OPTIMASI *QUERY* PADA SISTEM INFORMASI PENCATATAN AKTIFITAS PERUBAHAN DATA NILAI MAHASISWA

(Skripsi)

Oleh

Nilaliliana Prihatin

JURUSAN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
2017

ABSTRACT

QUERY OPTIMIZATION FOR INFORMATION SYSTEM FOR STUDENTS GRADE DATA CHANGE ACTIVITY RECORDING

BY

NILALILIANA PRIHATIN

Query optimization using the index can improve query performance. The prior research explained that relational operators combined with indexing strategy in sub query has a better performance compared to using the other methods. This research using indexing strategy and nested query for activity listing information system for student grades data changes. This system is used to listing activites done by student grades manager. The process of system testing conducted several experiments such as comparing data using index and data that did not use the index (table scan) to several conditions, namely the data on the amount of 517, 980, 2369, and 4221. Nested query testing and regular query or query were tested directly using the same amount of data and applied on subjects data search. Indexing strategy that is applied to the big amount of data will generate faster execution time than a search without indexing strategy, in contrary to the small amount of data indexing strategy does not really affect the search process. The use of nested query in the subjects data search will be faster than the time pencarianya regular query or query directly.

Key word: query optimization, index, nested query.

ABSTRAK

OPTIMASI *QUERY* PADA SISTEM INFORMASI
PENCATATAN AKTIFITAS PERUBAHAN DATA NILAI MAHASISWA

Oleh

NILALILIANA PRIHATIN

Optimasi *query* menggunakan *index* dapat meningkatkan performa *query*. Penelitian sebelumnya menjelaskan bahwa *operator* relasional dikombinasikan dengan strategi pengindeksan di sub *query* memiliki kinerja yang lebih baik dibandingkan dengan menggunakan metode lainnya. Penelitian ini menerapkan strategi pengindeksan dan *query nested* pada sistem informasi pencatatan aktifitas peubahan data nilai mahasiswa. Sistem informasi pencatatan aktifitas perubahan data nilai mahasiswa merupakan sistem informasi yang digunakan untuk mencatat aktifitas yang dilakukan oleh pengelola nilai. Proses uji coba sistem dilakukan beberapa kali percobaan diantaranya adalah membandingkan data yang menggunakan *index* dan data yang tidak menggunakan *index (table scan*) pada beberapa kondisi data yaitu pada jumlah data 517, 980, 2369, dan 4221. Pengujian *query nested* dan *query* biasa atau *query* langsung diuji menggunakan jumlah data yang sama dan diterapkan pada pencarian data mata kuliah. Strategi pengindeksan yang diterapkan pada jumlah data yang besar akan menghasilkan waktu eksekusi yang lebih cepat dibandingkan pencarian tanpa strategi pengindeksan, sebaliknya pada data yang kecil strategi pengindeksan tidak begitu mempengaruhi proses pencarian. Penggunaan *query nested* dalam pencarian data mata kuliah akan lebih cepat waktu pencarianya dibandingkan dengan *query* biasa atau *query* langsung.

Kata kunci: optimasi *query, index, query nested.*

OPTIMASI *QUERY* PADA SISTEM INFORMASI PENCATATAN AKTIFITAS PERUBAHAN DATA NILAI MAHASISWA

Oleh
NILALILIANA PRIHATIN

Skripsi

Sebagai Salah Satu Syarat untuk Memperoleh Gelar

SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam

JURUSAN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2017

Judul Skripsi	:	**OPTIMASI *QUERY* PADA SISTEM INFORMASI PENCATATAN AKTIFITAS PERUBAHAN DATA NILAI MAHASISWA**
Nama Mahasiswa	:	Nilaliliana Prihatin
Nomor Pokok Mahasiswa	:	1217051048
Jurusan	:	Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Dwi Sakethi, M.Kom.	**Febi Eka Febriansyah, M.T.**
NIP.196806111199802 1 001	NIP. 19800219 200604 1 001

2. Mengetahui,

Ketua Jurusan Ilmu Komputer,

Dr. Ir. Kurnia Muludi, M.S.Sc

NIP.196406161989021001

MENGESAHKAN

1. Tim Penguji

Ketua : **Dwi Sakethi, M.Kom.** .....

Sekretaris : **Febi Eka Febriansyah, MT.** .....

Penguji
Bukan Pembimbing : **Wamiliana, Dra., MA, Ph.D.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Prof. Warsito, S.Si., D.E.A., Ph.D.
NIP. 197102121995121001

Tanggal Lulus Ujian Skripsi : **23 Januari 2017**

PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya yang berjudul **"Optimasi *Query* Pada Sistem Informasi Pencatatan Aktifitas Perubahan Data Nilai Mahasiswa"** merupakan karya saya sendiri dan bukan hasil karya orang lain. Semua tulisan yang tertuang di skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti bahwa skripsi saya merupakan hasil penjiplakan atau dibuat orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, Januari 2017

Nilaliliana Prihatin
NPM. 1217051048

RIWAYAT HIDUP

Penulis dilahirkan pada tanggal 13 Juni 1994 di Desa Budi Lestari, Kecamatan Tanjung Bintang, Kabupaten Lampung Selatan, Lampung. Penulis adalah anak ketiga dari empat bersaudara dari pasangan Bapak Hasanudin dan Ibu Sudaryati.

Penulis menempuh pendidikan formal pertama kali di SD Negeri 2 Budi Lestari pada tahun 2000 sampai tahun 2004, Kemudian penulis pindah dan melanjutkan dan menyelesaikan pendidikan dasarnya di SD Negeri 1 Setia Bumi, Kecamatan Gunung Terang, Kabupaten Tulang Bawang Barat tahun 2006. Pendidikan menengah pertama di SMP Lestari Tanjung Bintang diselesaikan penulis pada tahun 2009. Pendidikan menengah atas di SMA Negeri 1 Tanjung Bintang diselesaikan penulis pada tahun 2012.

Pada tahun 2012, penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung yang masuk melalui jalur SBMPTN. Pada bulan Januari – Febuari tahun 2015, penulis melakukan kerja praktik di Lembaga Penelitian dan Pemberdayaan Masyarakat Universitas Lampung. Juli – September tahun 2015, penulis melaksanakan Kuliah Kerja Nyata (KKN) di Desa Setia Agung, Kecamatan Gunung Terang,

Kabupaten Tulang Bawang Barat. Selama kuliah penulis terdaftar sebagai Sekretaris Bidang keilmuan Himpunan Mahasiswa Jurusan Ilmu Komputer pada periode 2013-2014.

MOTTO

“Allah tidak membebani seseorang
melainkan sesuai dengan kesanggupanya”
(Al-Baqoroh : 286)

“Hasil tidak akan pernah menghianati proses”
(Nilaliliana Prihatin)

PERSEMBAHAN

Kupersembahkan karya ini untuk :

Allah SWT sang maha pencipta yang telah memberikan ridho-Mu, memberikan kemudahan serta kelancaran sehingga salah satu perjalanan hidup ini dapat terselesaikan.

Mamak dan Bapak, yang telah mendidik, merawat, membesarkan, mendukung dan selalu memberikan semangat. Terima kasih atas cinta, kasih sayang, perhatian, serta pengorbanan yang telah kalian berikan untukku.

Ayuk Linda, Ayuk Dian, dan Adikku, Nita, serta Seluruh Keluargaku, yang selalu memberikan semangat dan dukungannya.

Keluarga Ilmu Komputer 2012,
Terima kasih atas dukungan dan doanya selama ini.

Almamater Tercinta,
Universitas Lampung.

SANWACANA

Puji syukur penulis haturkan kehadirat Allah *subhanahuwata'ala* atas berkat rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan penulisan skripsi ini. Skripsi ini disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung dengan Judul skripsi adalah "Optimasi *Query* Pada Sistem Informasi Pencatatan Aktifitas Perubahan Data Nilai Mahasiswa".

Dalam kesempatan ini penulis ingin menyampaikan terima kasih kepada semua pihak yang telah memberikan bantuan, dukungan dan semangat sehingga penulis dapat menyelesaikan skripsi ini. Karena dalam penulisan skripsi ini penulis banyak menghadapi kendala dan masalah. Ungkapan terima kasih penulis ucapkan kepada:

1. Allah SWT yang telah memberikan rahmat dan karunia-Nya selama penulis melakukan penelitian sampai dengan terselesaikannya skripsi ini.
2. Bapak Dwi Sakethi, S.Si, M.Kom selaku Pembimbing Utama yang telah membimbing penulis dan memberikan semangat kepada penulis sampai terselesaikannya skripsi ini.
3. Bapak Feby Eka Febriansyah, M.T. selaku Pembimbing Kedua yang telah memberikan saran, motivasi, semangat, serta membimbing penulis selama proses pembuatan skripsi.
4. Ibu Wamiliana Dra., MA, Ph.D selaku Pembahas yang telah memberikan saran, serta masukan-masukan yang bermanfaat selama proses pembuatan skripsi.
5. Seluruh Dosen dan Staf Jurusan Ilmu Komputer FMIPA Universitas Lampung yang telah membimbing, memberikan ilmu, membantu, serta

sebagai motivator bagi penulis, selama penulis menempuh pendidikan di Jurusan Ilmu Komputer.

6. Mamak dan Bapak yang selalu mendukung baik moril dan materil serta doa yang tak hentinya untuk mengiringi penulis.
7. Bibik, Oom, Haqqi, Intan dan Mbah terima kasih untuk waktu, kesabaran yang tak terhingga serta tempat yang telah disediakan.
8. Ayuk Linda, Ayuk Dian, Kak Rasna, Kak John, Nita serta Icha dan Dina yang selalu memberikan doa dan dukungan serta selalu menghibur penulis setiap hari.
9. Sahabat tercinta dan terkasih ku Dian, Cindona, Muji, Astuti, dan Taqiya yang selalu memberikan warna-warni di kehidupan penulis.
10. Sahabat-sahabat seperjuangan Nurul, Erika, Riska, Rani, Erlina, Nafi, Anita, yuni, Niko, Arif, Roni, Dipa serta rekan-rekan angkatan 2012 lainnya dan D3 Informatika 2012 Jurusan Ilmu Komputer yang menjadi teman seperjuangan untuk menyelesaikan masa studi di Jurusan Ilmu Komputer.
11. Almamater Tercinta.

Penulis menyadari bahwa dalam skripsi ini masih terdapat banyak kekurangan karena masih terbatasnya kemampuan, pengalaman, dan pengetahuan penulis. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan sebagai bahan perbaikan untuk tulisan-tulisan yang akan datang. Penulis berharap semoga skripsi ini dapat bermanfaat bagi semua pihak.

Bandar Lampung, Januari 2017

Nilaliliana Prihatin

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sistem informasi pencatatan aktifitas perubahan data nilai mahasiswa merupakan sistem informasi yang melakukan perekaman data dari semua aktifitas perubahan yang pernah dilakukan pada data nilai mahasiswa. Perubahan yang dimaksud meliputi *edit* dan *delete*. *Edit* merupakan aktifitas perubahan isi atau nilai, sedangkan *delete* merupakan perubahan dari data yang berisi nilai ke data yang kosong.

