MODEL QUERY REALTIME *DATABASE* UNTUK TRANSAKSI ONLINE

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4
Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26
Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimkasud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan hak dan/atau tanpa ijin Pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).
3. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah).
4. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah).

MODEL QUERY REALTIME *DATABASE* UNTUK TRANSAKSI ONLINE

Anastasia Lidya Maukar
(Universitas Presiden, Cikarang)

Fitri Marisa
(Universitas Widyagama Malang)

Anik Vega Vitianingsih
(Universitas Dr. Soetomo)

Erri Wahyu Puspitarini
(STMIK Yadika Bangil)

President University

MODEL QUERY *REALTIME DATABASE* UNTUK TRANSAKSI *ONLINE*

Anastasia Lidya Maukar
Fitri Marisa
Anik Vega Vitianingsih
Erri Wahyu Puspitarini

Editor
Johan K. Runtuk
Yuseva
Mohammad Shihab

Desain Cover
Jessie Angelica

ISBN 978-623-6655-87-0

Book Cover Photo
https://pin.it/2Tbezs1

Penerbit
President University
Anggota APPTI 007.112.1.04.2020
Jalan Ki Hajar Dewantara, Mekarmukti, Cikarang Utara Bekasi,
Jawa Barat 17550
Email: lrpmpu@president.ac.id

PRAKATA

Buku ini membahas improvisasi model kombinasi MD5 dengan string pengolah kata pada pemrograman untuk memberi keamanan terhadap *database*. Keberadaan *database* sangat rawan dengan injeksi dan penyusupan yang dilakukan pihak luar. Untuk itu diperlukan metode-metode yang dapat dikembangkan untuk meningkatkan keamanan *database*. Buku ini diharapkan dapat membantu pembaca memahami dan menerapkan enkripsi data untuk mengamankan *database*.

Model dibangun dengan rancangan *coding* dan simulasi sistem berbasis web. Model diterapkan dalam sebuah *prototype* data penerimaan mahasiswa baru dengan data dummy yang menyesuaikan data penerimaan mahasiswa baru. Hasil dapat dilihat pada data hasil enkripsi yang ada pada data password.

Penulis mengucapkan terima kasih kepada semua rekan dosen dan mahasiswa Teknik Industri Universitas Presiden yang turut mendukung dalam penulisan buku ini. Semoga buku ini dapat memberikan manfaat bagi para pembaca.

Cikarang, April 2022

Penulis

DAFTAR ISI

PRAKATA v

DAFTAR ISI vi

BAB 1 PENDAHULUAN 1

BAB 2 BASIS DATA 5

2.1 Pengertian Data dan *Database* 5

2.2 Jenis *Database* 6

2.2.1 *Database* terdistribusi 6

2.2.2 *Database* Relasional (Relational *Database*) 9

2.2.3 *Object Oriented Database* 13

2.2.5 Opened Source *Database* 18

2.2.6 Cloud *Database* 19

2.3 Definisi *Database* dalam Disiplin Ilmu yang Terkait 20

2.4 SQL *Server* 21

2.5 Pemrograman .Net Framework 26

BAB 3 *SERVER* PULSA 29

3.1 *Server* I Pulsa 29

3.2 Sistem Finansial *Server* Pulsa *Reload* 30

BAB 4 KONSEP APLIKASI BERBASIS WEBSITE 31

4.1 Pengertian dan Konsep Website 31

4.2 Konsep Aplikasi Berbasis *Website* 33

4.3 Teknologi Web Service 34

4.4 *Script web dan .Net* 36

BAB 5 PERANCANGAN *QUERY REALTIME* APLIKASI WEB 38

5.1 Penjelasan Metodologi Peracangan 39

5.2 Hasil Perancangan 43

5.2.1 Desain Model dan Bentuk Formula 43

5.3 Perancangan Aplikasi Keuangan *Server* Pulsa 45
5.4 Desain *Database* 55
5.5 Implementasi Model Query Realtime dalam Aplikasi Keuangan 56
BAB 6 PENUTUP 59
DAFTAR PUSTAKA 60

BAB 1

PENDAHULUAN

Pada beberapa tahun terakhir Indonesia marak dengan keberadaan *server* pulsa *reload Server* pulsa di Indonesia bisa dikatakan menjadi salah satu bisnis yang memiliki potensi besar dalam menopang perekonomian. *Server* pulsa *reload* adalah sebuah bidang usaha jasa di bidang Teknologi Informasi yang menjual pulsa secara elektrik dari kartu prabayar. Cara kerja *server* pulsa *reload* adalah dengan mendistribusikan pulsa prabayar dari operator seluler yang didapatkan dari dealer resmi melalui *chip-chip* yang telah diisi oleh pulsa, kemudian diolah dalam aplikasi yang berbasis *short message service (sms) gateway* dan berbantuan beberapa perangkat modem sebagai tempat *chip,* setelah itu dengan mekanisme *parsing*, pulsa didistribusikan kepada *end user* melalui aplikasi tersebut.

Aplikasi *server reload* hingga saat ini dibangun dan dikembangkan oleh banyak *vendor,* dari yang berbasis desktop, web, maupun .Net. Namun seiring dengan perkembangannya aplikasi .Net yang saat ini paling banyak pengguna karena memiliki kelebihan lebih ringan dan lebih cepat dalam proses *parsing*. Salah satu vendor aplikasi *server* .Net yang terbesar adalah Orisinil.com yang didirikan tahun 2008 oleh Yusuf Arif Rahmanto. Produk aplikasi orisinil dikenal dengan Otomax. Otomax dibangun dengan menggunakan software C# yang berbasis .Net dengan *database* SQL *Server* Express. Produk ini memiliki bebrapa kelebihan, yaitu menawarkan kecepatan tinggi dalam proses pengisian pulsa, cara pengoperasian yang mudah dan adanya fitur sudah terintegrasi (ww.indotel.co.id/server-pulsa-otomax. 2022).

Jika ditinjau dari konten yang sudah disajikan oleh aplikasi *server* pulsa *reload,* biasanya selama ini konten sebatas pada pengisian pulsa dan data saldo pelanggan. Begitu juga pada Otomax, yaitu aplikasi sebatas pada mekanisme pengisian pulsa dan perhitungan saldo pelanggan. Sementara para pengusaha pulsa sangat membutuhkan pengolahan keuangan lengkap berupa *General Ledger* (GL) untuk menghasilkan informasi lengkap mengenai laba rugi, dan pergerakan keuangan.

Selama ini para pengusaha pulsa khususnya pengguna Otomax *reload* mengembangkan aplikasi GL secara terpisah dengan aplikasi *server*, sementara sumber data yang diolah adalah berasal dari database *server*, dimana pada Otomax yang digunakan adalah SQL *Server Express*. Dengan proses yang terpisah ini mengakibatkan terjadi ketidakakuratan pada pengolahan data keuangan. Kondisi tersebut juga menyebabkan tidak bisa mendapatkan informasi keuangan secara *real-time*, sementara itu transaksi berjalan dalam hitungan detik yang otomatis perubahan data keuangan mengikuti perubahan transaksi tersebut.

Dengan adanya tantangan yang ada maka aplikasi Otomax *reload* membutuhkan sebuah aplikasi pelengkap yang mencatat keuangan lengkap secara realtime. Oleh karena itu dalam penelitian ini akan dikembangkan model teknik *query database real-time* dengan menganalisis alur relasi *database* Otomax *reload* dan kemudian mengambil *field-field* kunci yang dibutuhkan dengan pendekatan *query.* Teknik ini akan menghasilkan rumus-rumus *query* yang kemudian hasilnya disimpan dalam *database* MySQL untuk diolah dalam aplikasi pengolahan keuangan. Rumus-rumus *query* berfungsi sebagai penjembatan antara data Otomax (dalam SQL *Server*) dan data keuangan (dalam MySQL). Dengan demikian pengolahan data keuangan diperoleh dari sumber data yang *update* secara *real-time.*

Berangkat dari permasalahan yang sudah diuraikan diatas, maka dirumuskan permasalahan sebagai berikut:

1. Bagaimana membangun dan mendesain model teknik *query database real-time* untuk mengolah data finansial pada *server* pulsa *reload* yang berbasis .Net?
2. Bagaimana mengimplementasikan model teknik *query database real-time* dalam sebuah aplikasi untuk mengolah data finansial pada *server* pulsa *reload* yang berbasis .Net?

Tujuan penelitian ini adalah untuk:

1. Membangun dan mendesain model teknik *query database real-time* untuk mengolah data finansial pada *server* pulsa *reload* yang berbasis .Net?
2. Mengimplementasikan model teknik query *database real-time* dalam sebuah aplikasi untuk mengolah data finansial pada *server* pulsa *reload* yang berbasis .Net?

Ruang lingkup dalam penelitian ini adalah sebagai berikut:

1. Studi kasus dilakukan pada aplikasi Otomax *reload*
2. Teknik yang digukanan adalah pendekatan *query database* dalam bahasa SQL.
3. *Database Engine* sumber berasal dari SQL *Server* Express, dan *database engine* pengolah adalah MySQL.
4. Penelitian ini juga menghasilkan aplikasi finansial untuk *server* pulsa.
5. Uji coba terhadap penelitian ini akan dilakukan 5 *server* pulsa yang menggunakan Otomax.

BAB 2
BASIS DATA

2.1 Pengertian Data dan *Database*

Dengan kata sederhana, data dapat berupa fakta yang terkait dengan obyek apa pun yang dipertimbangkan. Misalnya, nama anda, usia, tinggi, berat, dan lain-lain adalah beberapa data yang terkait dengan Anda. Gambar, file gambar, file pdf, dan sebagainya juga dapat dianggap sebagai data.

Basis data atau *database* adalah kumpulan data yang sistematis. Kumpulan data tersebut mendukung penyimpanan elektronik dan manipulasi data. *Database* membuat pengelolaan data menjadi mudah.

Mari kita bahas contoh *database*: Direktori telepon *online* menggunakan *database* untuk menyimpan data orang, nomor telepon, dan detail kontak lainnya. Penyedia layanan listrik Anda menggunakan *database u*ntuk mengelola penagihan, masalah terkait klien, menangani data kesalahan, dll.

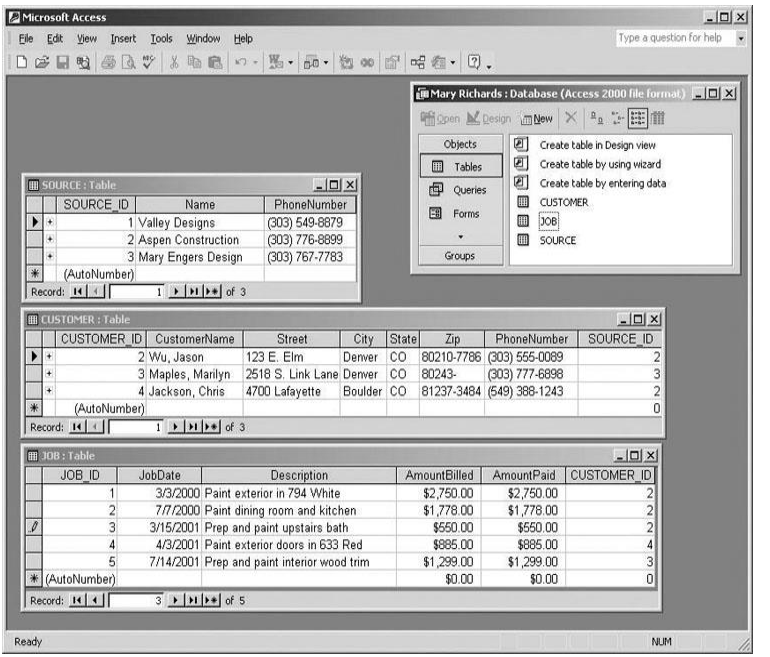
Seperti halnya Facebook, Facebook juga perlu menyimpan, memanipulasi, dan menyajikan data yang terkait dengan anggota, data pertemanan, aktivitas anggota, pesan, iklan, dan banyak lagi. Masih banyak contoh dalam aplikasi sehari-hari untuk penggunaan *database.*

Saat ini, kebanyakan *databases* dapat menyimpan simpanan data yang cukup besar dengan berabgai macam tipe data, di antaranya:

- Scalar data: names, dates, phone numbers
- Pictures
- Audio

- Video

Gambar 2.1 menyajikan sebuah contoh *database* dalam Microsoft Access, di mana simpanan datanya berupa tabel dan dalam contoh ini terdapat tiga buah tabel yaitu tabel *souce, job* dan *customer*. Setiap tabel memuat data yang terkait, sebagai contoh, tabel *customer* memiliki data tentang *customer id, customer name, Strett, city, state, Zip, Phone number*, *dan source.*



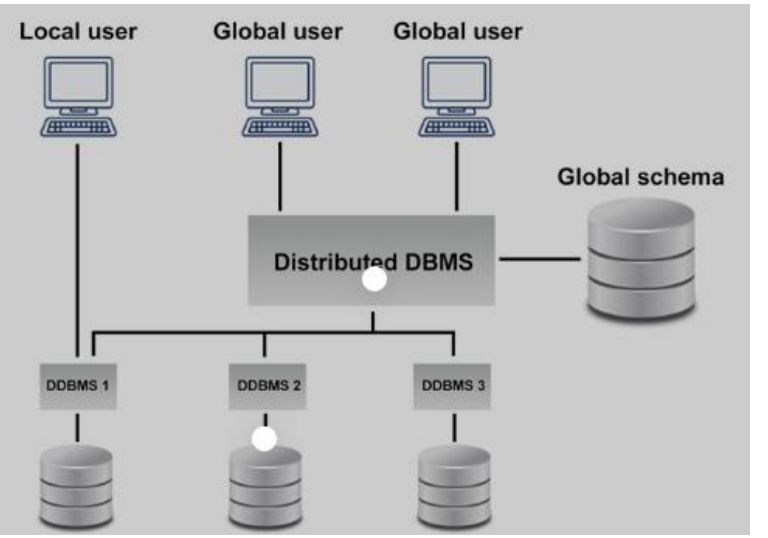
Gambar 2.1 Access *Database*

2.2 Jenis *Database*

Dalam pembahasan ini kita akan mendeskripsikan beberapa jenis *database* yang populer.

