MENGKONTRUKSI ULANG *DATABASE* SECARA OTOMATIS HINGGA DALAM BENTUK 3NF (THIRD NORMAL FORM)

(Studi Kasus : Database Anggota Ikatan Mahasiswa Muhammadiyah UIN Malang)

SKRIPSI

Oleh:

M. IQBAL FAHMI
NIM. 07650085

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG

2013

MENGKONTRUKSI ULANG *DATABASE* SECARA OTOMATIS HINGGA DALAM BENTUK 3NF (THIRD NORMAL FORM)

(Studi Kasus : Database Anggota Ikatan Mahasiswa Muhammadiyah UIN Malang)

SKRIPSI

Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:

M. IQBAL FAHMI
NIM. 07650085

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
2013

MENGKONTRUKSI ULANG *DATABASE* SECARA OTOMATIS HINGGA DALAM BENTUK 3NF (THIRD NORMAL FORM)

(Studi Kasus : Database Anggota Ikatan Mahasiswa Muhammadiyah UIN Malang)

SKRIPSI

Oleh:

M. IQBAL FAHMI
NIM. 07650085

Telah Disetujui untuk Diuji
Malang, 11 Januari 2013

Dosen Pembimbing I,	Dosen Pembimbing II,
M. Ainul Yaqin, M.Kom	Syahiduz Zaman, M,Kom
NIP. 197610132006041004	NIP. 197005022005011005

Mengetahui,
Ketua Jurusan Teknik Informatika

Ririen Kusumawati, M.Kom
NIP. 197203092005012002

MENGKONTRUKSI ULANG *DATABASE* SECARA OTOMATIS HINGGA DALAM BENTUK 3NF (THIRD NORMAL FORM)

(Studi Kasus : Database Anggota Ikatan Mahasiswa Muhammadiyah UIN Malang)

SKRIPSI

Oleh:

M. Iqbal Fahmi
NIM. 07650085

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)

Tanggal, 19 Januari 2013

Susunan Dewan Penguji:		Tanda Tangan
1. Penguji Utama	: ZAINAL ABIDIN, M.KOM NIP. 197606132005011004	( )
2. Ketua Penguji	: M. AMIN HARIYADI, M.T NIP. 196701182005011001	( )
3. Sekretaris Penguji	: M. AINUL YAQIN, M.KOM NIP. 197610132006041004	( )
4. Anggota Penguji	: SYAHIDUZ ZAMAN, M,KOM NIP. 197005022005011005	( )

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik Informatika

Ririen Kusumawati, M.Kom
NIP. 197203092005012002

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini :

Nama	: M. Iqbal Fahmi
NIM	: 07650085
Fakultas / Jurusan	: Sains Dan Teknologi / Teknik Informatika
Judul Penelitian	: MENGKONTRUKSI ULANG DATABASE SECARA OTOMATIS HINGGA DALAM BENTUK 3NF (THIRD NORMAL FORM)

Menyatakan dengan sebenar-benarnya bahwa hasil penelitian saya ini tidak terdapat unsur-unsur penjiplakan karya penelitian atau karya ilmiah yang pernah dilakukan atau dibuat oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata hasil penelitian ini terbukti terdapat unsur-unsur jiplakan, maka saya bersedia untuk mempertanggung jawabkan, serta diproses sesuai peraturan yang berlaku.

Malang, 2 Pebruari 2013

Yang Menyatakan,

M. Iqbal Fahmi

NIM. 07650085

MOTTO

وَلْتَكُن مِّنكُمْ أُمَّةٌ يَدْعُونَ إِلَى ٱلْخَيْرِ وَيَأْمُرُونَ بِٱلْمَعْرُوفِ وَيَنْهَوْنَ عَنِ ٱلْمُنكَرِ ۚ وَأُوْلَٰٓئِكَ هُمُ ٱلْمُفْلِحُونَ ﴿١٠٤﴾

"Dan hendaklah ada di antara kamu segolongan umat yang menyeru kepada kebajikan, menyuruh kepada yang ma'ruf dan mencegah dari yang munkar, merekalah orang-orang yang beruntung"

(Al-Imron: 104)

Lembar Persembahan

Lillahi maa fii as-Samaawaati wal ardh

Sesungguhnya apa yang ada pada diri hamba adalah milik-MU ya Allah. Shalat hamba, ibadah, hidup dan mati hamba berada dalam arungan perahu yang kau ridhoi.

Sungguh syukur ini hanya untuk-Mu ya Allah
Telah menciptakan dan melahirkan hamba dalam keluarga penuh kasih sayang
Untukmu Sang Pemegang kunci surgaku nan penuh kasih sayang, Ibu Nashihah
Bagimu Sang Bijaksana dan Penyabar, Ayah Faizin
Peluk kasih dan kecupan sayang Ayah dan Ibu adalah segala-galanya
Do'a yang terlantun takkan mampu Ananda membalas sampai kapanpun
Semoga Ananda tak lelah untuk berbakti

Mas ku, Ahmad Rizal & Rosyih Ridlo
Saudara seperjuangan, Imam Habibi
Indahnya pelukan persaudaraan kita
Usapan di kepala ini takkan pernah hilang

Para immawan/i IMM Koms Revivalis, Pelopor dan Reformer UIN Maliki Malang
Terimakasih, dukungan kalian merupakan motivasi yang sangat luar biasa

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahnya Skripsi yang berjudul *"Mengkontruksi Ulang Database Secara Otomatis Hingga Dalam Bentuk 3NF (Third Normal Form)"* ini dapat diselesaikan. Dan semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW yang senantiasa memberikan cahaya petunjuk kepada kita.

Selanjutnya penulis haturkan ucapan terima kasih seiring do'a dan harapan *jazakumullah ahsanal jaza'* kepada semua pihak yang telah membantu terselesaikannya skripsi ini. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang tak terhingga kepada:

1. Bapak M. Ainul Yaqin, M.Kom dan Bapak H. Syahiduz Zaman, M.Kom selaku dosen pembimbing skripsi, yang telah banyak memberikan bimbingan serta memberikan masukan positif kepada penulis dalam menyelesaikan skripsi ini.
2. Prof. Dr. H. Imam Suprayogo selaku Rektor UIN Maulana Malik Ibrahim Malang yang telah banyak memberikan pengetahuan dan pengalaman yang berharga.
3. Prof. Drs. Sutiman Bambang Sumitro, SU., DSc. selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Ibu Ririen Kusumawati, M.Kom selaku ketua jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
5. Seluruh Dosen Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah memberikan ilmu yang tak terbatas sebagai bekal hidup.
6. Seluruh keluarga yang selalu mendoakan, memberikan motivasi dan dorongan dalam penyelesaian skripsi ini.
7. Teman-teman Teknik Informatika angkatan 2007. Terima kasih atas segala bantuan, dukungan, motivasi, dan kebersamaannya selama ini. Semoga Allah

SWT memberikan balasan yang setimpal atas jasa dan bantuan yang telah diberikan.

8. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah banyak membantu dalam penyelesaian skripsi ini.

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun.

Malang, 11 Januari 2013

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

M. Iqbal Fahmi. 2013. 07650085**. Mengkontruksi Ulang Database Secara Otomatis Hingga Dalam Bentuk 3NF (Third Normal Form).** Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing : (I) M. Ainul Yaqin, M.Kom (II) Syahiduz Zaman, M.Kom

Kata Kunci : *ketergantungan fungsional, redundansi, anomali*

Mengkontruksi ulang *database* secara otomatis hingga dalam bentuk 3NF adalah suatu sistem terkomputerisasi yang dirancang untuk administrator dalam menormalisasi basis data. Normalisasi merupakan langkah yang sangat penting dilakukan oleh administrator basis data. Langkah ini dilakukan untuk menganalisis relasi basis data yang bertujuan untuk menciptakan suatu kumpulan relasi basis data yang minim akan redundansi dan anomali.

Proses normalisasi basis data secara otomatis sangat memudahkan dan mempercepat proses normalisasi. Pada proses ini, masukkan yang dibutuhkan berupa tabel dari basis data. Masukkan tersebut kemudian diproses oleh sistem untuk mendapatkan ketergantungan fungsional dengan membandingakn data dari tiap attribut kemudian ketergantungan fungsional diubah menjadi matrik ketergantungan fungsional dan kunci penentu. Dari matrik tersebut, dilakukan pencarian ketergantungan parsial dan ketergantungan transitif kumudian jika ditemukan makan tabel diuraikan menjadi tabel-tabel baru yang sudah tidak mengandung ketergantungan parsial dan transitif pada tabel tersebut. Tabel-tabel yang sudah dinormalisasi kemudian diubah menjadi kode-kode *query* MySql sehingga pengguna tinggal menjalankannya pada sistem MySql.

Dari hasil uji coba aplikasi yang dilakukan pada basis data anggota didapatkan basis data baru yang mampu mengurangi redundansi data dan anomali yang terjadi pada basis data lama sehingga pengguna mendapatkan basis data yang lebih efektif dan efisien.

ABSTRACT

M. Iqbal Fahmi. 2013. 07650085**. Repeat Database Automatically constructing Up In Form 3NF (Third Normal Form).** Informatic Department Science and Technology Faculty of Islamic State University of Malang Maulana Malik Ibrahim. Supervisor : (I) M. Ainul Yaqin, M.Kom (II) Syahiduz Zaman, M.Kom

Keyword : *functional dependency, redundancy, anomaly*

Construct the database automatically reset to the form 3NF is a computerized system that is designed for database administrators in normalizing. Normalization is a very important step performed by the database administrator. This step is carried out to analyze the relationship database that aims to create a set of database relationships be minimal redundancy and anomalies.

The process of database normalization is automatically greatly facilitate and accelerate the process of normalization. In this process, enter the required form of tables from the database. Insert is then processed by the system to obtain the functional dependence of the data from each attribute membandingakn then converted into functional dependency matrix functional dependencies and key determinants. From these matrices, to search for partial and transitive dependencies dependency kumudian if found dining table broken down into new tables that are not contained partial and transitive dependencies in the table. The tables are normalized and then converted into MySql query code so users stay MySql running on the system.

From the test results conducted on the application of database members obtained a new database that can reduce data redundancy and anomalies that occur in the old database so that users get the database more effectively and efficiently.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Basis data (*database*) merupakan komponen utama sistem informasi, karena semua informasi untuk pengambilan keputusan berasal dari data di basis data. Pengelolaan basis data yang buruk dapat mengakibatkan ketidaktersediaan data penting yang digunakan untuk menghasilkan informasi yang diperlukan dalam pengambilan keputusan agar data dapat dikelola dengan baik maka diperlukan satu sistem manajemen dalam pengelolaan data yang disebut sistem manajemen basis data atau DBMS (*Basis data Management System*), yaitu perangkat lunak untuk mendefinisikan, menciptakan, mengelola dan mengendalikan pengaksesan basis data. Tujuan utama sistem manajemen basis data adalah menyediakan lingkungan yang nyaman dan efisien untuk penyimpanan dan pengambilan data dari basis data.

