update dimensi kurikulum periode

  MERGE INTO warehouse.d_kurik_period p
```

```
  USING (SELECT DISTINCT tahun,semester
           FROM wahib.kuliah
         )wp
  ON (p.id_periode=wp.tahun||wp.semester)

  WHEN MATCHED THEN
    UPDATE
       SET p.tahun=wp.tahun,
           p.semester=wp.semester
  WHEN NOT MATCHED THEN


INSERT (p.id_periode,p.tahun,p.semester)
       VALUES
(wp.tahun||wp.semester,wp.tahun,wp.semester);

   --insert update fakta kurikulum
  MERGE INTO warehouse.f kurikulum k
  USING (SELECT n.nomor as nomor,
                k.tahun,k.semester,
                mt.jurusan as id_jur,

          mt.nomor as id matkul,n.na
           FROM nilai n,kuliah k,matakuliah mt
          WHERE n.kuliah=k.nomor            and
k.matakuliah=mt.nomor
            and n.na is not null
          )wk
  ON (k.id_fact=wk.nomor)

  WHEN MATCHED THEN
    UPDATE
       SET k.id periode=wk.tahun||wk.semester,
           k.id jurusan=wk.id jur,
           k.id_matkul=wk.id_matkul,
           k.nilai=wk.na
  WHEN NOT MATCHED THEN
       INSERT (k.id fact,k.id periode,
               k.id_jurusan,k.id_matkul,k.nilai
               )

    VALUES (wk.nomor,wk.tahun||wk.semester,
               wk.id_jur,wk.id_matkul,wk.na
    );
    commit;
end;
```

3.4.2.3 Proses ETL Analisa Kedisiplinan dan Analisa Minat Mahasiswa Terhadap Matakuliah Tertentu.

Pada proses ETL analisa disiplin dan analisa minat mahasiswa terhadap matakuliah tertentu di atas, proses dimulai dengan load data ke tabel dimensi jurusan, dimensi matakuliah, dimensi mahasiswa dan dimensi periode, kemudian proses load di lanjutkan pada tabel fakta minat_matkul, kemudian di lakukan proses transformasi data dari tabel sumber ke tabel fakta minat_matkul melalui pemanggilan fungsi UBAH_NILAI dan fungsi NEGASI_ABSENSI. Berikut Listing program analisa kedisiplinan dan analisa minat mahasiswa terhadap matakuliah tertentu.

```
declare
begin

  --insert update dimensi minat_matkul jurusan
  MERGE INTO warehouse.d_minat_jurusan j
  USING (
        SELECT nomor,jurusan
          FROM wahib.jurusan
          )wj
  ON (j.id_jurusan=wj.nomor)

  WHEN MATCHED THEN
    UPDATE
       SET j.jurusan=wj.jurusan
  WHEN NOT MATCHED THEN
       INSERT (j.id_jurusan,j.jurusan)
       VALUES (wj.nomor,wj.jurusan);


     --insert update dimensi minat_matkul matakuliah
  MERGE INTO warehouse.d_minat_matkul mt
  USING (
        SELECT nomor,matakuliah
        FROM   wahib.matakuliah
         )wmt
  ON (mt.id_matkul=wmt.nomor)

  WHEN MATCHED THEN
    UPDATE
       SET mt.matakuliah=wmt.matakuliah
  WHEN NOT MATCHED THEN
```

```
       INSERT (mt.id matkul,mt.matakuliah)
       VALUES (wmt.nomor,wmt.matakuliah);

--insert update dimensi minat_matkul mahasiswa
 MERGE INTO warehouse.d_minat_mhs m
  USING (

       SELECT nrp,nama
        FROM wahib.mahasiswa
        )wm


 ON (m.nrp=wm.nrp)

  WHEN MATCHED THEN
    UPDATE
       SET m.nama=wm.nama
  WHEN NOT MATCHED THEN
       INSERT (m.nrp,m.nama)
       VALUES (wm.nrp,wm.nama);

     --insert update dimensi minat_matkul periode
MERGE INTO warehouse.d_minat_period p
USING (
        SELECT DISTINCT tahun,semester

 FROM wahib.kuliah
         )wp
  ON (p.id_periode=wp.tahun||wp.semester)

  WHEN MATCHED THEN
    UPDATE
       SET p.tahun=wp.tahun,
           p.semester=wp.semester
  WHEN NOT MATCHED THEN
       INSERT (p.id_periode,p.tahun,p.semester)
       VALUES
(wp.tahun||wp.semester,wp.tahun,wp.semester);


  --fakta minat thd matakuliah & analisa kedisiplinan
    MERGE INTO warehouse.f_minat_matkul fm
  USING (SELECT a.nomor as id,
                k.tahun as tahun,
                k.semester as semester,
                mk.jurusan as jur id,
                mk.nomor as matkul_id,
                m.nrp as nrp,
```

```
                a.status as status,
                mk.jam as jam


               FROM wahib.eis_absensi_mahasiswa a,
                wahib.mahasiswa m,
                 wahib.matakuliah mk,
                 wahib.kuliah k
          WHERE m.nomor=a.mahasiswa
            and k.nomor=a.kuliah
            and mk.nomor=k.matakuliah
            and a.status is not null
            )wfm
  ON (fm.id_minat=wfm.id)
  WHEN MATCHED THEN

    UPDATE
       SET fm.id periode=wfm.tahun||wfm.semester,
           fm.id_jurusan=wfm.jur_id,
           fm.id_matkul=wfm.matkul_id,
           fm.nrp=wfm.nrp,
           fm.absensi=UBAH_NILAI(wfm.status), --fungsi
ubah nilai--> merubah nilai Huruf menjadi nilai
Angka(di gunakan analisa kedisiplinan)
          fm.pelanggaran=NEGASI_ABSENSI(wfm.status),--
fungsi ini membalik nilai fm.absensi yg digunakan unt.
mendapatkan nilai rangking pelanggaran
           fm.jam=wfm.jam,

fm.pelanggaran_perjam=NEGASI_ABSENSI(wfm.status)*wfm.j
am
  WHEN NOT MATCHED THEN
       INSERT (
              fm.id_minat,fm.id_periode,fm.id_jurusan,

fm.id_matkul,fm.nrp,fm.absensi,fm.pelanggaran,
              fm.jam,fm.pelanggaran_perjam
              )
       VALUES (
              wfm.id,wfm.tahun||wfm.semester,
              wfm.jur_id,wfm.matkul_id,wfm.nrp,

UBAH_NILAI(wfm.status),NEGASI_ABSENSI(wfm.status),

wfm.jam,NEGASI ABSENSI(wfm.status)*wfm.jam
              );
      commit;
end;
```

Fungsi yang di panggil dalam proses ETL analisa Kedisiplinan dan Analisa Minat mahasiswa terhadap matakuliah tertentu adalah :

Fungsi NEGASI_ABSENSI(memberi nilai status absensi)
Berikut listing program fungsi NEGASI_ABSENSI :

```
create or replace function NEGASI_ABSENSI(N CHAR)
return number is
  Result number;
Begin

  if N='A' then
     Result:=1;
  elsif N='H' then
     Result:=0;
  elsif N='I' then
     Result:=0;
  elsif N='S' then
     Result:=0;
  else
     Result:=1;
  end if;
  return(Result);
end NEGASI_ABSENSI;
```

Fungsi NEGASI_ABSENSI di buat untuk memperoleh nilai angka pada status absensi Alfa(A) mahasiswa,hal ini diperlukan untuk proses agregasi yakni menghitung jumlah jam_pelanggaran mahasiswa selama satu semester.

Fungsi UBAH_NILAI(memberi nilai status absensi)
Berikut listing program fungsi UBAH_NILAI :

```
create or replace function UBAH_NILAI(X CHAR) return
number is
  Result number;

begin

  if X='A' then
     Result:=0;
  elsif X='H' then
```

```
      Result:=1;

elsif X='I' then

      Result:=1;
  elsif X='S' then
      Result:=1;
  else
      Result:=0;
  end if;

  return(Result);

end UBAH_NILAI;
```

Fungsi UBAH_NILAI merupakan ke balikan dari fungsi NEGASI_ABSENSI, jika pada fungsi NEGASI_ABSENSI status absensi Alfa(A) nlainya adalah 1 dan status Hadir (H) bernilai 0, maka pada fungsi UBAH_NILAI status absensi Hadir(H) bernilai 1 dan status Alfa(A) bernilai 0, karena fungsi ini digunakan untuk menghitung(agregasi) Kedisiplinan mahasiswa berdasarkan rata-rata kehadiran mahasiswa.

3.4.2.4 Proses ETL Analisa Recommended Beasiswa

Pada proses ETL analisa recommendasi beasiswa bagi mahasiswa, proses dimulai dengan load data dari tabel kuliah ke tabel dimensi periode, dari tabel mahasiswa data di load ke tabel dimensi mahasiswa, kemudian proses load di lanjutkan dari tabel eis_absensi mahasiswa ke tabel tempdss1, dilanjutkan load data dari tabel tempdss1 ke table tempdss2 yang sebelumnya telah melalui proses transformasi dengan memanggil fungsi NEGASI_ABSENSI.

Melalui proses cleaning data (ekstrak data) dari tabel f_prestasi dan tabel tempdss2 output data kemudian di alirkan ke tabel f_dss, dari tabel f_dss inilah di lakukan perhitungan dengan teknik fuzzy dengan menggunakan field IPS,PELANGGARAN dan PENGHASILAN_ORTU sebagai variabel dari fuzzy, hasil dari perhitungan fuzzy di alirkan ke tabel beasiswa, pada tabel inilah yang nantinya menjadi sumber dari analisa recomendasi

beasiswa bagi mahasiswa. Berikut listing program analisa rekomendasi beasiswa.