2.2.1 *Database* terdistribusi

Database terdistribusi adalah jenis *database* yang memiliki kontribusi dari *database* umum dan informasi yang ditangkap oleh komputer lokal. Dalam sistem basis data jenis ini, data tidak berada di satu tempat dan didistribusikan di berbagai organisasi.



Gambar 2.2 *Distributed Database*
Sumber: https://phoenixnap.com/kb/distributed-*database*)

Bagian dari *database* disimpan di beberapa lokasi fisik dan pemrosesan didistribusikan di antara beberapa node *database (*lihat Gambar 2.2). Sistem manajemen basis data terdistribusi terpusat atau *distributed database management system* (DDBMS) mengintegrasikan data secara logis sehingga dapat dikelola seolah-olah semuanya disimpan di lokasi yang sama. DDBMS menyinkronkan semua data secara berkala dan memastikan bahwa pembaruan dan penghapusan data yang dilakukan di satu lokasi akan secara otomatis tercermin dalam data yang disimpan di tempat lain. Sebaliknya, *database* terpusat terdiri dari file *database* tunggal atau *single database* yang terletak di satu situs menggunakan jaringan tunggal.

Fitur *database* terdistribusi.

Ketika dalam sebuah koleksi, *database* terdistribusi secara logis saling terkait satu sama lain, dan mereka sering mewakili satu *database* logis. Dengan *database* terdistribusi, data disimpan secara fisik di beberapa situs dan dikelola secara independen. Prosesor di setiap situs dihubungkan oleh jaringan, dan mereka tidak memiliki konfigurasi multiprosesor.

Kesalahpahaman yang umum adalah bahwa *database* terdistribusi adalah sistem *file* yang terhubung secara longgar. Pada kenyataannya, ini jauh lebih rumit dari itu. *Database* terdistribusi menggabungkan pemrosesan transaksi, tetapi tidak identik dengan sistem pemrosesan transaksi. Secara umum, *database* terdistribusi mencakup fitur-fitur berikut:

a. Lokasi independen
b. Pemrosesan kueri terdistribusi
c. Manajemen transaksi terdistribusi
d. Perangkat keras independen
e. Sistem operasi independen
f. Jaringan independen
g. Transparansi transaksi
h. DBMS independen
i. Arsitektur *database* terdistribusi
j. *Database* terdistribusi bisa homogen atau heterogen.

Dalam sistem basis data terdistribusi homogen, semua lokasi fisik memiliki perangkat keras dasar yang sama dan menjalankan sistem operasi dan aplikasi basis data yang sama. Sistem basis data terdistribusi homogen tampak bagi pengguna sebagai satu sistem, dan mereka dapat lebih mudah untuk dirancang dan dikelola. Agar sistem *database* terdistribusi menjadi homogen, struktur data di setiap lokasi harus identik atau kompatibel. Aplikasi *database* yang digunakan di setiap lokasi juga harus identik atau kompatibel.

Dalam *database* terdistribusi heterogen, perangkat keras, sistem operasi atau aplikasi *database* mungkin berbeda di setiap lokasi. Situs yang berbeda dapat menggunakan skema dan perangkat lunak yang berbeda, meskipun perbedaan dalam skema dapat mempersulit pemrosesan *query* dan transaksi. Node yang berbeda mungkin memiliki perangkat keras, perangkat lunak, dan struktur data yang berbeda, atau mungkin berada di lokasi yang tidak kompatibel. Pengguna di satu lokasi mungkin dapat

membaca data di lokasi lain tetapi tidak mengunggah atau mengubahnya. Basis data terdistribusi heterogen seringkali sulit digunakan, membuatnya tidak layak secara ekonomi untuk banyak bisnis.

Keuntungan dari *database t*erdistribusi

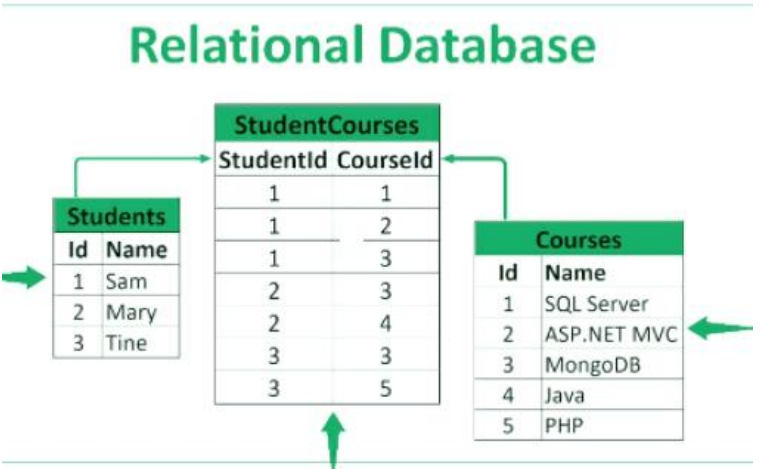
Ada banyak keuntungan menggunakan *database* terdistribusi. Basis data terdistribusi mampu pengembangan modular, artinya sistem dapat diperluas dengan menambahkan komputer baru dan data lokal ke situs baru dan menghubungkannya ke sistem terdistribusi tanpa gangguan. Ketika kegagalan terjadi di *database* terpusat, sistem berhenti total. Namun, ketika komponen gagal dalam sistem basis data terdistribusi, sistem akan terus berfungsi dengan kinerja yang berkurang hingga kesalahan diperbaiki. *Administrator* dapat mencapai biaya komunikasi yang lebih rendah untuk sistem basis data terdistribusi jika data terletak dekat dengan tempat yang paling sering digunakan. Ini tidak mungkin dalam sistem terpusat.

Jenis *database* terdistribusi.

Data yang direplikasi digunakan untuk membuat instance data di berbagai bagian *database*. Dengan menggunakan data yang direplikasi, *database* terdistribusi dapat mengakses data yang identik secara lokal, sehingga menghindari lalu lintas. Data yang direplikasi dapat dibagi menjadi dua kategori: data hanya-baca dan data yang dapat ditulis.

2.2.2 *Database* Relasional (Relational *Database*)

Jenis *database* ini mendefinisikan hubungan *database* dalam bentuk tabel. Ini juga disebut *Database Management System* (DBMS) Relasional, yang merupakan jenis DBMS paling populer di pasar (lihat Gambar 2.3). Contoh RDBMS antara lain *database* MySQL, Oracle, dan Microsoft SQL *Server*.



Gambar 2.3 ***Relational Database***
Sumber: https://www.pragimtech.com/blog/mongodb-tutorial/relational-and-non-relational-databases/

Database relasional memelihara data dalam tabel, menyediakan cara yang efisien, intuitif, dan fleksibel untuk menyimpan dan mengakses informasi terstruktur. Tabel, juga dikenal sebagai relasi, terdiri dari kolom yang berisi satu atau lebih kategori data, dan baris, juga dikenal sebagai catatan tabel, berisi sekumpulan data yang ditentukan oleh kategori. Aplikasi mengakses data dengan menentukan *query* yang menggunakan operasi seperti proyek untuk mengidentifikasi atribut, memilih untuk mengidentifikasi tupel, dan bergabung untuk menggabungkan relasi. Model relasional untuk manajemen *database* dikembangkan oleh ilmuwan komputer IBM Edgar F. Codd pada tahun 1970.

Bagaimana Basis Data Relasional Bekerja?

Database relasional menyediakan lingkungan dari mana data dapat diakses atau disusun kembali dalam berbagai cara yang berbeda tanpa perlu mengatur ulang tabel *database*. Setiap tabel memiliki pengidentifikasi unik atau kunci utama yang mengidentifikasi informasi dalam tabel, dan setiap baris berisi contoh data unik untuk kategori yang ditentukan oleh kolom.

Misalnya, tabel mungkin memiliki kunci utama 'Nama Depan' dan baris dengan contoh spesifik seperti 'John, Paul, George, dan Ringo.

Koneksi logis antara tabel yang berbeda kemudian dapat dibuat dengan menggunakan kunci asing - bidang dalam tabel yang terhubung ke data kunci utama dari tabel lain. Sistem Manajemen Basis Data Relasional sering menggunakan SQL atau bahasa kueri terstruktur untuk mengumpulkan data untuk laporan dan untuk kueri interaktif. Jadi dalam contoh kita, Nama Depan mungkin ditautkan ke tabel Peran dengan peran data Vokal Utama, Gitar Bass, Drum, dan Gitar Utama.

Bagaimana Data dalam Sistem Basis Data Relasional Diorganisasikan?

Model relasional dari basis data relasional memisahkan struktur data logis dari struktur penyimpanan fisik, memungkinkan administrator basis data untuk mengelola penyimpanan data fisik tanpa memengaruhi akses ke data tersebut sebagai struktur logis. Perbedaan juga berlaku untuk operasi *database* - operasi logis memungkinkan aplikasi untuk menentukan konten yang dibutuhkannya, dan operasi fisik menentukan bagaimana data tersebut harus diakses, kemudian menjalankan tugasnya.

Apa Keuntungan Basis Data Relasional?

Keuntungan utama dari *database* relasional adalah struktur tabularnya yang dijelaskan secara formal, dari mana data dapat dengan mudah disimpan, dikategorikan, ditanyakan, dan disaring tanpa perlu mengatur ulang tabel *database*. Manfaat lebih lanjut dari *database* relasional meliputi:

a. *Skalabilitas*: Data baru dapat ditambahkan terlepas dari catatan yang ada.
b. *Kesederhanaan*: Kueri kompleks mudah dilakukan pengguna dengan SQL.

c. *Akurasi Data*: Prosedur normalisasi menghilangkan anomali desain.
d. *Integritas Data*: Pengetikan data yang kuat dan pemeriksaan validitas memastikan akurasi dan konsistensi.
e. *Keamanan*: Data dalam tabel dalam RDBMS dapat membatasi akses ke pengguna tertentu.
f. *Kolaborasi*: Beberapa pengguna dapat mengakses *database* yang sama secara bersamaan.

Apa itu Sistem Manajemen Basis Data Relasional/*Relational Database Management System (RDBMS)*?

Sistem Manajemen Basis Data Relasional atau RDBMS adalah kumpulan program dan kemampuan berbasis tabel yang menyediakan antarmuka antara pengguna dan aplikasi dan basis data, menawarkan cara sistematis untuk membuat, memperbarui, menghapus, mengelola, dan mengambil data. Sebagian besar sistem manajemen basis data relasional menggunakan bahasa pemrograman SQL untuk mengakses basis data dan banyak yang mengikuti properti *Atomicity, Consistency, Isolation,* dan *Durability* (ACID) dari *database:*

Atomicity: Jika ada pernyataan dalam transaksi yang gagal, seluruh transaksi gagal dan *database* dibiarkan tidak berubah.

Consistency: Transaksi harus memenuhi semua protokol yang ditentukan oleh sistem -- tidak ada transaksi yang diselesaikan sebagian.

Isolation: Tidak ada transaksi yang memiliki akses ke transaksi lain yang belum selesai. Setiap transaksi bersifat independen.

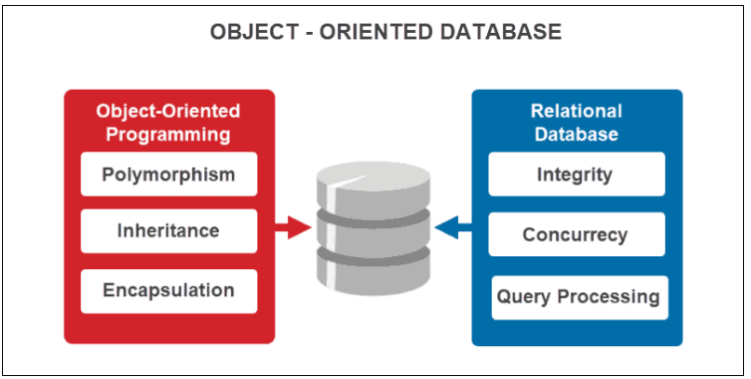
Durability atau Daya Tahan: Setelah transaksi dilakukan, transaksi akan tetap dilakukan melalui penggunaan log transaksi dan cadangan.

Apa Perbedaan Antara Basis Data Relasional dan Non Relasional?

Basis Data Non Relasional, atau basis data NoSQL, menyimpan dan mengatur data dengan cara selain model relasi tabular yang digunakan dalam basis data relasional. Dimana *database* relasional menyimpan data dalam baris dan kolom, memiliki aturan ketat mengenai variasi data dan hubungan tabel, dan mengikuti properti ACID yang ketat, *database* non relasional menawarkan struktur data yang lebih fleksibel berdasarkan model BASE (Basically Available, Soft state, Eventual Consistency) : Pada dasarnya Tersedia menjamin ketersediaan data - akan ada respons terhadap permintaan apa pun, tetapi tanpa jaminan konsistensi; Soft State menjamin bahwa status sistem dapat berubah seiring waktu; dan Konsistensi Akhirnya menjamin bahwa sistem pada akhirnya akan menjadi konsisten setelah berhenti menerima input.