Dalam ajaran Islam juga dianjurkan tentang pentingnya efisiensi dalam kehidupan manusia. Efisiensi dalam hal bekerja, keuangan, perkataan dan perbuatan dalam setiap aktifitas dilakukan oleh manusia dalam menjalankan kehidupan sehari-harinya. Setiap manusia dianjurkan untuk membiasakan diri selalu efektif, artinya dalam menjalankan kehidupannya janganlah melakukan perbuatan yang sia-sia. Dalam tafsir al-Qurthubi dijelaskan bahwa perbuatan yang sia-sia adalah perbuatan yang tiada manfaat dan juga tidak ada keburukan dalam perbuatan tersebut. (Tafsir al-Qurthubi:2008). Dalam Firman Allah yang dijelaskan dalam Surat al-Israa' ayat 29 yang berbunyi:

وَلَا تَجۡعَلۡ يَدَكَ مَغۡلُولَةً إِلَىٰ عُنُقِكَ وَلَا تَبۡسُطۡهَا كُلَّ ٱلۡبَسۡطِ فَتَقۡعُدَ مَلُومٗا مَّحۡسُورٗا ٢٩

Artinya: *"Dan janganlah kamu jadikan tanganmu terbelenggu pada lehermu dan janganlah kamu terlalu mengulurkannya karena itu kamu menjadi tercela dan menyesal."*(Q.S Al-Israa' 17:29)

Dalam ayat ini dijelaskan bahwa dalam melaksanakan aktifitas sehari-hari dalam kehidupan di dunia hendaklah harus mempertimbangkan asas manfaatnya. Efisiensi yang dimaksudkan dalam ayat ini adalah di dalam proses pengambilan keputusan, berkata dan berbuat dalam kehidupan haruslah memperhitungkan asas manfaat dari hal tersebut, janganlah kemudian apa yang dilakukan nantinya akan mendatangkan keburukan dan menjadikanya hal yang sia-sia belaka. Dalam manajemen basis data langkah ini sangatlah berguna karena di dalam membangun sistem diperlukan perancangan basis data yang efektif dan efisien sehingga memudahkan dalam proses penyimpanan dan pengambilan data dan menghasilkan sistem basis data yang baik.

Untuk menghasilkan sebuah sistem basis data yang baik maka diperlukan suatu proses analisa yang tidak mudah bagi kebanyakan orang. Hal ini membutuhkan pengetahuan yang kuat tentang dasar-dasar sistem manajemen basis data. Terkadang proses ini membutuhkan waktu yang lama dan dokumen yang lengkap tentang sistem yang akan dibuat tergantung pada tingkat kompleksitas sistem tersebut. Salah satu bagian yang sering menjadi kendala adalah redundansi data (Ramakrishnan, 2004). Redundansi pada basis data dapat mengakibatkan penumpukan informasi yang berulang dan ketidakkonsistenan data sehingga basis data menjadi kurang efisien dalam penggunaan ruang penyimpanan

dan memperlambat pengaksesan data maka dari itu, untuk mengatasi masalah tersebut dibutuhkan proses normalisasi basis data.

Normalisasi dalam basis data merupakan langkah penting dalam pembuatan basis data. Proses melakukan hal itu secara manual sangat sulit, perlu mendatangkan orang-orang terampil dan membutuhkan waktu yang cukup banyak, selain itu manusia dapat membuat kesalahan dan tentunya akan mengeluarkan biaya yang cukup banyak dalam melakukan normalisasi. Maka dari itu, untuk mengatasi permasalahan-permasalahan di atas peneliti merasa perlu membangun suatu aplikasi yang bisa digunakan sebagai alat yang dapat mengkontruksi ulang tabel dengan cepat dan akurat secara otomatis.

1.2 Rumusan Masalah

Berdasarkan penjelasan pada latar belakang di atas, maka rumusan masalah dalam penelitian ini yaitu, Bagaimana mengotomatiskan kontruksi ulang basis data sehingga berbentuk 3NF?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk membuat aplikasi atau *tool* yang dapat mengkontruksi ulang basis data secara otomatis sehingga berbentuk 3NF.

1.4 Manfaat Penelitian

Adapun penelitian ini diharapkan memberikan manfaat berupa kemudahan kepada perancang basis data untuk mengambil keputusan dalam proses perancangan basis data.

1.5 Batasan Masalah

Batasan masalah yang diberikan dalam dalam pembuatan aplikasi ini adalah sebagai berikut:

1. Aplikasi dibangun dengan menggunakan bahasa pemrograman Java.
2. Basis data yang digunakan untuk uji coba adalah basis data MySQL.
3. Basis data yang dinormalisasi minimal dalam bentuk normal pertama (1NF).
4. Aplikasi ini diuji menggunakan basis data anggota Ikatan Mahasiswa Muhammadiyah UIN Malang.

1.6 Metode Penelitian

Metode yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Pada tahap ini dilakukan pencarian dan pemahaman literatur yang berhubungan dengan permasalahan pembuatan aplikasi Kontruksi Ulang Basis Data yang bisa diperoleh dalam bentuk *paper*, *textbook*, dan hasil *browsing* di internet yang terkait dengan penelitian ini.

2. Perumusan masalah dan penyelesaiannya

   Tahap ini meliputi perumusan masalah, batasan-batasan masalah, dan penyelesaiannya serta penentuan parameter untuk mengukur hasilnya.

3. Perancangan Perangkat Lunak

   Pada tahap ini dilakukan perancangan perangkat lunak untuk menerapkan permasalahan dan penyelesaiannya pada tahap sebelumnya.

4. Pembuatan Perangkat Lunak

Pada tahap ini dilakukan pembuatan perangkat lunak sesuai dengan perancangan perangkat lunak yang telah dilakukan, dengan melakukan konversi algoritma menjadi kode program yang siap dieksekusi.

5. Uji Coba dan Evaluasi Hasil

Tahap ini meliputi uji coba terhadap sistem yang telah dirancang untuk pembuatan aplikasi ini serta kesesuaian dengan program jadinya, dan evaluasi dari setiap percobaan.

6. Penyusunan laporan penelitian

Pada tahap ini dilakukan penulisan laporan penelitian yang merupakan dokumentasi dari konsep atau teori penunjang, perancangan perangkat lunak, pembuatan perangkat lunak dan dokumentasi dari uji coba dan analisis, serta kesimpulan dan saran.

1.7 Sistematika Penyusunan Skripsi

Sistematika dalam penulisan skripsi ini akan dibagi menjadi beberapa bab sebagai berikut :

BAB I PENDAHULUAN

Pendahuluan adalah bab pertama dari skripsi yang memuat tentang berbagai latar belakang tentang pentingnya dilakukan penelitian ini, untuk apa penelitian tersebut dan mengapa penelitian itu harus dilakukan. Oleh karena itu, bab pendahuluan terdiri atas : Latar belakang masalah, tujuan penelitian, manfaat penelitian, rumusan masalah, batasan masalah, metode penelitian dan sistematika penyusunan skripsi.

BAB II KAJIAN PUSTAKA

Bab dua berisi tentang argumentasi ilmiah yang dipakai sebagai referensi. Bahan pustaka yang digunakan diperoleh dari berbagai sumber seperti : Jurnal penelitian, laporan penelitian, buku teks, diskusi ilmiah, maupun temuan-temuan hasil *browsing* di internet. Berikutnya mengkaji hasil temuan pustaka yang berhubungan dengan konsep-konsep yang dipermasalahkan dan akan dipakai dalam analisis, termasuk mengkaji tentang teori-teori keislaman yang berkaitan dengan penelitian ini.

BAB III ANALISIS DAN PERANCANGAN

Bab tiga berisi analisis dan perancangan sistem secara terstruktur, yang dilengkapi dengan beberapa diagram. Selain itu akan dilakukan pembuatan aplikasi yang dibangun sesuai dengan permasalahan dan batasannya yang telah dijabarkan pada bab pertama.

BAB IV HASIL DAN PEMBAHASAN

Bab empat membahas tentang implementasi dari aplikasi yang dibuat secara keseluruhan. Serta melakukan pengujian terhadap aplikasi yang dibuat untuk mengetahui apakah aplikasi tersebut telah berjalan sesuai dengan yang diharapkan, termasuk relevansi dalam kajian keislamannya.

BAB V PENUTUP

Penutup berisi kesimpulan dan saran dari hasil penelitian. Kesimpulan merupakan pernyataan singkat yang dijabarkan dari hasil penelitian dan pembahasan, untuk membuktikan kebenaran dari temuan

pustaka yang diperoleh sekaligus menjawab tujuan penelitian. Sedangkan saran adalah rekomendasi untuk penelitian selanjutnya, yang didasarkan atas pengalaman dan pertimbangan dari hasil penelitian yang dilakukan.

BAB II
TINJAUAN PUSTAKA

2.1 Pengertian Basis Data

Menurut Connolly (2002), basis data adalah kumpulan data yang saling terhubung secara logis, yang dapat digunakan secara bersama, dan deskripsi dari data ini dirancang untuk memenuhi kebutuhan. Menurut Date (2000), basis data adalah sebuah koleksi dari data yang tahan lama yang digunakan oleh sistem aplikasi dari perusahaan tertentu.

Jadi basis data adalah kumpulan data tetap yang tersimpan dan saling terhubung secara logis yang digunakan oleh beberapa sistem aplikasi untuk memenuhi kebutuhan informasi sebuah perusahaan atau organisasi.

2.2 Model Relasional

Model relasional adalah model data yang didasarkan pada konsep matematika dari relasi, yang secara fisik direpresentasikan sebagai tabel. Codd menggunakan terminologi yang diambil dari matematika, terutama teori set dan logika predikat.

Dalam model relasional, semua data terstruktur secara logis dalam sejumlah relasi. Setiap relasi memiliki nama, dan terdiri dari sejumlah atribut data. Kelebihan dari model relasional adalah struktur logis datanya yang sederhana.

Tabel 2.1 Istilah penting dalam model relasional

Jenis	**Deskripsi**
Relasi	sebuah tabel dengan kolom dan baris.
Atribut/*Field*	nama kolom dalam sebuah relasi.
Domain	sekumpulan nilai yang diijinkan untuk satu atau lebih atribut.
Tuple/*Record*	baris dari sebuah relasi.
Degree	jumlah atribut pada sebuah relasi.
Cardinality	jumlah tuple pada sebuah relasi.

Basis data relasional	kumpulan relasi yang telah dinormalisasi dengan nama relasi yang berbeda.
Skema relasi	set terbatas dari kumpulan atribut. Contoh: skema relasi R memiliki atribut-atribut: A, B, dan C,maka ditulis R = A B C.
Skema basis data relasional	kumpulan skema relasi yang masing-masing memiliki nama yang berbeda.
Superkey	sebuah atribut atau serangkaian atribut yang secara unik mengidentifikasi tuple-tuple pada suatu relasi.
Candidate key	sebuah superkey sedemikian sehingga tidak ada subsetyang merupakan superkey dalam relasi tersebut.
Primary key	candidate key yang dipilih untuk mengidentifikasi tuple-tuple secara unik dalam sebuah relasi.
Foreign key	sebuah atribut atau serangkaian atribut dalam satu relasi yang sama dengan candidate key pada relasi yang lain.
Null	representasi nilai sebuah atribut yang tidak diketahui nilainya atau tidak dapat dipakai untuk sebuah tuple.