```
declare

begin

    --insert update dimensi dss_periode

MERGE INTO warehouse.d_dss_period p
USING (SELECT DISTINCT tahun,semester

FROM wahib.kuliah
 )wp
  ON (p.id_periode=wp.tahun||wp.semester)

  WHEN MATCHED THEN
    UPDATE
       SET p.tahun=wp.tahun,
           p.semester=wp.semester
  WHEN NOT MATCHED THEN
       INSERT (p.id_periode,p.tahun,p.semester)
       VALUES
(wp.tahun||wp.semester,wp.tahun,wp.semester);

  --insert update dimensi dss_mahasiswa
  MERGE INTO warehouse.d_dss_mhs m
  USING (SELECT nrp,nama,
                (penghasilan+penghasilan_ibu)as
penghasilan_ortu
           FROM wahib.mahasiswa
           )wm
  ON (m.nrp=wm.nrp)

  WHEN MATCHED THEN
    UPDATE
       SET m.nama=wm.nama,
           m.penghasilan_ortu=wm.penghasilan_ortu
  WHEN NOT MATCHED THEN
       INSERT (m.nrp,m.nama,m.penghasilan ortu)
       VALUES (wm.nrp,wm.nama,wm.penghasilan_ortu);

  --insert update  tempdss1
  MERGE INTO warehouse.tempdss1 t
  USING (SELECT eis.nomor,k.tahun,
                k.semester,m.nrp,mt.jam,eis.status
           FROM wahib.eis_absensi_mahasiswa eis,
```

```
                wahib.kuliah k,wahib.mahasiswa
m,wahib.matakuliah mt
          where eis.kuliah=k.nomor
            and k.matakuliah=mt.nomor
            and eis.mahasiswa=m.nomor
            and eis.status is not null
            )wt
  ON (t.id_temp=wt.nomor)

  WHEN MATCHED THEN
    UPDATE
       SET
t.id_tempdss2=wt.tahun||wt.semester||wt.nrp,
           t.tahun=wt.tahun,
           t.semester=wt.semester,
           t.nrp=wt.nrp,
           t.jam=wt.jam,
           t.absensi=NEGASI_ABSENSI(wt.status),

t.pelanggaran_perjam=NEGASI_ABSENSI(wt.status)*wt.jam
  WHEN NOT MATCHED THEN
       INSERT (
              t.id temp,t.id tempdss2,
              t.tahun,t.semester,t.nrp,
              t.jam,t.absensi,t.pelanggaran_perjam
              )
       VALUES (
              wt.nomor,wt.tahun||wt.semester||wt.nrp,
              wt.tahun,wt.semester,wt.nrp,wt.jam,
              NEGASI_ABSENSI(wt.status),
              NEGASI_ABSENSI(wt.status)*wt.jam
              );


    --insert update  tempdss2
  MERGE INTO warehouse.tempdss2 t2
  USING (
         SELECT id tempdss2,tahun,semester,
                nrp,sum(pelanggaran_perjam)as
pelanggaran
           FROM warehouse.tempdss1
         GROUP BY id_tempdss2,tahun,semester,nrp
         ) wt2
  ON (t2.id_tempdss2=wt2.id_tempdss2)

  WHEN MATCHED THEN
    UPDATE
       SET t2.tahun=wt2.tahun,
```

```
          t2.semester=wt2.semester,
          t2.nrp=wt2.nrp,
          t2.pelanggaran=wt2.pelanggaran
  WHEN NOT MATCHED THEN
       INSERT (
              t2.id tempdss2,t2.tahun,
              t2.semester,t2.nrp,t2.pelanggaran
              )
            VALUES (
              wt2.id tempdss2,wt2.tahun,
              wt2.semester,wt2.nrp,wt2.pelanggaran
              );


     --insert update  fakta dss
  MERGE INTO warehouse.f dss fd
  USING (SELECT f.id f pres,f.id periode,
                f.nrp,f.ips,t.pelanggaran
           FROM warehouse.f_prestasi f,
                warehouse.tempdss2 t
          WHERE id_f_pres=id_tempdss2
          ) wfd

  ON (fd.id_dss=wfd.id_f_pres)

  WHEN MATCHED THEN
    UPDATE
       SET fd.id periode=wfd.id_periode,
           fd.nrp=wfd.nrp,
           fd.ips=wfd.ips,
           fd.pelanggaran=wfd.pelanggaran
  WHEN NOT MATCHED THEN
       INSERT
(fd.id dss,fd.id_periode,fd.nrp,fd.ips,fd.pelanggaran)
       VALUES
(wfd.id_f_pres,wfd.id_periode,wfd.nrp,wfd.ips,wfd.pela
nggaran);
    --insert update fakta beasiswa
  MERGE INTO warehouse.beasiswa b

  USING(SELECT f.id_dss,p.tahun,p.semester,
               m.nrp,m.nama,round(f.ips,2) as ip,
               f.pelanggaran,m.penghasilan_ortu
          FROM warehouse.d dss period p,
               warehouse.d dss mhs m,warehouse.f_dss f
         WHERE f.id periode=p.id periode
           and f.nrp=m.nrp and f.pelanggaran is not
null
```

```
           and f.ips is not null
           )wb
  ON (b.id_mhs=wb.id_dss)

  WHEN MATCHED THEN
    UPDATE

       SET b.tahun=wb.tahun,
           b.semester=wb.semester,
           b.nrp=wb.nrp,
           b.nama=wb.nama,
           b.ips=wb.ip,
           b.ip_kurang=IP_KURANG(wb.ip),
           b.ip_baik=IP_BAIK(wb.ip),
           b.ip sangat baik=IP ISTIMEWA(wb.ip),
           b.jam_pelanggaran=wb.pelanggaran,

b.tidak_disiplin=TIDAK_DISIPLIN(wb.pelanggaran),

b.kurang_disiplin=KURANG_DISIPLIN(wb.pelanggaran),
b.disiplin=DISIPLIN(wb.pelanggaran),
b.penghasilan_ortu=wb.penghasilan_ortu,

b.p kurang=KURANG MAMPU(wb.penghasilan ortu),
           b.p_mampu=MAMPU(wb.penghasilan_ortu),
           b.p_kaya=KAYA(wb.penghasilan_ortu),

b.rangking all=IP ISTIMEWA(wb.ip)*DISIPLIN(wb.pelangga
ran)*KURANG_MAMPU(wb.penghasilan_ortu),

b.rangking_ip_disiplin=IP_ISTIMEWA(wb.ip)*DISIPLIN(wb.
pelanggaran)
  WHEN NOT MATCHED THEN
       INSERT (

b.id_mhs,b.tahun,b.semester,b.nrp,b.nama,b.ips,b.ip_ku
rang,

b.ip baik,b.ip_sangat_baik,b.jam_pelanggaran,b.tidak_d
isiplin,

b.kurang_disiplin,b.disiplin,b.penghasilan_ortu,

b.p kurang,b.p_mampu,b.p_kaya,b.rangking_all,b.rangkin
g ip disiplin
             )
```

```
VALUES (
wb.id_dss,wb.tahun,wb.semester,wb.nrp,wb.nama,wb.ip,

IP_KURANG(wb.ip),IP_BAIK(wb.ip),IP_ISTIMEWA(wb.ip),

wb.pelanggaran,TIDAK_DISIPLIN(wb.pelanggaran),


KURANG_DISIPLIN(wb.pelanggaran),DISIPLIN(wb.pelanggara
n),wb.penghasilan_ortu,KURANG_MAMPU(wb.penghasilan_ort
u),

MAMPU(wb.penghasilan_ortu),KAYA(wb.penghasilan_ortu),
IP_ISTIMEWA(wb.ip)*DISIPLIN(wb.pelanggaran)*KURANG_MAM
PU(wb.penghasilan_ortu),

IP_ISTIMEWA(wb.ip)*DISIPLIN(wb.pelanggaran)
                );
  commit;
end;
```

Adapun fungsi yang di panggil dalam proses ETL untuk analisa recomendasi beasiswa yang merupakan variabel dari analisa recomendasi beasiswa adalah:

Fungsi NEGASI_ABSENSI(memberi nilai status absensi)
Berikut listing program fungsi NEGASI_ABSENSI :

```
create or replace function NEGASI_ABSENSI(N CHAR)
return number is
  Result number;
begin

  if N='A' then
     Result:=1;
  elsif N='H' then
     Result:=0;
  elsif N='I' then
     Result:=0;
  elsif N='S' then
     Result:=0;
  else
     Result:=1;
  end if;
```

```
return(Result);
end NEGASI_ABSENSI;
```

Fungsi DISIPLIN (Menentukan Miu(u) Variabel Disiplin)
Berikut listing program fungsi DISIPLIN :

```
create or replace function DISIPLIN(DIS NUMBER) return
number is
  Result float;
begin

  if DIS<10 then
     Result:=1;
  elsif DIS>=10 and DIS<=19 then
     Result:=(19-DIS)/9;
  elsif DIS>19 then
     Result:=0;
  else
     Result:=0;
  end if;

  return(Result);
end DISIPLIN;
```

Fungsi KURANG_DISIPLIN (Menentukan Nilai Miu(u) Variabel Kurang Disiplin)
Berikut listing program fungsi KURANG DISIPLIN :

```
create or replace function KURANG_DISIPLIN(DIS NUMBER)
return number is
  Result float;
begin

  if DIS<=10 and DIS>=30  then
     Result:=0;
  elsif DIS>=10 and DIS<=19 then
     Result:=(DIS-10)/9;
  elsif DIS>19 and DIS<30 then
     Result:=(30-DIS)/11;
  else
     Result:=1;
  end if;

  return(Result);
end KURANG_DISIPLIN;
```

Fungsi TIDAK_DISIPLIN (Menentukan Nilai Miu(u) Variabel Tidak Disiplin)
Berikut listing program fungsi TIDAK_DISIPLIN :

```
create or replace function TIDAK_DISIPLIN(DIS NUMBER)
return number is
  Result float;
begin


if DIS<=19 then

     Result:=0;
  elsif DIS>19 and DIS <=30 then
     Result:=(DIS-19)/11;
  elsif DIS>30 then
     Result:=1;
  end if;

  return(Result);
end TIDAK_DISIPLIN;
```

Fungsi IP_ISTIMEWA (Menentukan Nilai Miu(u) Variabel IP yang Cumlaud)
Berikut listing program fungsi IP_ISTIMEWA :

```
create or replace function IP_ISTIMEWA(IPS NUMBER)
return number is
  hasil float;
begin
  if IPS<=3 then
     hasil:=0;
  elsif IPS<=3.3 and IPS>3 then
     hasil:=(IPS-3)/0.3;
  elsif IPS>3.3 then
     hasil:=1;
  else
     hasil:=0;
  end if;

  return(hasil);
end IP_ISTIMEWA;
```

Fungsi IP_BAIK (Menentukan Nilai Miu(u) Variabel IP yang Cukup Baik)
Berikut listing program fungsi IP_BAIK :