2.2.3 *Object Oriented Database*

Jenis *database* ini mendukung penyimpanan semua tipe data seperti yang terlihat pada Gambar 2.4. Data disimpan dalam bentuk objek. Objek yang akan disimpan dalam basis data memiliki atribut dan metode yang menentukan apa yang harus dilakukan dengan data. PostgreSQL adalah contoh DBMS relasional berorientasi objek.



Gambar 2.4 *Object Oriented Database*
Sumber: https://phoenixnap.com/kb/object-oriented-*database*

Database berorientasi objek adalah jenis sistem manajemen *database*. Sistem manajemen basis data yang berbeda menyediakan fungsionalitas tambahan. Basis data berorientasi objek menambahkan fungsionalitas basis data ke bahasa pemrograman objek, menciptakan basis kode yang lebih mudah dikelola. *Database* objek dikelola oleh sistem manajemen *database* berorientasi objek/*Object Oriented Database Management System* (OODBMS). Basis data menggabungkan konsep pemrograman berorientasi objek dengan prinsip basis data relasional.

Object adalah blok bangunan dasar dan turunan dari kelas, di mana tipenya adalah bawaan atau ditentukan pengguna.

Class menyediakan skema atau cetak biru untuk objek, mendefinisikan perilaku.

Method adalah yang menentukan perilaku kelas.

Pointer membantu mengakses elemen *database* objek dan membangun hubungan antar objek.

Karakteristik utama dari objek dalam OODBMS adalah kemungkinan tipe yang dibuat oleh pengguna. Objek yang dibuat dalam proyek atau aplikasi disimpan ke dalam *database* apa adanya.

Database berorientasi objek secara langsung menangani data sebagai objek yang lengkap. Semua informasi datang dalam satu paket objek yang tersedia secara instan alih-alih beberapa tabel. Sebaliknya, blok bangunan dasar dari *database* relasional, seperti PostgreSQL atau MySQL, adalah tabel dengan tindakan berdasarkan koneksi logis antara data tabel.

Karakteristik ini membuat *database* objek cocok untuk proyek dengan data kompleks yang memerlukan pendekatan berorientasi objek untuk pemrograman. Sistem manajemen

berorientasi objek menyediakan fungsionalitas yang didukung yang melayani pemrograman berorientasi objek di mana objek kompleks adalah pusatnya. Pendekatan ini menyatukan atribut dan perilaku data menjadi satu entitas.

Konsep Pemrograman Berorientasi Objek.

Database berorientasi objek berhubungan erat dengan konsep pemrograman berorientasi objek. Empat ide utama dari pemrograman berorientasi objek adalah:

a. *Polymorphism*
b. *Inheritance*
c. *Encapsulation*
d. *Abstraction*

Keempat atribut ini menggambarkan karakteristik kritis dari sistem manajemen berorientasi objek.

a. *Polymorphism*

   Polimorfisme adalah kemampuan suatu objek untuk mengambil beberapa bentuk. Kemampuan ini memungkinkan kode program yang sama untuk bekerja dengan tipe data yang berbeda. Baik mobil maupun sepeda bisa pecah, tetapi mekanismenya berbeda. Dalam contoh ini, jeda tindakan adalah *Polimorfisme*. Tindakan yang ditentukan adalah polimorfik — hasilnya berubah tergantung pada kinerja kendaraan mana.

b. *Inheritance.*

   Warisan menciptakan hubungan hierarkis antara kelas terkait sambil membuat bagian kode dapat digunakan kembali. Mendefinisikan tipe baru mewarisi semua bidang dan metode kelas yang ada plus memperluasnya lebih jauh. Kelas yang ada adalah kelas induk, sedangkan kelas turunan adalah kelas induk. Misalnya, kelas induk yang disebut Kendaraan akan memiliki kelas anak Mobil dan Sepeda. Kedua kelas anak mewarisi informasi dari kelas induk dan memperluas kelas induk dengan informasi baru tergantung pada jenis kendaraan.

c. *Encaptilation.*

   Enkapsulasi (*Encaptilation*) adalah kemampuan untuk mengelompokkan data dan mekanisme ke dalam satu objek untuk memberikan perlindungan akses. Melalui proses ini, potongan informasi dan detail tentang cara kerja suatu objek disembunyikan, menghasilkan keamanan data dan fungsi. Kelas berinteraksi satu sama lain melalui metode tanpa perlu mengetahui cara kerja metode tertentu. Sebagai contoh, sebuah mobil memiliki karakteristik dan tindakan deskriptif. Anda dapat mengubah warna mobil, tetapi model atau mereknya adalah contoh sifat yang tidak dapat diubah. Kelas merangkum semua informasi mobil menjadi satu entitas, di mana beberapa elemen dapat dimodifikasi sementara beberapa tidak.

d. *Abstraction.*

   Abstraksi (*Abstraction*) adalah prosedur yang mewakili hanya fitur data penting untuk fungsionalitas yang dibutuhkan. Proses memilih informasi penting sementara informasi yang tidak perlu tetap tersembunyi. Abstraksi membantu mengurangi kompleksitas model data dan memungkinkan penggunaan kembali. Misalnya, ada berbagai cara agar komputer terhubung ke jaringan. Sebuah web browser membutuhkan koneksi internet. Namun, jenis koneksi tidak relevan. Koneksi yang dibuat ke internet mewakili abstraksi, sedangkan berbagai jenis koneksi mewakili implementasi abstraksi yang berbeda.

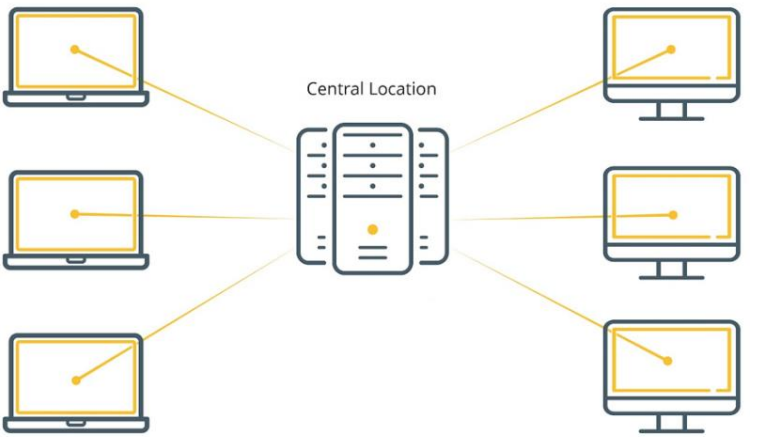
2.2.4 *Centralized Database (CDB)*

Ini adalah lokasi terpusat, dan pengguna dari berbagai latar belakang dapat mengakses data ini. Jenis *database* komputer ini menyimpan prosedur aplikasi yang membantu pengguna mengakses data bahkan dari lokasi yang jauh. *Database* Terpusat (CDB) menyediakan hub utama untuk menyimpan data di mana jutaan entri disimpan dan dipelihara. Organisasi besar, institusi (sekolah & universitas), Perusahaan bisnis (bursa saham) menggunakan sistem *database* terpusat, karena ini adalah cara mudah untuk menyimpan data. Sebuah CPU atau komputer *mainframe* digunakan untuk beroperasi sebagai *database* terpusat.

Ini menyediakan fasilitas konektivitas *database* ke komputer lain untuk menyimpan data mereka ke *database* terpusat (lihat Gambar 2.5).

Contoh *Database* Terpusat.

Misalkan ada aplikasi yang berjalan di sebuah rumah sakit. Semuanya memiliki aplikasi tetapi tidak ada *database* yang diinstal pada komputer yang menjalankan aplikasi tersebut. Mereka semua terhubung ke komputer *server* utama tempat semua data disimpan.



Gambar 2.5 *Centralized Database*
Sumber: https://phoenixnap.com/kb/object-oriented-*database*

Keuntungan dari Basis Data Terpusat.

Keuntungan utama adalah keamanan di mana *server* tersedia untuk komputer yang terhubung secara lokal.

a. Hanya orang internal yang dapat menggunakan *database*. Setiap masalah pada komputer *server* dapat ditangani dan diselesaikan dengan mudah.
b. Konektivitas ke *database* lebih cepat karena jaringan lokal menyediakan koneksi yang lebih baik.
c. Basis data tunggal sehingga pencadangan dan pemulihan lebih cepat daripada basis data online.

d. Desain *database* mudah dan mudah ditangani oleh database *administrator* (DBA).
e. Mudah untuk mengatur peran dan aturan pada *database* yang diakhiri oleh DBA.

Kekurangan *Database* Terpusat.

a. Basis data terpusat sepenuhnya bergantung pada konektivitas jaringan.
b. Fasilitas cadangan segera diperlukan ketika koneksi terputus ke *database* terpusat.
c. Traffic dan beban penyimpanan yang tinggi pada *server*, dapat menyebabkan crash atau kegagalan jika aktivitas pendinginan atau aktivitas pemeliharaan dilakukan.
d. *Database* memiliki catatan yang lebih besar perangkat penyimpanan tinggi mungkin diperlukan.
e. DBA diperlukan setiap kali berada di dekat *server* atau *database* terpusat.

Administrator Basis Data atau DBA

Database Administrator (DBA) adalah satu-satunya orang yang dapat menangani atau memelihara komputer *database*. Semua peran pengguna akan diatur oleh DBA, jadi tidak ada yang bisa memanipulasi *database* kecuali DBA. Dia harus mengurus hal-hal berikut untuk komputer *database* seperti suhu, permintaan per menit, cadangan, siapa yang dapat mengakses *database d*an aspek lain yang benar-benar perlu dipertimbangkan.

2.2.5 Opened Source *Database*

Database semacam ini menyimpan informasi yang berhubungan dengan operasi (lihat Gambar 2.6). Hal ini terutama digunakan di bidang pemasaran, hubungan karyawan, layanan pelanggan, *database*. Pada beberapa tahun terakhir ini konferensi Basis Data Sumber Terbuka, kami merayakan teknologi basis data sumber terbuka yang tidak sesuai dengan ranah MySQL®, MongoDB®, MariaDB®, atau PostgreSQL dengan menampilkannya di jalurnya sendiri. *Lagu Other Opened Source*

Databases yang diberi nama glamor! Sebagai juara solusi *database opened source* yang tidak memihak, kami merangkul semua jenis *database* open source, dan bangga mempersembahkan salah satu acara terbesar yang didedikasikan untuk setiap dan semua *open source database* (OSDB).

Gambar 2.6 *Opened Source Database*
Sumber: https://mobilemonitoringsolutions.com/global-open-source-*database*-software-market-2020-development-analysis-mysql-sqlite/

2.2.6 Cloud *Database*

Basis data *cloud (*lihat Gambar 2.7) adalah basis data yang dioptimalkan atau dibangun untuk lingkungan yang tervirtualisasi. Ada begitu banyak keuntungan dari *database* cloud, beberapa di antaranya dapat membayar untuk kapasitas penyimpanan dan bandwidth. Ini juga menawarkan skalabilitas sesuai permintaan, bersama dengan ketersediaan tinggi.

Gambar 2.7 *Cloud Database*
Sumber: https://mobilemonitoringsolutions.com/global-open-source-*database*-software-market-2020-development-analysis-mysql-sqlite/

2.3 Definisi *Database* dalam Disiplin Ilmu yang Terkait

Data adalah fakta yang dapat direkam dan memiliki arti secara implisit. Sedangkan kumpulan data yang memiliki hubungan secara implisit itu disebut *Database*. (Cahyono, 2006:10). Menurut Ramakrishnan dan Gehrke (2003) dalam Simarmata & Paryudi (2006:1) menyatakan "basis data sebagai kumpulan data, umumnya mendeskripsikan aktivitas satu organisasi atau lebih yang berhubungan". Dalam *database* juga dikenal istilah DBMS (*Database Management Systems)* yaitu sekumpulan program yang memungkinkan pengguna untuk membuat dan memelihara suatu *database*. (Cahyono, 2006:10). Bisa juga dikatakan bahwa DBMS merupakan perangkat *General Purpose Software System* yang berfungsi untuk mewadahi proses-proses dalam *database* seperti pendefinisian, pembuatan, sharing, maupun manipulasi *database*.

Dalam *database*, dikenal istilah *Entity Relationship* (ER). Menurut Octafian (2011:2) "Entitas adalah sesuatu atau objek

dalam dunia nyata yang dapat dibedakan dari objek lain, misalnya mahasiswa dan matakuliah. Entitas digambarkan dalam basis data dengan kumpulan atribut. Atribut merupakan karateristik dari entitas itu sendiri, Sebagai contoh, entitas mahasiswa memiliki atribut seperti nomor induk mahasiswa (NIM), nama, alamat, kota, program studi, indeks presrasi (IP), dan lain-lainnya.