2.3 Efisiensi Sistem Basis Data

Tujuan utama sistem manajemen basis data adalah menyediakan lingkungan yang nyaman dan efisien untuk penyimpanan dan pengambilan data dari basis data. Efisiensi merupakan suatu ukuran keberhasilan yang dinilai dari besarnya sumber atau biaya untuk mencapai hasil dari kegiatan yang dijalankan. Efisiensi juga bisa dimaknai suatu ukuran dalam membandingkan rencana penggunaan masukan dengan penggunaan yang direalisasikan.

Dalam Islam, efisiensi sendiri diartikan sebagai sebuah aktifitas yang tidak sia-sia, yang mempunyai asas manfaat di dalamnya. Sebagaimana Firman Allah SWT dalam surat al-Mu'minuun ayat 3, yang berbunyi:

وَٱلَّذِينَ هُمۡ عَنِ ٱللَّغۡوِ مُعۡرِضُونَ ۝٣

Artinya : *" Dan orang-orang yang menjauhkan diri dari (perbuatan dan perkataan) yang tiada berguna."* (Q.S al-Mu'minuun 23 : 3)

Dalam ayat di atas dijelaskan bahwa setiap manusia dianjurkan untuk membiasakan diri selalu efektif, artinya dalam menjalankan kehidupannya

janganlah melakukan perbuatan yang sia-sia. Dalam tafsir al-Qurthubi dijelaskan bahwa perbuatan yang sia-sia adalah perbuatan yang tiada manfaat dan juga tidak ada keburukan dalam perbuatan tersebut. (Tafsir al-Qurthubi:2008).

Dalam membangun sistem basis data, efisiensi sangatlah diperlukan agar tidak dikategorikan berlebihan dalam alokasi memori dan media penyimpan data. Sesuatu yang berlebih-lebihan dilarang dalam Islam sebagaimana Firman Allah SWT dalam surat al-Israa' ayat 26-27 yang berbunyi :

وَءَاتِ ذَا ٱلْقُرْبَىٰ حَقَّهُۥ وَٱلْمِسْكِينَ وَٱبْنَ ٱلسَّبِيلِ وَلَا تُبَذِّرْ تَبْذِيرًا ﴿٢٦﴾ إِنَّ ٱلْمُبَذِّرِينَ كَانُوٓا۟ إِخْوَٰنَ ٱلشَّيَٰطِينِ ۖ وَكَانَ ٱلشَّيْطَٰنُ لِرَبِّهِۦ كَفُورًا ﴿٢٧﴾

Artinya : *"Dan berikanlah kepada keluarga-keluarga yang dekat akan haknya, kepada orang miskin dan orang yang dalam perjalanan dan janganlah kamu menghambur-hamburkan (hartamu) secara boros. Sesungguhnya pemboros-pemboros itu adalah saudara-saudara syaitan dan syaitan itu adalah sangat ingkar kepada Tuhannya".* (Q.S al-Israa' 17: 26-27)

Dalam ayat di atas dijelaskan bahwa umat manusia diperintahkan oleh Allah SWT agar tidak melakukan sesuatu hal secara berlebihan-lebihan karena sifat berlebih-lebihan itu adalah sebagian dari sifat syetan. Sifat berlebih-lebihan dapat membuat kehancuran sebagaimana para syetan dan kelak syetan akan menemaninya di dalam neraka. (Tafsir al-Qurthubi:2008).

Dalam proses membangun sistem basis data jika tidak diikuti dengan analisis yang cermat dalam penyusunanya nanti akan mengakibatkan adanya redudansi data (menyimpan data yang sama di beberapa tabel). Sehingga untuk menghilangkan redudansi data ini dan memastikan ketergantungan data yang ada sudah benar (hanya menyimpan data yang berhubungan ke dalam sebuah tabel), Maka diperlukan proses normalisasi basis data.

2.4 Normalisasi

Normalisasi adalah suatu teknik formal untuk menganalisis relasi berdasarkan *primary key* (*candidate key*) dan ketergantungan fungsional (Connolly, 2002). Teknik tersebut mencakup serangkaian aturan yang dapat digunakan untuk menguji relasi individu sehingga basis data dapat dinormalisasi pada derajat tertentu. Ketika syarat tidak terpenuhi, relasi yang tidak sesuai syarat harus diuraikan ke dalam beberapa relasi dimana secara individu memenuhi syarat-syarat normalisasi. Proses normalisasi merupakan metode formal yang mengidentifikasi beberapa relasi berdasarkan *primary* atau *candidate key* dan ketergantungan fungsional pada atribut-atributnya (Connolly, 2002). Tujuan dari proses normalisasi adalah untuk menghilangkan redundansi data (menyimpan data yang sama di beberapa tabel) dan memastikan ketergantungan data yang ada sudah benar (hanya menyimpan data yang berhubungan ke dalam sebuah tabel).

Normalisasi mendukung para perancang basis data dengan memberikan serangkaian percobaan yang dapat digunakan dalam relasi individual sehingga skema relasi tersebut dapat dinormalisasi ke dalam bentuk yang lebih spesifik untuk menghindari kejadian yang memungkinkan terjadinya *update anomaly* (Connolly, 2002). Normalisasi biasanya dilakukan dalam beberapa tahap. Masing-masing tahap berkaitan dengan bentuk normal tertentu yang telah diketahui sifat-sifatnya. Dalam pemodelan data, sangatlah penting untuk mengenal bentuk normal pertama (1NF) yang merupakan kondisi kritis untuk membuat relasi-relasi yang diperlukan. Bagian bentuk normal yang lain hanya merupakan pilihan (*optional)*. Akan tetapi, untuk menghindari terjadinya *update anomaly*, maka

dalam proses normalisasi setidaknya harus sampai pada bentuk normal ketiga (3NF) (Connolly, 2002). Kemudian, oleh R. Boyce dan E. F. Codd memperkenalkan definisi yang lebih kuat dari bentuk normal ketiga yaitu Boyce-Codd Normal Form (BCNF). Bentuk normal form yang lebih tinggi lagi diperkenalkan kemudian sebagai bentuk normal keempat (4NF) dan kelima (5NF). Namun, bentuk normal ini jarang digunakan pada beberapa situasi sekarang ini (Connolly, 2002).

2.4.1 Redundansi Data dan Update Anomaly

Tujuan utama perancangan basis data relasional adalah untuk mengelompokkan atribut-atribut ke dalam relasi agar supaya redundansi data minimal sehingga dapat mengurangi kapasitas penyimpanan file yang dibutuhkan oleh relasi yang diimplementasikan. Relasi yang mempunyai redundansi data akan menimbulkan masalah yang disebut *update anomaly*, yang dikelompokkan menjadi *insertion, deletion,* dan *modification anomaly* (Connolly, 2002).

Tabel 2.2 Relasi *Staff* dan *Branch*

a. *Staff*

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beach	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

b. *Branch*

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Tabel 2.3 Relasi *StaffBranch*

staffNo	sName	Position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beach	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

a. Insertion Anomaly

Menurut Connolly (2002), ada 2 jenis utama *insertion anomaly*, dimana dapat digambarkan dengan menggunakan relasi *StaffBranch* pada Tabel 2.3.

1) Memasukkan anggota baru staf ke dalam relasi *StaffBranch*, perincian cabang dimana staf ditempatkan perlu dimasukkan. Sebagai contoh, memasukkan rincian staf baru yang akan ditempatkan pada cabang nomor B007, perincian cabang nomor B007 perlu dimasukkan dengan benar, sehingga perincian cabang konsisten dengan nilai cabang B007 di *tuple* lain dalam relasi *StaffBranch*. Relasi yang ditunjukkan pada tabel 2.2 tidak mengalami data yang tidak konsisten, sebab hanya memasukkan nomor cabang dengan tepat untuk masing-masing staf dalam relasi *Staff*. Sedangkan rincian cabang nomor B007 disimpan dalam basis data sebagai *single tuple* dalam relasi *Branch*.

2) Memasukkan rincian cabang baru yang tidak memiliki nomor anggota staf ke dalam relasi *StaffBranch*, perlu memasukkan nilai *null* ke dalam atribut staf, seperti *staffNo*. Akan tetapi, *staffNo* adalah *primary key* dari relasi *StaffBranch*, sehingga dengan memasukkan nilai *null* pada *staffNo* akan melanggar konsep integritas data, dan hal ini tidak dibolehkan. Oleh karena itu, nilai *null* tidak bisa dimasukkan pada stafNo ke dalam cabang

baru pada relasi *StaffBranch*. Perancangan yang ditunjukkan pada tabel 2.2 mencegah masalah ini karena rincian cabang yang dimasukkan pada relasi *Branch* terpisah dari perincian staf.

b. Deletion Anomaly

Jika menghapus sebuah *tuple* dari relasi *StaffBranch* yang merepresentasikan anggota lama staf yang dilokasikan pada sebuah cabang, perincian mengenai cabang juga akan terhapus dari basis data. Sebagai contoh, jika menghapus *tuple* dari staf dengan nomor SA9 (Mary Howe) dari relasi *StaffBranch*, perincian yang berhubungan dengan nomor cabang B007 juga akan terhapus dari basis data. Perancangan pada relasi yang ditunjukkan pada tabel 2.2 mencegah masalah ini, sebab *tuple* cabang disimpan secara terpisah dari *tuple* staf dan hanya atribut *branchNo* yang berhubungan dengan kedua relasi tersebut. Jika menghapus *tuple* nomor staf SA9 dari relasi *Staff*, perincian pada nomor cabang B007 tetap tidak berdampak dalam relasi *Branch* (Connolly, 2002).

c. Modification Anomaly

Jika ingin mengubah nilai dari satu atribut untuk cabang tertentu pada relasi *StaffBranch*, sebagai contoh alamat untuk nomor cabang B003, *tuple* dari semua staf yang ditempatkan pada cabang tersebut perlu diubah (*update*). Jika perubahan ini tidak mengubah semua *tuple* pada relasi *StaffBranch* dengan tepat, maka basis data akan menjadi tidak konsisten. Pada contoh ini, cabang nomor B003 akan mempunyai alamat yang berbeda untuk *tuple* staf yang berbeda (Connolly, 2002).

2.4.2 Ketergantungan Fungsional

Salah satu konsep utama dalam normalisasi adalah *functional dependency* (ketergantungan fungsional), yang menggambarkan hubungan antara atribut-atributnya.

a. Definisi Ketergantungan Fungsional

Menurut Silberschatz (2002), ketergantungan fungsional memegang peranan penting dalam membedakan perancangan basis data yang baik dengan basis data yang buruk. Ketergantungan fungsional merupakan sebuah jenis batasan yang menggeneralisasikan *key*.