```
create or replace function IP_BAIK(IPS NUMBER) return
number is
  hasil float;
begin
  if IPS<=2.7 and IPS>=3.3 then
     hasil:=0;
  elsif IPS>2.7 and IPS<=3 then
     hasil:=(IPS-2.7)/0.3;
  elsif IPS>3 and IPS<3.3 then
        hasil:=(3.3-IPS)/0.3;
  else
        hasil:=0;
  end if;

  return(hasil);
end IP_BAIK;
```

Fungsi IP_KURANG (Menentukan Nilai Miu(u) Variabel IP yang di Kategorikan Kurang)
Berikut listing program fungsi IP_KURANG :

```
create or replace function IP_KURANG(IPS NUMBER)
return number is
  hasil float;

begin
  if IPS <= 2.7 then
     hasil:=1;
  elsif IPS>2.7 and IPS<=3 then
     hasil:=(3-IPS)/0.3;
  elsif IPS > 3 then
     hasil:=0;
  else
     hasil:=0;
  end if;

  return(hasil);
end IP_KURANG;
```

Fungsi KAYA (Menentukan Nilai Miu(u) Variabel Penghasilan_Ortu yang memiliki penghasilan Lebih)
Berikut listing program fungsi KAYA :

```
create or replace function KAYA(ORTU NUMBER) return
number is
  hasil float;
begin
    if  ORTU<2000000 then
       hasil:=0;
    elsif ORTU>2000000 and ORTU<=3000000 then
       hasil:=(ORTU-2000000)/1000000;

elsif ORTU>3000000 then
       hasil:=1;
    else
       hasil:=1;
    end if;
  return(hasil);
end KAYA;
```

Fungsi MAMPU (Menentukan Nilai Miu(u) Variabel Penghasilan_Ortu yang memiliki penghasilan Cukup)
Berikut listing program fungsi MAMPU :

```
create or replace function MAMPU(ORTU NUMBER) return
number is
  hasil float;
begin
    if ORTU <=700000 and ORTU >=3000000 then

hasil:=0;

    elsif ORTU>700000 and ORTU<=2000000 then
       hasil:=(ORTU-700000)/1300000;
    elsif ORTU >2000000 and ORTU<3000000 then
       hasil:=(3000000-ORTU)/1000000;
    else
       hasil:=0;
    end if;

  return(hasil);
end MAMPU;
```

Fungsi KURANG_MAMPU (Menentukan Nilai Miu(u) Variabel Penghasilan_Ortu yang berpenghasilan Kurang)
Berikut listing program fungsi KURANG_MAMPU :

```
create or replace function KURANG_MAMPU(ORTU NUMBER)
return number is
  hasil float;
begin
    if ORTU <=700000 then
       hasil:=1;
    elsif ORTU>700000 and ORTU<=2000000 then
       hasil:=(2000000-ORTU)/1300000;
    elsif ORTU >2000000 then
       hasil:=0;
    else
       hasil:=0;
    end if;
  return(hasil);
end KURANG_MAMPU;
```

3.4.3 Agregasi Data

Setelah semua proses ektraksi, transformasi dan loading data selesai di lakukan ke table tujuan(data warehouse), proses selanjutnya adalah melakukan agregasi data, sehingga di dapatkan analisa data yang terkelompok berdasarkan periode data.

Dari Tabel fakta Prestasi diperoleh table- table Analisa:

Tabel 3.7 Agregasi analisa prestasi

Nama Tabel Analisa	**Query Agregasi(Material View)**
F_PRESTASI_JUR_SMT	SELECT tahun,semester,jurusan,count(*) as jml_MHS,sum(ips) as jml_IPS,avg(ips) as rata_rata_IPS,(avg(ips)/4) * 100 as prosentase_IP FROM d_pres_period d,d_pres_jurusan j ,d_pres_mhs m,f_prestasi f WHERE d.id_periode=f.id_periode and f.id_jurusan=j.id_jurusan and f.nrp=m.nrp GROUP BY tahun,semester,jurusan

F_PRESTASI_JUR_TAHUNAN	SELECT tahun,jurusan,count(*) as jml_MHS,sum(ips) as jml_IPS,avg(ips) as rata_rata_IPS,(avg(ips)/4) * 100 as prosentase_IP FROM d_pres_period d,d_pres_jurusan j ,d_pres_mhs m,f_prestasi f WHERE d.id_periode=f.id_periode and f.id_jurusan=j.id_jurusan and f.nrp=m.nrp GROUP BY tahun,jurusan
• Analisa Rangking Indeks Prestasi di peroleh dari Tabel-Tabel : d_pres_period, d_pres_mhs , d_pres_jurusan , f_prestasi	

Dari Table Fakta Kurikulum di peroleh beberapa table Analisa :

Tabel 3.8 Agregasi analisa kurikulum

Nama Tabel Analisa	**Query Agregasi(Material View)**
F_KURIK_JUR_SEMESTER	SELECT p.tahun,p.semester,j.jurusan,mt.matakuliah,count(*) as jumlah_MHS,sum(f.nilai) as jml_nilai,((sum(f.nilai)/count(*))/100) * 100 as persentase FROM f_kurikulum f,d_kurik_jurusan j,d_kurik_period p,d_kurik_matkul mt WHERE f.id_periode=p.id_periode and f.id_jurusan=j.id_jurusan and f.id_matkul=mt.id_matkul GROUP BY tahun,semester,jurusan,matakuliah
F_KURIK_JUR_TAHUNAN	SELECT p.tahun ,j.jurusan,mt.matakuliah,count(*) as jumlah_MHS ,sum(f.nilai) as jml_nilai,((sum(f.nilai)/count(*))/100) * 100 as persentase FROM f_kurikulum f,d_kurik_jurusan j ,d_kurik_period p,d_kurik_matkul mt WHERE f.id_periode=p.id_periode and f.id_jurusan=j.id_jurusan and f.id_matkul=mt.id_matkul GROUP BY tahun,jurusan,matakuliah

Dari relasi tabel-tabel dimensi minat_matkul dan tabel fakta minat_matkul, di peroleh beberapa analisa sebagai berikut :

1. Analisa Minat Terhadap Matakuliah Tertentu

Tabel 3.9 Agregasi analisa minat matkul

Nama Tabel Analisa	Query Agregasi(Material View)
F_MINAT_MATKUL_SMT	SELECT p.tahun,p.semester,mt.matakuliah,j.jurusan ,count(*) as jml_absensi,sum(f.absensi) as hadir ,(count(*) - sum(f.absensi)) as pelanggaran ,(sum(f.absensi) / count(*)) * 100 as porsentase FROM d_minat_period p, f_minat_matkul f , d_minat_matkul mt,d_minat_jurusan j WHERE f.id_periode=p.id_periode and f.id_matkul=mt.id_matkul and f.id_jurusan=j.id_jurusan GROUP BY tahun,semester,matakuliah,jurusan

2. Analisa Kedisiplinan

Tabel 3.10 Agregasi analisa kedisiplinan

Nama Tabel Analisa	Query Agregasi(Material View)
F_DISIPLIN_SEMESTER	SELECT p.tahun , p.semester , j.jurusan ,count(*) as total_Absensi ,sum(f.absensi) as hadir ,(count(*) - sum(f.absensi)) as pelanggaran ,(sum(f.absensi) / count(*)) * 100 as persentase FROM d_minat_period p , d_minat_jurusan j , f_minat_matkul f WHERE f.id_periode=p.id_periode and f.id_jurusan=j.id_jurusan GROUP BY jurusan,tahun,semester

F_DISIPLIN_TAHUNAN	SELECT p.tahun , j.jurusan ,count(*) as total_Absensi ,sum(f.absensi) as hadir ,(count(*) - sum(f.absensi)) as pelanggaran ,(sum(f.absensi) / count(*)) * 100 as persentase FROM d_minat_period p , d_minat_jurusan j , f_minat_matkul f WHERE f.id_periode=p.id_periode and f.id_jurusan=j.id_jurusan GROUP BY jurusan,tahun

3. Analisa Ranking Pelanggaran

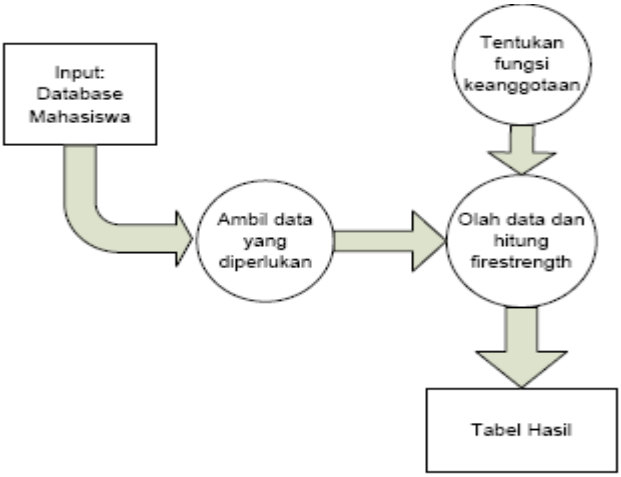
Tabel 3.11 Agregasi analisa rangking pelanggaran

Nama Tabel Analisa	Query Agregasi(Material View)
RANGKING_PELANGGARAN	SELECT p.tahun,p.semester, j.jurusan,m.nrp,m.nama, sum(f.pelanggaran_perjam) as Pelanggaran_Perjam FROM d_minat_period p,d_minat_jurusan j, d_minat_mhs m,f_minat_matkul f WHERE f.id_periode=p.id_periode and f.id_jurusan=j.id_jurusan and f.nrp=m.nrp GROUP BY tahun,semester,jurusan,m.nrp,m.nama

Data-data dari tabel agregasi inilah yang nantinya akan menjadi sumber analisa dari data warehouse, yang kemudian akan di sampaikan kepada user berupa report data berbasis website.

3.4.4 Fuzzy Database untuk Rekomendasi Beasiswa

Variabel yang di gunakan untuk menghitung rekomendasi beasiswa adalah indeks prestasi, pelanggaran da penghasilan orang tua mahasiswa. Secara umum Algoritma sistem yang dibuat adalah seperti gambar diagram di bawah ini:



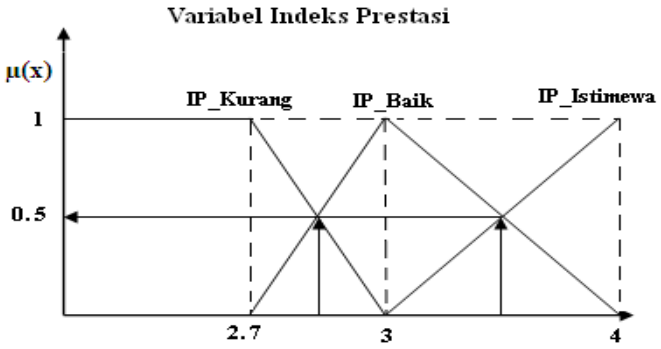
Gambar 3.8 Algoritma Diagram sistem

Dari gambar diatas terlihat ada 4 proses, yaitu:

1. Input database mahasiswa ,yaitu menentukan database, tabel dan field asal. nama database, tabel dan field ditentukan pada saat proses ETL.
2. Ambil data yang di perlukan, hanya data yang akan menjadi variable penentuan beasiswa saja yang akan dip roses dengan fuzzy.
3. Menentukan fungsi keanggotaan, fungsi keanggotaan di peroleh dari variable-variabel fuzzy yang telah di petakan dan telah di tentukan batasan-batasanya.(Lihat gambar 3.9 – 3.14).
4. Menghitung firestrength. Fire strength menunjukan seberapa besar rekomendasi yang diberikan oleh sistem (fire strength ini memiliki nilai berkisar antara 0-1). Nilai 1 menunjukkan rekomendasi penuh, apabila fire strength mendekati nilai 0, maka mahasiswa tersebut semakin tidak direkomendasi. Penghitungan fire strength ini membutuhkan data Mu (μ) dari Tabel Data Mu (μ) dan variable linguistik dari query yang ada.

Berikut ini adalah variable yang di pakai untuk menentukan ekomendasi beasiswa.

1. Variabel Indeks Prestasi



Gambar 3.9 Variabel Indeks Prestasi

Fungsi keanggotaan dari variable indeks prestasi adalah :

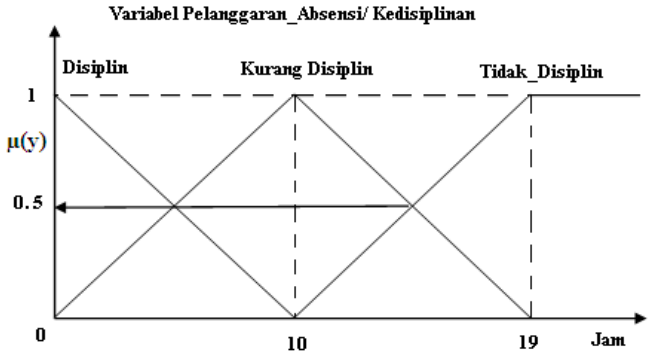
$$\mu IP\_Istemewa[x] = \begin{cases} 0 & x <= 3 \\ \frac{x-3}{1} & 3 =< x <= 4 \\ 1 & x = 4 \end{cases}$$

$$\mu IP\_Baik[x] = \begin{cases} 0 & x <= 2.7 \text{ atau } x = 4 \\ \frac{x-2.7}{0.3} & 2.7 =< x <= 3 \\ \frac{4-x}{1} & 3 =< x <= 4 \end{cases}$$

$$\mu IP\_Kurang[x] = \begin{cases} 0 & x >= 3 \\ \frac{3-x}{0.3} & 2.7 =< x <= 3 \\ 1 & x <= 2.7 \end{cases}$$

Gambar 3.10 Fungsi keanggotaan variable indeks prestasi

2. Variabel Pelanggaran (Absensi)



Gambar 3.11 Variabel Pelanggaran

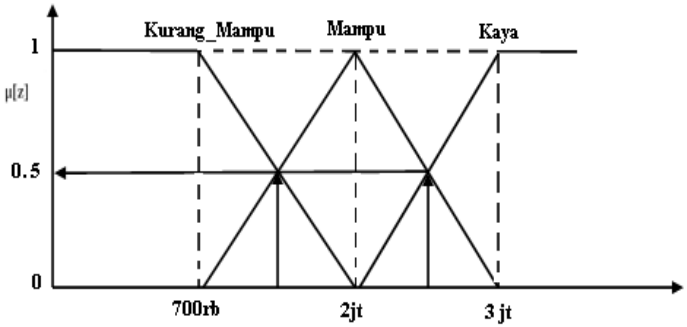
Fungsi keanggotaan dari variable Pelanggaran adalah :

$$\mu Disiplin[y] = \begin{cases} 0; & y>=10 \\ \frac{10-y}{10}; & 0=<y<=10 \\ 1; & y=0 \end{cases} \quad \mu Kurang\_Disiplin[y] = \begin{cases} 0; & 19=<y<=0 \\ \frac{y-10}{10}; & 0=<y<=10 \\ \frac{19-y}{9}; & 10=<y<=19 \end{cases}$$

$$\mu Tidak\_Disiplin[y] = \begin{cases} 0; & y<=10 \\ \frac{y-10}{9}; & 10=<y<=19 \\ 1; & y>=19 \end{cases}$$

Gambar 3.12 Fungsi keanggotaan variabel Pelanggaran

3. Variabel Penghasilan Ortu



Gambar 3.13 Variabel Pelanggaran

Fungsi keanggotaan dari variable Penghasilan Ortu adalah :

$$\mu Kurang\_Mampu[z] = \begin{cases} 0; & z >= 2jt \\ \frac{2jt - z}{1{,}3jt}; & 700rb =< z =< 2jt \\ 1; & z <= 700rb \end{cases}$$

$$\mu Mampu[z] = \begin{cases} 0; & z <= 700rb \text{ atau } z >= 3jt \\ \frac{z - 700rb}{1jt}; & 700rb =< z =< 2jt \\ \frac{3jt - z}{1jt}; & 2jt =< z =< 3jt \end{cases}$$

$$\mu Kaya[z] = \begin{cases} 0; & z <= 2jt \\ \frac{z - 2jt}{1jt}; & 2jt =< z =< 3jt \\ 1; & z >= 3jt \end{cases}$$

Gambar 3.14 Fungsi keanggotaan variabel Penghasilan Ortu

3.4.5 Membuat Report Data

Langkah terakhir dalam pemuatan data warehouse adalah membuat report data, dalam tugas akhit ini report data akan di tampilkan berbasis website dengan memanfaatkan teknologi PHP-ORACLE.

3.4.5.1 Membuat Koneksi PHP ke ORACLE

Berikut ini adalah penggalan script untuk koneksi ke database Oracle mengunakan PHP.