Setiap entitas memiliki atribut yang unik yang biasa disebut sebagai *identifier* atau *primary key.* Sebagai contoh, entitas mahasiswa memiliki nomor induk mahasiswa (NIM) sebagai identifier, sedangkan mata kuliah memiliki kode matakuliah sebagai *identifier.*

Relasi adalah hubungan antara beberapa entitas. Misalnya: relasi menghubungkan mahasiswa dengan mata kuliah yang diambilnya." Jenis hubungan/relasi dapat bermacam-macam, yaitu:

- *Many to one*
- *One to many*
- *Many to many*

Relasi mahasiswa dan matakuliah dapat digambarkan sebagai berikut: bahwa mahasiswa bisa mengambil beberapa mata kuliah dan matakuliah bisa diambil dan dihadiri oleh beberapa mahasisa.

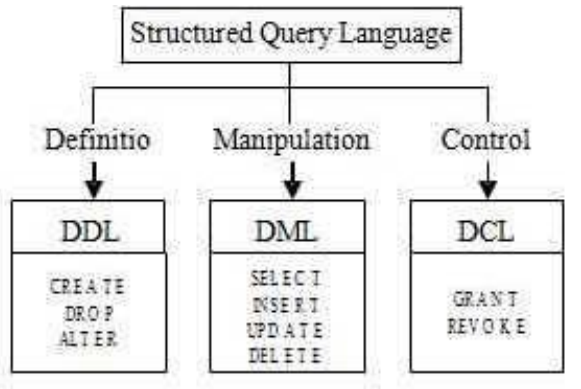
Bisa juga dikatakan bahwa "*Entity relationship* (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antarobjek (Simarmata & Paryudi, 2006:59).

2.4 SQL *Server*

SQL (ess-que-el) adalah singkatan dari *Structured Query Languag*e. SQL digunakan untuk berkomunikasi dengan *database*. Menurut ANSI (American National Standards Institute), ini adalah

bahasa standar untuk sistem manajemen basis data relasional. SQL merupakan bahasa *query* standar yang dipergunakan untuk mengakses basis data relasional.

Statement SQL secara garis besar dibagi menjadi 3 kategori yaitu *Data Definition Languages* (DDL), *Data Manipulation Languages* (DML), dan *Data Control Language* (DCL). (Dedianto, 2013: 2). Struktur SQL dapat digambarkan pada Gambar 2.8. Pernyataan SQL digunakan untuk melakukan tugas-tugas seperti memperbarui data pada *database*, atau mengambil data dari *database*. Beberapa sistem manajemen basis data relasional umum yang menggunakan SQL adalah: Oracle, Sybase, Microsoft SQL *Server*, Access, Ingres, dll.



Gambar 2.8. Struktur SQL

Meskipun sebagian besar sistem basis data menggunakan SQL, sebagian besar juga memiliki ekstensi kepemilikan tambahan yang biasanya hanya digunakan pada sistem mereka. Namun, perintah SQL standar seperti "*Select", "Insert", "Update", "Delete", "Create",* dan *"Drop*" dapat digunakan untuk menyelesaikan hampir semua hal yang perlu dilakukan dengan *database*.

Ketika berpikir tentang SQL, kebanyakan orang pada umumnya akan berpikir tentang *Data Manipulation Language*

(DML) sebagai salah aspek dalam *language.* DML memperbolehkan pengguna untuk melakukan *insert, delete, modify*, dan *retrieve* informasi. Adapun syntax perintah dari DML adalah:

- Select
- Delete
- Insert
- Update

SQL dapat digunakan pada:

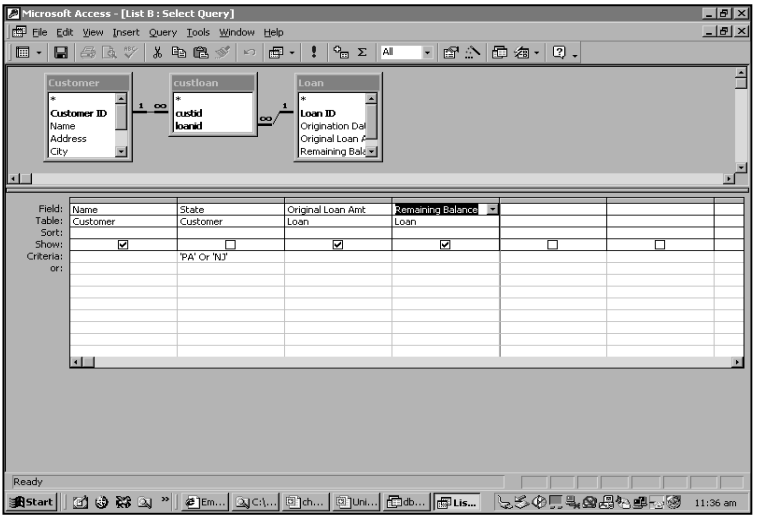
- Stand-alone within a DBMS command
- Embedded in triggers and stored procedures
- Scripting or programming languages

Sebuah DBMS harus menyediakan paling tidak satu DML. Beberapa alternatif atau pilihan dari DML adalah sebagai berikut:

- Query/Update language (e.g., SQL)
- Query-by-Example
- Query-by-Form

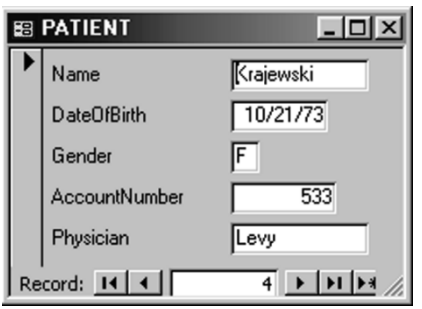
Contoh dari Query/Update Language
SELECT Name, Age
FROM PATIENT
WHERE Physician = 'Levy'

Contoh Query by Example (QBE) dapat dilihat pada Gambar 2.9.



Gambar 2.9 Contoh QBE pada Microsoft Access

Sedangkan contoh query by form dapat dilihat pada Gambar 2.10



Gambar 2.10 Contoh Query by Form pada Microsoft Access

Perintah select dapat dijelasakan dalam syntax secara umum seperti di bawah ini

select [**distinct**] target-list
from tuple variable list
where qualification
[**order by** target list subset]
[**group by** target list subset]
[**having** set-qualification]
the items in square brackets ([ ]) are electives.

Daftar dari kualifikasi dapat berupa sebuah kombinasi boolean (and, or, not) dari salah satu *selection* dan *join clauses.* Sebuah *selection clause* merupakan perbandingan antara sebuah *indexed tuple variable* dan sebuah *constant*. Adapun *comparison operator* adalah {=, <>, <=, >=, >, <}.

Berikut adalah pembahasan dalam memahami SQL *Server*. "SQL *SERVER* adalah sistem manajemen *database* relasional (RDBMS) yang dirancang untuk aplikasi dengan arsitektur client*/server*. Istilah client, server, dan client*/server* dapat digunakan untuk merujuk kepada konsep yang sangat umum atau hal yang spesifik dari perangkat keras atau perangkat lunak. Pada level yang sangat umum. (Mustofa, 2012:1) *CLIENT* adalah setiap komponen dari sebuah sistem yang meminta layanan atau sumber daya (*resource)* dari komponen sistem lainnya. *SERVER* adalah setiap komponen sistem yang menyediakan layanan atau sumber daya ke komponen sistem lainnya". (Mustofa, 2013: 1)

Dalam SQL Server juga dikenal dengan istilah Relational *Database Management System (*RDBMS). RDBMS adalah dasar untuk SQL, dan untuk semua sistem *database* modern seperti Microsoft SQL *Server*, IBM DB2, Oracle, MySQL, dan Microsoft Access. Data dalam RDBMS disimpan dalam objek *database* yang disebut tabel. TABEL adalah kumpulan data entri terkait dan terdiri dari kolom dan baris.

SQL memiliki peran penting bagi pembangunan aplikasi sistem. Berikut beberapa fungsi SQL yaitu:

1. Mengakses dan memanipulasi *database*.
2. Mengeksekusi query terhadap *database*.
3. Mengambil, menyisipkan, memperbarui, dan menghapus data dari *database.*
4. Membuat tabel dan *database* baru.
5. Membuat prosedur yang tersimpan dalam *database*.
6. Mengatur hak akses pada tabel, prosedur, dan list data.

2.5 Pemrograman .Net Framework

Issa (2012: 44) mengatakan "*Have you ever thought of some great idea for a product but you couldn't bring it to life because technology wasn't on your side? Or maybe thought, "there's got to be an easier way!" Maybe you are a programmer that wanted to make a security system but then thought using a PC is too expensive to run a simple system? The answer is Microsoft's .NET Micro Framework!".* Pendapat tersebut dengan kata lain bahwa pemrograman .Net Framework merupakan Kerangka kerja yang menyediakan sejumlah besar pustaka pemrograman komputer dan mendukung beberapa bahasa pemrograman serta interoperabilitas yang baik sehingga memungkinkan bahasa-bahasa tersebut berfungsi satu dengan lain dalam pengembangan sistem. NET Framework berjalan pada lingkungan perangkat lunak melalui *Common Language Runtime* (CLR), dan bukan perangkat keras secara langsung. Hal ini memungkinkan aplikasi yang dibuat di atas, dapat berjalan pada perangkat keras apapun yang didukung oleh .NET Framework.

Dalam Issa (2012: 44) dijelaskan beberapa keunggulan .Net framework khususnya Microsoft framework antara lain:

1. Berjalan dalam platform yang gratis yaitu visual C# express dengan teknologi high-end IDE.
2. .Net framework adalah teknologi yang bersifat Open source dan Free.
3. Memiliki kemampuan Debugging yang handal.
4. Telah diujicobakan dalam berbagai produk komersil dan telah terjamin kehandalannya.
5. Memiliki banyak *Bus Drivers* (SPI, UART, I2C, dll)

Dari sisi pola pengkodean .Net Framework memiliki pola *Model, View & Controller* (MVC). Kelebihan dari pola MVC adalah kemudahan dalam memelihara kode yang telah kita buat karena modul program terbagi menjadi 3 bagian:

1. Model merupakan logika bisnis utama. Di dalamnya terdapat kode untuk data persistence dan perhitungan logika bisnis utama utama.
2. *View* menangani masalah-masalah yang berkaitan dengan tampilan (user interface).
3. *Controller* melakukan respon terhadap action yang dilakukan oleh user.

.Net Framework juga memiliki kelebihan hemat waktu, dimana pemrogram tidak perlu menghabiskan banyak waktu untuk menulis kode program, programmer bisa menggunakan fungsi atau class bawaan dari framework yang kita gunakan, seperti:

1. Modul Generator, akan menghasilkan modul yang kita inginkan menjadi cepat (menghasilkan MVC).
2. *Object Relation Mapping* (ORM), memungkinkan mempraktiskan syntax SQL yang spesifik untuk *database* tertentu. Sehingga progammer tidak perlu melakukan query terhadap tabel, hanya melakukan query terhadap objek yang telah didefinisikan oleh ORM.
3. Ketersediaan *Plug-in*, sehingga *programmer* dapat menggunakan *plug-in* yang tersedia pada komunitas sesuai kebutuhan.

.Net Framework juga memiliki kemudahan dalam melakukan debug program. Framework biasanya dilengkapi dengan fasilitas debug program, yang salah satunya berfungsi untuk melakukan analisa terhadap program yang sedang berjalan atau memeriksa kesalahan-kesalahan pada program yang dibuat. Dengan demikian .Net Framework sangat menguntungkan dari sisi kekayaan tools dan penyediaan kebutuhan kerja bagi performansi aplikasi yang dibuat. Selain itu dengan modul MVC membuat .Net Framework juga memiliki kelebihan pada kecepatan akses.

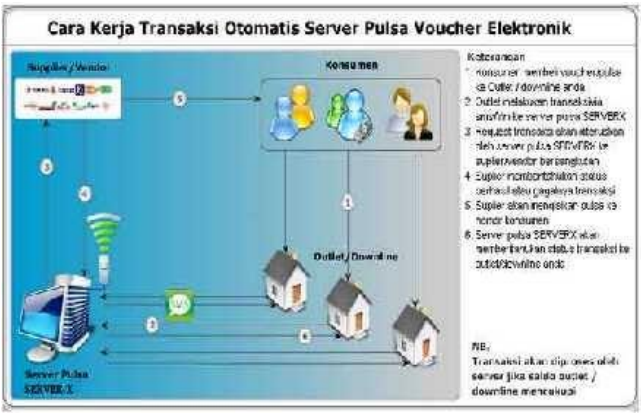
BAB 3

SERVER PULSA

3.1 *Server* I Pulsa

Dalam pembahasan ini, akan dijelaskan tentang pengertian dan mekanisme *server* pulsa *reload Server* pulsa adalah sebuah sistem pengisian pulsa elektrik yang dilakukan dengan berbantuan komputer, modem, dan handphone yang terangkai kerja dengan perangkat lunak yang bertugas melakukan parsing dengan menangkap dan menyesuaikan kode-kode pengiriman dari operator seluler, kemudian diteruskan kepada pelanggan.

Ide dasar dari pembuatan *server* pulsa elektrik ini adalah dengan semakin maraknya distributor penjualan pulsa elektrik yang pada dasarnya menggunakan format SMS dan dial, ternyata pada proses pengisian dirasa tidak efektif jika dilakukan oleh distributor pulsa, dikarenakan jumlah transaksi per hari sekurang- kurangnya 200 transaksi per hari. Untuk itu diperlukan sebuah mekanisme komputerisasi yang dapat melakukan proses pengisian secara cepat dan dalam jumlah yang banyak. Maka dibuatlah sistem *reload* pulsa untuk mengefektifkan proses pengisian pulsa oleh distributor. Sistem kerja *server* pulsa dapat dilihat pada Gambar 3.1.