Sistem informasi pencatatan perubahan data nilai mahasiswa membantu mengevaluasi dan analisis aktifitas yang pernah dilakukan oleh pengelola nilai mahasiswa. Sistem informasi yang dapat mencatat aktifitas perubahan data dibutuhkan untuk membantu pengelola nilai mengingat aktifitas yang dilakukan selama proses perubahan data pada nilai mahasiswa. Sistem informasi pencatatan aktifitas perubahan data nilai diharapkan dapat membantu pencocokan data pencatatan aktifitas yang ada di sistem pencatatan otomatis pada sebuah sistem informasi akademik.

Pencatatan otomatis yang dilakukan sistem informasi akademik akan berjalan ketika seorang pengelola nilai masuk kedalam sistem melalui proses *login* dan melakukan perubahan pada data. Perubahan data yang otomatis direkam tabel *trigger* perlu dicocokkan dengan sistem informasi pencatatan aktifitas perubahan data nilai karena kebocoran rahasia *username* dan *password* bisa saja terjadi serta digunakan oleh orang yang tak berhak memiliki hak akses dengan menggunakan hak akses yang dimiliki orang lain.

Proteksi pencatatan aktifitas yang disediakan pada *database* yang ada di sistem informasi akademik dikhawatirkan tidak membantu secara efektif. Adanya kemungkinan penyalahgunaan *login* membuat kesalahpahaman pengguna akses sistem. Untuk mengendalikan ancaman perubahan-perubahan nilai yang tidak dinginkan perlu adanya sistem informasi yang dapat mencatat perubahan data dan mempunyai waktu pencarian yang cepat untuk mendapatkan informasi dengan waktu sedikit.

Pada sistem informasi, perubahan data sangat mungkin dilakukan oleh pengelola nilai. Perubahan data dilakukan ketika data dirasa kurang sesuai dengan keadaan yang semestinya. Perekaman aktifitas perubahan data yang dilakukan akan disimpan di sebuah *database* dan data yang disimpan akan sangat banyak. Pertambahan data merupakan hal yang wajar, pertambahan data dalam beberapa periode waktu akan meningkat sangat cepat. Ketika sebuah *database* memiliki data yang sangat banyak maka proses akses ke *database* akan menurun. Proses akses dan pencarian data yang diperlukan sangat membantu ketika data dalam sistem menjadi sangat banyak. Namun,

efisiensi waktu akses data dalam proses pencarian di data yang sangat banyak akan berkurang. Proses pencarian data membutuhkan metode yang dapat membuat proses pencarian menjadi lebih efisien dan efektif pada data berukuran besar.

Contoh dari persoalan pencarian data adalah penggunaan *"select"*. Penggunaan *"select"* dalam pencarian berukuran 517 data memerlukan waktu sekitar 0,0039 detik, sedangkan untuk data sebanyak 4000 data memerlukan waktu sebanyak 0,0146 detik. Berdasarkan contoh dari persoalan tersebut menunjukkan bahwa semakin besar data, waktu untuk proses pencarian akan semakin lama, begitu pula sebaliknya.

Sistem informasi yang dapat mencatat segala aktifitas perubahan data nilai mahasiswa diperlukan untuk kepentingan pengelola nilai. Waktu akses yang dibutuhkan harus bisa diminimalisir menggunakan teknik pengoptimalan *query* untuk bisa mendapatkan informasi dari sistem informasi pencatatan aktifitas secara cepat.

1.2.Rumusan Masalah

Berdasarkan latar belakang yang diuraikan di atas, rumusan masalah yang dibuat adalah sebagai berikut.

1. Bagaimana membuat sistem pencatatan aktifitas.
2. Bagaimana meminimalisir waktu pencarian data menggunakan *query optimization.*

1.3.Batasan Masalah

Adapun batasan masalahnya yaitu sistem ini:

1. Administrasi melakukan pencatatan aktifitas yang dilakukan oleh pengelola nilai dalam lingkup pribadi.
2. Optimasi dilakukan pada proses pencarian data.
3. Sistem berbasis web, menggunakan bahasa pemrograman PHP, dan menggunakan pengolah basis data *Xampp.*

1.4.Tujuan

Tujuan penelitian ini adalah menghasilkan sistem informasi pencatatan aktifitas yang dapat membantu pengawasan perubahan pada *database* dan mengimplementasikan *query optimization* pada sistem yang dibuat.

1.5.Manfaat

1. Memberikan kemudahan untuk analisis aktifitas yang telah dilakukan oleh pengelola nilai.
2. Mempermudah pengelola nilai mengingat perubahan nilai terakhir yang dilakukan.
3. Mengefisienkan waktu pencarian dengan data yang sangat banyak.

BAB II

TINJAUAN PUSTAKA

2.1 Query optimizer

Optimasi merupakan suatu langkah untuk mengoptimalkan waktu menjadi lebih efisien. Ketika sebuah *query* diberikan pada sistem *database*, optimasi penting dilakukan untuk memilih strategi yang efisien untuk mengevaluasi ekspresi relasi yang ditentukan. *Query optimization* adalah suatu proses untuk menganalisis *query*, menentukan sumber-sumber apa saja yang digunakan oleh *query* tersebut dan apakah penggunaan dari sumber tersebut dapat dikurangi tanpa merubah *output*. *Query optimization* dapat juga dikatakan sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu *query* untuk membuat evaluasi tersebut menjadi lebih efektif, mencakup beberapa teknik seperti transformasi *query* ke dalam bentuk logika yang sama, memilih jalan akses yang optimal dan mengoptimumkan penyimpanan data. Tujuan dari *query optimization* adalah menemukan jalan akses yang termurah untuk meminimumkan total waktu pada saat proses sebuah *query* (Maulani dkk, 2015).

Query optimizer adalah bagian dari DBMS (*Database Management System*) yang berfungsi mengoptimasi *query*. Proses yang biasanya terjadi dalam

optimizer adalah *optimizer* memeriksa semua ekspresi-ekspresi aljabar yang sama yang diberikan *query* dan memilih salah satunya yang memiliki harga taksiran paling rendah. Tugas dari *optimizer* adalah untuk mentransformasikan inisial ekspresi *query* ke dalam sebuah rencana evaluasi yang menghasilkan *record* yang sama.

Keuntungan *optimizer* adalah dapat mengakses semua informasi statistik dari sebuah *database*. Selain itu *optimizer* juga dapat dengan mudah untuk melakukan optimisasi kembali apabila informasi statistik sebuah *database* berubah dan *optimizer* dapat menangani strategi yang berbeda-beda dalam jumlah besar yang tidak mungkin dilakukan oleh manusia. *Input* dari *optimizer* adalah sebuah *tree* yang sudah mengalami proses *parsing* di dalam *query* parser. *Tree* tersebut biasanya disebut dengan *parse tree*. Sedangkan *output* dari *optimizer* adalah berupa rencana eksekusi (*execution plan*) yang siap untuk dikirimkan ke dalam kode *query generator* dan *query processor* untuk diproses untuk mendapatkan hasil akhir dari *query* tersebut (Ermattita, 2009).

2.2 *Query performent*

Menurut Oktavia dan Sujarwo (2014) penggunaan *index* pada tabel dapat meningkatkan performa *query*. Tabel yang digunakan dalam percobaan menggunakan dua pengindeksan. Eksperimen yang telah dilakukan menunjukkan bahwa menggunakan *operator* relasional dikombinasikan dengan strategi pengindeksan di sub *query* memiliki kinerja yang lebih baik

dibandingkan dengan menggunakan metode yang sama tanpa strategi pengindeksan dan juga metode lainnya.

Optimalisasi sub *query* dapat dilakukan dengan menerapkan strategi pengindeksan ke tabel sesuai dengan kondisi yang digunakan dalam *query*. Untuk aplikasi bisnis yang membutuhkan waktu proses yang cepat dalam mengambil data dari *database* dari waktu pemrosesan dalam menyimpan tanggal, lebih baik untuk mengindeks diterapkan di semua tabel yang digunakan oleh aplikasi, dan juga semua sub *query* lebih baik menggunakan operator relasional. Sebaliknya, untuk aplikasi bisnis yang diperlukan waktu proses yang cepat dalam menyimpan data dari waktu proses yang cepat dalam mengambil data, lebih baik untuk tidak diterapkan pengindeksan dan untuk sub *query* lebih baik menggunakan *IN* atau *EXISTS*, karena jika pengindeksan diterapkan, waktu proses untuk menyimpan data akan meningkat di setiap operasi *INSERT* yang juga otomatis memperbarui indeks dalam tabel terkait.

2.3 Index

Index adalah objek pada MySQL yang berisi data yang terurut dari nilai-nilai pada satu atau lebih *field* dalam suatu tabel. Penggunaan *index* pada *database* merupakan salah satu teknik pembuatan *database* yang baik. Hal ini terutama sangat berguna pada implementasi *database* dengan skala VLDB (*Very Large Database*) atau OLDB (*Online Large Database*). Saat *database* dibuat tanpa menggunakan *index*, maka kinerja *server database* dapat menurun secara

drastis. Hal ini dikarenakan *resource* komputer banyak digunakan untuk pencarian data atau pengaksesan *query* SQL dengan metode *table-scan.*

Cecilia dan Mihai (*2011*) mengemukakan bahwa *index* pada tabel memungkinkan *SQL server* untuk mendapatkan hasil pencarian tanpa mencari seluruh data di dalam tabel. Tabel dengan menggunakan *index* adalah cara terbaik untuk mengurangi *logical read and disk input/output* karena menyediakan mekanisme pencarian terstruktur. Ada dua cara berbeda untuk mendefinisikan *index* yaitu:

a. *Like a dictionary*: kamus merupakan daftar kata-kata yang disusun menurut abjad. Indeks didefinisikan seperti kamus adalah sekumpulan data di-*order*

   secara *leksikografi*. Untuk alasan ini pencarian di indeks tidak akan mencakup semua baris tapi akan lebih mudah berdasarkan data yang di-*order*.

b. *Like a book index*: pendekatan ini membuat indeks tidak akan mengubah tata letak data dalam tabel, tetapi hanya seperti indeks buku akan posisi data dalam tabel untuk posisi yang sesuai dalam tabel. Indeks didefinisikan dengan cara ini akan berisi data di kolom diindeks dan jumlah baris yang sesuai dari data.

Cara penulisan untuk membuat *index* dalam *database* dapat dilihat seperti berikut:

```
CREATE UNIQUE INDEX index_name
ON tabel_name (column1, column2, ….);
```

Kode 2.1. Kode Program Pembuatan *Index*.