Menurut Connolly (2002), ketergantungan fungsional mendeskripsikan hubungan antar atribut dalam sebuah relasi. Sebagai contoh, jika A dan B adalah atribut dari relasi R, maka B secara fungsional tergantung pada A (ditulis A B), jika masing-masing nilai dari A diasosiasikan dengan tepat satu nilai dari B (di mana A dan B masing-masing terdiri dari satu atau lebih atribut).

Ketergantungan fungsional merupakan semantik dari atribut dalam sebuah relasi.Semantik menunjukkan bagaimana atribut-atribut berelasi satu sama lain, dan menspesifikasikan saling ketergantungan antar atribut. Ketika ketergantungan fungsional dilakukan, maka ketergantungan dikenal sebagai *constraint*/ batasan di antara atribut. Relasi dengan atribut A dan B di mana atribut B tergantung dengan atribut A dapat ditulis A → B.

Jika ingin mengetahui nilai A dan mengamati relasi tersebut, maka hanya akan ditemukan satu nilai B pada semua *tuple* dari nilai A yang diberikan.

Oleh karena itu ketika dua *tuple* mempunyai nilai yang sama pada A, B juga mempunyai nilai yang sama. Pada contoh di atas A merupakan *determinant* dari B. *Determinant* menunjukkan atribut atau kumpulan atribut yang berada di sebelah kiri tanda panah dalam ketergantungan fungsional.

Contoh relasi R sebagai berikut (Silberschatz, 2002):

Tabel 2.4 Tabel contoh Relasi R

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Dapat dilihat ketergantungan fungsional mana saja yang memenuhi relasi r tersebut. Ketergantungan fungsional A → C terpenuhi, di mana dua *tuple* yang mempunyai nilai a_1 pada A mempunyai nilai C yang sama yaitu c_1, dua *tuple* yang mempunyai nilai a2 pada A mempunyai nilai C yang sama juga yaitu c_2 dan tidak terdapat pasangan lain dari *tuple* berbeda yang mempunyai nilai A yang sama. Sedangkan ketergantungan fungsional C → A tidak terpenuhi, di mana dua *tuple* yang mempunyai nilai C yang sama (c_2) mempunyai nilai A yang berbeda yaitu a_2 dan a_3. Dengan demikian, telah ditemukan dua *tuple* t1 dan t2 dengan $t_1[C] = t_2[C]$ tetapi $t_1[A] \neq t_2[A]$ (Mata-Toledo, 2007).

b. Inference Rule untuk Ketergantungan Fungsional

Dalam mengidentifikasi ketergantungan fungsional pada suatu relasi, biasanya perancang basis data memulai dengan menspesifikasikan saling ketergantungan tersebut secara nyata, namun ada banyak ketergantungan fungsional lainnya. Akhirnya, untuk menspesifikasi semua ketergantungan

yang mungkin untuk proyek basis data merupakan hal yang tidak praktis. Oleh karena itu, ada sebuah pendekatan yang membantu mengidentifikasi ketergantungan fungsional secara menyeluruh dalam suatu relasi, yang dikenal dengan sejumlah *inference rule* yang dapat digunakan untuk mengambil ketergantungan baru dari sejumlah ketergantungan yang diberikan. Tujuh aturan yang dikenal sebagai *inference rule* untuk ketergantungan fungsional adalah sebagai berikut (Connolly, 2002):

Tabel 2.5 Tabel *inference rule*

1. Reflektif	jika B bagian dari A, maka A ⟶ B
2. Augmentasi	jika A ⟶ B, maka A,C ⟶ B,C
3. Transitif	jika A ⟶ B dan B ⟶ C, maka A ⟶ C
4. Self-Determination	A ⟶ A
5. Dekomposisi	jika A ⟶ B,C, maka A ⟶ B dan A ⟶ C
6. Union	jika A ⟶ B dan A ⟶ C, maka A ⟶ B,C
7. Composisi	jika A ⟶ B dan C ⟶ D, maka A,C ⟶ B,D

c. Minimal Set dari Ketergantungan Fungsional

Sejumlah ketergantungan fungsional X minimal jika memenuhi kondisi berikut (Connolly, 2002):

1) Setiap ketergantungan X mempunyai atribut tunggal dari *right-hand side*.
2) Ketergantungan A → B pada X tidak dapat diganti dengan ketergantungan C → B, di mana C adalah bagian dari A dan masih mempunyai sejumlah ketergantungan yang ekuivalen dengan X.
3) Tidak dapat menghilangkan ketergantungan X dan masih mempunyai sejumlah ketergantungan yang ekuivalen dengan X.

2.4.3 Bentuk Normal Pertama (1NF)

Menurut Connolly (2002), bentuk normal pertama (1NF) adalah sebuah relasi dimana irisan tiap baris dan kolom mengandung satu dan hanya satu nilai. Proses normalisasi awalnya dimulai dari memindahkan data ke dalam tabel dengan baris dan kolom, dimana tabel tersebut masih dalam bentuk tidak normal (UNF). Tujuan bentuk normal pertama (1NF) adalah untuk menghilangkan bentuk tidak normal (UNF) dengan mengidentifikasi dan menghilangkan grup yang berulang (*repeating groups*) di dalam tabel. Grup yang berulang (*repeating groups*)merupakan sebuah atribut atau sekumpulan atribut dalam tabel yang mengandung beberapa nilai untuk satu kejadian.

Menurut Connolly (2002), ada dua pendekatan umum untuk menghilangkan grup yang berulang dari tabel yang tidak normal:

a. Pendekatan yang pertama, menghilangkan grup yang berulang dengan memasukkan data yang tepat pada kolom kosong dari baris yang mengandung data yang berulang. Dengan kata lain, kolom yang kosong diisi dengan menggandakan data yang tidak berulang, jika diperlukan. Tabel yang dihasilkan, merupakan suatu relasi yang mengandung nilai tunggal pada irisan dari tiap baris dan kolom dan tabel ini sudah dalam bentuk normal pertama.
b. Pendekatan yang kedua, menghilangkan grup yang berulang dengan menempatkan data yang berulang bersamaan dengan salinan atribut *key* yang asli. *Primary key* diidentifikasikan untuk relasi yang baru. Terkadang tabel yang tidak normal mengandung lebih dari satu grup yang berulang, atau grup berulang di dalam grup yang berulang juga. Dalam kasus tersebut, pendekatan

ini digunakan berulang kali sampai tidak ada grup yang berulang lagi. Apabila sudah tidak ada grup yang berulang lagi, maka sudah berada dalam bentuk normal pertama.

Tabel 2.6 Tabel *ClientRental* yang belum dinormalisasi

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-00	31–Aug - 01	350 C	O40	Tina Murphy
	PG	16	ovar Dr, Glasgow	1-Sep-01	1-Sep-02	450	CO93	Tony Shaw
CR65	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Jul-00	31 – Aug - 01	350 C	O40	Tina Murphy
	PG	36	anor Rd Glasgow	10-Oct-00	1-Dec-01	375	CO93	Tony Shaw
	PG	16	ovar Dr, Glasgow	1-Sep-01	1-Sep-02	450	CO93	Tony Shaw

clientNo dalam tabel *ClientRental* diidentifikasi sebagai atribut *key*. Struktur dari grup yang berulang yaitu (*propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName*).

Akibatnya, terdapat beberapa nilai pada irisan untuk baris dan kolom tertentu. Sebagai contoh, ada dua nilai untuk *propertyNo* (PG4 dan PG16) untuk klien yang bernama John Kay. Untuk mengubah tabel yang tidak normal ke dalam 1NF, dipastikan hanya ada nilai tunggal pada irisan tiap baris dan kolom. Oleh karena itu grup yang berulang harus dihilangkan.

Dengan menggunakan pendekatan pertama, grup yang berulang dihilangkan dengan memasukkan data klien yang tepat ke dalam masing-masing baris. Hasil dari bentuk normal pertama dari relasi *ClientRental* dapat dilihat pada tabel 2.7. *Candidate key* untuk relasi *ClientRental* diidentifikasi sebagai *key* gabungan (*composite key*) yang terdiri dari (*clientNo*, *propertyNo*), (*clientNo*, *rentStart*) dan (*propertyNo*, *rentStart*). *Primary key* yang dipilih pada relasi ini

adalah (*clientNo*, *propertyNo*) dan agar lebih jelas atribut yang dipilih sebagai *primary key* diletakkan di sebelah kiri pada relasi. Atribut *rentFinish* tidak tepat sebagai sebuah komponen dari *candidate key* karena mengandung nilai *null*.

Tabel 2.7 Relasi *ClientRental* pada bentuk normal pertama (1NF)

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-00	31 – Aug - 01	350 C	O40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-01	1-Sep-02	450	CO93	Tony Shaw
CR65	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Jul-00	31 – Aug - 01	350 C	O40	Tina Murphy
CR65	PG36	Aline Stewart	2 Manor Rd Glasgow	10-Oct-00	1-Dec-01	375	CO93	Tony Shaw
CR65	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Sep-01	1-Sep-02	450	CO93	Tony Shaw

Relasi *ClientRental* dapat didefinisikan sebagai berikut : *ClientRental* (*clientNo*, *propertyNo*, *cName*, *pAddress*, *rentStart*, *rentFinish*, *rent*, *ownerNo*, *oName*). Relasi *ClientRental* berada dalam bentuk normal pertama bila terdapat nilai tunggal pada irisan untuk tiap baris dan kolom. Relasi mengandung data yang menggambarkan klien, properti yang disewakan, dan pemilik properti, dimana diulang beberapa kali. Sebagai hasil, relasi *ClientRental* mengandung redundansi data yang signifikan. Jika diimplementasikan, maka relasi 1NF tersebut akan terjadi *update anomaly*. Untuk mengatasi hal tersebut, maka relasi harus diubah ke dalam bentuk normal kedua.

Dengan menggunakan pendekatan kedua, grup yang berulang dihilangkan dengan menempatkan data yang berulang bersamaan dengan salinan atribut *key* yang asli (*clientNo*) dalam relasi yang terpisah, seperti yang ditunjukkan pada tabel 2.8. *Primary key* diidentifikasi untuk relasi yang baru, dan hasil dari relasi 1NF adalah sebagai berikut:

Client (*clientNo*, *cName*)

PropertyRentalOwner (*clientNo*, *propertyNo*, *pAddress*,*rentStart*, *rentFinish*, *rent*, *ownerNo*, *oName*)

Kedua relasi *Client* dan *PropertyRentalOwner* berada dalam bentuk 1NF bila terdapat nilai tunggal pada irisan untuk tiap baris dan kolom. Relasi *Client* mengandung data yang menggambarkan klien dan relasi *PropertyRentalOwner* mengandung data yang menggambarkan properti yang disewa oleh klien dan pemilik properti. Akan tetapi, relasi yang ditunjukkan pada tabel 2.8 mengandung beberapa redundansi data yang menyebabkan *update anomaly* (Connolly, 2002).