Sumber: http://www.ipulsamedia.com/
Gambar 3.1 Sistem Kerja *Server* Pulsa *Reload*

Dari Gambar 3.1 dapat dijelaskan secara singkat bahwa ada beberapa komponen penting yang harus ada dalam *server* pulsa yaitu: (1) handphone Penerima pesan perintah dari *reseller* untuk diteruskan ke s*oftwar*e aplikasi agar dilakukan proses *parsing*, (2) modem/Handphone tempat stok pulsa dari operator seluler yang akan diambil oleh reseler melalui aplikasi, (3) *handphone* pengirim yang bertugas untuk mengirimkan laporan pengisian pulsa kepada r*eseller*, (4) aplikasi i pulsa yang bertugas untuk melakukan parsing kode sesuai permintaan reseler dan ketersediaan stok pulsa.

3.2 Sistem Finansial *Server* Pulsa *Reload*

Pada Aplikasi *server reload* umumnya terdapat sistem perhitungan keuangan namun masih terbatas pada posisi saldo dan jumlah laba/kerugian kotor. Sementara kebutuhan akan perhitungan keuangan lengkap sangat diperlukan untuk mengetahui informasi-informasi penting seperti posisi hutang pelanggan, jumlah piutang, laba bersih, jumlah kesalahan pengisian operator yang disebabkan *human error*. Oleh karena itu diperlukan aplikasi pengembangan untuk meneruskan proses perhitungan keuangan yang bersumber pada *database server* sehingga dapat ter-update secara *real-time.*

BAB 4

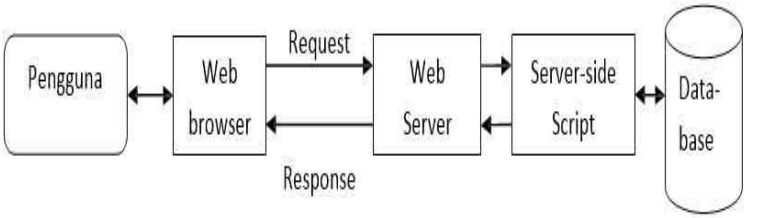
KONSEP APLIKASI BERBASIS WEBSITE

4.1 Pengertian dan Konsep Website

Merujuk pendapat Defrizal (2006) bahwa *World Wide Web* (WWW atau Web1) merupakan salah satu "*killer applications*" yang menyebabkan populernya Internet. Kehebatan Web adalah kemudahannya untuk mengakses informasi, yang dihubungkan satu dengan lainnya melalui konsep *hypertext*. Dan juga menurut sebuah survey pengguna WWW yang diadakan oleh *Graphics, Visualization, and Usability* (GVU) Center, 84% pengguna menganggap akses WWW tidak tergantikan.

Badan usaha komersial, badan penelitian, kampus atau organisasi pengguna jaringan komputer lainnya mulai menggunakan WWW sebagai basis dari aplikasi–aplikasi penting dan kritikal dalam kegiatan operasionalnya. Melihat fenomena tersebut, maka dapat disimpulkan bahwa keberadaan teknologi WWW memang sangat menentukan bagi kemajuan dunia internet yang berdampak langsung pada pesatnya penyebaran informasi di dunia.

Web merupakan dokumen visual di Internet dengan fasilitas hiperteks untuk menampilkan data berupa teks, gambar, bunyi, animasi dan data multimedia lainnya. Untuk menampilkan halaman Web digunakan program aplikasi *web browser* yang umumnya terdapat pada sistem operasi komputer yang berbasis visual seperti yang terlihat pada Gambar 4.1.



Gambar 4.1 Konsep Website

Web dinamis memiliki kemampuan layaknya pemrograman dan kemampuan untuk dapat mengakses *database* yang terdapat pada *server*. Lebih lanjut web dinamis dikembangkan sebagai aplikasi berbasis web yang umumnya pada lingkungan intranet. Dengan kemampuan pemrograman pada web, pengembangan web komunitas berkembangan pesat.

Portal web (*website portal*) adalah sebuah situs web yang menyediakan beragam informasi dari berbagai sumber dengan cara (*format/layout*) yang seragam. Biasanya, setiap sumber informasi mendapat area khusus pada halaman *website portal* dalam menampilkan informasi. Biasanya pengguna *website portal* dapat mengkonfigurasi informasi mana yang akan ditampilkan. Terlepas dari fitur mesin pencari standar pada website portal, portal web juga menawarkan layanan lainnya seperti e-mail, berita, harga saham, informasi, *database* dan hiburan. Portal web menyediakan dan memberikan tampilan yang konsisten dan juga memberikan kontrol akses dan prosedur untuk beberapa aplikasi dan *database* (www.websiteflashmedia.com).

Komunitas atau Jejaring sosial (*social network*) adalah bentuk struktur sosial yang terdiri dari simpul-simpul yang saling terkait dan terikat oleh satu atau lebih tipe hubungan yang spesifik. Simpul-simpul yang dimaksudkan disini dapat berupa individu maupun organisasi. (www.wikipedia.com). Jejaring social yang berbentuk website antara lain www.facebook.com, www.twitter.com.

Beberapa penelitian tentang web komunitas antara lain (Marisa, 2010) menghasilkan web komunitas untuk sivitas akademika perguruan tinggi untuk memberi wadah komunikasi antar mahasiswa, dosen, maupun staf, (Hardana, et al., 2014) menghasilkan aplikasi web komunitas untuk mewadahi komunitas para dosen selaku peneliti, (Mardiyanto, et al., 2012) menghasilkan rancang bangun *website* pertemanan untuk memfasilitasi mahasiswa pendidikan jarang jauh untuk saling berkomunikasi, dan masih banyak lagi bidang yang menggunakan web sebagai media informasi.

4.2 Konsep Aplikasi Berbasis *Website*

Keberadaan smartphone android menjadi salah satu alat komunikasi yang banyak digunakan masyarakat. Beberapa keuntungan samrtphone android antara lain jarga terjangkau, mudah mengoperasikan, serta dukungan open source yang membuat perkembangan software maupun fitur android berkembang sangat pesat (Ragil, 2013). Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak.

Ardiansyah (2013) menggolongkan sejarah perkembangan sistem operasi Android sejak diperkenalkan ke publik pada tanggal 5 November 2007 adalah sebagai berikut 1) Android Beta, 2) Android versi 1.x, 3) Android 1.1 Bender, 4) Android 1.5 Cupcake, 5) Android 1.6 Donut, 6) Android versi 2.x, 7) Android 2.2 Froyo atau Frozen Yoghurt, 8) Android 2.3 Gingerbread , 9) Android versi 3.x, 10) Android versi 4.x, 11) Android 4.1 Jelly Bean, 12) Android 4.2 Jelly Bean , 13) Android 4.3 Jelly Bean, 14) Android 4.4 KitKat , 15) Android versi 5.x

Banyak penelitian tentang aplikasi Android, dimana umumnya aplikasi yang dibuat untuk kebutuhan sehari-hari, antara lain penelitian yang menghasilkan aplikasi cek tagihan listrik yang ditanam playstore sehingga dapat digunakan oleh siapapun (Agan & Susanto, 2013), menghasilkan aplikasi *Geographic Information System* (GIS) yang berjalan dalam Android (Harjo, 2011), Aplikasi kamus bahasa Sunda menggunakan Android (Moharom, et al., 2013), aplikasi penghitungan biaya listrik menggunakan android (Ragil, 2013) dan aplikasi sosial media yang dilengkapi dengan *geotagging* untuk komunitas kampus (Brata, et al., 2012). Maka dengan banyaknya aplikasi yang ada, dimungkinkan untuk dapat dibangun aplikasi *Tracer Study* dalam platform Andorid.

4.3 Teknologi Web Service

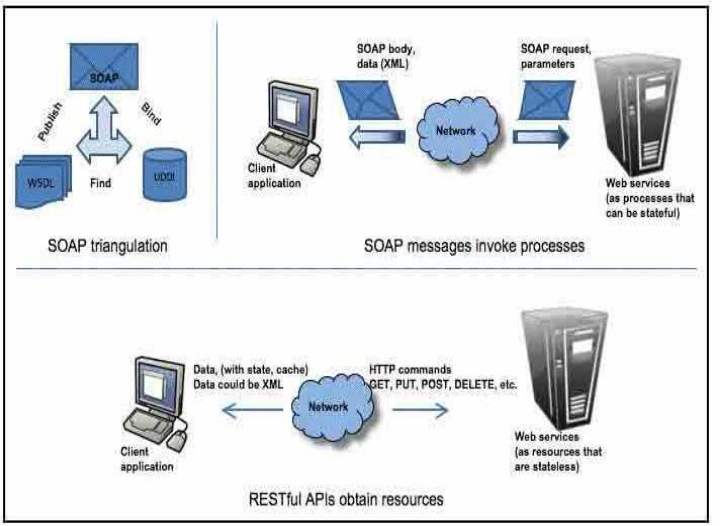
Web Service adalah konsep distribusi yang menyediakan informasi melalui web dengan menggunakan format XML dengan standar protokol HTTP. Web service merupakan kunci integrasi untuk aplikasi-aplikasi yang berbeda platform, bahasa, dan sistem. Dengan kata lain kita dapat menyebut web service sebagai "titik temu bisnis" (Natsir, et al., 2013).

Ada perbedaan – perbedaan mendasar antara *web service* dengan *web site*. Perbedaan tersebut dapat dilihat pada tabel perbandingan yang tercantum pada Tabel 4.1.

Web Site	*Web Service*
1. Memiliki *web interface*	1. Tidak memiliki *interface* yang bagus
2.Dibuat untuk berinteraksi langsung dengan *user*	2. Dibuat untuk berinteraksi langsung dengan applikasi yang lain baik beda *Operation system* (OS)/ Konsep sekalipun.
3. Dibuat untuk bekerja pada *web browser.*	3. Dibuat untuk bekerja pada semua tipe client applikasi / perangkat *device*

Tabel 4.1. Perbandingan *Web Site* dan *Web Service*

Pada *web service*, diperlukan pengelolaan dan interkoneksi *database* dan URL dengan menggunakan mekanisme SOAP dan REST seperti yang terlihat pada Gambar 4.2. *Simple Object Access Protocol* SOAP). SOAP adalah protokol untuk pertukaran informasi dengan desentralisasi dan terdistribusi. Peran SOAP di dalam teknologi *web service* adalah sebagai protokol pemaketan untuk pesan-pesan yang digunakan secara bersama oleh aplikasi-aplikasi penggunanya. Sedangkan *REpresentational State Transfer* (REST) merupakan sebuah Teknik pada arsitektur software untuk sistem terdistribusi seperti World Web Wide.



Sumber Wicaksono, et al., 2014
Gambar 4.2 Mekanisme SOAP dan REST

4.4 *Script web dan .Net*

Pembahasan ini akan diawali dengan karakteristik dan dasar pemrograman *Personal Home Page* (PHP). Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (Situs personal). PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak pemrogram yang tertarik untuk ikut mengembangkan PHP.

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.

2. Web *Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.
5. PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system.

Sintak Dasar PHP.

```
<html>
<head>
<title>contoh 1 </title>
</head>
<body>
<? php
// INI CONTOH format mencetak dengan print
print "CONTOH MENCETAK menggunakan print";
//INI CONTOH memberlakukan perintah HTML dalam PHP
echo "<br>";
// INI CONTOH format mencetak dengan print
echo "CONTOH MENCETAK menggunakan echo";
?>
</body>
</html>
```

Deklarasi Variable.

```
<html>
<head>
<title>contoh 1 </title>
</head>
<body>
<? php
// INI CONTOH format mencetak dengan print
print "CONTOH MENCETAK menggunakan print";
```

BAB 5

PERANCANGAN *QUERY REALTIME* APLIKASI WEB

Salah satu metodologi yang digunakan dalam perancangan *query real time* untuk aplikasi web adalah dengan menggunakan pendekatan *System Development Life Cycle* (SDLC). Dalam SDLC ada lima tahapan utama yaitu *planning*, *analysis, design, implementation,* dan *support/maintenance*. Pada tahap perencanaan, biasanya suatu organisaasi akan berfokus pada pengenalan permasalahan. Selanjutnya dilakukan investigasi, pemahaman masalah dan pengenalan kebutuhan atau *solution requirements* pada tahap analisis. Pada tahap perancangan, solusi perbaikan akan didesain, atau merancangan sistem usulan berdasarkan *solution requirement*. Solusi juga akan dispesifikasikan secara rinci pada tahap ini. Sistem baru yang dapat menyelesaikan masalah dibangun dan diinstal pada tahap implementasi. Akhirnya, sistem akan digunakan oleh *user,* dirawat, dan dikembangkan terus/*enhanced* pada tahap *support/maintenance*.

Kegiatan penelitian ini dimulai dengan pengkajian literatur tentang query *database,* karakteristik .Net, SQL *Server express* dan MySQL. Selanjutnya dilakukan analisis alur *database* Otomax yang berada di SQL *Server Express*. Kemudian disusun model *Query database* dalam bentuk perintah-perintah *query* untuk mengambil *field-filed* yang diperlukan untuk pengolahan keuangan. Setelah itu model diimplementasikan dengan membuat aplikasi keuangan. Hasil dari teknik *query* yang diperoleh adalah berupa data yang selalu tersaji secara update sesuai data dalam SQL server express otomax tersimpan dalam MySQL yang merupakan bahan yang diolah dalam aplikasi keuangan.