Menurut Cecilia dan Mihai (2011) terdapat dua jenis *index* yang didefinisikan yaitu:

a. *Clustered index*

   Secara fisik *record* disusun sesuai dengan *index*. *Clustered index* hanya dapat diterapkan sebanyak satu kali pada satu tabel dan *primary key* pada sebuah tabel akan menjadi *clustered index* pada tabel tersebut. Dapat dikatakan *clustered index* tersebut adalah *primary key* dari sebuah tabel. Penggunaan *clustered index* akan lebih baik dibanding *non-clustered index* pada saat memperbaharui dan menghasilkan data.

b. *Non clustered index*

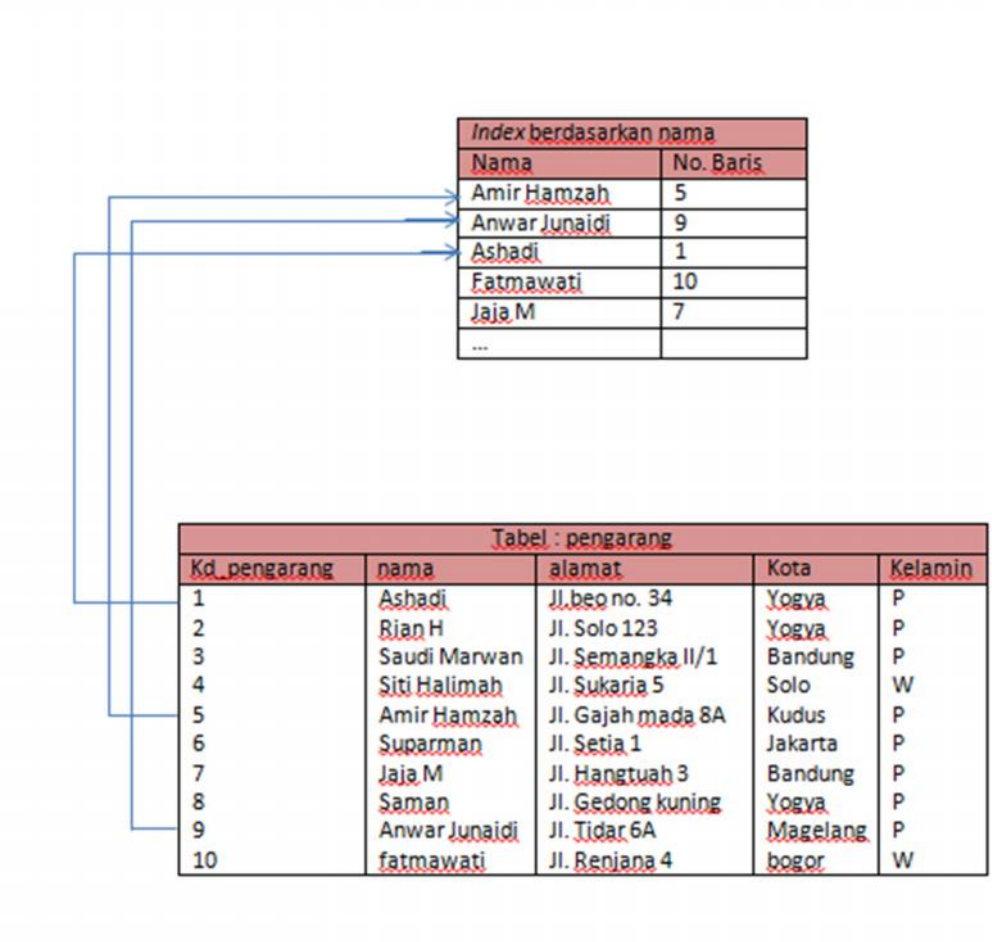
   Secara fisik susunan *record* tidak berpengaruh. *Non clustered index* menyimpan pointer yang menunjuk ke *record* dari tabel dengan data yang terpisah dengan *index*. Sebuah tabel dapat memiliki beberapa *non-clustered index* tetapi hanya memiliki satu *clustered index* yang digunakan pada sebuah *database*. *Non clustered index* akan selalu bergantung pada *clustered index* pada sebuah *database*. *Non clustered index* lebih baik diterapkan pada kolom-kolom yang juga sering digunakan oleh pengguna pada saat pencarian data.

Tujuan menciptakan indeks adalah:

1. *Index* dapat meningkatkan kinerja.
2. *Index* menjamin bahwa suatu kolom bersifat unik.

Dengan adanya *index*, pencarian suatu data yang didasarkan kolom yang di*index* akan dapat dilakukan dengan cepat. Namun, penggunaan *index* juga mempunyai kekurangan yaitu dapat memperlambat proses

penambahan dan penghapusan baris pada tabel, karena pada saat terjadi penambahan atau penghapusan baris, *index* perlu diperbaharui (Kadir, 2001)



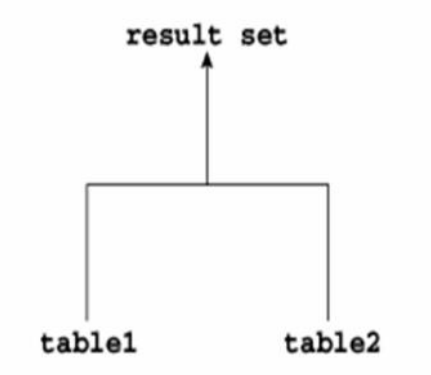
Gambar 2.1 Gambaran *Index* Menurut Nama (Kadir, 2001).

2.4 *Join*s

Join tabel adalah penggabungan tabel-tabel menggunakan *query* yang dilakukan melalui kolom/*key* tertentu yang memiliki nilai terkait untuk mendapatkan satu set data dengan informasi lengkap. Pengaksesan data atau pencarian data dengan menggunakan *Query* atau *Join* pada *database* perlu

memperhatikan ketepatgunaan implementasi dari data itu sendiri serta waktu prosesnya.

Join tree dapat dilihat pada Gambar 2.2:



Gambar 2.2. *Join Tree*

Jenis jenis *join* yang umum digunakan:

a. *Inner join*

Inner join berfungsi untuk menampilkan satu atau lebih *field* pada dua atau lebih tabel yang berbeda, dengan mengacu kepada sebuah *field* yang memiliki data yang sama.

```
SELECT (select_list) FROM TABELA A INNER JOIN
TABELB B ON A.key=B.Key;
```

Kode 2.2 Kode Program *Inner Join*

b. *Natural Join*

Operasi *natural join* adalah operasi *equi join* yang memiliki kesamaan dalam semua *field* yang memiliki nama yang sama dalam tabel R dan tabel S. Syarat penggunaan natural join adalah dua tabel harus mempunyai minimal satu nama kolom atau *field* dan tipe data yang sama.

c. *Straight join*

Straight join merupakan operator MySQL yang digunakan untuk menggabungkan dua tabel secara menyeluruh tanpa ada kaitan data. Perintah STRAIGHT_JOIN sama seperti JOIN, namun STRAIGHT_JOIN tidak mengenal klausa WHERE.

```
SELECT * FROM customers STRAIGHT_JOIN products
```

Kode 2.3 Kode Program *Straight Join*

Meskipun tabel *customers* dengan tabel *products* tidak ada relasi (kesamaan data) maka dengan perintah *STRAIGHT_JOIN* keduanya tetap bisa digabungkan.

d. *Outer join*

1. *Left join*

Left outer join berfungsi untuk menampilkan satu atau lebih *field* pada dua atau lebih tabel yang berbeda, dengan mengacu kepada sebuah *field* yang memilki data yang sama. Data yang ditampilkan adalah data yang ada di dalam tabel sebelah kiri atau pertama kali disebutkan dalam perintah.

```
SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON
A.Key = B.Key;
```

Kode 2.4 Kode Program *Left Join*

atau

```
SELECT <select_list> FROM TableA A LEFT OUTER JOIN
TableB B ON A.Key = B.Key;
```

Kode 2.5 Kode Program *Left Outer Join*

2. *Right join*

Right outer join berfungsi menampilkan satu atau lebih *field* pada dua atau lebih tabel yang berbeda, dengan mengacu kepada sebuah *field* yang memiliki data yang sama. Data yang ditampilkan adalah data

yang ada di tabel di sebelah kanan atau yang urutan kedua disebutkan pada perintah SQL.

```
SELECT <select_list> FROM TableA A RIGHT JOIN TableB B
ON A.Key = B.Key;
```

Kode 2.6 Kode Program *Right Join*

atau

```
SELECT <select_list> FROM TableA A RIGHT OUTER JOIN
TableB B ON A.Key = B.Key;
```

Kode 2.7 Kode Program *Right Outer Join*

2.5 Nested query

Nested query atau *query* bersarang adalah *query* yang memiliki *query* lain di dalamnya. *Sub Query* merupakan pernyataan *Select* yang merupakan bagian dari pernyataan *Insert, Select*. *Nested query* digunakan untuk menangani masalah dalam *query* yang kompleks, bahkan nilai yang akan dilakukan perintah *select* atau *insert* tidak diketahui (Maulani dkk, 2015).

2.6 Pencatatan

Menurut Kamus Besar Bahasa Indonesia (KBBI) catat atau mencatat adalah menuliskan sesuatu untuk peringatan (dalam sebuah buku catatan); menuliskan apa yang sudah ditulis atau diucapkan orang lain; menyalin; memasukkan ke dalam daftar; memperoleh atau mencapai hasil; memasukkan (suara atau ujaran) ke dalam pita perekam. Dalam KBBI pencatatan didefinisikan sebagai proses, cara perbuatan mencatat.

2.7 Sistem

Secara umum sistem dapat didefinisikan sebagai sekumpulan hal atau kegiatan atau elemen atau sub sistem yang saling bekerja sama atau dihubungkan dengan cara-cara tertentu sehingga membentuk satu kesatuan untuk melaksanakan suatu fungsi guna mencapai suatu tujuan (Sutanta, 2003).

2.8 Sistem informasi

Sistem informasi dapat didefinisikan sebagai berikut:

1. Suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan informasi.
2. Sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan dan/atau untuk mengendalikan organisasi.
3. Suatu sistem di dalam organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung operasi bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan laporan yang diperlukan (Ladjamudin, 2013).

2.9 Tujuan Sistem Informasi

Tujuan dari sistem informasi adalah menghasilkan informasi. Informasi adalah data yang di oleh menjadi bentuk yang berguna bagi para pemakainya. Informasi harus didukung oleh 3 pilar sebagai berikut:

a. *Relevance*

b. *Timeliness*

c. *Accurate.*

Output yang tidak didukung ketiga pilar ini tidak dapat dikatakan sebagai informasi yang berguna (Jogiyanto, 2003).

2.10 Elemen sistem

Beberapa elemen yang membentuk sebuah sistem menurut Kadir (2003) diantaranya adalah sebagai berikut:

1. Tujuan

   Setiap sistem memiliki tujuan (*goal*), dimana tujuan antara satu sistem dengan sistem lain berbeda-beda. Tujuan inilah yang menjadi pemotivasi yang mengarahkan sistem. Begitu pula yang berlaku pada Sistem Informasi. Walaupun begitu tujuan yang umum ada tiga macam yaitu:

   a. Untuk mendukung fungsi kepengurusan manajemen.
   b. Untuk mendukung pengembalian keputusan manajemen.
   c. Untuk mendukung operasi perusahaan.

2. Masukan (*input*)

   Masukan (*input*) sistem adalah segala sesuatu yang masuk kedalam sistem dan selanjutnya menjadi bahan untuk diproses. Masukan dapat berupa hal-hal berwujud (tampak secara fisik) maupun yang tidak tampak. Contoh: masukan yang berwujud adalah informasi.

3. Proses

   Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna, misalnya berupa informasi dan produk.

4. Luaran (*output*)

   Luaran merupakan hasil dari pemrosesan. Pada sistem informasi luaran bisa berupa informasi, saran, cetakan laporan, dan sebagainya.

5. Mekanisme Pengendalian dan Umpan Balik

   Mekanisme pengendalian (kontrol mekanisme) diwujudkan dengan menggunakan umpan balik (*feedback*), yang menciptakan luaran. Umpan balik ini digunakan untuk mengendalikan baik masukan maupun proses. Tujuannya adalah untuk mengatur agar sistem berjalan sesuai dengan fungsinya.

6. Batas

   Batasan (*boundary*) sistem adalah pemisah antara sistem dan daerah luar sistem (lingkungan). Batasan sistem menentukan konfigurasi, ruang lingkup atau kemampuan sistem.