```
$username=”WAREHOUSE”;
$pwd=”ORACLE”;
$db=”AKADEMIK”;
$conn=oci_connect($username,$pwd ,$db);
```

3.4.5.2 Membuat Report Analisa Indeks Prestasi

Berikut ini adalah Penggalan listing program analisa indeks prestasi yang membandingkan semua jurusan di PENS-ITS.

```
<?php

include ("..\jpgraph\src\jpgraph.php");
include ("..\jpgraph\src\jpgraph_line.php");
include ("..\jpgraph\src\jpgraph_bar.php");
include ("..\jpgraph\src\jpgraph_flags.php");
include ("koneksi.php");

$dataIT= array();
$dataELKA=array();
$dataELIN=array();
$dataTELKOM=array();
$dataMK=array("","","","","","","","");
$dataTK=array("","","","","","","","","","");
$dataMB=array("","","","","","","","","","","","");
$dataTahun= array();
$dataSmt=array();
$dataLabel=array();

//mengambil nilai bplot1
$queryIT = "select tahun,semester,rata_rata_ips from
f_prestasi_jur_smt where jurusan='Teknik Informatika'
order by tahun desc,semester desc";
$hasil_IT=oci_parse($conn,$queryIT);
```

```
oci_execute($hasil_IT);

while ($data1 = oci_fetch_array($hasil_IT))
{
       array_unshift($dataIT,


$data1['RATA_RATA_IPS']);
array_unshift($dataTahun, $data1['TAHUN']);
       array_unshift($dataSmt, $data1['SEMESTER']);
       array_unshift($dataLabel, $data1['TAHUN']."
Semester ".$data1['SEMESTER']);
}

//mengambil nilai bplot2
$queryELKA = "select tahun,semester,rata_rata_ips from
f_prestasi_jur_smt where jurusan='Elektronika' order
by tahun desc,semester desc";
$hasil_ELKA=oci_parse($conn,$queryELKA);

oci_execute($hasil_ELKA);

while ($data2 = oci_fetch_array($hasil_ELKA))
{
       array_unshift($dataELKA,
$data2['RATA_RATA_IPS']);
}
//mengambil nilai bplot3
$queryTELKOM = "select tahun,semester,rata_rata_ips
from f_prestasi_jur_smt where jurusan='Telekomunikasi'
order by tahun desc,semester desc";

$hasil_TELKOM=oci_parse($conn,$queryTELKOM);
oci_execute($hasil_TELKOM);

while ($data3 = oci_fetch_array($hasil_TELKOM))
{
       array_unshift($dataTELKOM,
$data3['RATA_RATA_IPS']);
}

//mengambil nilai bplot4
$queryELIN = "select tahun,semester,rata_rata_ips from
f_prestasi_jur_smt where jurusan='Elektro Industri'
order by tahun desc,semester desc";

$hasil_ELIN=oci_parse($conn,$queryELIN);
oci_execute($hasil_ELIN);
```

```
while ($data4 = oci_fetch_array($hasil_ELIN))
{
      array_unshift($dataELIN,
$data4['RATA_RATA_IPS']);
}

//mengambil nilai bplot5
$queryMK = "select tahun,semester,rata_rata_ips from
f_prestasi_jur_smt where jurusan='Mekatronik' order by
tahun asc,semester asc";

$hasil_MK=oci_parse($conn,$queryMK);
oci_execute($hasil_MK);

while ($data5 = oci_fetch_array($hasil_MK))
{
      array_push($dataMK, $data5['RATA_RATA_IPS']);
}

//mengambil nilai bplot6
$queryTK = "select tahun,semester,rata_rata_ips from
f_prestasi_jur_smt where jurusan='Teknik Komputer'
order by tahun asc,semester asc";

$hasil_TK=oci_parse($conn,$queryTK);
oci_execute($hasil_TK);

while ($data6 = oci_fetch_array($hasil_TK))
{
      array_push($dataTK, $data6['RATA_RATA_IPS']);
}

//mengambil nilai bplot7
$queryMB = "select tahun,semester,rata_rata_ips from
f_prestasi_jur_smt where jurusan='Multimedia
Broadcasting' order by tahun asc,semester asc";

$hasil_MB=oci_parse($conn,$queryMB);
oci_execute($hasil_MB);

while ($data7 = oci_fetch_array($hasil_MB))
{
      array_push($dataMB, $data7['RATA_RATA_IPS']);

}

//membuat basic graph
```

```
$graph = new Graph(2000,500);
$graph->SetScale("textlin");
//$graph->Set90andMargin(220,40,80,10);
$graph->SetMargin(50,20,70,180);
$graph->SetMarginColor('white:0.9');


$graph->SetColor('green@0.8');
// menampilkan plot batang dari data IT
$bplot1 = new BarPlot($dataIT);
$bplot1->SetFillColor("red@0.1");
$bplot1->value->SetFont(FF_TIMES,FS_NORMAL,9);
$bplot1->value->SetAngle(70);
$bplot1->value->show();
$bplot1->value->setFormat('%.2f');

. . . . . .