5.1 Penjelasan Metodologi Peracangan

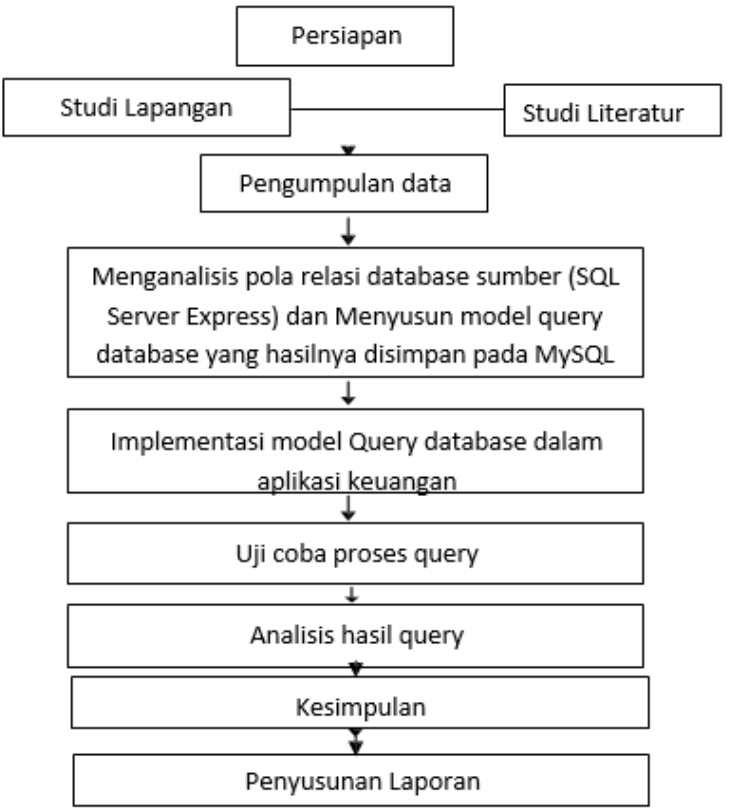
Tahapan-tahapan penelitian ini adalah sebagaimana ditampilan dalam diagram alir penelitian pada Gambar 5.1.

Tahapan rancangan penelitian dijabarkan sebagai berikut:

1. Persiapan

   Kegiatan pada tahapan ini terdiri dari:

   a. Pengurusan ijin penelitian kepada pihak pengusana *server* AR-PulsaBiz di wilayah Malang.
   b. Perusahaan *Server Reload* AR-PulsaBiz sebagai obyek penelitian.
   c. Pengurusan ijin peminjaman Laboratorium Teknik Informatika Universitas Widyagama Malang sebagai tempat penelitian. Proses penyiapan bahan dan peralatan meliputi:
      - Komputer operasional
      - Jaringan: LAN dan Internet
      - Software: Sistem Operasi, C# dan microsoft framework, SQL *Server* Express, Visual Basic 6, Mozilla firefox, XAMPP (Apache, MySql)



Gambar 5.1 Diagram Alir Metodologi Penelitian

2. Pengumpulan data
   Meliputi kegiatan mengumpulkan semua kebutuhan data dalam penelitian, antara lain:
   a. Tujuan dibangun aplikasi keuangan GL realtime bagi *server* pulsa.
   b. Selanjutnya mengidentifikasi kebutuhan data-data yang meliputi pencatatan dan mekanisme *server* pulsa *reload*
3. Pembuatan model Teknik *Query* yang meliputi kegiatan sebagai berikut:
   a. Menganalisis relasi *database* sumber.
   b. Mencatat kunci-kunci penting dalam kebutuhan data dalam sistem keuangan yang akan dibuat.

c. Membuat rumus-rumus query dalam proses pengambilan field *yang* dibutuhkan.

4. Implementasi model *query database real-time* dengan aplikasi keuangan.
   Berdasarkan model yang sudah dibuat selanjutnya dilakukan proses perancangan aplikasi keuangan dengan bahasa Visual Basic 6.

5. Uji coba aplikasi finansial yang difokuskan pada proses *query* pengambilan *database real-time*.
   Pengujian sistem dilakukan dengan mengoperasikan aplikasi dengan 20 user di laboratorium. *User* melakukan proses posting data untuk menguji keberhasilan *query.*

6. Analisis Hasil Pengukuran.
   Setelah dilakukan proses posting transaksi, selanjutnya hasil posting dilihat dalam *database* MySql. Hasil dalam MySQL dibandingkan dengan data sumber dengan mengambil sample secara acak melihat data saldo 5 *reseller* yang terdaftar dalam program. Jika hasil yang ada dilaporan keuangan sama dengan yanga da dalam *database* sumber maka hasil dianggap valid.
   a. Lokasi Penelitian
      Penelitian ini dilakukan pada obyek penelitian CV ARPulsaBiz yang berlokasi di kabupaten Malang.
   b. Variabel yang diamati
      Variabel yang diamati mengacu pada kerangka perspektif query *database* khususnya keuangan dalam *server* pulsa, yaitu:
   c. Data *Reseller* (kode, dan nama *reseller*, posisi saldo, jumlah hutang dan piutang)
   d. Kehandalan proses *query,* dengan patokan ketepatan dan kecepatan proses *query*.
   e. Model yang digunakan

Model yang digunakan dalam penyelesaian permasalahan adalah dengan pendekatan *query database* yang diimplementasikan secara *real-time*.

f. Rancangan Penelitian

Rancangan penelitian ini adalah penelitian kasus yang dikembangkan dalam bentuk model query *database* dalam bentuk perintah query dan aplikasi finansial *server* pulsa *reload* yang merupakan obyek tempat model query diimplementasikan.

g. Teknik pengumpulan data

Dalam proses pengumpulan data, digunakan teknik:

i. Wawancara, untuk mempelajari data-data mekanisme *server* dan keuangannya.

ii. Dokumentasi, untuk memperoleh data form kebutuhan dalam pengolahan keuangan bagi *reseller* seperti pembayaran, permintaan saldo, dan hutang piutang.

h. Analisis data

Analisis data dilakukan dengan melakukan proses *posting* data hutang dan piutang reseller, kemudian dicari konsistensinya dengan data sumber, sebagai contoh mengecek posisi saldo beberapa *user*, jika data konsisten dengan data sumber maka proses *query* dianggap valid. Selanjutnya analisis data digunakan untuk menyimpulkan kehandalan *query* yang sudah dibuat.

9. Kesimpulan

Dari hasil analisis maka dapat disimpulkan mengenai model *query database realtime* yang sudah dibangun, seberapa jauh mengefektifkan proses penghitungan finansial *server* pulsa *reload* yang berbasis .Net.

10. Penyusunan Laporan

Setelah proses penelitian dilakukan selanjutnya dibuat laporan hasil penelitian berupa buku laporan, sinopsis penelitian lanjut, jurnal.

5.2 Hasil Perancangan

Pada bagian ini akan dijelaskan beberapa hal penting berkenaan dengan hasil penelitian antara lain adalah membuat model *query realtime*, perancangan Aplikasi keuangan *server* pulsa, dan implementasi serta pengujian sistem. Perincian pada tahap ini meliputi (1) desain model dalam bentuk formula *query,* (2) perancangan keuangan *server* pulsa dalam bentuk bagan *Data Flow Diagram* (DFD) dan desain basis data, (3) Implementasi model *query* dalam aplikasi keuangan *server* pulsa. Pembahasan ketiga tahap perancangan secara rinci akan disajikan pada tiap sub bab selanjutnya.

5.2.1 Desain Model dan Bentuk Formula

Desain model *query real-time* yang diterjemahkan dalam bentuk formula query dari aplikasi keuangan *server* pulsa. Bentuk formula dapat dilihat pada Gambar 5.2.

```
//Model Untuk deposit reseller.

cSql = "Select Kode,Kode_Reseller,Tanggal,Jumlah"
 cSql = cSql & " From Mutasi"
 cSql = cSql & " Where Keterangan Like 'Deposit%'"
 cSql = cSql & " And Year(Tanggal)>=" & Mid(dTgl.Value, 1, 4) & ""
 cSql = cSql & " And Month(Tanggal)>=" & Mid(dTgl.Value, 6, 2) & ""
 cSql = cSql & " And Day(Tanggal)>=" & Mid(dTgl.Value, 9, 2) & ""
 cSql = cSql & " And Kode_Reseller Like '%" & cKode.Text & "%'"
 cSql = cSql & " Order By Kode"
 Set dbOtomax = objOtomax.SQL("PulsaBiz", cSql)
 If Not dbOtomax.EOF Then
  PrgBar.Max = dbOtomax.RecordCount
  PrgBar.Value = 0
  While Not dbOtomax.EOF
   PrgBar.Value = PrgBar.Value + 1
   PrgBar.Refresh
   lbCaption.Caption = UCase(dbOtomax!Kode_Reseller)
   lbCaption.Refresh
```

```
    ObjData.Update GetDsn, "Deposit_Reseller", "Faktur='" &
dbOtomax!Kode & "'", _
    Array("Faktur", "Tgl", "TglJam", "Reseller", "Jumlah", "Sisa", _
    "Operator", "Waktu"), _
```

```
cSql = "SELECT     a.bank, a.tgl_proses, a.jumlah, a.keterangan,
a.kode_tiket, c.kode_reseller, d.nama"
 cSql = cSql & " FROM        data_bank AS a LEFT OUTER JOIN"
 cSql = cSql & "    tiket_deposit AS b ON a.kode_tiket = b.kode LEFT
OUTER JOIN"
 cSql = cSql & "                inbox AS c ON b.kode_inbox = c.kode LEFT
OUTER JOIN"
 cSql = cSql & "                reseller AS d ON c.kode_reseller = d.kode"
 cSql = cSql & " WHERE     (a.kode_tiket <> '')"
 cSql = cSql & " And Year(a.tgl_proses)=" & Mid(dTgl.Value, 1, 4) & ""
 cSql = cSql & " And Month(a.tgl_proses)=" & Mid(dTgl.Value, 6, 2) &
""
 cSql = cSql & " And Day(a.tgl_proses)=" & Mid(dTgl.Value, 9, 2) & ""
 cSql = cSql & " Order By a.tgl_proses"

 Set dbOtomax = objOtomax.SQL("PulsaBiz", cSql)
 If Not dbOtomax.EOF Then
  PrgBar.Max = dbOtomax.RecordCount
  PrgBar.Value = 0
  While Not dbOtomax.EOF
   PrgBar.Value = PrgBar.Value + 1
   PrgBar.Refresh
   lbCaption.Caption = UCase(dbOtomax!Nama)
   lbCaption.Refresh
```

Gambar 5.2 Desain Model Formula Query Realtime *Database*

Beberapa kelompok skrip *database* yang penting dapat disajikan dalam Gambar 5.3 dan Gambar 5.4

```
cSql = "Select Kode,Kode_Reseller,Tanggal,Jumlah"
  cSql = cSql & " From Mutasi"
  cSql = cSql & " Where Keterangan Like 'Deposit%'"
  cSql = cSql & " And Year(Tanggal)>=" & Mid(dTgl.Value, 1, 4) & ""
  cSql = cSql & " And Month(Tanggal)>=" & Mid(dTgl.Value, 6, 2) & ""
  cSql = cSql & " And Day(Tanggal)>=" & Mid(dTgl.Value, 9, 2) & ""
  cSql = cSql & " And Kode_Reseller Like '%" & cKode.Text & "%'"
  cSql = cSql & " Order By Kode"
  Set dbOtomax = objOtomax.SQL("PulsaBiz", cSql)
  If Not dbOtomax.EOF Then
    PrgBar.Max = dbOtomax.RecordCount
    PrgBar.Value = 0
    While Not dbOtomax.EOF
      PrgBar.Value = PrgBar.Value + 1
      PrgBar.Refresh
      lbCaption.Caption = UCase(dbOtomax!Kode_Reseller)
      lbCaption.Refresh

      ObjData.Update GetDsn, "Deposit_Reseller", "Faktur='" & dbOtomax!Kode & "'", _
      Array("Faktur", "Tgl", "TglJam", "Reseller", "Jumlah", "Sisa", _
      "Operator", "Waktu"), _
      Array(dbOtomax!Kode, dbOtomax!Tanggal, dbOtomax!Tanggal, UCase(dbOtomax!Kode_Reseller),
     | dbOtomax!Jumlah, "&Jumlah-Lunas", _
      pbOperator, GetWaktu)

      dbOtomax.MoveNext
    Wend
  End If
```