7. Lingkungan

   Lingkungan adalah segala sesuatu yang berada diluar sistem. Lingkungan bisa berpengaruh terhadap operasi sistem dalam arti bisa merugikan atau menguntungkan sistem itu sendiri.

2.11 *Database*

Secara umum, *database* atau basis data berarti koleksi data yang saling terkait. Secara praktis, basis data dapat dianggap sebagai suatu penyusun

data yang terstruktur yang disimpan dalam media pengingat (*hard disk*) yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat (Kadir, 2008).

Pengguna sistem basis data dapat melakukan berbagai operasi antara lain:

a. Menambahkan *file* baru ke sistem basis data .
b. Mengosongkan berkas.
c. Menyisipkan data kesuatu berkas.
d. Mengambil data yang ada di suatu berkas.
e. Mengubah data pada suatu berkas.
f. Menghapus data pada suatu berkas.
g. Menyajikan suatu informasi yang diambil dari sejumlah berkas.

2.12 MySQL

MySQL didefinisikan sebagai sistem manajemen *database*. *Database* merupakan struktur penyimpanan data untuk menambah, mengakses dan memproses data yang disimpan dalam sebuah *database*. Selain itu MySQL dapat dikatakan sebagai basis data terhubung (RDBMS). Server *database* MySQL mempunyai kecepatan akses tinggi, mudah digunakan dan andal. MySQL dikembangkan untuk menangani *database* yang besar secara cepat dan telah sukses digunakan. Fitur utama MySQL (Kustiahningsih dan Anamisa, 2011) adalah:

1. Dapat bekerja dalam berbagai *platform.*
2. Menyediakan mesin penyimpan transaksi dan non transaksi.
3. Mempunyai *library* yang dapat ditempelkan pada aplikasi yang berdiri sendiri sehingga aplikasi tersebut dapat digunakan pada komputer yang

tidak mempunyai jaringan dan mempunyai sistem *password* yang fleksibel dan aman.

4. Dapat menangani basis data dalam skala besar.

2.13 *Trigger*

Trigger adalah blok SQL atau prosedur yang berhubungan dengan tabel, *view*, skema atau *database* yang dijalankan secara *implicit* pada saat terjadi sebuah *event*. *Trigger* berisi program yang dihubungkan dengan suatu tabel atau *view* yang secara otomatis melakukan suatu aksi ketika suatu baris di dalam tabel atau *view* dikenai oprerasi *insert*, *update* atau *delete*. *Trigger* dibuat sesuai dengan keperluan (Kustiahningsih dan Anamisa, 2011).

a. Waktu menjalankan *trigger*

   Waktu menjalankan *trigger* dibagi menjadi dua yaitu:

   a. *Before: trigger* diaktifkan sebelum dihubungkan suatu operasi.
   b. *After: trigger* diaktifkan setelah dihubungkan dengan suatu operasi.
   c. *Instead of: trigger* dijalankan untuk *view*.

b. *Trigger event*

   Menurut Kadir (2008) *Trigger event* ada tiga jenis yaitu:

   a. *Insert*: *Trigger* diaktifkan ketika sebuah *record* baru disisipkan ke dalam tabel.

b. *Update*: *Trigger* dijalankan ketika terdapat operasi pengubahan sebuah baris dalam sebuah tabel.

c. *Delete*: *Trigger* dijalankan ketika terdapat operasi penghapusan sebuah baris pada sebuah tabel.

c. Batasan *Trigger*

Batasan yang berlaku bagi *Trigger* (Kadir, 2008):

a. Tidak dapat melibatkan prosedur tersimpan.

b. Tidak dapat melibatkan pernyataan SQL dinamis *PREPAE*.

c. Tidak dapat melibatkan pernyataan-pernyataan yang terkait dengan transaksi (*START TRANSACTION*, *COMMIT*, atau *ROLLBACK).*

d. Penulisan sintak

Penulisan pembuatan *trigger* adalah sebagai berikut:

```
CREATE [DEFINER ={USER | CURRENT_USER}]
TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW trigger_stmt
```

Kode 2.8 Kode Program Pembuatan *Trigger*

Contoh pembuatan *trigger* untuk tabel Mahasiswa

```
CREATE TRIGGER ins_mhs AFTER INSERT ON mahasiswa
FOR EACH ROW INSERT INTO log_mahasiswa VALUES (‘tambah
data’, NOW ());
```

Kode 2.9 Kode Program Pembuatan *Trigger* Tabel Mahasiswa

e. Kegunaan *trigger*

Hal-hal yang dapat dilakukan *trigger* (Husni, 2004):

1. Melakukan pemabatasan data secara otomatis untuk memastikan bahwa *user* hanya memasukan nilai-nilai valid ke dalam kolom.

2. Mengurangi perawatan aplikasi, karena perubahan pada suatu *trigger* secara otomatis berpengaruh pada semua aplikasi yang menggunakan tabel-tabel yang mengandung *trigger* tanpa harus melakukan kompilasi dan link ulang.
3. Pencatatan otomatis terhadap perubahan tabel. Sebuah aplikasi dapat membuat catatan perubahan dengan *trigger* yang berjalan saat tabel dimodifikasi.

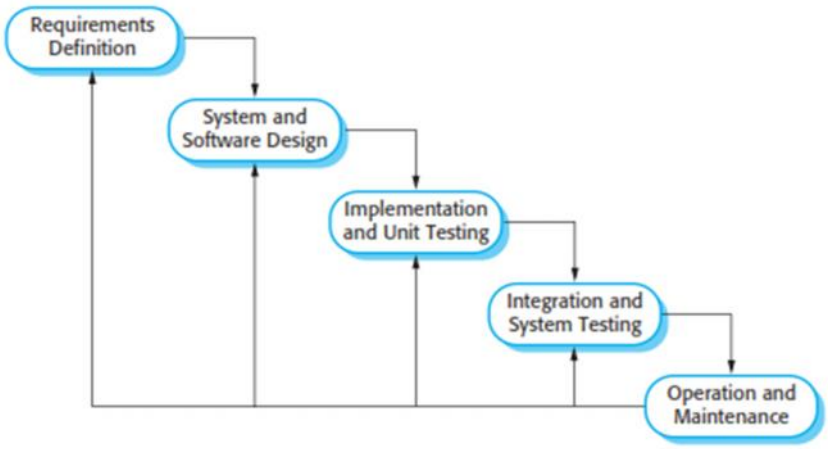
2.14 Metode pengembangan sistem

Metodologi yang digunakan dalam penelitian ini adalah metode *Waterfall*, Selain itu juga digunakan desain alur proses dengan *Data Flow Diagram* (DFD), dan desain hubungan antar data dengan *Entity Relationship Diagram* (ERD).

a. **Metode *Waterfall***

Model proses *Waterfall* atau *linier sequential model* merupakan model klasik yang bersifat sistematis, yang artinya berurutan atau secara linier dalam membangun *software* (Pressman, 2014). Sebuah Model *Waterfall* memacu tim pengembang untuk merinci apa yang seharusnya perangkat lunak lakukan (mengumpulkan dan menentukan kebutuhan sistem) sebelum sistem tersebut dikembangkan (Simarmata, 2010).

Tahapan yang dilakukan dalam pengembangan aplikasi ini sesuai dengan Metode *Waterfall* disajikan pada Gambar 2.3.



Gambar 2.3. Tahapan dalam *Waterfall Model* (Sommerville, 2011).

Tahapan utama dari *Waterfall Model* dicerminkan dalam kegiatan pengembangan dasar sebagai berikut (Sommerville, 2011):

1. *Requirement Definition*

   Requirement adalah tahap untuk mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh sistem yang akan dibangun. Tahap ini harus dikerjakan secara lengkap untuk dapat menghasilkan desain yang lengkap.

2. Sistem *and Software Design*

   Proses desain sistem mengalokasikan persyaratan, baik perangkat keras atau perangkat lunak sistem dengan membentuk arsitektur sistem secara keseluruhan. Desain perangkat lunak melibatkan identifikasi dan menggambarkan abstraksi sistem perangkat lunak yang mendasar dan hubungan-hubungannya.

3. *Implementation and Unit Testing*

   Implementation and Unit Testing merupakan tahapan menerjemahkan desain sistem ke dalam kode program dengan menggunakan bahasa pemrograman tertentu. Sistem yang dibangun kemudian diuji secara unit.

4. *Integration and* Sistem *Testing*

   Integration and Sistem *Testing* merupakan tahap penyatuan unit-unit program kemudian diuji secara keseluruhan (sistem *testing*).

5. *Operation and Maintenance*

   Operation and Maintenance merupakan tahap mengoperasikan program di lingkungannya dan melakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya.

Model ini memungkinkan pemecahan misi pengembangan yang rumit menjadi beberapa langkah logis dengan beberapa langkah yang pada akhirnya akan menjadi produk akhir yang siap pakai. Pada akhirnya, pendekatan ini membuat perangkat lunak yang lebih besar, mudah diatur dan selesai tepat pada waktunya tanpa biaya yang berlebihan (Simarmata, 2010).

b. *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) merupakan diagram yang menggambarkan aliran data dalam sistem, sumber dan tujuan data, proses yang mengolah data tersebut, dan tempat penyimpanan datanya (Yasin, 2012).

Ladjamudin (2013) juga menjelaskan untuk memudahkan analisa dimulai dengan diagram-diagram salah satunya adalah Diagram Konteks. Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks

merupakan level tertinggi dari DFD yang menggambarkan seluruh *input* ke sistem atau *output* dari sistem:

Simbol-simbol DFD disajikan pada Tabel 2.2.

Tabel 2.1. Tabel simbol-simbol DFD

Elemen DFD	Simbol Gene and Sarson	Simbol De Marco and Jourdan	Fungsi
Proses			Menunjukan pemrosesan data atau informasi yang terjadi di dalam sistem
Data Flow			Menunjukan arah aliran dokumen antar bagian yang terkait pada suatu sistem
Data Store			Tempat menyimpan dokumen arsip
Entitas			Menunjukan entitas atau bagian yang terlibat yang melakukan proses

c. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan relationship data (Ladjamudin, 2004).

Seperti DFD, ERD juga menggunakan simbol-simbol khusus untuk menggambarkan elemen-elemen ERD. Berikut elemen-elemen yang terdapat pada ERD:

1. Entitas (*Entity*)

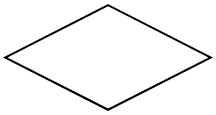
   Entitas adalah objek apa saja yang ada di dalam sistem dimana data tersimpan atau dimana terdapat data. Entitas diberi nama dengan kata benda dan dapat dikelompokan dalam empat jenis nama, yaitu orang, benda, lokasi, kejadian (terdapat unsur waktu di dalamnya).

   Simbol entitas disajikan pada Gambar 2.4.

Gambar 2.4. Simbol Entitas

2. Relasi (*Relationship*)

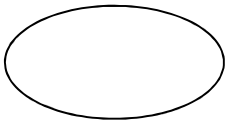
   Relasi adalah hubungan alamiah yang terjadi antar entitas. Pada umumnya *relationship* diberi nama dengan kata kerja dasar sehingga memudahkan untuk pembacaan relasinya. Simbol relasi disajikan pada Gambar 2.5.



Gambar 2.5. Simbol Relasi

3. Atribut (*Attribute*)

   Atribut atau *field* adalah suatu karakteristik yang biasa digunakan untuk menggambarkan seluruh atau sebagian dari *record*. Kata lain dari atribut adalah elemen data (Ladjamudin, 2004). Simbol atribut disajikan pada Gambar 2.6.