Tabel 2.8 Relasi *Client* dan *PropertyRentalOwner* pada 1NF

a. Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

b. PropertyRentalOwner

clientNo	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	6 Lawrence St, Glasgow	1-Jul-00	31 – Aug – 01	350	CO40	Tina Murphy
CR76	PG16	5 Novar Dr, Glasgow	1-Sep-01	1-Sep-02	450	CO93	Tony Shaw
CR65	PG4	6 Lawrence St, Glasgow	1-Jul-00	31 – Aug – 01	350	CO40	Tina Murphy
CR65	PG36	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	375	CO93	Tony Shaw
CR65	PG16	5 Novar Dr, Glasgow	1-Sep-01	1-Sep-02	450	CO93	Tony Shaw

2.4.4 Bentuk Normal Kedua (2NF)

Menurut Connolly (2002), bentuk normal kedua (2NF) adalah sebuah relasi yang berada dalam bentuk normal pertama dimana setiap atribut *non-primary key* bergantung fungsional secara penuh pada *primary key*. Sebuah

ketergantungan fungsional A → B dikatakan bergantung fungsional secara penuh apabila pemindahan beberapa atribut dari A menyebabkan tidak adanya ketergantungan lagi. Sebuah ketergantungan fungsional A → B dikatakan tergantung sebagian (*partial dependent*) jika terdapat beberapa atribut yang bisa dihilangkan dari A dan ketergantungan.yang ada masih terjaga. Sebagai contoh, ketergantungan fungsional berikut:

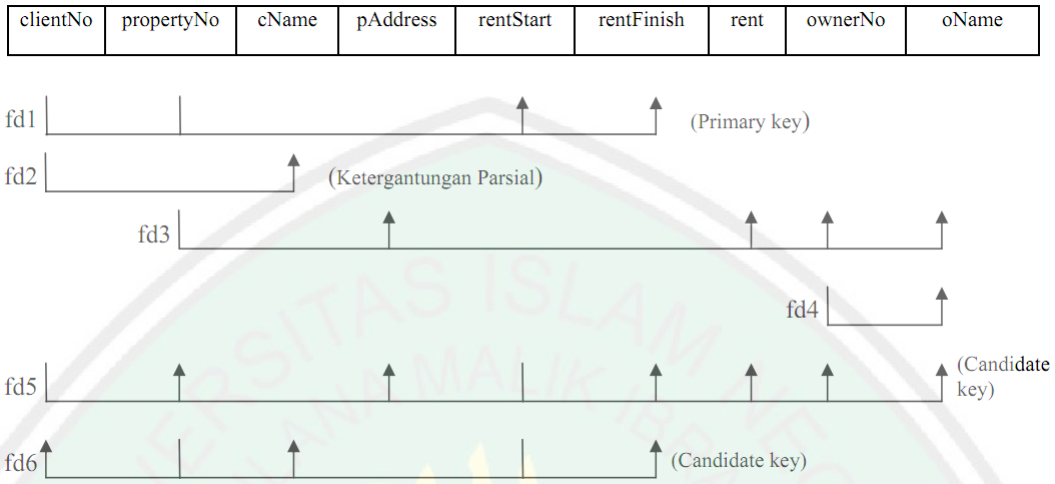
staffNo, *sName branchNo*

Ketergantungan fungsional dikatakan benar bila tiap nilai (*staffNo*, *sName*) dihubungkan dengan nilai tunggal dari *branchNo*. Akan tetapi, tidak bisa dikatakan mempunyai ketergantungan fungsional secara penuh, karena *branchNo* juga mempunyai ketergantungan fungsional terhadap bagian dari (*staffNo*, *sName*), yaitu *staffNo*.

Jika masih ada ketergantungan sebagian, atribut yang bergantung fungsional dari suatu relasi dihilangkan dengan menempatkan atribut tersebut ke dalam relasi yang baru bersamaan dengan salinan *determinant* atribut-atributnya. Proses normalisasi dari relasi bentuk normal pertama ke bentuk normal kedua dapat dilihat pada contoh berikut dengan menggunakan relasi *ClientRental* pada tabel 2.8.

Pada gambar 2.1 ditunjukkan ketergantungan fungsional (fd1 sampai fd6) dari relasi *ClientRental* dengan (*clientNo, propertyNo*) sebagai *primary key*. Walaupun relasi *ClientRental* mempunyai tiga *candidate key*, namun yang diperhatikan adalah hubungan di antara ketergantungan tertentu terhadap *primary key*, karena definisi bentuk normal kedua hanya menyatakan hubungan terhadap

primary key dari sebuah relasi.



Gambar 2.1 Ketergantungan Fungsional pada relasi *ClientRental* (Connolly, 2002)

Relasi *ClientRental* mempunyai ketergantungan fungsional sebagai berikut:

fd1 *clientNo, propertyNo → rentStart, rentFinish* (*Primary Key*)

fd2 *clientNo → cName* (*Partial dependency*)

fd3 *propertyNo → pAddress, rent, ownerNo, oName* (*Partial dependency*)

fd4 *ownerNo → oName* (*Transitive dependency*)

fd5 *clientNo, rentStart→ propertyNo, pAddress, rentFinish, rent, ownerNo, oName* (*Candidate key*)

fd6 *propertyNo, rentStart → clientNo, cName, rentFinish* (*Candidate key*)

Proses normalisasi pada relasi *ClientRental* dimulai dengan mengecek apakah relasi tersebut berada dalam bentuk normal kedua dengan mengidentifikasi keberadaan ketergantungan sebagian terhadap *primary key*. Atribut Klien (*cName*) bergantung sebagian pada *primary key*, yaitu hanya pada atribut *clientNo* (ditunjukkan pada fd2). Atribut properti (*pAddress*, *rent*, *ownerNo*, *oName*)

bergantung sebagian pada *primary key*, yaitu hanya pada atribut *propertyNo* (ditunjukkan pada fd3). Atribut sewa properti (*rentStart* dan *rentFinish*) bergantung penuh pada semua *primary key*, yaitu atribut *clientNo* dan *propertyNo* (ditunjukkan pada fd1).

Gambar 2.1 menunjukkan adanya ketergantungan transitif pada *primary key* (ditunjukkan pada fd4). Walaupun ketergantungan transitif dapat juga menyebabkan *update anomaly*, tetapi keberadaanya tidak mempengaruhi sebuah relasi dalam 2NF. Ketergantungan seperti ini akan dihilangkan pada bentuk normal ketiga (3NF).

Identifikasi ketergantungan sebagian dalam relasi *ClientRental* menunjukkan bahwa relasi tersebut tidak berada dalam bentuk normal kedua. Untuk mengubah relasi *ClientRental* ke bentuk normal kedua perlu dibuat relasi baru dimana atribut *non-primary key* dihilangkan bersamaan dengan salinan dari bagian *primary key* yang memiliki ketergantungan fungsional secara penuh. Hasil dari pembuatan relasi baru adalah *Client*, *Rental*, dan *PropertyOwner*, seperti yang ditunjukkan pada tabel 2.9. Ketiga relasi ini sudah dalam bentuk normal kedua dimana setiap atribut *non-primary key* bergantung fungsional secara penuh terhadap *primary key* pada relasi tersebut. Relasi yang dibentuk adalah sebagai berikut:

Client	(*clientNo*, *cName*)
Rental	(*clientNo*, *propertyNo*, *rentStart*, *rentFinish*)
PropertyOwner	(*propertyNo*, *pAddress*, *rent*, *ownerNo*, *oName*)

Tabel 2.9 Relasi dalam 2NF yang diperoleh dari relasi *ClientRental*

a. Client

clienNo	cName
CR76	John Kay
CR56	Aline Stewart

b. Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-00	31- Aug - 01
CR76	PG16	1-Sep-01	1-Sep-02
CR65	PG4	1-Jul-00	31- Aug - 01
CR65	PG36	10-Oct-00	1-Dec-01
CR65	PG16	1-Sep-01	1-Sep-02

c. PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw

2.4.5 Bentuk Normal Ketiga (3NF)

Menurut Connolly (2002), bentuk normal ketiga (3NF) adalah sebuah relasi yang berada dalam bentuk pertama dan kedua dimana tidak ada atribut *non-primary key* yang bergantung secara transitif terhadap *primary key* atau *candidate key*. Meskipun relasi bentuk normal kedua memiliki redundansi yang lebih sedikit dibanding dengan bentuk normal pertama, namun masih memungkinkan terjadinya *update anomaly*. *Update anomaly* disebabkan karena sebuah ketergantungan transitif, oleh sebab itu perlu dihilangkan ketergantungan tersebut dengan bentuk normal ketiga. Ketergantungan transitif adalah suatu kondisi dimana A, B, C merupakan atribut-atribut dari sebuah relasi yang bila A → B dan B → C, sehingga C bergantung transitif terhadap A melalui B (dengan syarat A tidak bergantung fungsional terhadap B atau C).

Proses normalisasi dari 2NF ke 3NF adalah untuk menghilangkan ketergantungan transitif. Apabila ketergantungan transitif masih ada, perlu dihilangkan atribut sebuah relasi yang memiliki ketergantungan transitif dengan menempatkan atribut tersebut pada sebuah relasi baru bersama dengan salinan *determinant*. Untuk lebih jelasnya dapat dilihat pada contoh berikut. Ketergantungan fungsional untuk relasi *Client*, *Rental*, dan *PropertyOwner* diperoleh dari contoh sebelumnya (seperti yang ditunjukkan pada tabel 2.7):

Client

fd2	*clientNo cName*	(*primary key*)

Rental

fd1	*clientNo, propertyNo rentStart, rentFinish*	(*primary key*)
fd5’	*clientNo, rentStart propertyNo, rentFinish*	(*candidate key*)
fd6’	*propertyNo, rentStart clientNo, rentFinish*	(*candidate key*)

PropertyOwner

fd3	*propertyNo pAddress, rent, ownerNo, oName*	(*partial dependency*)
fd4	*ownerNo oName*	*(transitive dependency)*

Semua atribut *non-primary key* dalam relasi *Client* dan *Rental* memiliki ketergantungan fungsional hanya pada *primary key* dan kedua relasi ini tidak memiliki ketergantungan transitif, sehingga kedua relasi ini sudah berada dalam bentuk normal ketiga (3NF). Ketergantungan fungsional (fd) yang diberi tanda petik (seperti fd5’), menunjukkan bahwa ketergantungan tersebut telah diubah dengan membandingkan ketergantungan fungsional yang aslinya. Semua atribut *non-primary key* dalam relasi *PropertyOwner* bergantung fungsional terhadap

primary key, kecuali *oName*, yang juga tergantung pada *ownerNo* (seperti ditunjukkan pada fd4). Hal ini merupakan salah satu contoh ketergantungan transitif, yang terjadi ketika atribut *non-primary key* (*oName*) tergantung pada satu atau lebih atribut *non-primary key* (*ownerNo*). Ketergantungan transitif telah diidentifikasi sebelumnya pada gambar 2.1. Untuk mengubah relasi *PropertyOwner* ke bentuk normal ketiga, perlu dihilangkan ketergantungan transitif ini dengan membuat dua relasi baru yaitu *PropertyForRent* dan *Owner*, seperti yang ditunjukkan pada tabel 2.8. Relasi baru yang terbentuk tersebut adalah:

PropertyForRent (*propertyNo*, *pAddress*, *rent*, *ownerNo*)

Owner (*ownerNo*, *oName*)