. . . . . .
```

3.4.5.3 Membuat Report Analisa Kedisiplinan

Berikut ini adalah penggalan listing program analisa kedisiplinan yang membandingkan semua jurusan di PENS-ITS.

```
<?php

include ("..\jpgraph\src\jpgraph.php");
include ("..\jpgraph\src\jpgraph_line.php");
include ("..\jpgraph\src\jpgraph_bar.php");
include ("..\jpgraph\src\jpgraph_flags.php");
include ("koneksi.php");

$dataIT= array("");
$dataELKA=array();
$dataELIN=array("");
$dataTELKOM=array();
$dataMK=array("","","");
$dataTK=array("","","","","");
$dataMB=array("","","","","","","");
$dataTahun= array();
$dataSmt=array();
$dataLabel=array();

//mengambil nilai bplot1
```

```
$queryELKA = "select tahun,semester,jurusan,persentase
from f_disiplin_semester where jurusan='Elektronika'
order by tahun desc,semester desc";

$hasil_ELKA=oci_parse($conn,$queryELKA);
oci_execute($hasil_ELKA);

while ($data2 = oci_fetch_array($hasil_ELKA))
{
       array_unshift($dataELKA, $data2['PERSENTASE']);
       array_unshift($dataTahun, $data2['TAHUN']);
       array_unshift($dataSmt, $data2['SEMESTER']);
       array_unshift($dataLabel, $data2['TAHUN']."
Semester ".$data2['SEMESTER']);
}
//mengambil nilai bplot2
$queryTELKOM ="select
tahun,semester,jurusan,persentase from
f_disiplin_semester where jurusan='Telekomunikasi'
order by tahun desc,semester desc";

$hasil_TELKOM=oci_parse($conn,$queryTELKOM);
oci_execute($hasil_TELKOM);

while ($data3 = oci_fetch_array($hasil_TELKOM))
{
       array_unshift($dataTELKOM,
$data3['PERSENTASE']);
}

//mengambil nilai bplot3
$queryELIN ="select tahun,semester,jurusan,persentase
from f_disiplin_semester where jurusan='Elektro
Industri' order by tahun asc,semester asc";


$hasil_ELIN=oci_parse($conn,$queryELIN);
oci_execute($hasil_ELIN);

while ($data4 = oci_fetch_array($hasil_ELIN))
{
       array_push($dataELIN, $data4['PERSENTASE']);
}

//mengambil nilai bplot4
$queryIT = "select tahun,semester,jurusan,persentase
from f_disiplin_semester where jurusan='Teknik
Informatika' order by tahun asc,semester asc";
```

```
$hasil_IT=oci_parse($conn,$queryIT);
oci_execute($hasil_IT);

while ($data1 = oci_fetch_array($hasil_IT))
{
       array_push($dataIT, $data1['PERSENTASE']);

}
//mengambil nilai bplot5
$queryMK = "select tahun,semester,jurusan,persentase
from f_disiplin_semester where jurusan='Mekatronik'
order by tahun asc,semester asc";

$hasil_MK=oci_parse($conn,$queryMK);
oci_execute($hasil_MK);

while ($data5 = oci_fetch_array($hasil_MK))
{
       array_push($dataMK, $data5['PERSENTASE']);}
//mengambil nilai bplot6
$queryTK ="select tahun,semester,jurusan,persentase
from f_disiplin_semester where jurusan='Teknik
Komputer' order by tahun asc,semester asc";

$hasil_TK=oci_parse($conn,$queryTK);
oci_execute($hasil_TK);

while ($data6 = oci_fetch_array($hasil_TK))
{
       array_push($dataTK, $data6['PERSENTASE']);
}

//mengambil nilai bplot7
$queryMB ="select tahun,semester,jurusan,persentase
from f_disiplin_semester where jurusan='Multimedia
Broadcasting' order by tahun asc,semester asc";

$hasil_MB=oci_parse($conn,$queryMB);
oci_execute($hasil_MB);

while ($data7 = oci_fetch_array($hasil_MB))
{
       array_push($dataMB, $data7['PERSENTASE']);

}

//membuat basic graph
$graph = new Graph(1500,500);
```

```
$graph->SetScale("textlin");
//$graph->Set90andMargin(220,40,80,10);
$graph->SetMargin(50,20,100,180);
$graph->SetMarginColor('white:0.9');
$graph->SetColor('green@0.8');
//$graph->SetShadow();

// menampilkan plot batang dari data ELKA
$bplot1 = new BarPlot($dataELKA);
$bplot1->SetFillColor("orange@0.1");
$bplot1->value->SetFont(FF_TIMES,FS_NORMAL,9);
$bplot1->value->SetAngle(70);
$bplot1->value->show();
$bplot1->value->setFormat('%.2f o/o');
. . . . . .
. . . . . .
```

3.4.5.4 Membuat Report Analisa Beasiswa

Berikut ini adalah listing program analisa beasiswa tahun 2008 semester 1 untuk semua jurusan di PENS-ITS.

```
<?php

include ("..\jpgraph\src\jpgraph.php");
include ("..\jpgraph\src\jpgraph_bar.php");
include ("..\jpgraph\src\jpgraph_flags.php");
include ("koneksi.php");

$dataPersen= array();
$dataMhs=array();
$dataSmt=array();
$dataLabel=array();

//mengambil nilai bplot1
$query = "SELECT
tahun,semester,nrp,nama,ips,ip_sangat_baik,jam_pelangg
aran,disiplin,penghasilan_ortu,p_kurang,rangking_all,r
angking_all*100 as persentase FROM beasiswa WHERE
rangking_all>0 and tahun=2008 and semester=1 order by
tahun,semester,rangking_all asc";

$hasil=oci_parse($conn,$query);
oci_execute($hasil);

while ($data = oci_fetch_array($hasil))
```

```
{
       array_unshift($dataPersen,
$data['PERSENTASE']);
       array_unshift($dataMhs, $data['NAMA']);


array_unshift($dataSmt, $data['SEMESTER']);
       array_unshift($dataLabel, "(".$data['TAHUN']."-
".$data['SEMESTER']."-".$data['NRP'].")"." "."
".$data['NAMA']);
}
//membuat basic graph
$graph = new Graph(750,1000);
$graph->SetScale("textlin");
//rotasi 90*
$graph->Set90andMargin(300,40,80,10);
$graph->SetMarginColor('white:0.9');
$graph->SetColor('red@0.8');

// menampilkan plot batang dari data jumlah penduduk
$bplot = new BarPlot($dataPersen);
$bplot->SetFillGradient("blue","cyan",GRAD_MIDVER);
$bplot->value->show();
$bplot->SetWidth(0.3);
$graph->add($bplot);

//Judul dan keterangan
$graph->title->Set("Rekomendasi Beasiswa Mahasiswa
PENS-ITS Tiap Semester(%)");
$graph->xaxis->SetTickLabels($dataLabel);

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Stroke();
oci_close();