Gambar 2.6. Simbol Atribut

4. Derajat Relasi (*Degree of Relationship*)

   Derajat dari relasi adalah jumlah entitas yang berpartisipasi dalam suatu relationship.

5. Kardinalitas (*Cardinality*)

   Kardinalitas relasi menunjukan jumlah maksimum tupel yang dapat berelasi dengan entitas pada entitas yang lain. Terdapat tiga macam kardinalitas relasi yaitu:

   - *one-to-one*

     Satu elemen di entitas (A) tepat berasosiasi dengan satu elemen di entitas (B). Contoh: pegawai dengan *workstation*.

   - *one-to-many*

     Satu elemen di entitas (A) berasosiasi dengan nol, satu atau lebih elemen yang ada di entitas (B), tetapi untuk satu elemen di entitas (B) hanya berelasi dengan satu elemen di entitas (A). Contoh: departemen dan projek.

   - *many-to-many*

     Satu elemen di entitas (A) berasosiasi dengan nol, satu atau lebih elemen di entitas (B), dan satu elemen di entitas (B) berasosiasi dengan nol, satu atau lebih elemen di entitas (A). Contoh: pegawai dengan projek.

2.15 Desain sistem

Desain sistem adalah proses dari penggambaran, pengaturan, dan susunan dari komponen sebuah sistem pada kedua tingkat keterkaitan dan tingkat detail dengan melihat perancangan sistem yang akan diajukan (Satzinger,

dkk 2009). Tahap desain sistem mempunyai dua tujuan utama, yaitu sebagai berikut:

a. Untuk memenuhi kebutuhan kepada pemakai sistem.
b. Untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap kepada pemrogram komputer dan ahli-ahli teknik lainnya yang terlibat.

Kedua tujuan ini lebih condong pada desain sistem yang terinci, yaitu pembuatan rancang bangun yang jelas dan lengkap yang nantinya digunakan untuk pembuatan progam komputernya, untuk mencapai tujuan ini, analis sistem harus dapat mencapai sasaran-sasaran sebagai berikut:

1. Desain sistem harus berguna, mudah dipahami dan nantinya mudah digunakan. Ini berarti bahwa data harus mudah ditangkap, metode-metode harus mudah diterapkan dan informasi harus mudah dihasilkan serta mudah dipahami dan digunakan.
2. Desain sistem harus dapat mendukung tujuan utama perusahaan sesuai dengan yang didefinisikan pada tahap perencanaan sistem yang dilanjutkan pada tahap analisis sistem.
3. Desain sistem harus efisien dan efektif untuk dapat mendukung pengolahan transaksi, pelaporan manajemen dan mendukung keputusan yang akan dilakukan oleh manajemen, termasuk tugas-tugas yang lainnya yang tidak dilakukan oleh komputer.
4. Desain sistem harus dapat mempersiapkan rancang bangun yang terinci untuk masing-masing komponen dari sistem informasi meliputi data, informasi, simpanan data, metode-metode, prosedur-prosedur, orang-orang, perangkat keras, perangkat lunak dan pengendalian *internal*.

2.16 HTML

Dokumen HTML (*Hyper Text Markup Language*) adalah file teks murni yang dapat dibuat dengan editor teks sembarang. Dokumen ini dikenal sebagai web page. *File-file* HTML ini berisi instruksi-instruksi yang kemudian diterjemahkan oleh *browser* yang ada di komputer *client* (*user*) sehingga isi informasinya dapat di tampilkan secara visual di komputer pengguna.

HTML dikenal sebagai standar bahasa yang digunakan untuk menampilakan dokumen web. Hal-hal yang bisa dilakukan HTML (Kustiyahningsih dan Anamisa, 2011) yaitu:

a. Mengontrol tampilan dari web page dan kontennya.
b. Mempublikasikan dokumen secara *online* sehingga bisa diakses dari seluruh dunia.
c. Membuat *online* form yang bisa digunakan untuk menangani pendaftaran, transaksi secara *online*.
d. Menambahkan objek, seperti *image, audio, video* dan *juga Java applet* dalam dokumen HTML.

2.17 PHP

PHP (Resminya: *Hypertext Preprosessor*) merupakan bahasa pemrograman *open source* yang digunakan secara luas terutama untuk pengembangan web dan dapat disimpan dalam bentuk HTML. Web tidak hanya memberikan informasi tetapi terjalin interaksi dan menjadikan web bersifat dinamis dan

diintegrasikan dengan *web server Apache*, PWS *(personal web server)* dan IIS *(internet information service).*

PHP (*Personal Home Page*) sebagai alternatif lain memberikan solusi sangat murah dan dapat berjalan di berbagai jenis *platform*. PHP adalah skrip bersifat *server-side* yang ditambahkan ke dalam HTML. Sifat *server-side* berarti pengerjaan skrip dilakukan di *server*, baru kemudian hasilnya dikirimkan ke *browser* (Kustiahningsih dan Anamisa, 2011).

2.18 XAMPP

XAMPP adalah sebuah *software web server Apache* yang di dalamnya sudah tersedia *database server* MySQL dan dapat mendukung pemrograman PHP. XAMPP merupakan *software* yang mudah digunakan, gratis dan mendukung instalasi di *Linux* dan *Windows*. Keuntungan lainnya adalah menginstal satu kali sudah tersedia *Apache* Web *Server*, MySQL *Database Server*, PHP *Support* (PHP 4 dan PHP 5) dan beberapa *module* lainnya (Ibrahim, 2008).

2.19 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah prosedur eksekusi sebuah program atau sistem dengan tujuan untuk menemukan kesalahan serta pengujian perangkat lunak ini juga digunakan untuk memastikan apakah sistem telah bekerja sesuai dengan spesifikasinya (Nidhra dan Dondeti, 2012).

Menurut Zohrahayati (2007) Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah sebagai berikut:

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan masalah.
2. *Test Case* yang baik adalah yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Ada dua teknik pengujian yang dapat digunakan untuk menguji perangkat lunak, yaitu teknik *black box* dan *white box testing*. *Black box testing* merupakan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinsikan. *White box testing* merupakan pengujian untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya. Metode pengujian perangkat lunak yang digunakan dalam penelitian ini adalah metode pengujian kotak hitam atau *black box testing*.

2.20 Black box Testing

Black box Testing adalah suatu metode yang digunakan untuk menguji perangkat lunak dari segi spesifikasi fungsionalitas tanpa menguji desain dan kode program. Pengujian *black box* ini dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sudah sesuai dengan yang dibutuhkan (Sukamto dkk, 2011).

Gries and Schneider (2005) mengatakan bahwa tujuan dari pengujian *black box* ini yaitu:

1. Menemukan fungsi yang hilang atau tidak benar
2. Kesalahan *interface*
3. *Error* pada struktur data atau akses *eksternal database,*
4. *Error* pada kinerja
5. Dan batasan dari suatu data.

Keuntungan dari pengujian dengan menggunakan metode *black box* adalah:

1. Penguji tidak harus menguasai pemrograman
2. Kesalahan dari perangkat lunak ataupun *bug* sering kali ditemukan oleh kelompok penguji yang berasal dari pengguna.
3. Hasil dari pengujian menggunakan metode *black box* dapat memperjelas kerancuan yang mungkin timbul dari proses eksekusi sebuah perangkat lunak.
4. Proses pengujian lebih cepat dibandingkan dengan pengujian *white box.*

BAB III

METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian dilakukan di lingkungan Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Waktu penelitian dilaksanakan pada semester genap tahun ajaran 2015/2016.

3.2 Lingkungan Pengembangan

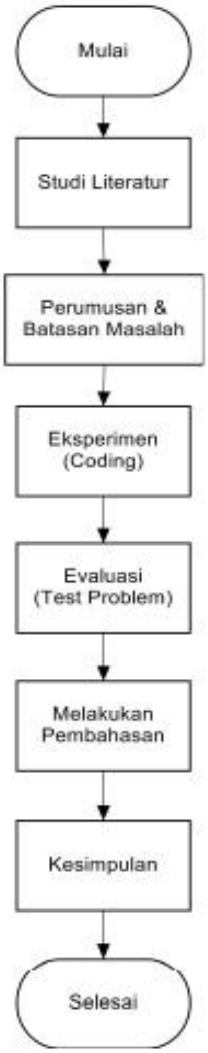
Lingkungan pengembangan yang digunakan pada penelitian ini adalah sebagai berikut:

1. Perangkat lunak: OS Windows 8 Ultimate 32 bit, Xampp, Browser, Notepad++.
2. Perangkat keras: Laptop ASUS X450C series, Processor Intel(R) Celeron(R) CPU 1017U @ 1.60GHz, RAM 2 GB.

3.3 Metode Penelitian

Kualitas keilmuan terlihat dari hasil penelitian yang diperoleh berdasarkan pada proses-proses penelitian yang telah direncanakan dengan baik. Proses penelitian digunakan agar hasil penelitian mencapai optimasi pada berbagai keputusan riset.

Gambar 3.1. menjelaskan bagaimana proses penelitian ini dilaksanakan.



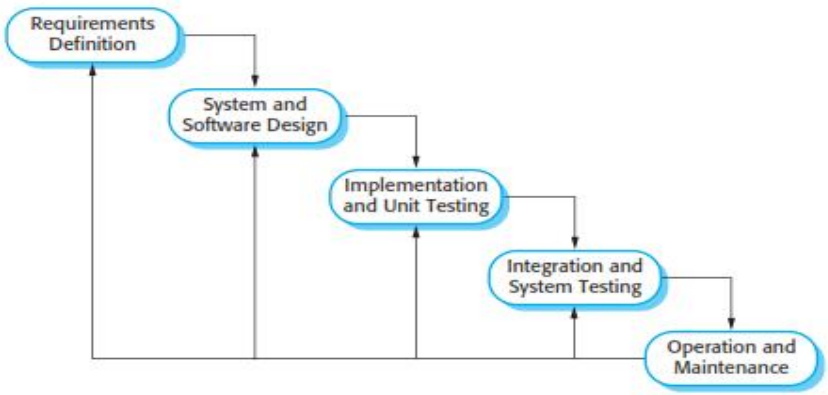
Gambar 3.1. Proses Penelitian

Dalam penelitian ini digunakan metode studi literatur, dimana diperlukan sumber baca seperti buku-buku dan jurnal-jurnal yang berkaitan dengan optimasi *query* dan pengembangan sistem pencatatan aktifitas. Tujuan dari metode studi literatur ini adalah untuk memperoleh sumber referensi sehingga memudahkan pelaksanaan penelitian.

3.4 Metode Pengembangan Sistem

Waterfall Model adalah *"Linear Sequential Model"*. Model ini kadang-kadang disebut dengan *"classic life cycle"*. Disebut dengan *Waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. *Waterfall* menunjukkan sebuah pendekatan sistematis untuk pengembangan perangkat lunak (Pressman , 2014).

Model ini dimulai dari level kebutuhan sistem lalu menuju ke tahap *analisis, desain, coding, testing, dan maintenance*.



Gambar 3.2. Fase-Fase Pengembangan Sistem *Waterfall* (Sommerville, 2011).