Tabel 2.10 Relasi 3NF yang diperoleh dari relasi *PropertyOwner*

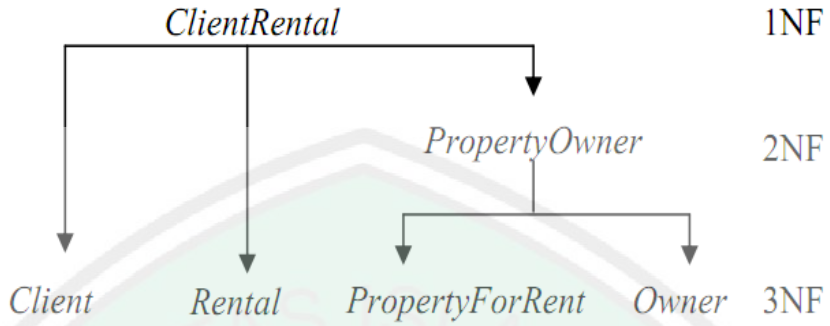
a. PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

b. Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

Relasi *PropertyForRent* dan *Owner* sudah dalam bentuk normal ketiga dimana sudah tidak ada lagi ketergantungan transitif terhadap *primary key*. Relasi *ClientRental* pada tabel 2.6 telah diubah dengan proses normalisasi menjadi empat relasi dalam bentuk normal ketiga. Gambar 2.2 menunjukkan proses dimana relasi asli 1NF didekomposisikan menjadi relasi 3NF. Hasil dari relasi 3NF yang terbentuk adalah:



Gambar 2.2 Dekomposisi relasi *ClientRental* (Connolly, 2002)

Relasi asli *ClientRental* ditunjukkan pada tabel 2.6 dapat dibuat kembali dengan menggabungkan relasi *Client*, *Rental*, *PropertyForRent*, dan *Owner*. Hal ini dapat dilakukan dengan mekanisme *primary key* atau *foreign key*. Sebagai contoh, atribut *ownerNo* adalah *primary key* pada relasi *Owner*, dan juga ada pada relasi *PropertyForRent* sebagai *foreign key*. Atribut *ownerNo* bertindak sebagai *primary key* atau *foreign key* mengijinkan penggabungan antara relasi *PropertyForRent* dan *Owner* untuk mengidentifikasi nama dari *PropertyOwner*.

Atribut *clientNo* adalah *primary key* dari relasi *Client* dan juga ada pada relasi *Rental* sebagai *foreign key*. Perlu diketahui, atribut *clientNo* pada relasi *Rental* memiliki dua peran sebagai *foreign key* dan sebagai bagian *primary key* dari relasi ini. Sama halnya dengan atribut *propertyNo* adalah *primary key* dari relasi *PropertyForRent*, dan juga berada pada relasi *Rental* yang bertindak sebagai *foreign key* dan bagian *primary key* untuk relasi ini. Relasi *Client*, *Rental*, *PropertyForRent* dan *Owner* dapat dilihat pada tabel 2.11.

Tabel 2.11 Relasi keseluruhan 3NF yang diperoleh dari relasi *ClientRental*

a. Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

b. Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-00	31- Aug – 01
CR76	PG16	1-Sep-01	1-Sep-02
CR65	PG4	1-Jul-00	31- Aug – 01
CR65	PG36	10-Oct-00	1-Dec-01
CR65	PG16	1-Sep-01	1-Sep-02

c. PropertyForRent

PropertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

d. Owner

ownerN	ame
CO40	Tina Murphy
CO93	Tony Shaw

2.5 Menampilkan Ketergantungan

Untuk menampilkan ketergantungan menggunakan tiga struktur, Ketergantungan Graf/ Dependency Graph, Ketergantungan Matriks/ Dependency Matrix (DM), dan Matriks Graf Berarah/Directed Graph Matrix(DG) untuk menunjukkan dan memanipulasi ketergantungan antara atribut dari relasi (Bahmani, 2008).

2.5.1 Diagram Ketergantungan Graf

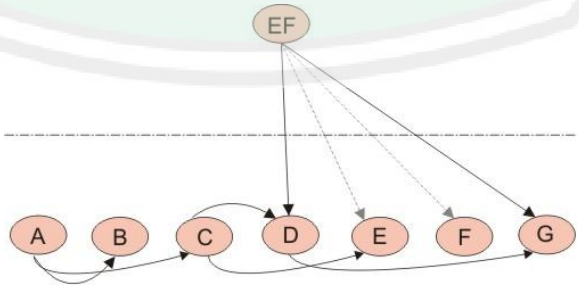
Dengan ketergantungan fungsional kita bisa memantau semua hubungan antara atribut yang berbeda dari tabel. Kita dapat menunjukkan grafik ketergantungan dengan menggunakan satu set simbol-simbol sederhana. Dalam

grafik, panah adalah simbol yang paling penting yang digunakan. Selain itu, dalam cara kita menunjukkan grafik relasi, garis (putus-putus) horisontal memisahkan key sederhana (yaitu, atribut) dari kunci komposit (yaitu, kunci terdiri dari lebih dari satu atribut). Sebuah grafik ketergantungan yang dihasilkan menggunakan aturan berikut.

a. Setiap atribut dari tabel dikelilingi dan semua atribut dari tabel dibuat pada tingkat terendah (yaitu, bawah) dari grafik.
b. Sebuah garis horizontal digambarkan di atas semua atribut.
c. Setiap kunci komposit (jika ada) dikelilingi dan semua tombol komposit tenggelam di atas garis horizontal.
d. Semua panah ketergantungan fungsional diambil.
e. Semua aturan ketergantungan refleksivitas yang digambar menggunakan panah putus-putus (misalnya AB → A, AB → B).

Pertimbangkan sekumpulan ketergantungan fungsional pada Contoh 2 untuk relasi r.

Contoh 2: FD = {A → BCD, C → DE, EF → DG, D →G}



Gambar 2.3 : Menampilkan Graf (Bahmani, 2008).

Jika kita dapat memperoleh semua ketergantungan antara kunci penentu kita dapat memproduksi semua ketergantungan antara semua atribut dari relasi.

Ketergantungan ini diwakili dengan menggunakan Ketergantungan Matriks (DM). Menggunakan algoritma *path finding* dan aturan transitivitas Armstrong ketergantungan baru ditemukan dari kumpulan ketergantungan yang ada. Ini adalah dasar dari algoritma normalisasi yang akan dibahas di bawah ini (Bahmani, 2008).

2.5.2 Ketergantungan Matriks

Menurut Bahmani (2008) Dari Ketergantungan Graf, Ketergantungan Matrix (DM) yang sesuai dihasilkan sebagai berikut:

i. Mendefinisikan matriks DM [n] [m], di mana

   n = jumlah Kunci Penentu.

   m = jumlah Kunci Sederhana.

ii. Anggaplah bahwa $\beta \subseteq \alpha$, $\alpha \not\subset \gamma$ dan

   $\beta, \gamma \in$ {kumpulan kunci Sederhana}

   $\alpha \in$ {kumpulan Kunci Penentu}

iii. Menetapkan elemen DM sebagai berikut:

   Jika $\alpha \rightarrow \beta$ ➔ DM[α][β] = 2,

   jika $\alpha \rightarrow \gamma$ ➔ DM[α][γ] = 1,

   jika tidak ➔ DM[α][γ] = 0,

DM untuk Contoh 1 ditunjukkan pada Tabel 2.12.

Tabel 2.12 Menunjukkan Ketergantungan Matrik

	A	B	C	D	E	F	G
A	2	1	1	1	0	0	0
C	0	0	2	1	1	0	0
D	0	0	0	2	0	0	1
EF	0	0	0	1	2	2	1

2.5.3 Matriks Graf Berarah

Matriks Graf Berarah (DG) untuk kunci penentu digunakan menunjukkan semua kemungkinan ketergantungan langsung antara kunci penentu. DG adalah matriks n × n dimana n adalah jumlah kunci penentu. Proses penentuan elemen mengikuti matriks ini. Unsur-unsur dari matriks DG pada awalnya diatur ke nol. Mulai dari baris pertama dari ketergantungan matriks DM, matriks ini diselidiki dalam pendekatan baris utama. Misalkan kita sedang menyelidiki baris yang sesuai ke kunci penentu x. Jika semua kunci sederhana yang terdiri dari x tergantung pada kunci penentu selain x maka x juga tergantung pada kunci penentu itu (aturan augmentasi Armstrong). Ketergantungan kunci sederhana untuk kunci penentu diwakili oleh non-nol dalam matriks DM.

Sebagai contoh, anggaplah bahwa FD = {AB→ E, BC → A, DE → A}. Matriks ketergantungan yang sesuai dan matriks graf berarah awalnya ditunjukkan pada Gambar 2.4.

	A	B	C	D	E
AB	2	2	0	0	1
BC	1	2	2	0	0
DE	1	0	0	2	2

(a). Ketergantungan Matrik

	AB	BC	DE
AB	0	0	0
BC	0	0	0
DE	0	0	0

(b). Matrik Graf berarah

Gambar 2.4: Inisialisasi Matrik DM dan DG (Bahmani, 2008)

Pada bagian (a) Gambar 2.5, kita mulai dengan baris pertama dari matriks DM. Kunci penentu dari baris ini adalah AB. A dan B adalah himpunan bagian dari AB yang muncul dalam kolom satu dan dua dari matriks. Dalam baris satu, kolom satu dan dua keduanya bukan nol. Oleh karena itu AB tergantung pada AB. Mengingat baris kedua, kolom satu dan dua keduanya bukan nol juga. Oleh karena itu, AB tergantung pada BC. Namun, untuk baris ketiga, itu bukanlah

kasus bahwa A dan B tergantung pada DE. Oleh karena itu, nilai -1 diletakkan di persimpangan baris DE dan kolom AB dalam matriks DG bagian(b) dari gambar 2.5

	A	B	C	D	E
AB	2	2	0	0	1
BC	1	2	2	0	0
DE	1	0	0	2	2

(a)

	AB	BC	DE
AB	1	-1	-1
BC	1	1	-1
DE	-1	-1	1

(b)

Gambar 2.5: Matrik DM dan DG (Bahmani, 2008)

Setelah menghasilkan matriks DG kita mengalihkan perhatian kita terhadap menemukan semua jalur yang mungkin antara semua pasangan. Matriks ini akan menampilkan semua dependensi transitif antara kunci penentu. Ada banyak algoritma *path finding* seperti algoritma Prim, Kruskal, dan Warshal. Jika ada jalur dari node x ke node y berarti y transitif tergantung pada x. Sebagai contoh, Gambar 2.6 menunjukkan determinan dependensi transitif lengkap kunci yang sesuai dengan grafik DM dari Gambar 2.4.

	AB	BC	DE
AB	1	-1	-1
BC	1	1	-1
DE	-1	-1	1

Gambar 2.6: Kunci Penentu Ketergantungan Transitif (Bahmani, 2008)

Dari Gambar 2.6 kita dapat mengurangi bahwa AB tergantung pada BC. Di sisi lain, E tergantung pada AB. Oleh karena itu, E tergantung pada BC. Artinya, BC→ AB, AB → E => BC → E. ketergantungan yang diakui melalui prosedur penutupan ketergantungan yang disajikan pada Gambar 2.7.