Gambar 5.3 Skrip Visual basic untuk *Query Real-time*

5.3 Perancangan Aplikasi Keuangan *Server* Pulsa

Dalam tahap ini dilakukan proses perancangan aplikasi keuangan *server* pulsa *reload* dalam format desain sistem informasi terstruktur menggunakan *Context Diagram* dan *Data Flow Diagram* (DFD). Gambar 5.5 adalah *context diagram* dari aplikasi yang telah dibangun.

```
cSql = "SELECT     a.bank, a.tgl_proses, a.jumlah, a.keterangan, a.kode_tiket, c.kode_reseller, d.nama"
 cSql = cSql & " FROM         data_bank AS a LEFT OUTER JOIN"
 cSql = cSql & "    tiket_deposit AS b ON a.kode_tiket = b.kode LEFT OUTER JOIN"
 cSql = cSql & "                      inbox AS c ON b.kode_inbox = c.kode LEFT OUTER JOIN"
 cSql = cSql & "                      reseller AS d ON c.kode_reseller = d.kode"
 cSql = cSql & " WHERE     (a.kode_tiket <> '')"
 cSql = cSql & " And Year(a.tgl_proses)=" & Mid(dTgl.Value, 1, 4) & ""
 cSql = cSql & " And Month(a.tgl_proses)=" & Mid(dTgl.Value, 6, 2) & ""
 cSql = cSql & " And Day(a.tgl_proses)=" & Mid(dTgl.Value, 9, 2) & ""
 cSql = cSql & " Order By a.tgl_proses"

 Set dbOtomax = objOtomax.SQL("PulsaBiz", cSql)
 If Not dbOtomax.EOF Then
   PrgBar.Max = dbOtomax.RecordCount
   PrgBar.Value = 0
   While Not dbOtomax.EOF
     PrgBar.Value = PrgBar.Value + 1
     PrgBar.Refresh
     lbCaption.Caption = UCase(dbOtomax!Nama)
     lbCaption.Refresh

     UpdRekTiketDeposit ObjData, dbOtomax!Kode_Tiket, dbOtomax!Bank, dbOtomax!Tgl_Proses,
     dbOtomax!Kode_Reseller & " : " & dbOtomax!Nama, dbOtomax!Jumlah

     dbOtomax.MoveNext
   Wend
 End If
End Sub
```

Gambar 5.4 Skrip *Query Real-Time* dalam Visual Basic lanjutan

Data flow diagram (DFD) atau diagram alir data adalah alat bantu atau *tools* untuk memodelkan proses dengan menggambarkan aliran data dalam suatu sistem serta aktivitas atau pemrosesan yang dilakukan oleh sistem tersebut. Diagram aliran data dipopulerkan pada akhir 1970-an, yang muncul dari buku Structured Design, oleh pelopor komputasi Ed Yourdon dan Larry Constantine. Mereka mendasarkannya pada model komputasi "grafik aliran data" oleh David Martin dan Gerald Estrin. Konsep desain terstruktur lepas landas di bidang rekayasa perangkat lunak, dan metode DFD juga berkembang pesat. Ini menjadi lebih populer di kalangan bisnis, karena diterapkan pada analisis bisnis, daripada di kalangan akademis.

Tiga sistem simbol yang umum dinamai menurut penciptanya:

- Yourdon dan Coad
- Yourdon dan DeMarco
- Gane dan Sarson

Satu perbedaan utama dalam simbol mereka adalah bahwa Yourdon-Coad dan Yourdon-DeMarco menggunakan lingkaran untuk proses, sementara Gane dan Sarson menggunakan persegi panjang dengan sudut membulat, kadang-kadang disebut pelega tenggorokan. Ada variasi simbol lain yang digunakan juga, jadi hal penting yang harus diingat adalah jelas dan konsisten dalam bentuk dan notasi yang anda gunakan untuk berkomunikasi dan berkolaborasi dengan orang lain.

Menggunakan aturan atau pedoman DFD konvensi apa pun, simbol menggambarkan empat komponen diagram aliran data yaitu:

- *Entitas eksternal*: sistem luar yang mengirim atau menerima data, berkomunikasi dengan sistem yang digambarkan. Mereka adalah sumber dan tujuan informasi yang masuk atau keluar dari sistem. Mereka mungkin organisasi atau orang luar, sistem komputer atau sistem bisnis. Mereka juga dikenal sebagai terminator, sumber dan tenggelam atau aktor. Mereka biasanya digambar di tepi diagram.
- *Proses*: setiap proses yang mengubah data, menghasilkan output. Itu mungkin melakukan perhitungan, atau mengurutkan data berdasarkan logika, atau mengarahkan aliran data berdasarkan aturan bisnis. Label pendek digunakan untuk menjelaskan prosesnya, seperti “Kirim pembayaran”.
- *Penyimpanan data*: file atau repositori yang menyimpan informasi untuk digunakan nanti, seperti tabel *database* atau formulir keanggotaan. Setiap penyimpanan data menerima label sederhana, seperti "Pesanan".
- *Aliran data*: rute yang diambil data antara entitas eksternal, proses, dan penyimpanan data. Ini menggambarkan antarmuka antara komponen lain dan

ditampilkan dengan panah, biasanya diberi label dengan nama data pendek, seperti "Detail penagihan".

Beberapa konsep yang berkontribusi pada DFD antara lain:

- *Object Oriented Analysis and Design* (OOAD), dikemukakan oleh Yourdon dan Peter Coad untuk menganalisis dan mendesain sebuah aplikasi atau sistem.
- Metode *Structured Systems Analysis and Design* (SSADM), sebuah metode waterfall untuk menganalisis dan merancang sistem informasi. Pendekatan dokumentasi yang ketat ini kontras dengan pendekatan tangkas modern seperti Scrum dan Metode Pengembangan Sistem Dinamis (DSDM.)
- Tiga pakar lain yang berkontribusi terhadap peningkatan metodologi DFD ini adalah Tom DeMarco, Chris Gane, dan Trish Sarson. Mereka bekerja sama dalam kombinasi yang berbeda untuk menjadi penentu utama dari simbol dan notasi yang digunakan untuk diagram aliran data.

Simbol atau notasi dalam DFD ada empat buah yaitu: proses, simpanan data, entitas luar, dan aliran data (dapat dilihat pada Tabel 5.1).

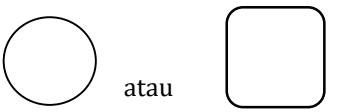
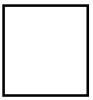
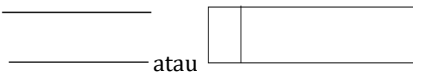
Proses merupakan pekerjaan atau kegiatan yang dilakukan terhadap data yang masuk (input) untuk menghasilkan data keluaran (output). Proses hanya menunjukkan kegiatannya saja, tidak merujuk orang/pihak yang melakukan, contoh: proses pembayaran, proses cetak kartu rencana studi (KRS), dan sebagainya.

Entitas luar merupakan pihak di dalam atau di luar organisasi (bisa orang atau departemen atau organisasi) yang mengirimkan input ke sistem atau menerima output dari sistem, menunjukkan batasan dari sistem dan hal ini tidak sama dengan entitas dalam basis data. Contoh: Bank, Sekretariat, kepala program studi, Biro administrasi akademik, dan lain-lain.

Simpanan data merupakan kumpulan data, baik berupa file atau basis data tapi tidak selalu berupa file atau basis data, Contoh: file Mahasiswa, data KRS, dan sebagainya,

Aliran data menyatakan data masukan ke suatu proses atau data keluaran dari suatu proses, Aliran data juga menyatakan update data dalam suatu file, basis data atau simpanan data yang lain. contoh: Matakuliah yang diambil, jumlah yang sks, jumlah pembayaran semesteran. Saldo terakhir, dan lain-lain.

DFD digunakan untuk secara grafis mewakili aliran data dalam sistem informasi bisnis. DFD menggambarkan proses yang terlibat dalam sistem untuk mentransfer data dari input ke penyimpanan file dan pembuatan laporan.

Simbol	Arti
	Proses
	Entitas luar
	Simpanan data
	Aliran data

Tabel 5.1 Simbol atau Notasi dari DFD

DFD dapat dibagi menjadi logis dan fisik. Diagram aliran data logis menggambarkan aliran data melalui sistem untuk melakukan fungsionalitas tertentu dari suatu bisnis. Diagram aliran data fisik menggambarkan implementasi aliran data logis.

DFD secara grafis mewakili fungsi, atau proses, yang menangkap, memanipulasi, menyimpan, dan mendistribusikan data antara sistem dan lingkungannya dan antara komponen

sistem. Representasi visual menjadikannya alat komunikasi yang baik antara Pengguna dan perancang Sistem. Struktur DFD memungkinkan mulai dari gambaran umum dan memperluasnya ke hierarki diagram terperinci. DFD sering digunakan karena alasan berikut:

1. Aliran informasi logis dari sistem
2. Penentuan persyaratan konstruksi sistem fisik
3. Kesederhanaan notasi
4. Penetapan persyaratan sistem manual dan otomatis.

Langkah-langkah dalam membuat DFD adalah sebagai berikut:

1. Identifikasi entitas dalam dan luar yang terlibat dalam sistem
2. Identifikasi semua input dan output yang berhubungan dengan entitas tersebut.
3. Gambarkan diagram konteksnya atau *context diagram* (DFD level yang paling atas)
4. Jika perlu gambarkan diagram berjenjang (diagram dekomposisi)
5. Identifikasi simpanan data.
6. Gambarkan DFD untuk level 0
7. Jika perlu gambarkan DAD untuk level 1 sampai dengan level yang dibutuhkan.

Adapun atura-aturan dalam pembuatan DFD adalah sebagi berikut:

1. Minimal salah satu ujung suatu aliran data adalah proses. Jadi tidak boleh ada aliran data:
   - dari entitas ke entitas
   - dari entitas ke simpanan data
   - dari simpanan data ke entitas
   - dari simpanan data ke simpanan data
2. Data yang mengalir dalam setiap level DAD harus konsisten (jumlah data masuk dan keluar dalam suatu proses harus konsisten),

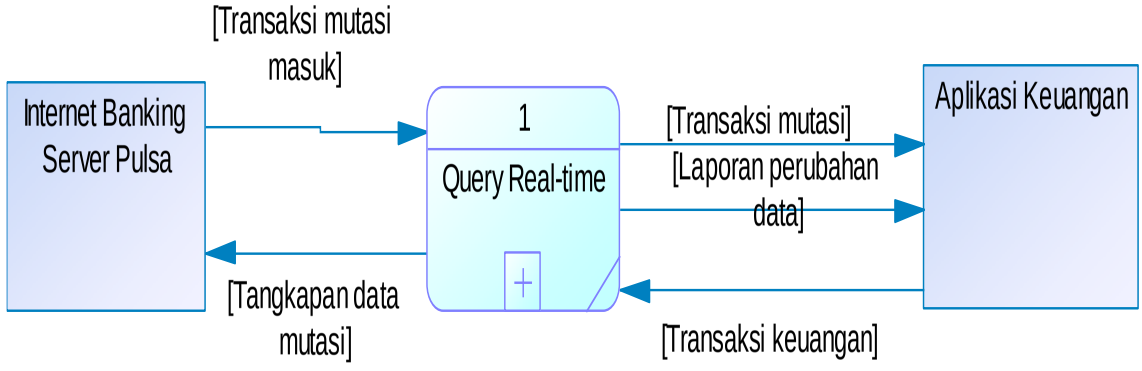
3. Suatu proses harus punya output
   Blackhole: proses tanpa output
4. Suatu proses harus punya input
   Miracle: proses tanpa input
5. Suatu proses harus punya cukup input untuk menghasilkan output,
   Gray hole: proses tanpa input memadai untuk menghasilkan output
6. Proses-proses yang hanya melewatkan data tanpa melakukan pemrosesan thd data tersebut sebaiknya tidak digambarkan
7. Data yang berasal dari sumber yang sama dan mengalir dengan tujuan yang sama dapat digambarkan dalam satu aliran data (aliran data komposit)
8. Jangan gunakan aliran data menyebar untuk DAD yang penting. Sebaiknya aliran data dipisahkan menurut komponen-komponennya
9. Sebaiknya simpanan data diberi nama sesuai dengan nama yang dipakai dalam model data (diagram ER). Kata DATA tidak perlu dipakai.
10. Untuk mengurangi kompleksitas gambar, simbol-simbol proses, entitas, simpanan data dapat dibuat duplikatnya,

Keuntungan penggunaan DFD sebagai pemodelan proses adalah sebagai berikut:

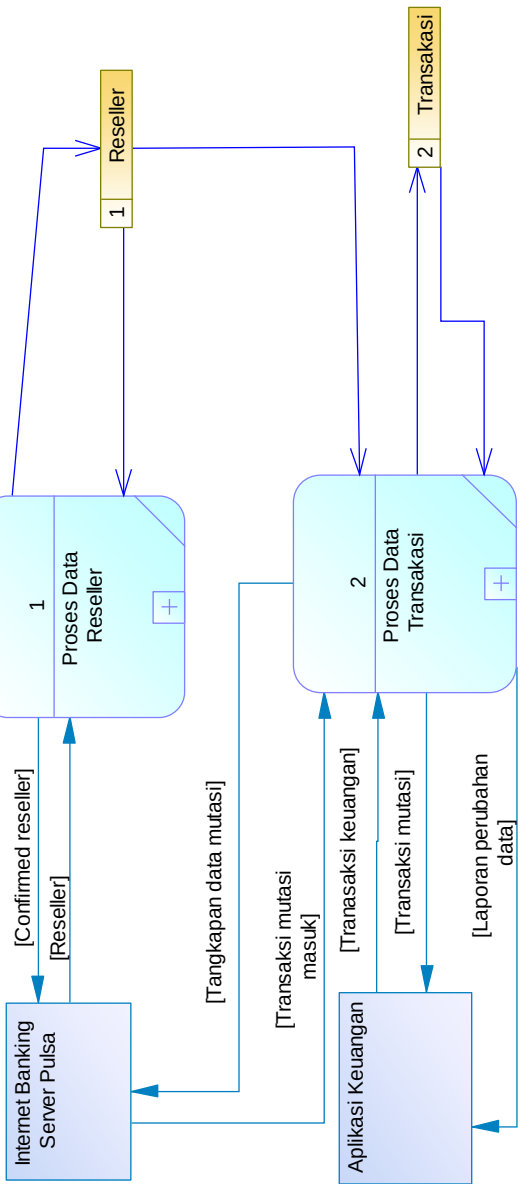
- Membantu memahami implementasi teknis sistem terlalu dini
- Membantu untuk memahami keterkaitan sistem dan subsistem
- Membantu dalam mengkomunikasikan pengetahuan sistem saat ini kepada pengguna
- Membantu menganalisis sistem.