Fase dalam Metode *Waterfall*

Tahapan tahapan dari metode *Waterfall* adalah sebagai berikut:

1. Fase Perencanaan Sistem

   Pada fase perencanaan sistem ini peneliti harus merencanakan tentang *project* apa yang akan dibuat atau dengan kata lain mendefinisikan masalah yang harus dipecahkan. Bagaimana cara membuat sistem pencatatan aktifitas dan membuat optimasi *query* sehingga diharapkan mampu mengefisienkan waktu pencarian pada sistem.

1. Fase Analisis

   Pada fase analisis harus mengetahui kebutuhan-kebutuhan apa saja yang akan di gunakan dalam pembuatan sistem, dan *software* yang dibutuhkan harus bisa didapatkan dalam fase ini. Terdapat analisis kebutuhan yang digunakan dalam pengembangan sistem ini yaitu berupa perangkat keras laptop beserta spesifikasi processor: Processor Intel(R) Celeron(R) CPU 1017U @ 1.60GHz, RAM 2 GB. Spesifikasi *software* yang digunakan adalah sebagai berikut: Sistem Operasi Windows 8 Ultimate, Aplikasi *local server* XAMPP 3.2.1, Notepad++, *Browser Google* Chrome, Microsoft Visio 2010.

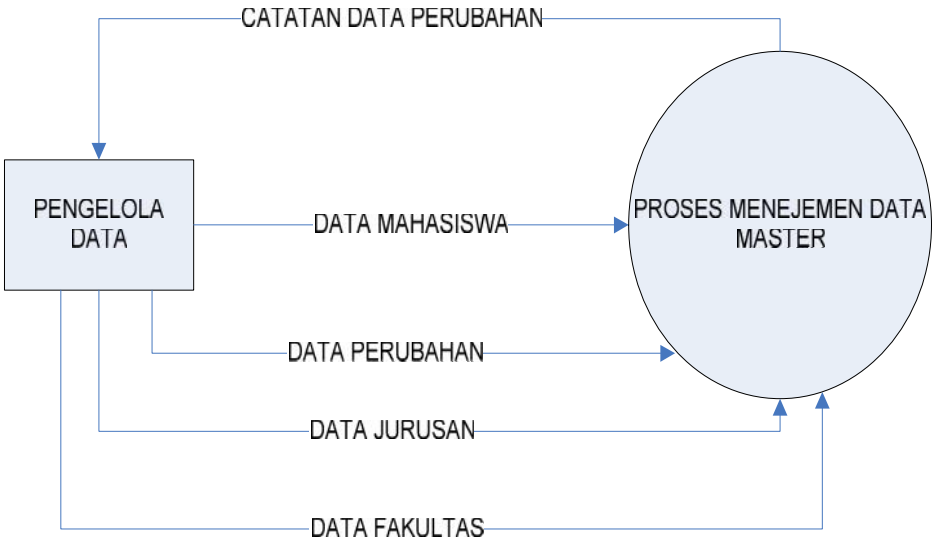
2. Fase Desain

   Pada fase ini akan dilakukan desain pada sistem sebelum melakukan pengkodean. Tahap ini bertujuan untuk memberikan gambaran apa yang harus dikerjakan dan bagaimana tampilannya. Tahap ini membantu dalam menspesifikasikan kebutuhan *hardware* dan sistem serta mendefinisikan arsitektur sistem secara keseluruhan. Menurut Satzinger dkk (2009) *data*

flow diagram merupakan model grafikal sistem yang menunjukkan semua kebutuhan utama suatu sistem informasi pada satu diagram yang di dalamnya terdapat penjelasan mengenai *input* dan *output*, proses dan penyimpanan data.

a. *Data Flow Diagram Level 0*

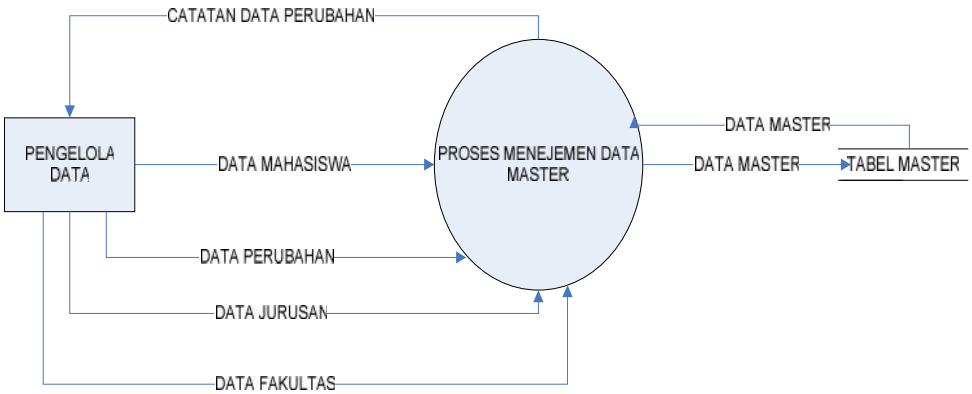
Berikut ini merupakan *data flow diagram* (DFD) untuk sistem informasi pencatatan perubahan data nilai pada level 0 dan level 1. DFD pada level 0 juga sering disebut sebagai *Sequence Diagram.*



Gambar 3.3. *Data Flow Diagram Level 0*

Pada Gambar 3.3. dijelaskan aliran data dari entitas pengelola data ke sistem, dan sistem ke entitas admin. Aliran data dari admin ke sistem adalah sebagai berikut: data mahasiswa, data nilai, data mata kuliah, data jurusan dan data perubahan. Sedangkan aliran data dari sistem ke admin adalah data mahasiswa, data nilai, data mata kuliah, data jurusan dan data perubahan.

b. Data Flow Diagram Level 1



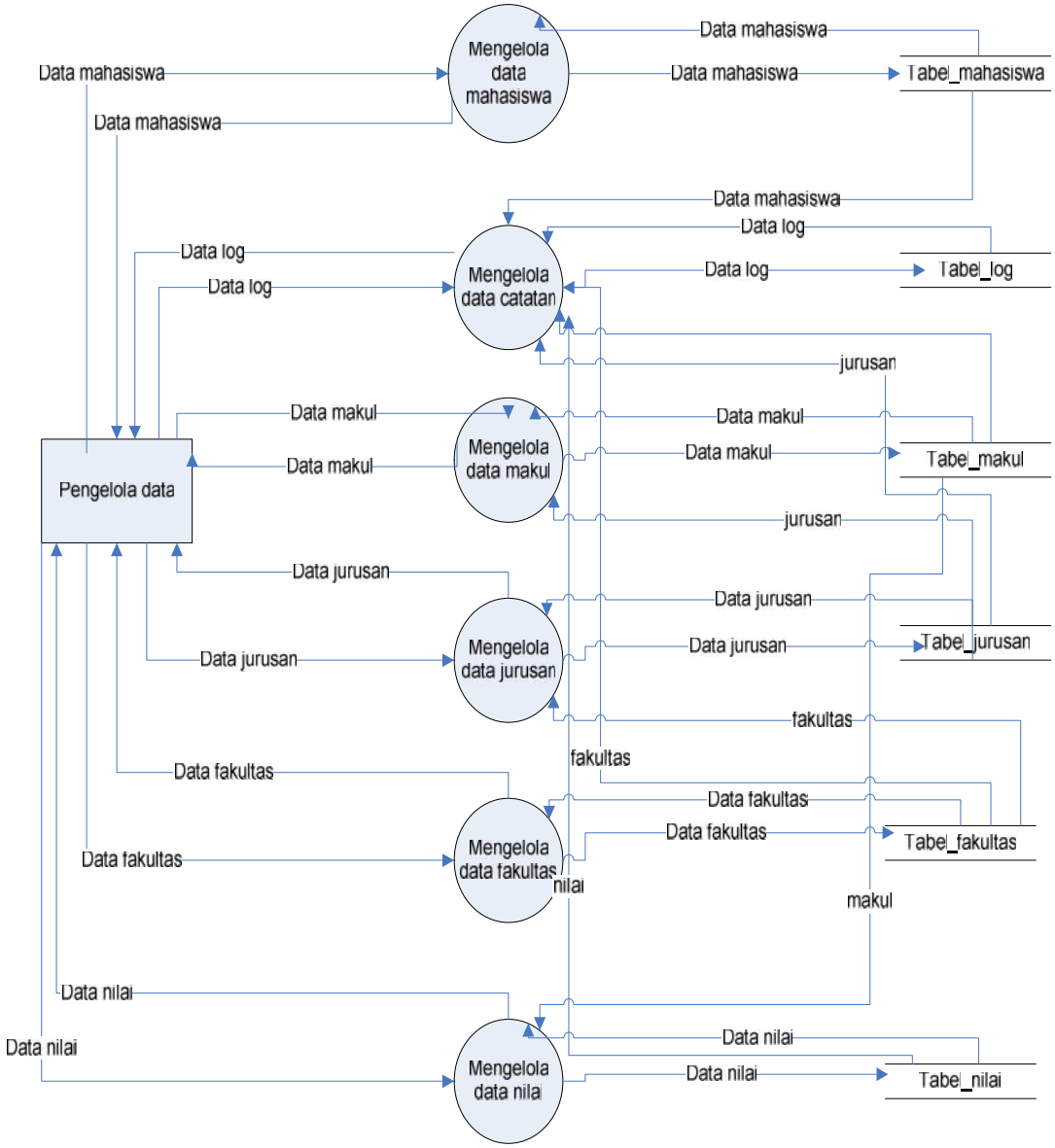
Gambar 3.4. *Data Flow Diagram Level 1*

Penjelasan dari Gambar 3.4 dapat dipaparkan sebagai berikut:

Pengelola data dapat melakukan proses kelola data *master*. Data *master* yang dimaksud berupa: data catatan perubahan data, data mahasiswa, data jurusan, data fakultas, dan data nilai.

c. Data Flow Diagram Level 2

DFD level 2 dari proses manajemen data *master* disajikan pada Gambar 3.5.