	A	B	C	D	E
AB	2	2	0	0	1
BC	1	2	2	0	AB
DE	1	0	0	2	2

Gambar 2.7: E tergantung pada BC melalui AB (Bahmani, 2008)

Pada Gambar 2.7, E tergantung pada BC melalui AB. Ada kemungkinan bahwa E mungkin bergantung pada BC melalui beberapa kunci penentu lainnya, juga. Dalam hal ini tidak akan menjadi masalah kunci penentu yang digunakan dalam Gambar 2.7 untuk mewakili ketergantungan ini. Salah satu masalah untuk berhati-hati adalah bahwa dengan memperbarui matriks DM untuk mencerminkan ketergantungan transitif beberapa dependensi langsung dapat memudar. Pertimbangkan FD = {A → B, B → A dan B → C}. Matriks DM dan DG ditunjukkan pada Gambar 2.8.

	A	B	C
A	2	1	0
B	1	2	1

(a). Ketergantungan Matrik

	A	B
A	1	1
B	1	1

(b). Matrik Graf berarah

Gambar 2.8: Matrik DM dan DG (Bahmani, 2008)

Dengan menggunakan algoritma *path finding*, matriks diperbarui ditunjukkan pada bagian (a) dari Gambar 2.9. Seperti dapat dilihat dari bagian (a) dari Gambar 2.8, ketergantungan langsung C ke B telah memudar. Untuk mengatasi kekurangan ini, algoritma *Circular-Dependency* dirancang. Algoritma ini secara internal menggunakan algoritma rekursif FindOne. Yang terakhir akan menemukan ketergantungan langsung, jika ada, dan menggantikan yang transitif. Hal ini tercermin dalam bagian (b) Gambar 2.9 (Bahmani, 2008).

	A	B	C
A	2	1	B
B	1	2	A

(a)

	A	B	C
A	2	1	B
B	1	2	1

(b)

Gambar 2.9: C → B yang asli telah dikembalikan (Bahmani, 2008)

2.6 Perancangan Basis Data

Menurut Connolly (2002), perancangan basis data (*basis data design*) merupakan proses pembuatan suatu desain untuk sebuah basis data yang mendukung operasional dan sasaran suatu perusahaan.Ada 2 pendekatan untuk mendesain suatu basis data, antara lain:

a. Pendekatan *bottom-up*

Yang dimulai pada tingkat awal dari attribut, yaitu properti dari *entity* dan *relationship*, yang mana melalui analisis dari *asosiasi* antar atribut, dikelompokkan menjadi relasi yang merepresentasikan jenis-jenis *entity* dan relasi antar *entity*. Pendekatan ini cocok untuk mendesain basis data yang sederhana dengan jumlah attribut yang tidak banyak.

b. Pendekatan *top-down*

Digunakan pada basis data yang lebih kompleks, yang dimulai dengan pengembangan dari model data yang mengandung beberapa *entity* dan relasi tingkat tinggi dan kemudian memakai perbaikan *top-down* berturut-turut untuk mengidentifikasi *entity*, relasi dan attribut berkaitan tingkat rendah. Pendekatan ini biasanya digambarkan melalui *Entity Relationship* (*ER*).

2.6.1 Perancangan Basis Data Konseptual

Menurut Connoly (2002), proses perancangan basis data konseptual merupakan proses pembuatan sebuah model informasi yang digunakan dalam sebuah perusahaan, yang tidak tergantung pada semua masalah fisik. Awal tahap ini dimulai dengan pembuatan konseptual data model perusahaan yang secara keseluruhan bebas dari detil implementasi seperti DBMS yang digunakan, program aplikasi, bahasa pemograman, platform untuk hardware, tingkat kinerja, maupun bentuk masalah fisik lainnya.

Model data konseptual didukung dengan dokumentasi, termasuk sebuah kamus data, yang dihasilkan melalui pengembangan model. Berikut ini merupakan langkah- langkah dalam merancang basis data konseptual, antara lain (Connolly, 2002):

a. Identifikasi tipe *entity*

Ada satu metode dalam pengidentifikasian *entity* yaitu dengan mengambil spesifikasi kebutuhan *user*. Dari spesifikasi ini, tentukan kata benda atau frase kata benda yang disebutkan, misalnya noStaf, NmStaf, noProperti, AlamatProperti, sewa, jumlahKamar. Perlu untuk mencari objek utama seperti orang, tempat, atau konsep menarik, memisahkan kata benda tersebut dengan objek yang lainnya. Misalnya, noStaf, NmStaf dikelompokkan dengan sebuah objek atau *entity* yang disebut Staf dan noProperti, AlamatProperti, sewa, dan jumlahKamar dikelompokkan dengan sebuah *entity* yang disebut PropertiSewa.

b. Identifikasi tipe *relationship*

Tujuannya adalah untuk mengidentifikasi hubungan penting yang berada

diantara tipe *entity* yang telah diidentifikasi. Secara khusus, hubungan diindikasikan dengan kata kerja atau *verbal expression.*

c. Identifikasi dan hubungkan atribut-atribut dengan tipe *entity* atau *relationship*

Tujuannya adalah untuk mengasosiasikan atribut-atribut dengan *entity* yang tepat atau tipe *relationship*. Misalnya pada *entity* Staf, atribut nama dapat dikomposisikan lagi menjadi lebih rinci yaitu fNama, lNama. Begitu pun dengan *entity* PropertiSewa, atribut alamat dapat dikomposisikan menjadi jalan, kota, dan kode pos.

d. Menentukan *domain* atribut

Tujuan dari tahap ini adalah untuk menentukan *domain* untuk atribut-atribut dalam model data konseptual lokal. *Domain* adalah sekelompok nilai dari yang satu atau lebih atribut yang mengambil nilai mereka. Sebagai contoh, antara lain:

1) *Domain* atribut yang valid dari noStaf adalah panjangnya 5 karakter string, dimana dua karakter pertama sebagai huruf dan karakter berikutnya bisa berupa angka dari 1-999.
2) *Nilai* yang mungkin untuk atribut JenisKelamin yaitu bisa berupa "M" atau "F".

e. Menentukan atribut *candidate* dan *primary key*

Candidate *key* adalah sekumpulan atribut dari sebuah *entity* yang secara unik mengidentifikasi setiap kejadian dari *entity* tersebut. Pada saat memilih sebuah *primary key* diantara beberapa *candidate key*, perlu mengikuti aturan yang dapat membantu dalam pemilihan, antara lain:

1) *Candidate key* dengan kumpulan atribut yang minim
2) *Candidate key* yang perubahan nilainya kecil
3) *Candidate key* dengan karakter yang paling sedikit
4) *Candidate key* dengan nilai maximum terkecil
5) *Candidate key* yang paling mudah digunakan dari sisi *user*

f. Mempertimbangkan penggunaan dari konsep pemodelan perluasan

Tujuan dari tahap ini adalah untuk mempertimbangkan penggunaan dari konsep pemodelan perluasan, seperti *specialization*/*generalization*, *aggregation*, dan *composition*. Jika menggunakan pendekatan *specialization*, perlu membedakan antara *entity* dengan medefinisikan satu atau lebih *subclass* dari sebuah *superclass entity*. Jika menggunakan pendekatan *generalization*, dengan mengidentifikasi fitur yang umum diantara *entity*. *Aggregation* diperlukan apabila adanya hubungan "has– a" atau "is-part-of" diantara tipe *entity*, dimana satu merepresentasikan "whole" dan "the part" lainnya. *Composition* (tipe khusus dari *aggregation*) diperlukan apabila ada sebuah hubungan diantara tipe *entity* dimana terdapat suatu kepemilikan yang kuat diantara "whole" dan "part".

g. Pengecekkan model untuk redundansi

Pada tahap ini, ada dua aktivitas yang digunakan untuk menghilangkan redudansi, antara lain:

1) Memeriksa kembali hubungan *one-to-one* (1:1)

Identifikasi dua *entity* yang memiliki kesamaan objek dalam suatu perusahaan. Sebagai contoh, dua *entity* Klien dan Penyewa merupakan *entity* yang memiliki kesamaan, sehingga kedua *entity* ini

harus digabungkan bersama. Apabila memiliki perbedaan *primary key*, maka pilih salah satu yang akan menjadi *primary key* dan yang lainnya menjadi *alternate key*.

2) Menghilangkan hubungan yang redundan

Sebuah hubungan yang redundan adalah jika informasi yang sama dapat dihasilkan melalui hubungan yang lain. Sehingga hubungan yang redundan itu tidak diperlukan/dihilangkan.

h. Validasi model konseptual lokal bertentangan dengan transaksi *user*

Ada dua pendekatan untuk menentukan dimana model data konseptual lokal mendukung transaksi yang dibutuhkan *user*, antara lain:

1) Menjelaskan transaksi-transaksi tersebut

Pada pendekatan ini, mengecek semua informasi yang dibutuhkan oleh tiap transaksi yang dibutuhkan oleh model, dengan mendokumentasikan deskripsi dari kebutuhan-kebutuhan tiap transaksi.

2) Menggunakan *pathway* transaksi

Pendekatan kedua ini setiap transaksi direpresentasikan ke dalam bentuk diagram ER. Pendekatan ini mengijinkan perancang untuk memperlihatkan area dari model yang tidak dibutuhkan oleh transaksi dan area itu dimana kritis untuk mengadakan transaksi.

i. Mengulang kembali model data konseptual dengan *user*

Tujuan dari tahap ini adalah untuk mengulang kembali untuk meyakinkan dimana model yang telah dibuat telah benar untuk direpresentasikan. Apabila masih ada maka perlu untuk diulang tahap sebelumnya sampai model itu benar.

2.6.2 Perancangan Basis Data Logikal

Menurut Connolly (2002), perancangan basis data secara logikal merupakan proses pembuatan model informasi yang digunakan perusahaan berdasarkan pada model data khusus, tapi bebas dari DBMS tertentu dan masalah fisik lainnya. Tahap ini mematahkan model konseptual pada sebuah model logikal yang dipengaruhi oleh data model untuk tujuan basis data. Model data logikal merupakan sumber informasi untuk tahap perancangan fisikal, menyediakan suatu kendaraan bagi perancang basis data fisikal, untuk melakukan pertukaran yang sangat penting untuk perancangan basis data yang efisien.