Dalam *context diagram* terdapat dua entitas yaitu internet banking dari sever pulsa dan aplikasi keuangan. Kedua entitas berinteraksi dalam satu sistem *query real time.* Internet banking melakukan transaksi mutasi masuk dan menangkap mutasi. Sementara aplikasi keuangan melakukan posting transaksi,

menerima hasil posting dan laporan perubahan data. Proses ini kemudian dijabarkan lebih detil dalam DFD level 0 yang ada pada Gambar 5.6



Gambar 5.5 Desain *Context Diagram*

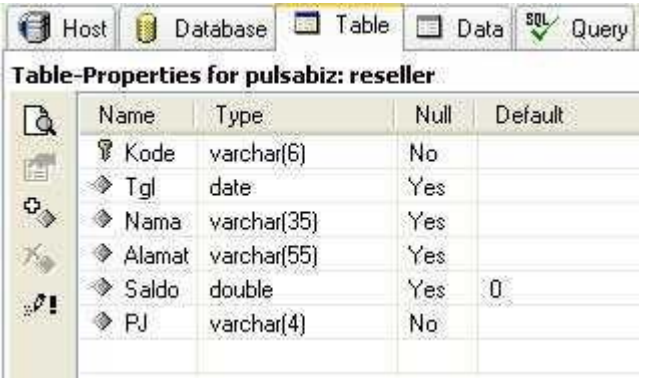


Gambar 5.6 Data *Flow Diagram* (DFD) Level 0 Aplikasi Real-*time* *Server* Pulsa *Reload*

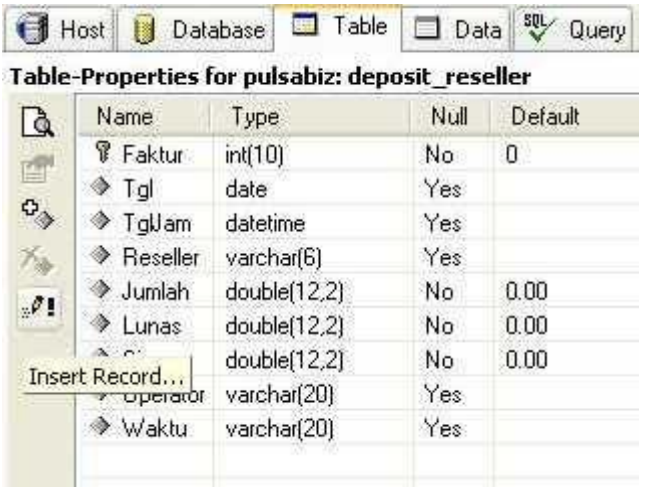
Dalam DFD, proses dalam sistem dipecah menjadi dua bagian yaitu proses data *reseller* dan proses data transaksi. Proses data *reseller* mengelola data transaksi *reseller y*ang disimpan dalam tabel reseler. Sedangkan proses data transaksi melibatkan data mutasi keuangan yang dismpan dalan tabel transaksi. Selain tabel reseler dan transaksi juga terdapat tabel buku besar yang terlibat secara tidak langsung dalam sistem. Namun tabel ini selalu terkait dengan setiap transaksi yang dilakukan dalam sistem.

5.4 Desain *Database*

Tahap berikutnya adalah desain *database* dari aplikasi *server* pulsa *reload* Terdapat dua tabel penting dalam proses ini yaitu tabel *Reseller* (Gambar 5.7) dan tabel deposit_reseller (Gambar 5.8).



Gambar 5.7 Tabel *Reseller*



Gambar 5.8 Tabel Deposit_Reseller

5.5 Implementasi Model Query Realtime dalam Aplikasi Keuangan

Pada tahap ini adalah implementasi dari coding pemrograman aplikasi keuangan *server* pulsa *reload* Terdapat beberapa modul yang dijabarkan pada menu-menu yang terlihat pada Gambar 5,9 sampai dengan Gambar 5.12

Gambar 5.9 Menu Posting Transaksi

Gambar 5.8 merupakan form yang menyajikan menu posting transaksi yang berfungsi sebagai *refresh* terhadap *record* terakhir sehingga dihasilkan data terakhir dari mutasi saldo bank, sebagaimana dijelaskan dalam Gambar 5.10.

Gambar 5.10 Hasil Posting.

Kemudian dilakukan pengecekan jumlah saldo dan update mutasi masing-masing *reseller* seperti Gmbar 5.11 dan Gambar 5.12.

Gambar 5.11 Filter sesuai *Reseller*

Gambar 5.12 Hasil pengecekan.

Saat *user* sudah terekam proses pembayarannya melalui cara otomatis, maka di area gambar 14 disediakan pengisian pembayaran yang manual atau tidak melalui tiket otomatis. Hal ini bertujuan untuk melengkapi historis pembayaran *reseller*.

Dengan demikian dapat disimpulkan bahwa pengujian terhadap teknik *query real-time* berhasil diterapkan pada aplikasi keuangan pulsa *reload* dengan cara merekam update mutasi bank dalam *internet banking*.

BAB 6

PENUTUP

Dengan membangun model dan implementasi *query real-time database* pada keuangan *server* pulsa berbasis .Net dapat membantu *server* pulsa berbasis .Net dalam merekam data keuangan pulsa agar lebih cepat dan akurat

Beberapa saran yang dapat dipertimbangkan untuk pengembangan *query real-time* yaitu dibutuhkan mekanisme backup yang otomatis berkala sehingga tidak membebani *server* dengan historis mutasi keuangan setiap harinya. Penelitian ini masih banyak kelemahan antara lain untuk mengetahui hasil mutasi harus dilakukan proses posting transaksi, maka pada pengembangan penelitian berikutnya dapat menghilangkan proses mutasi agar mempercepat proses.

DAFTAR PUSTAKA

Cahyono, S. (2006) Panduan Praktis Pemrograman *Database* Menggunakan MySQL dan Java. Bandung: Informatika.

Dedianto. Sistem Trigger *Database* Pada SIAKAD Informatika, (2013). *Jurnal Sistem dan Teknologi Informasi (JusTin).* 1(1). http://jurnal.untan.ac.id/index.php/justin/article/view/924/859

Filipova, N. and Filipov, F. (2008). Development Of Database For Distributed Information Measurement And Control System University of Economics, Varna, Bul. Kniaz Boris I.

Issa, G. (2012). *Beginners Guide to C# and the .NET Micro Framework.* USA: GHI Elecktonic LLC.

Liu, B., (2007). *Web Data Mining, Exploring Hyperlinks, Content, and Usage Data.* 1st Ed. Heidelberg: Springer.

Mardiyanto, S., Setyowati, Y. & Asmara, R. (2012). Rancang Bangun Website Pertemanan Menggunakan AJAX Framework untuk komunitas PJJ. http://repo.pens.ac.id/560/ diakses 20 Juni 2014.

Marisa, F. (2008). Pengembangan Web Based Learning dalam Matakuliah Algoritma Pemrograman 1 di STMIK Pradnya Paramita Malang. *JITIKA,* 3(1), 13-18.

Marisa, F. (2010). Analisis Website kamus Bahasa Jawa Online untuk Pembelajaran Bahasa Jawa Bagi Masyarakat. *DInamika DotCom,* 1(10).

Marisa, F. (2010). Desain Web Personal Sebagai Sarana Penyampaian Informasi bagi Sivitas Akademika STMIK Pradnya Paramita. *Teknnologi Informasi,* 1(1), 63-74.

Marisa, F. (2013). Educational Data Mining (Konsep dan Penerapan). *Teknologi Informasi* , 4(2), 90-97.

Marisa, F., Efendi, D. U. & Mumpuni, I. D. (2016). Tracer Study System Portal-Based Social Network To Optimize Data Collection On Higher Education Graduates*. ICITB Proceeding*, 19-24.

Marisa, F. & Hardianto, A. (2014). Model dan Implementasi Metode Enkripsi Kombinasi MD5 ddan Skrip Pengolah String pada Fitur Layanan PMB Online. *Dinamika DotCom,* 5(2).

Mengenal Server Pulsa Otomax dan Kelebihannya (2021) diakses dari https://www.indotel.co.id/server-pulsa-otomax/ pada Juli 2022.

Moharom, A., Cahyana, R. & B. (2013). Pengembangan Aplikasi Sunda Berbasis Android Menggunakan Metode Rapid Application Development (RAD). *Jurnal Algoritma-STT Garut,* 10(1), 1-11.

Mustofa, D. (2013). Pengertian dan Fungsi QLQ *Server*. http://ilmukomputerdananalisis.blogspot.com/2012/12/pengertian-dan-fungsi- sql-*server*.html" diakses tanggal 16 April 2014.

Nadezhda, Filipova dan Ficho Filipov. "Development of *Database* for Distributed Information Measurement and Control System" 2008. University of Economic.

Narwati, (2010). Pengelompokan Mahasiswa Menggunakan K-Means. *Jurnal Dinamika Informatika,* 2(2). https://doi.org/10.35315/informatika.v2i2.890

Natsir, F., Tamrin, H. & Rakhmadani, A., (2013). Implementasi Web Service pada Aplikasi Kamus Bahasa Indonesia. *IKOMUNITY,* 1(1), 27-36.

Octafian, D.T. (2011). Desain *Database* Sistem Informasi Pendualan Barang. *Jurnal Teknologi dan Informatika (TEKNOMATIKA)*. 1(2), 148-157.

Simarmata, J. & Paryudi, I. (2006). *Basis Data*. Yogyakarta: Andi Offset

Suhaidi, M. (2010). *Analisis dan Sistem Informasi Sistem Pengisia Pulsa Elektronik berbasis SMS (Studi kasus 4VR1.Cell Magelang.* Naskah publikasi. Amikom Yogyakarta. https://adoc.pub/analisis-dan-perancangan-sistem-informasi-pengisian-pulsa-el.html

Powell, T. (1998). *Web Site Engineering.* New Jersey: Prentice Hall.

Purnamasari, D. S. (2008). *Web Service Sebagai Solusi Integrasi Data Pada Sistem Informasi Akademik Universitas Bina Darma,* palembang: Universitas Bina Darma.

Ragil, O. (2013). *Rancang Bangun Aplikasi Android untuk Menghitung Biaya Listrik Rumah Tangga,* Skripsi, Universitas Negeri Semarang.

Renny, Chandra, R., Ruhama, S. & Sarjono, M. W. (2013). Exploring Indonesian Web Based Career Center Discrepancy of Web Popularity and Type of Services. *UACEE International Journal of Andvances in Computer Science and Its Applications,* 2(3), 212-216.

Tentang Penulis

Anastasia Lidya Maukar adalah dosen program studi Teknik Industri di Universitas Presiden, Cikarang, Jawa Barat. Penulis merupakan lulusan sarjana Teknik industri dari Universitas Surabaya, *Master of Science* untuk *Information System Development* dari University of Hertfordshire dan Magister Manajemen Teknik Industri dari Institut Teknologi Sepuluh Nopember, Surabaya. Bidang penelitian dan pengajaran dari penulis adalah tata letak pabrik atau fasilitas, basis data dan sistem informasi, statistik industri, serta sistem produksi.

Fitri Marisa adalah dosen bidang Ilmu Komputer di Universitas Widyagama Malang. Selama ini aktif mengajar bidang Datamining, Machine Learning dan Gamification. Penulis juga menjadi penulis di bidang pemrograman web dan book chapter dengan tema teknologi terbarukan. Penulis aktif sebagai peneliti baik skala nasional maupun Internasional dan pernah memenangkan *research grant* dari pemerintah India. Penulis kerap kali mendapatkan hibah penelitian dari kemenristekbrin dalam skim multi tahun.

Anik Vega Vitianingsih adalah Dosen Tetap Jurusan Informatika, Pemimpin Redaksi International Journal of Artificial Intelligence & Robotics Universitas Dr. Soetomo, Bidang peminatan adalah Analisis Spasial, dan Pemodelan Data Spasial, Kecerdasan Buatan dalam Sistem Informasi Geografis. Pengalaman menulis makalah sesuai bidangnya di Jurnal Scopus. Penulis juga telah menjadi *reviewer* untuk penerbit Taylor dan Francis Ltd, termasuk Journal of Transportation Safety and Security, IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India), dan International Journal of Injury Control and Safety Promotion, serta a pengulas di Penerbit Jurnal Ilmu Sosial Internasional Wiley-Blackwell Publishing Ltd.

Erri Wahyu Puspitarini adalah dosen bidang Sistem Informasi di STMIK Yadika Bangil. Penulis lulusan sarjana Sistem Informasi

dari STIKOM Surabaya dan Magister Manajemen Teknologi dari ITS Surabaya. Di bidang pengajaran aktif mengajar di bidang sistem informasi manajemen, research IT dan metodologi penelitian. Penulis juga aktif melakukan penelitian dan mendapatkan hibah dari kemenristekbrin.