?>
```

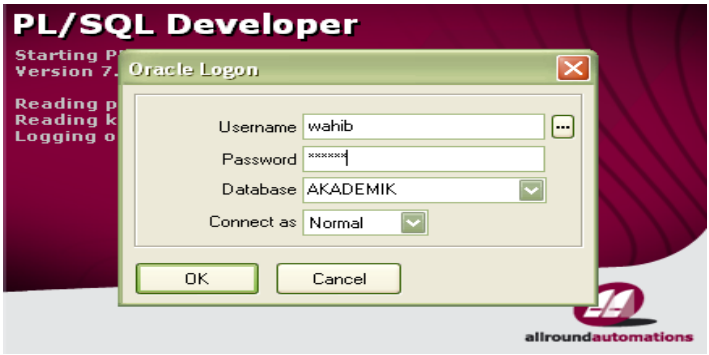
-- Halaman ini sengaja dikosongkan --

BAB IV
PENGUJIAN APLIKASI DAN ANALISA

Pengujian perangkat lunak bertujuan untuk mengetahui apakah aplikasi yang di bangun sudah berjalan sesuai dengan tujuan yang telah di rumuskan pada awal perencanaan.

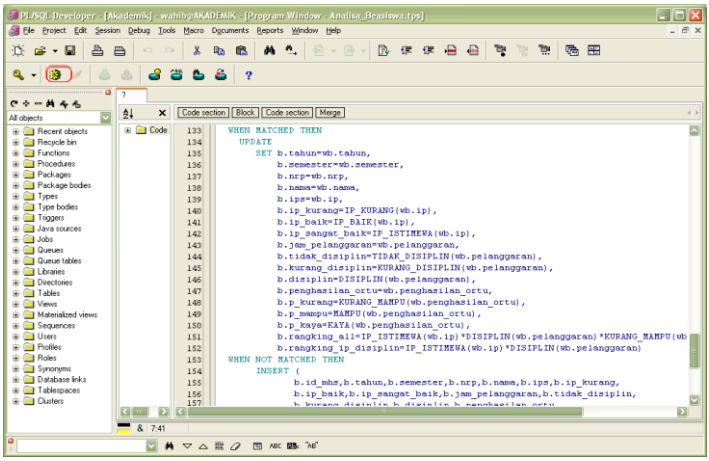
4.1 Uji Coba Sistem

Langkah pertama, pastikan semua tabel pada data warehouse recordnya kosong, jika belum kita bisa menhapus dulu seluruh isi tabel agar proses ETL nya bisa terlihat jelas. Kemudian jalankan aplikasi PL/SQL Developer dan login dengan username : *wahib* (karena data akademik di simpan pada schema user wahib) dan masukan password : *oracle*



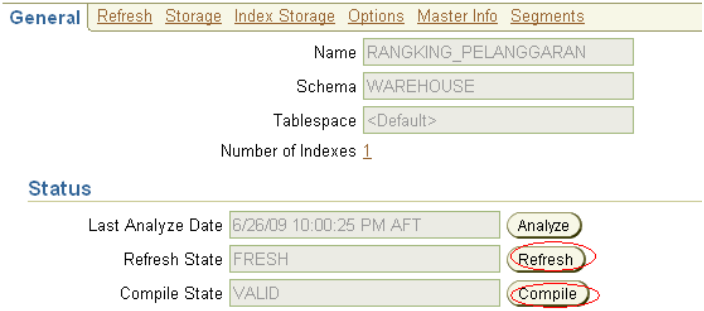
Gambar 4.1 Login PL/SQL Developer

Klick Project →Open → Akademik.prj , kemudian open File →Open → Program File → Pilih salah satu file.tps dari empat proses ETL yaitu Analisa_Prestasi, Analisa_Kurikulum, Analisa MinatMatkul&Disiplin dan Analisa_DSS_Beasiswa. Kemudian *execute* dengan menekan simbol yang kami lingkari dengan tanda merah di bawah ini.



Gambar 4.2 Menjalankan proses ETL

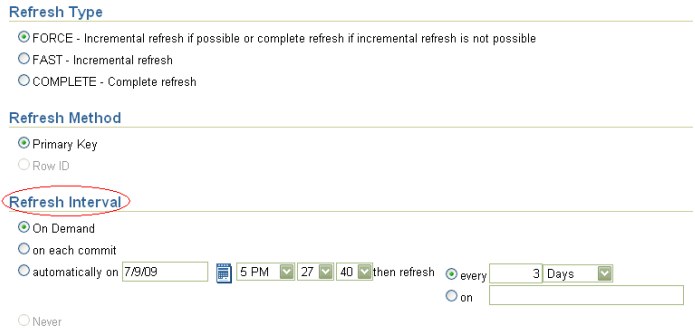
Setelah seluruh proses ETL di jalankan, langkah selanjutnya adalah merefresh Material View.



Gambar 4.3 Merefresh tabel view

Sebelumnya, kita *compile* duhulu untuk mengetahui data yang masuk valid atau tidak, kemudian kita bisa merefresh view dengan mengeklik button *refresh*, proses refresh ini digunakan untuk mengupdate data dalam tabel agregasi data warehouse.

Kita juga bisa mensetting agar proses refreshing dapat berjalan secara otomatis dengan merubah settingan berikut:



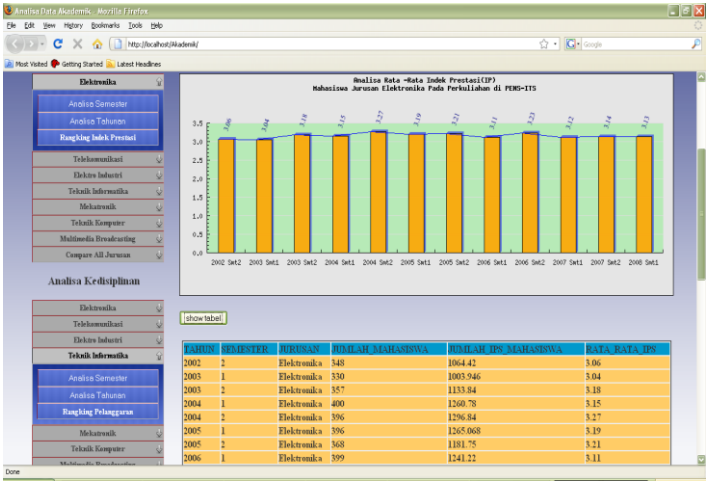
Gambar 4.4 Option proses refresh pada view

Perhatikan pada Refresh Interval yang kami tandai diatas

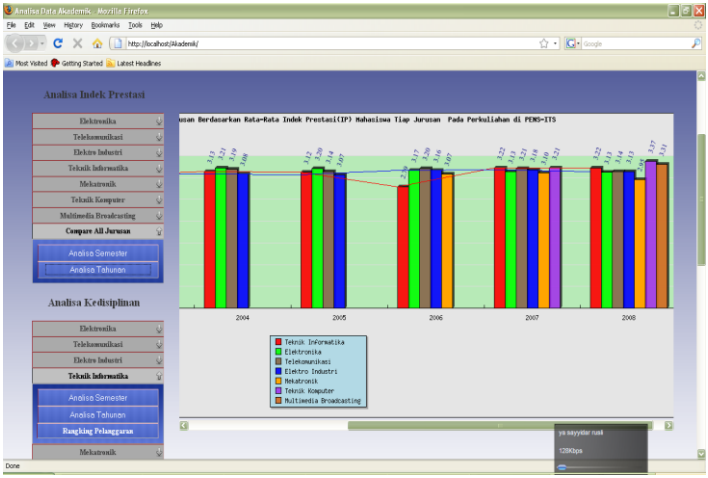
- On Demand : pada option ini proses refresh harus dilakukan secara manual.
- On Each Commit : pada option ini proses refresh akan dilakukan secara otomatis ketika ada inputan data dari tabel sumber.
- Automatically : pada option ini kita bisa menset proses refresh secara otomatis mulai tanggal dan jam tertentu, kemudian kita bisa menset proses refresh selanjutnya dengan ketentuan waktu yang kita inputkan.

Setelah seluruh proses dalam data warehouse selesai di jalankan, selanjutnya adalah menampilkan report dari beberapa analisa data dalam data warehouse yang telah kita bangun.

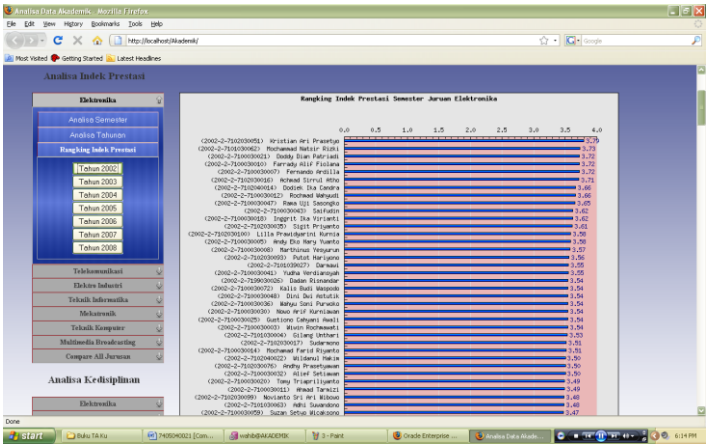
Menampilkan Report Analisa Indeks Prestasi



Gambar 4.5 Analisa rata-rata indeks prestasi



Gambar 4.6 Membandingkan rata-rata indeks prestasi semua jurusan



Gambar 4.7 Report rangking indeks prestasi

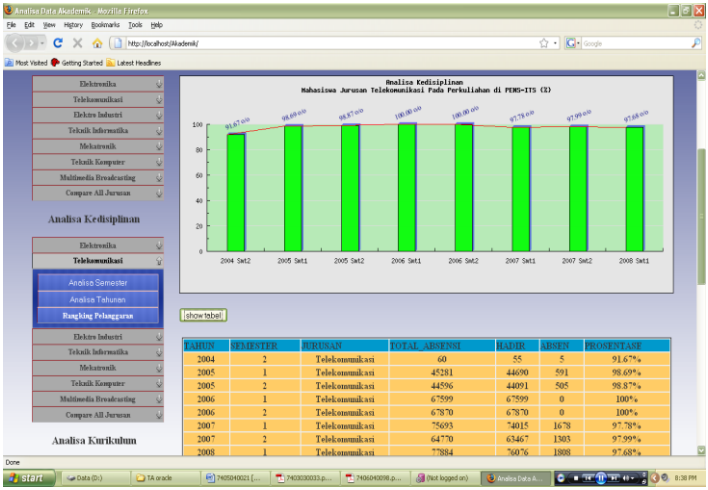
Untuk menghitung analisa indeks prestasi mahasiswa, parameter yang di gunakan adalah rata-rata indeks prestasi yang di peroleh mahasiswa tiap semester. Jika di rumuskan adalah :

Jumlah IPS mahasiswa / Jumlah Mahasiswa

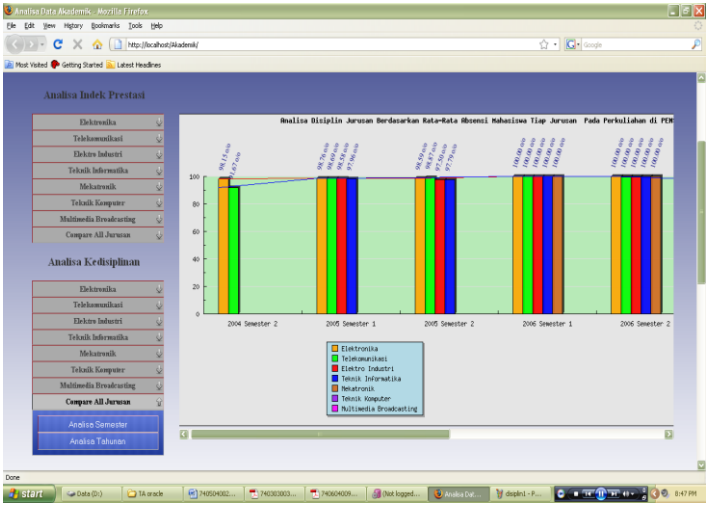
Pada Gambar 4.6 kita bisa melihat perbandingan rata-rata indeks prestasi antar jurusan.Informasi seperti ini tentu sangat penting bagi para pemimpin masing-masing jurusan sebagai acuan mengambil kebijakan di masa mendatang.

Pada Gambar 4.7 report menampilkan rangking indeks prestasi mahasiswa yang di peroleh pada tiap semester untuk semua kelas dalam satu jurusan.

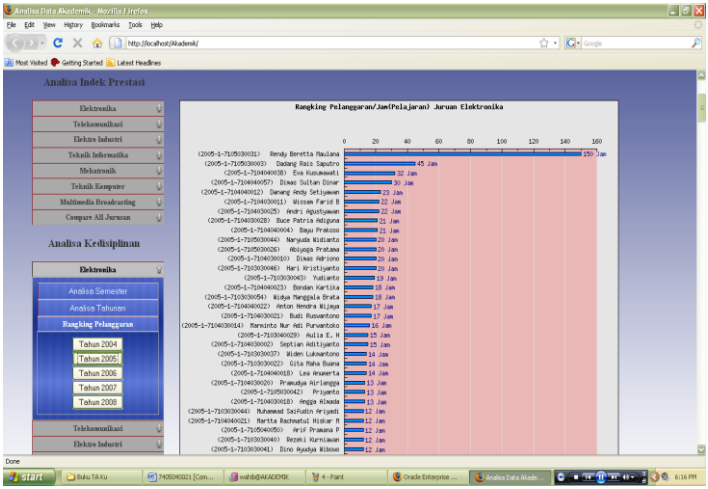
Menampilkan Report Analisa Kedisiplinan.



Gambar 4.8 Report rata-rata kehadiran mahasiswa



Gambar 4.9 Report rata-rata kehadiran mahasiswa All jurusan



Gambar 4.10 Report rangking jam_pelanggran

Untuk menghitung analisa kedisiplinan mahasiswa, parameter yang di gunakan adalah rata-rata absensi kehadiran mahasiswa pada setiap matakuliah terhadap keseluruhan absensi selama satu semester pada setiap jurusan. Jika di rumuskan adalah :

(Total Kehadiran / Total Absensi) x 100 %

dimana status Hadir, Izin dan Sakit masuk dalam kategori Kehadiran,karena penulis menganggap orang yang sakit dan izin tentu sudah memberikan keterangan yang benar kepada dosen maupun kepada managemen absensi.

Pada Gambar 4.9, data yang di tampilkan pada tahun 2004 hanya mencakup dua jurusan yaitu jurusan elektronika dan jurusan telekomunikasi, hal ini dikarenakan pada tahun itu untuk jurusan teknik informatika dan elektro industri absensi mahasiswanya belum akuntable (belum di inputkan kedalam database).

Pada Gambar 4.10 analisa yang ditampilkan adalah rangking banyaknya jam pelanggaran mahasiswa terhadap setiap

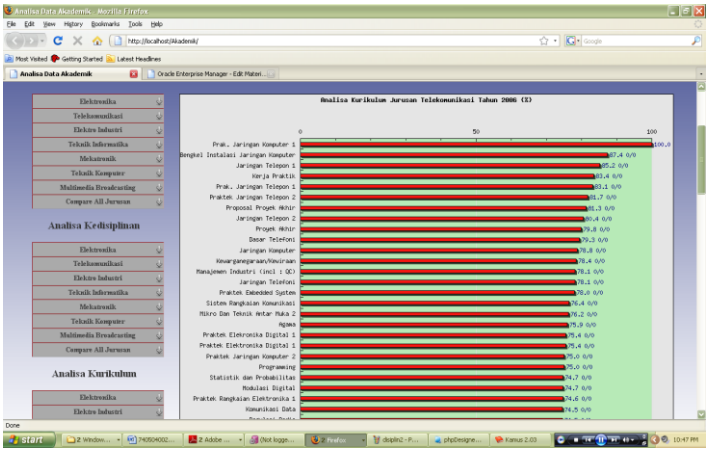
matakuliah dalam waktu satu semester, jika di rumuskan adalah sebagai berikut :

$$\sum (Absensi\ Alfa\ x\ jam\_matakuliah)$$

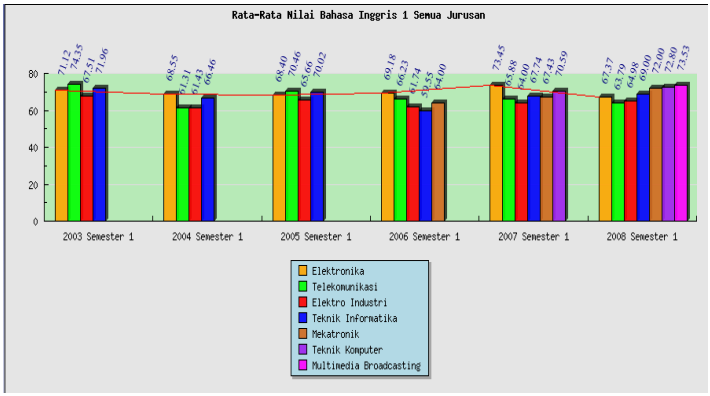
pada report di atas terdapat satu mahasiswa yang melakukan pelanggran absensi sampai 150 jam, hal ini di karenakan pada saat absensi mahasiswa tersebut telah mencapai 38jam nama mahasiswa tersebut belum di coret dari daftar absensi, padahal jika kita mengacu pada peraturan yang ada pada perkuliahan di PENS-ITS ketika mahasiswa melakukan pelanggaran absensi sebanyak 38jam maka mahasiswa tersebut akan di DO (droup out), hal ini bisa di karenakan managemen tetap memasukkan data absensi kepada mahasiswa yang telah meninggal, pindah, cuti, DO atau mahasiswa tersebut mendapat perlakuan kusus dari management.

Pada analisa kedisiplinan juga di temukan pola absensi yang janggal pada kegiatan akademis di PENS-ITS yakni pada tahun 2006 sama sekali tidak ada absensi Alfa,Izin atau Sakit semua mahasiswa status absensinya adalah Hadir.

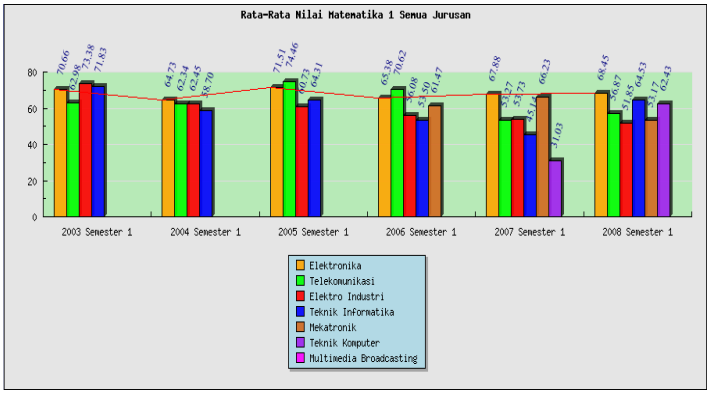
Menampilkan Report Analisa Kurikulum dan Matakuliah.



Gambar 4.11 Report analisa kurikulum



Gambar 4.12 Report Matakuliah Bahasa Inggris 1



Gambar 4.13 Report Matakuliah Matematika 1

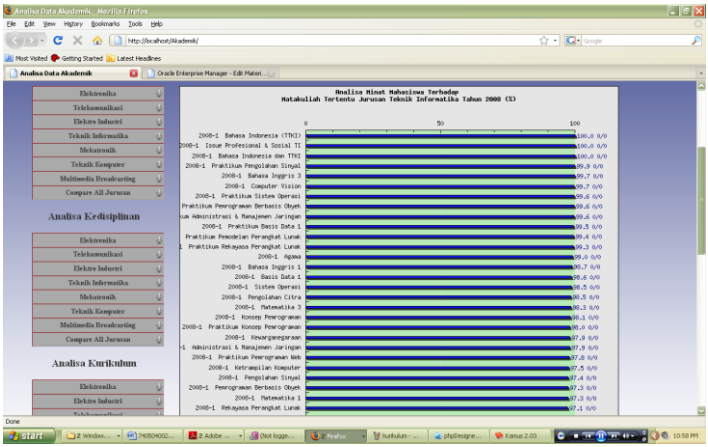
Analisa kurikulum dan matakuliah di ukur berdasarkan parameter rata-rata nilai mahasiswa pada matakuliah tertentu dalam satu semester, jika di rumuskan adalah

(Jumlah Nilai / Jumlah Mhs) x 100 %

Jumlah Nilai : total jumlah nilai angka yg di peroleh semua mahasiswa yang mengikuti matakuliah tertentu.
Jumlah Mhs : total jumlah mahasiswa yang ikut matakuliah tersebut.

Pada gambar 4.11 di atas di temukan ada sebuah matakuliah yang memiliki nilai rata-rata 100%, ini berarti setiap mahasiswa dalam mengikuti matakuliah tersebut mendapatkan nilai 100.

Menampilkan Report Analisa Minat Mahasiswa Tehadap Matakuliah Tetentu.



Gambar 4.14 Report analisa minat_matkul

Analisa minat mahasiswa terhadap matakuliah tertentu, di hitung dengan menggunakan parameter jumlah hadir mahasiswa, terhadap seluruh absensi mahasiswa, pada matakuliah tersebut, jika di rumuskan adalah sebagai berikut :

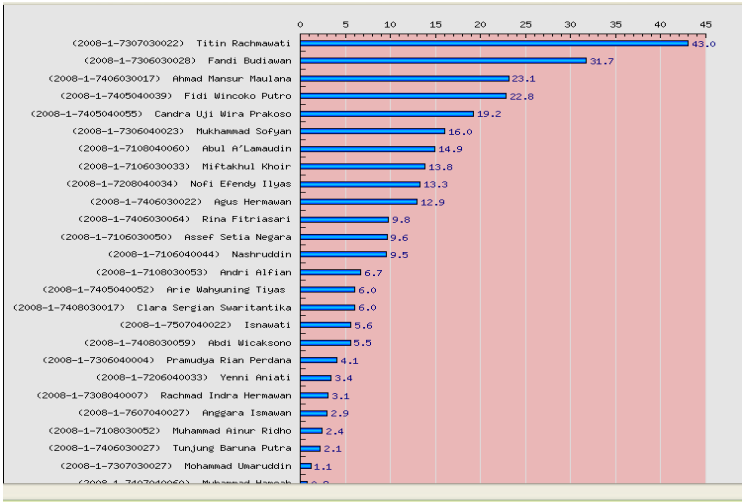
(Jumlah Absensi Hadir / Jumlah Total Absensi) x 100%

perhitungan di atas berlaku untuk tiap tiap matakuliah, yang kemudian hasilnya di rangking seperti pada Gambar 4.14

Pada analisa minat mahasiswa terhadap matakuliah tertentu yang di dasarkan pada absensi kehadiran mahasiswa ini, pada tahun 2006 terdapat ke janggalan yakni minat mahasiswa terhadap seluruh matakuliah pada tahun ini adalah 100%, hal ini

berarti tak ada satupun mahasiswa yang absen(Alfa) pada tahun 2006.

Menampilkan Report Analisa Rekomendasi Beasiswa



Gambar 4.15 Report analisa beasiswa

Gambar 4.15 menunjukkan report Analisa beasiswa dengan menggunakan tiga parameter yakni indeks prestasi, kedisiplinan dan penghasilkan orang tua, dalam perhitungannya juga menggunakan teknik fuzzy. Dalam perhitungan fuzzy setelah di peroleh nilai μ tiap variabel maka nanti yang akan di pilih adalah mahasiswa yang memiliki μ(IP cumlaud), μ(Disiplin tinggi) dan μ(Penghasilan Ortu Rendah). Dari ketiga variabel dapat di rumuskan

μ(IP cumlaud) x μ(Disiplin tinggi) x μ(Penghasilan Ortu Rendah).

Pada Gambar 4.15 tiga mahasiswa teratas yang berhak menerima beasiswa adalah Titin, Fandi dan Ahmad Mansyur jika

di hitung secara manual indeks prestasi, tingkat pelanggaran dan penghasilan orang tua dari ketiga mahasiswa tersebut adalah

Tabel 4.1 Variabel Beasiswa

Nama	IP	Jam Pelanggaran	Penghasilan_ORTU
Titin	3.43	0	500.000
Fandi	3.55	0	1.250.000
Ahmad	3.33	0	1000.000

Untuk menghitung μ dari ketiga tabel tersebut adalah menggunakan rumus :

$$\mu IP\_Istemewa[x] \begin{cases} 0 & x<=3 \\ \frac{x-3}{1} & 3=<x<=4 \\ 1 & x=4 \end{cases} \qquad \mu Disiplin[y] = \begin{cases} 0\ ; & y>=10 \\ \frac{10-y}{10}\ ; & 0=<y<=10 \\ 1\ ; & y=0 \end{cases}$$

$$\mu Kurang\_Mampu[z] = \begin{cases} 0\ ; & z>=2jt \\ \frac{2jt-z}{1,3jt}\ ; & 700rb=<z<=2jt \\ 1\ ; & z<=700rb \end{cases}$$

μIP_Titin[x]= $\frac{x - 3}{1}$ μIP_Titin[3.43]= $\frac{3.43 - 3}{1}$ = 0.43

μDisiplin_Titin[y]=1 μDisiplin_Titin[0]= 1

μK_Mampu_Titin[z]=1 μK_Mampu_Titin[500.000]=1

Rekomen_Beasiswa_Titin = *μ(IP_Istimewa) x μ(Disiplin) x μ(Kurang_Mampu) x 100%*
= 0.43 x 1 x 1 x 100%
= 43 %

Dengan perhitungan yang sama di peroleh hasil seperti pada tabel 4.2.

Tabel 4.2 Nilai µ Variabel Beasiswa

Nama	µIP	µDisiplin	µKurangMampu	Rekomendasi	Prosentase
Titin	0.43	0	1	0.43	43 %
Fandi	0.55	0	0.577	0.317	31.7 %
Ahmad	0.3	0	0.769	0.231	23.1 %

Membandingkan Kecepatan Query Datawarehouse (Skema Star) Dengan Database OLTP (Tabel Relational)

Pada pengujian ini skema star yang di gunakan adalah skema star pada analisa Kedisiplinan dan Matakuliah, skema ini dapat di lihat pada Gambar 3.6. Skema ini berisi semua data absensi mahasiswa mulai tahun ajaran 2004 sampai tahun 2008. Sedangkan pada tabel relational akses datanya ke tabel eis_absensi_mahasiswa yang merupakan tabel sumber dari tabel fakta_minat_matkul pada skema star. Kedua tabel memiliki jumlah row 1688385.

1. Mengukur Kecepatan Query Skema Star