Gambar 3.5. *Data Flow Diagram Level 2*

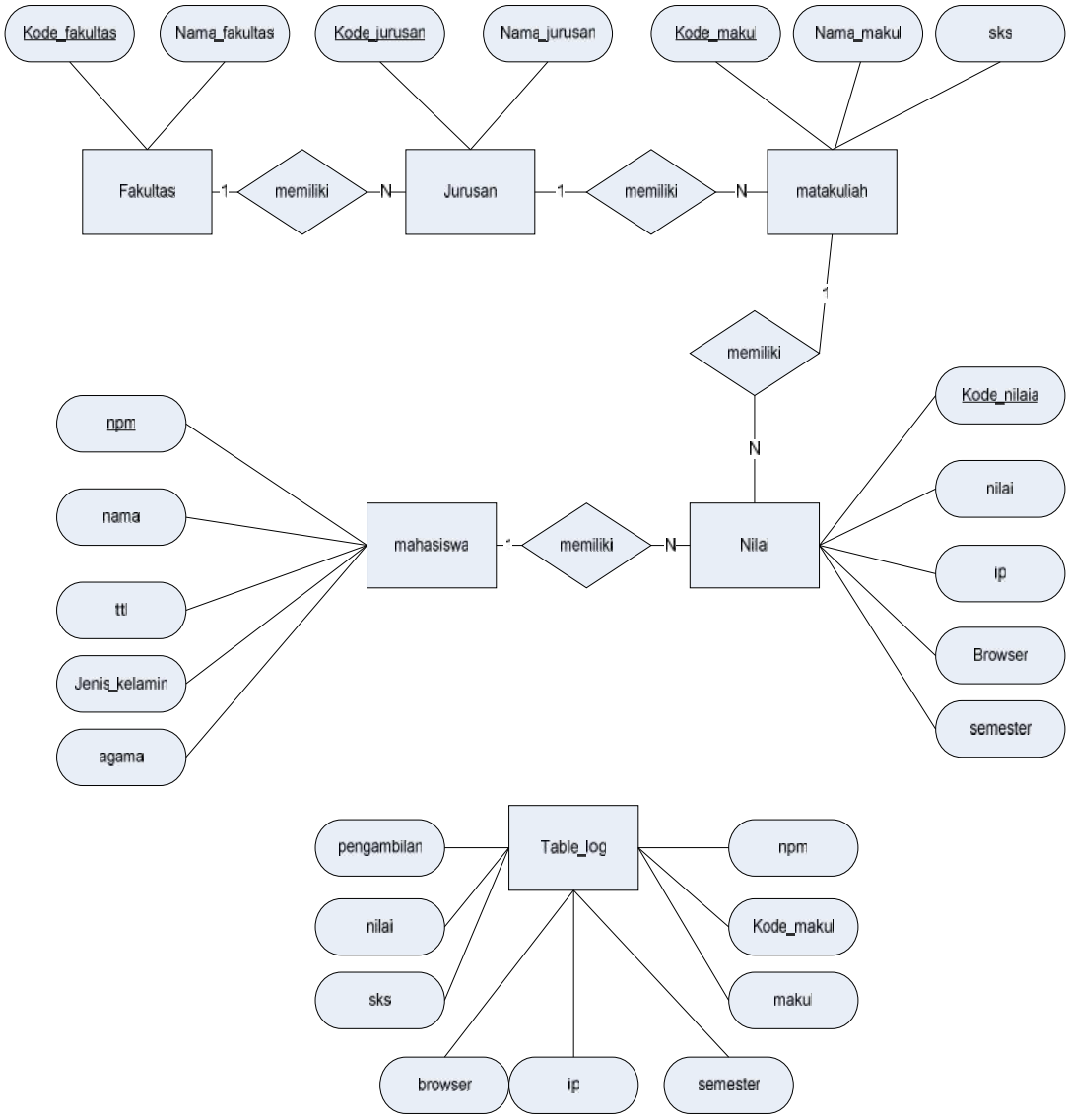
Penjelasan dari Gambar 3.5 dapat dipaparkan sebagai berikut:

1. Pengelola data dapat mengelola data mahasiswa termasuk data orang tua dan lain-lain.
2. Pengelola data dapat mengelola data catatan
3. Pengelola data dapat mengelola data mata kuliah
4. Pengelola data dapat mengelola data jurusan

5. Pengelola data dapat mengelola data fakultas
6. Pengelola data dapat mengelola data nilai mahasiswa

d. *Entity Relationship Diagram (ERD)*

Hubungan antar entitas data untuk sistem yang dibangun disajikan pada Gambar 3.3. Pada aplikasi sistem informasi pencatatan aktifitas perubahan data nilai terdapat 6 entitas yaitu fakultas, jurusan, mata kuliah, mahasiswa, nilai dan tabel pencatat. Entitas digambarkan dengan bentuk persegi panjang. Masing-masing entitas memiliki atribut yang digambarkan dengan bentuk *ellipse*. Atribut *key* dicirikan dengan adanya garis bawah pada nama atribut. Relasi atau hubungan antar entitas digambarkan dengan bentuk belah ketupat. Masing-masing relasi memiliki derajat relasi yang menyatakan banyaknya entitas yang berasosiasi dengan relasi tersebut.



Gambar 3.6. Entity Relationship Diagram

Berikut adalah rancangan struktur tabel dari *Entity Relationship Diagram* pada Gambar 3.6.

1. Tabel Fakultas

Tabel ini digunakan untuk menyimpan data fakultas.

Tabel 3.1. Tabel Fakultas

No	*Attribute*	*Type*	*Length*	*Ket.*
1	Kode_fakultas	*Varchar*	10	*Primary key*
2	Fakultas	*Varchar*	50	

2. Tabel Jurusan

Tabel jurusan digunakan untuk menyimpan data jurusan.

Tabel 3.2. Tabel Jurusan

No	*Attribute*	*Type*	*Length*	*Ket.*
1	id_jurusan	*Varchar*	10	*Primary key*
2	Jurusan	*Varchar*	100	
3	Kode_fakultas	*Varchar*	10	*Foreign key*

3. Tabel Mata kuliah

Tabel mata kuliah digunakan untuk menyimpan data mata kuliah.

Tabel 3.3. Tabel Mata kuliah

No	*Attribute*	*Type*	*Length*	*Ket.*
1	Kode_makul	*Varchar*	10	*Primary key*
2	Mata kuliah	*Varchar*	100	
3	Id_jurusan	*Varchar*	10	*Foreign key*
4	Jumlah_sks	*INT*	1	

4. Tabel mahasiswa

Tabel mahasiswa digunakan untuk menyimpan semua data mahasiswa.

Tabel 3.4. Tabel mahasiswa

No	*Attribute*	*Type*	*Length*	*Ket.*
1	NPM	*Char*	10	*Primary key*
2	Nama_mahasiswa	*Char*	100	
3	Ttl	*Varchar*	100	
4	Gender	*Char*	10	

5. Tabel Nilai

Tabel nilai digunakan untuk menyimpan data nilai dari setiap mahasiswa dan mata kuliah.

Tabel 3.5. Tabel Nilai

No	*Attribute*	*Type*	*Length*	*Ket.*
1	Id_nilai	*Varchar*	100	*Primary key*
2	Npm	*Char*	10	*Foreign key*
3	Id_mata kuliah	*Varchar*	100	*Foreign key*
4	Nilai	*Varchar*	100	
5	Tanggal	*Datetime*		
6	Ip	*Varchar*	100	
7	*Browser*	*Text*	100	
8	Semester	*Int*	2	

6. Tabel Pencatat

Tabel pencatat digunakan untuk menyimpan semua rekaman data dari tabel nilai.

Tabel 3.6. Tabel Pencatat

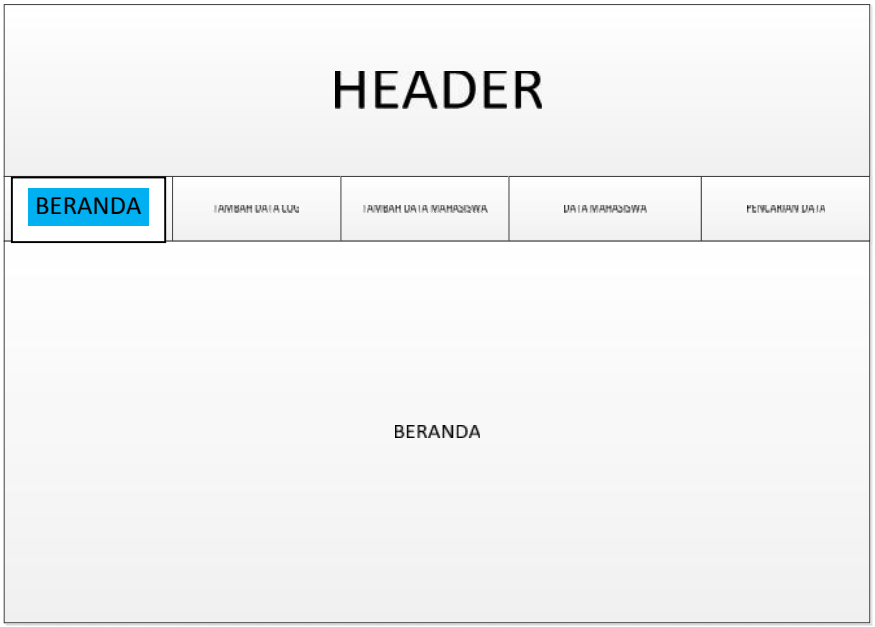
No	*Attribute*	*Type*	*Length*	*Ket.*
1	Id_log	Int	100	*Primary key*
2	Nilai	varchar	100	
3	Id_mata kuliah	varchar	100	*Foreign key*
4	Tanggal_perubahan	*Datetime*		
5	Npm	*Char*	10	*Foreign key*
6	Ip	*Varchar*	100	
7	Kejadian	*Varchar*	100	
8	*Nama_user*	*Varchar*	100	
9	*Browser*	*Text*	100	

e. Desain *Interface*

Desain *interface* sistem informasi pencatatan aktifitas perubahan data nilai mahasiswa adalah sebagai berikut :

1. Halaman Beranda

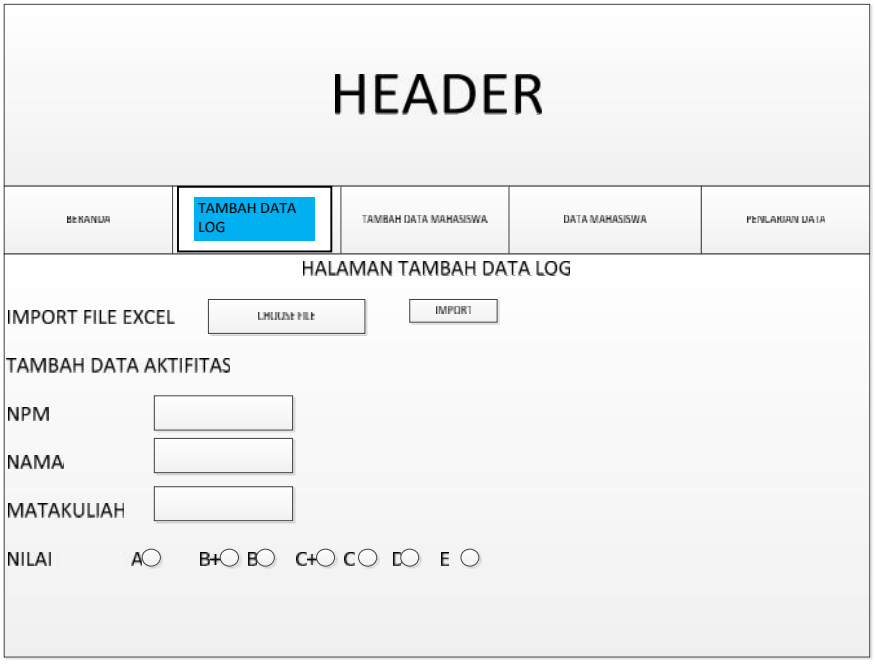
Halaman beranda sistem informasi pencatatan aktifitas perubahan data nilai mahasiswa ini digambarkan pada Gambar 3.6. Halaman beranda adalah halaman awal dari sebuah sistem informasi.



Gambar 3.7. Halaman Beranda Sistem

2. Halaman Tambah Data Log

Halaman data log sistem informasi pencatatan aktifitas perubahan data digambarkan pada Gambar 3.7.