```
set timing on
set autotrace on
select count(*) from(
SELECT p.tahun,p.semester,j.jurusan,m.nama,f.absensi
FROM f_minat_matkul f,d_minat_period p,
d_minat_jurusan   j,d_minat_mhs m
WHERE f.id_periode=p.id_periode and
f.id_jurusan=j.id_jurusan and f.nrp=m.nrp and
f.absensi is not null)
```

COUNT(*)
1688385

Elapsed: 00:00:00.89

Execution Plan

Plan hash value: 1614159527

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	38	1439 (5)	00:00:18
1	SORT AGGREGATE		1	38		
* 2	HASH JOIN		1713K	62M	1439 (5)	00:00:18
3	INDEX FAST FULL SCAN	PK_MINAT_MHS	5025	55275	5 (0)	00:00:01
* 4	HASH JOIN		1713K	44M	1404 (3)	00:00:17
5	MERGE JOIN CARTESIAN		112	896	4 (0)	00:00:01
6	INDEX FULL SCAN	PK_MINAT_JUR	8	24	1 (0)	00:00:01
7	BUFFER SORT		14	70	3 (0)	00:00:01
8	INDEX FAST FULL SCAN	PK_MINAT_PERIOD	14	70	0 (0)	00:00:01
9	BITMAP CONVERSION TO ROWIDS		1713K	31M	1370 (1)	00:00:17
10	BITMAP INDEX FAST FULL SCAN	F_MINAT_MATKUL_INDEX				

Gambar 4.16 Timing Query Skema Star

2. Mengukur kecepatan Query Pada Tabel Relational

```
set timing on
set autotrace on
select count(*) from(
select
from wahib.jurusan j,wahib.mahasiswa m,wahib.kuliah
k,wahib.eis_absensi_mahasiswa a,wahib.matakuliah mt

j.jurusan,m.nama,k.tahun,k.semester,mt.matakuliah,a.st
atus
where a.mahasiswa=m.nomor and a.kuliah=k.nomor and
k.matakuliah=mt.nomor and mt.jurusan=j.nomor and
a.status is not null)
```

COUNT(*)
1688385

Elapsed: 00:00:02.12

Execution Plan

Plan hash value: 4133344043

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	36	1688 (10)	00:00:21
1	SORT AGGREGATE		1	36		
* 2	HASH JOIN		1691K	58M	1688 (10)	00:00:21
3	INDEX FAST FULL SCAN	PK_MHS	5025	20100	4 (0)	00:00:01
* 4	HASH JOIN		1691K	51M	1654 (8)	00:00:20
* 5	HASH JOIN		7983	163K	38 (8)	00:00:01
6	NESTED LOOPS		5985	65835	14 (8)	00:00:01
7	TABLE ACCESS FULL	MATAKULIAH	5985	47880	13 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	PK_JURUSAN	1	3	0 (0)	00:00:01
9	TABLE ACCESS FULL	KULIAH	7983	79830	22 (0)	00:00:01
* 10	TABLE ACCESS FULL	EIS_ABSENSI_MAHASISWA	1691K	17M	1587 (6)	00:00:20

Gambar 4.17 Timing Query Tabel Relational

Dari gambar 4.16 dan gambar 4.17 menunjukkan bahwa, semakin banyak relasi antar tabel, waktu yang di perlukan untuk menampilkan informasi kepada user, juga bertambah. Relasi tabel pada skema star yang terpusat pada satu tabel fakta yang di kelilingi beberapa tabel dimensi terbukti memiliki kecepatan query yang lebih baik.

.

-- Halaman ini sengaja dikosongkan --

BAB V
PENUTUP

5.1. KESIMPULAN

Berdasarkan hasil uji coba dan analisa sistem dapat di simpulkan bahwa :

- Analisa data akademik berbasis data warehouse di PENS-ITS dapat menjadi salah satu solusi untuk mengatur dan mengelola data akademik menjadi asset yang berperan dalam pengambilan keputusan, pembuatan pelaporan, hingga analisis ke depan.
- Dari hasi pengukuran menunjukkan performance proses query data warehouse (Skema Star) dalam menampilkan informasi kepada user lebih baik dari pada Tabel Relational yang banyak memiliki relasi antar tabel.

5.2. SARAN

- Sumber data sangat penting dalam pembangunan data warehouse ini untuk memperoleh hasil analisa yang tepat dan akurat. Untuk itu kedepan di harapkan PENS-ITS dapat menyajikan informasi data yang lebih baik lagi. seperti data penghasilan orang tua.

-- Halaman ini sengaja dikosongkan --

DAFTAR PUSTAKA

[1] Paulraj Ponniah 2001, *Data Warehousing Fundamental*.Wiley-Interscience Publication.

[2] Jeffrey A. Hoffer, Mary B. Prescott, dan Fred R.McFadden. 2005. *Moderen Database Management*. Sevent Edition. Prentice Hall.

[3] Greenwald, Stackowiak,Dodge, Klein, shapiro dan Chelliah. 2005. *Professional Oracle Programming*.

[4] Amborowati Armadyah. 2008. *Perancangan Data Warehouse Pada Perpustakaan STIMIK AMIKOM Yogyakarta*.

[5] Arina Azima dan Yudgo Giri sucahyo. 2007. *Penggunaan data warehouse dan data mining untuk data akademik*.Setudi kasus pada universitas nacional.

[6] Tessy Badriyah. *Mendesain Data Warehouse*

[7] http://forums.oracle.com/forums/forum.jspa?forumID=75&start=0. *Forum Oracle PL/SQL*

[8] http://www.diskusiweb.com/forumdisplay.php?fid=18. *Forum Oracle Database Server*

[9] http://aditus.nu/jpgraph/documentation.php. *JpGrah Documentation*.

-- Halaman ini sengaja dikosongkan --

DAFTAR RIWAYAT HIDUP

Nama : Aminul Wahib
Alamat : RT 02 RW 01 Desa Besur Kec. Sekaran
Kab. Lamongan
No. HP : 085646493936
Email : wahib_it05@yahoo.com.

Riwayat Pendidikan :

- TK Besur-Sekaran (1990 – 1992)
- SDN Besur-Sekaran (1992 – 1998)
- MTS BU Besur-Sekaran(1998 – 2001)
- SMAN I Lamongan (2001 – 2004)
- D4 Teknik Informatika PENS – ITS (2005 – 2009)

Penulis telah mengikuti Seminar Tugas Akhir pada tanggal 24 Juli 2009 sebagai salah satu syarat untuk memperoleh gelar sarjana sain terapan (SST).