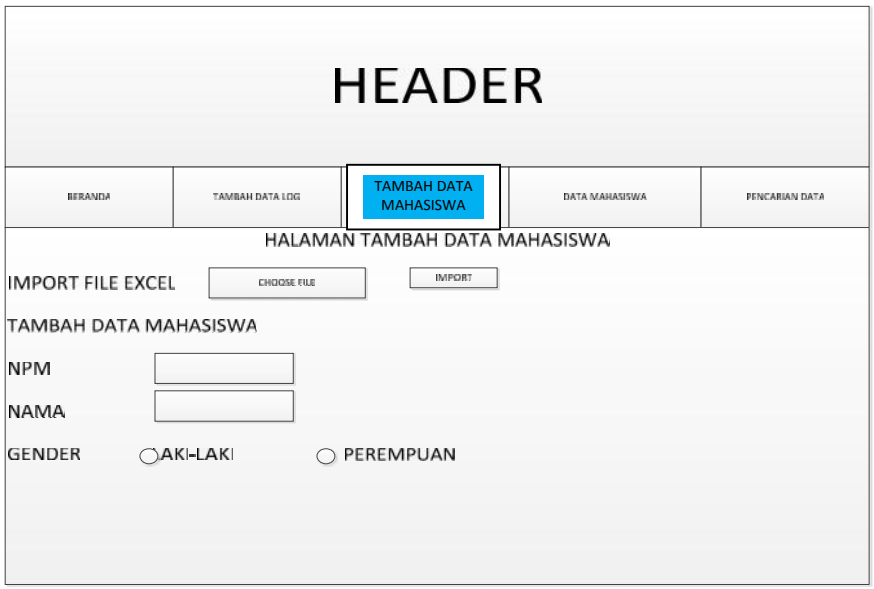


Gambar 3.8. Halaman Data Log Sistem

Pada halaman data log pengelola nilai dapat menyimpan aktifitas yang baru saja dilakukanya. Data yang disimpan adalah NPM, nama mahasiswa mata kuliah, nilai dan tanggal perubahan.

3. Halaman Tambah Data Mahasiswa

Halaman tambah data mahasiswa sistem informasi pencatatan aktifitas perubahan data digambarkan pada Gambar 3.8.

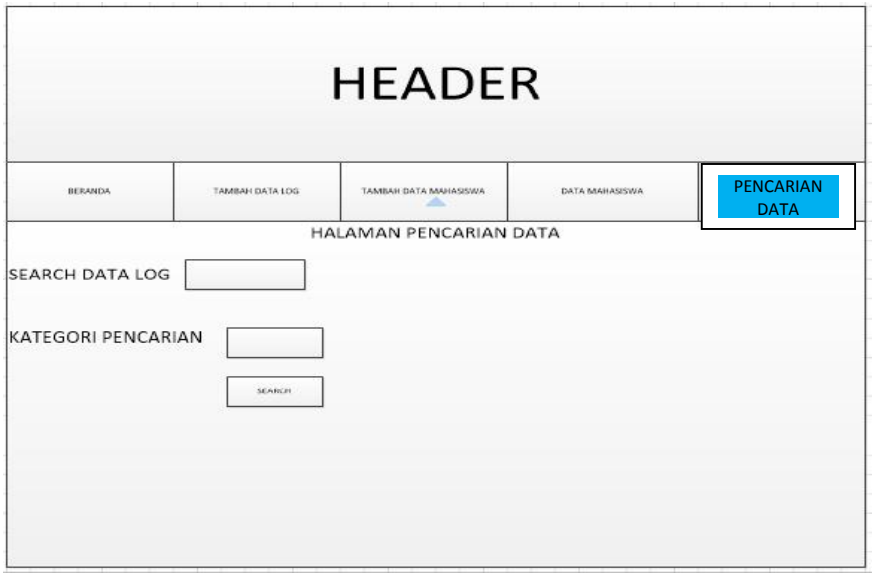


Gambar 3.9. Halaman Tambah Data Mahasiswa

Pada halaman tambah data mahasiswa pengelola nilai dapa tmelakukan penambahan data mahasiswa, mata kuliah jurusan dan fakultas. Pada penambahan data ini segala informasi tentang mahasiswa dapat di simpan dalam sebuah *database.*

4. Halaman Pencarian

Halaman pencarian sistem informasi pencatatan aktifitas perubahan data digambarkan pada Gambar 3.9.



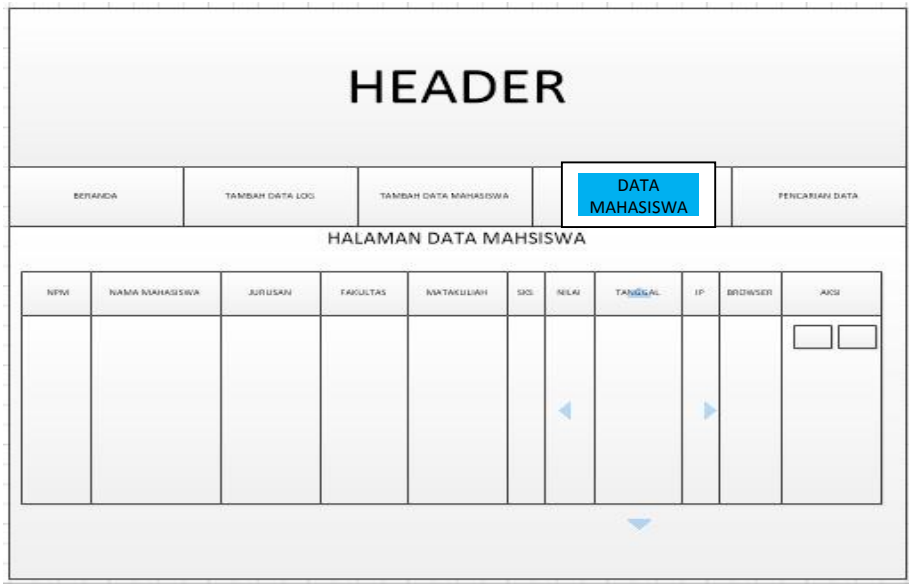
Gambar 3.10. Halaman Pencarian Sistem

Pada halaman pencarian pengelola nilai dapat melakukan pencarian catatan aktifitas dengan beberapa kategori yaitu:

1. Nama mahasiswa
2. NPM
3. Aktifitas kejadian
4. Tanggal perubahan data
5. IP yang digunakan
6. *Browser* yang digunakan

5. Halaman Data Log

Halaman data log sistem informasi pencatatan aktifitas perubahan data digambarkan pada Gambar 3.10.



Gambar 3.11. Halaman Data Log

Pada halaman data log pengelola nilai dapat melihat semua catatan yang pernah dilakukan.

3. Fase Implementasi

   Pada fase Implementasi ini mulai dilakukan pengkodean dengan bahasa pemrograman berbasis web yaitu PHP dan HTML. Penelitian ini menggunakan database MySQL serta menggunakan *web server Apache* yaitu XAMPP. Pembuatan *software* dipecah menjadi modul-modul kecil yang akan digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini dilakukan pemeriksaaan terhadap modul yang dibuat, apakah sudah memenuhi fungsi yang diinginkan.

4. Fase *Integration & Testing*

   Pada fase *Integration & Testing* ini dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian untuk mengetahui apakah

software yang dibuat telah sesuai dengan desainnya dan masih terdapat kesalahan atau tidak, jika sudah tidak ada kesalahan langsung masuk ke fase pemeliharaan.

5. Fase *Maintenance*

   Pada fase *maintenance software* yang sudah jadi dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis pada Optimasi *query* pada sistem informasi pencatatan aktifitas perubahan data nilai, maka didapatkan kesimpulan sebagai berikut:

1. Penggunaan *index* untuk optimasi *query* pada *database* akan bepengaruh pada jumlah data di atas 2000 *record*,
2. Penggunaan *query nested* yang digabungkan dengan penggunaan *index* akan membantu mempercepat waktu pencarian pada data mata kuliah,
3. Sistem informasi pencatatan aktifitas perubahan data dapat membantu pengelola nilai mencatat aktifitas yang dilakukan.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa saran sebagai berikut:

1. Optimasi dapat dilakukan menggunakan *query merge.*
2. Tampilan sistem dapat diperbaiki sehingga pengguna dapat dengan mudah menggunakan sistem.

DAFTAR PUSTAKA

Cecilia dan Mihai. 2011. Increasing Database Performance Using Indexes. *Database Sytems Jurnal Vol.II, No. 2.*

Ermattita. 2009. Analisis Opimasi Query Pada Data Mining, *Jurnal Sistem Informasi (JSI), Vol.1, No. 1*. *Universitas Sriwijaya*: Palembang.

Gries, David and Fred B. Schneider. 2005. *An Integrated Approach To Software Engineering*, Third Edition. Pankaj Jalote. Indian Institute of Technology Kanpur, India.

Husni. 2004. *Pemrograman Database dengan Delphi*. Yogyakarta ; Graha Ilmu.

Ibrahim. 2008. *Cara Praktis Membuat Website Dinamis Menggunakan Xampp*. Yogyakarta: Neotekno.

Jogiyanto. 2003. *System teknologi informasi*. Yogyakarta. Andi offset

Kadir, abdul. (2001), *Dasar Pemrograman Web Dinamis Menggunakan PHP*, C.V Andi Offset, Yogyakarta.

Kadir, Abdul. 2003. *Pengenalan Sistem Informasi*. Yogyakarta: Andi Offset.

Kadir, abdul. 2008. *Tuntunan praktis: belajar database menggunakan MySQL*. Yogyakarta; Andi.

Kustiahningsih, Yeni dan Devie Rosa Anamisa. 2011. *Pemrograman Basis Data Berbasisi Web Menggunakan PHP Dan Mysql*. Yogyakarta : Graha Ilmu

Ladjamudin, Al-Bahra Bin. 2004. *Konsep Sistem Basis Data Dan Implementasinya*. Yogyakarta. Graha Ilmu

Ladjamudin, Al-Bahra Bin. 2013. *Analisis dan Desain Sistem Informasi*. *Graha Ilmu*. Yogyakarta

Maulani, Berry, A. Haidar Mirza, dan Maria Ulfa. 2015. Analisis Perbandingan Optimasi Query Nested Join Dan Hash Join Pada Aplikasi Pencarian Data Berbasis Web.*Jurnal Ilmiah Vol. 1, No. 1.* Universitas Bina Darma. Palembang

Nidhra, Srinivas and Jagruthi Dondeti. 2012. Black Box And White Box Testing Techniques – A Literature Review. *International Journal Of Embedded System And Application (*IJESA). Vol.2, No.2. Mcgraw-Hill.

Oktavia Dan Sujarwo. 2014. *Evaluation Of Sub Query Performance In Sql Server.* Bina Nusantara University

Presman,R.S.2014. *Software Engineering: A Practitioner's Approach*. Newyork: Mcgraw-Hill.

Satzinger, John, Robert Jackson, Stephen Burd. 2009. *System Analys And Design: In A Changing World Fifthedition*. Thomson Course Technology. Canada.

Simarmata, Janner. 2010. *Rekayasa web*. Yogyakarta: Andi

Sommerville, Ian. 2011. *Software Engineering (Rekayasa Perangkat Lunak)*. Jakarta: Erlangga.

Sukamto, Rosa Ariani dan M.Shalahuddin. 2011. *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur Dan Berorintasi Objek).* Bandung: Modula

Sutanta, Edhy. 2003. *Sistem Informasi Manajemen.*yogyakarta. Graha Ilmu

Yasin, Verdi. 2012. *Rekayasa perangkat lunak berorientasi objek pemodelan, arsitektur dan perancangan (modeling, architecture and design)*. Jakarta : Mitra Wacana Media.

Zohrahayati. 2007. Perancangan Sistem Informasi Pelayanan Pelanggan Berbasis Jaringan Pada PT.PLN Wil.Sultenggo Cab.Gorontalo Kantor Jaya Tapa. *Jurnal Ichsan Gorontalo Vol.2. No.1. Feb-April*. Gorontalo.