Langkah-langkah dalam perancangan basis data logikal, sebagai berikut:

a. Memindahkan fitur yang tidak kompatibel dengan model relational

Membersihkanmodel data konseptual local untuk menghapus fitur yang tidak kompatibel dengan model relasi. Pada langkah ini kita mentransform struktur kedalam form yang lebih mudah ditangani oleh sistem. Langkah-langkahnya, terdiri dari:

1) *Remove many-to many* (*.*) *type binary relationship*
2) *Remove many-to many* (*.*) *type recursive relationship*
3) *Remove complex type relationship*
4) *Remove multi-valued*

b. Mengambil relasi untuk model data logikal lokal

Tujuannya adalah membuat relasi untuk model data *logical local* untuk menggambarkan entity-entiti, *relationship-relationship*, dan atribut-atribut yang

diidentifikasi. Struktur-struktur yang mungkin disajikan dalam model data:

1) *Type strong entity*
2) *Type weak entity*
3) *One-to-many (1:*) type binary relationship*
4) *One-to-one (1:1) type binary relationship*
5) *One-to-one (1:1) recursive relationship*
6) *Type superclass atau subclass*
7) *Many-to-many (*.*) binary relationship*
8) *Type complex relationship*
9) *Atribut-atribut Multi valued*

c. Validasikan relasi menggunakan normalisasi

Proses normalisasi terdiri atas tahap-tahap sebagai berikut:

1) *Bentuk normal pertama (1NF)*
2) *Bentuk normal kedua (2NF)*
3) *Bentuk normal ketiga (3NF)*
4) *Boyce-Codd Normal Form (BCNF)*
5) *Bentuk normal keempat (4NF)*
6) *Bentuk* normal kelima (5NF)

d. Validasikan relasi bertentangan dengan transaksi *user*

Tujuannya untuk meyakinkan bahwa relasi pada model data *logical local* mendukung transaksi yang diperlukan oleh tampilan.

e. Mendefinisikan *integrity constraints*

Tujuannya untuk mendefinisikan *integrity constraints* yang diberikan dalam

tampilan. *Integrity constraints* adalah *constraints-constraints* yang kita harapkan untuk menyatukan dan melindungi basis data agar tidak menjadi *inconsisten*.

f. Mengulang model data logikal lokal dengan user

Meyakinkan model data logikal lokal dan mendukung dokumentasi yang menjelaskan model tersebut benar untuk direpresentasikan.

2.6.3 Perancangan Basis Data Fisikal

Menurut Connolly (2002), perancangan basis data secara fisik merupakan proses pembuatan sebuah deskripsi dari implementasi basis data pada secondary storage yang menjelaskan basis relasi, organisasi file, dan indeks yang digunakan untuk memperoleh akses pada data yang efisien, dan masalah integritas lainnya yang berkaitan, dan menetukan mekanisme security. Tahap ini memungkinkan perancang untuk menentukan bagaimana basis data diimplementasikan. Antara rancangan logikal dan fisikal terdapat keterkaitan, hal ini disebabkan karena keputusan diambil selama rancangan fisikal untuk meningkatkan kinerja bisa mempengaruhi data model logikal. Kegiatan-kegiatan yang dilakukan, antara lain:

a. Menterjemahkan model data logical global untuk DBMS

1) Mendesain relasi dasar

Tujuannya untuk memutuskan bagaimana cara menggambarkan identitas relasi dasar dalam model data logikal global dalam target DBMS. Untuk setiap identifikasi relasi dalammodel data global, kita memiliki definisi terdiri dari:

a) Nama relasi

b) Daftar atribut-atribut dalam penyimpanan

c) *Primary key* (PK), *alternate key* (AK) dan *foreign keys* (FK)

d) Referensi *constraints integrity* untuk untuk setiap *foreign keys* yang diidentifikasi.

2) Mendesain representasi dari pengambilan data

Memutuskan cara merepresentasikan beberapa pengambilan data muncul dalam model data logical global pada sasaran DBMS. Atribut-atribut yang nilainya dapat ditemukan dengan memeriksa nilai dari atribut lain dikenal dengan *derived* atau *calculated attributes*.

3) Mendesain *entrerprise constraints*

Merancang batasan *enterprise* untuk sasaran DBMS. Sebagai contoh, dalam pembuatan kode SQL berikut ini:

```
CONSTRAINT StaffNotHandlingTooMuch
  CHECK (NOT EXISTS (SELECT staffNo FROM
        PropertyForRent GROUP BY staffNo HAVING
        COUNT(*) > 100))
```

b. Merancang representasi fisikal

1) Menganalisa transaksi

Tujuannya untuk mengerti fungsi dari transaksi yang akan dijalankan pada basis data dan untuk menganalisa transaksi-transaksi penting.

2) Memilih organisasi file

Tujuannya untuk menentukan organisasi file yang efisien untuk setiap relasi dasar. Salah satu tujuan penting dari perancangan basis data fisikal adalah untuk menyimpan data dengan cara yang efisien.

3) Pemilihan Index

Tujuannya untuk menyediakan mekanisme untuk spesifikasi *key* tambahan untuk relasi yang dapat digunakan untuk penerimaan data lebih efisien.

4) Estimasi kapasitas penyimpanan yang dibutuhkan

Tujuannya adalah untuk memperkirakan besarnya penyimpanan file yang dibutuhkan basis data.

c. Merancang pandangan *user*

Tujuannya untuk merancang pandangan dari *user* yang diidentifikasi selama pengumpulan kebutuhan dan analisa tahap dari daur hidup aplikasi basis data relasional.

d. Merancang mekanisme keamanan

Tujuannya untuk mendesain keamanan basis data yang dispesifikasi oleh *user*. Pada umumnya ada dua jenis keamanan basis data:

1) Keamanan sistem

Menangani akses dan penggunaan basis data pada level sistem, seperti nama *user* dan *password*.

2) Keamanan data

Menangani akses dan penggunaan objek basis data (seperti relasi dan tampilan) dan tindakan yang *user* dapat peroleh pada objek.

BAB III
ANALISIS DAN PERANCANGAN

Bab ini akan membahas tentang analisis dan perancangan aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF yang akan dibuat, meliputi pembangunan program terstruktur, overview diagram alur, dan algoritma pemrogramannya, serta bagaimana implementasi perancangan dan analisis program pada *use case*, *activity diagram*.

3.1 Analisis Sistem

Analisis sistem bertujuan untuk mengidentifikasi permasalahan-permasalahan yang ada pada sistem di mana aplikasi dibangun yang meliputi perangkat lunak (software), pengguna (user) serta hasil analisis terhadap sistem dan elemen-elemen yang terkait. Analisis ini diperlukan sebagai dasar bagi tahapan perancangan sistem. Analisis sistem desain dan implementasi ini meliputi deskripsi sistem, desain proses dan implementasi desain dan semua yang diperlukan dalam aplikasi.

3.1.1 Spesifikasi Aplikasi

Aplikasi yang akan dibangun pada skripsi ini adalah Aplikasi mengkontruksi ulang basis data secara otomatis hingga berbentuk 3NF berbasis dekstop. Aplikasi ini dibangun menggunakan bahasa pemrograman JAVA dan MySQL. Adapun spesifikasi dari aplikasi ini adalah sebagai berikut:

a. User memilih tabel pada basis data atau membuat tabel baru kemudian memasukkan ketergantungan fungsional dari tabel tersebut.
b. Dari tabel dan ketergantungan fungsional yang telah dimasukkan kemudian sistem mengubahnya ke dalam dua bentuk matrik yaitu, matrik DM dan DG, setelah itu dari matrik DG sistem memunculkan ketergantungan transitif kunci penentu dengan memeriksa tiap kunci penentu berdasarkan pada *inference rule*. Kemudian proses selanjutnya yaitu *dependency closure* yang digunakan untuk menampilkan semua kemungkinan ketergantungan fungsional yang belum nampak. Setelah itu *circular dependency* yang digunakan untuk menampilkan kembali ketergantungan fungsional yang memudar yang disebabkan oleh proses *circular dependency*. Kemudian sistem mencari ketergantungan sebagian (parsial) pada DM dimulai dari baris pertama sampai terakhir, jika ditemukan sistem akan memisahkan atribut tersebut sehingga terbentuk tabel-tabel baru. Setelah itu sistem mencari ketergantungan transitif dari DM dimulai dari baris pertama sampai terakhir dan jika ditemukan sistem memisahkan atribut tersebut menjadi tabel-tabel baru dan sudah berbentuk normal ketiga kemudian dari hasil tabel-tabel tersebut diubah menjadi kode query yang bisa dijalankan pada sistem MySql.

3.1.2 Spesifikasi Pengguna

Aplikasi ini ditujukan untuk administrator atau perancang basis data dalam pembuatan atau perancangan basis data baru berbentuk 3NF berdasarkan tabel dan ketergantungan fungsional.

3.1.3 Deskripsi Program

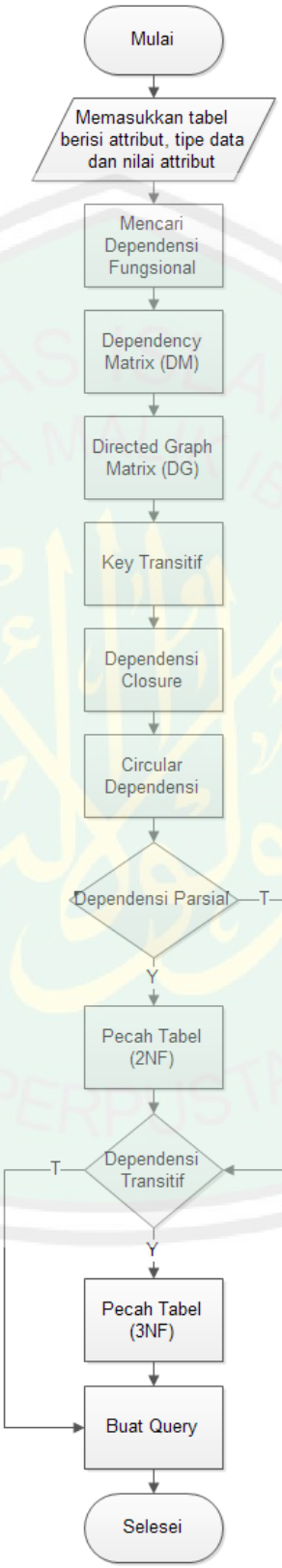
Program yang telah dirancang adalah aplikasi mengkontruksi ulang basis data secara otomatis hingga berbentuk 3NF berbasis dekstop, aplikasi ini nantinya akan digunakan sebagai *tool*/ alat dalam mengontruksi ulang basis data hingga berbentuk 3NF dengan menghilangkan ketergantungan parsial dan transitif. Sehingga dengan adanya aplikasi ini diharapkan bisa menghemat waktu, biaya, dan dapat berjalan dengan efektif serta menghasilkan basis data yang minim akan redundansi dan anomali.

3.1.4 Pembuatan Algoritma Program

Dalam pembuatan aplikasi mengkontruksi ulang basis data secara otomatis hingga berbentuk 3NF ini dalam pelaksanaannya, proses mengkontruksi ulang basis data didasarkan pada tabel dan ketergantungan fungsional. Di mana dari ketergantungan fungsional tabel akan ditemukan ketergantungan parsial dan transitif yang akan diolah menjadi tabel baru. Adapun alur program flowchart dari sistem ditunjukkan pada gambar 3.1.

3.2 Perancangan Sistem

Perancangan sistem untuk membangun aplikasi mengkontruksi ulang basis data secara otomatis hingga berbetuk 3NF ini adalah sebagai berikut:



Gambar 3.1 Algoritma Pemrograman

3.2.1 Alur Proses

Pada awalnya aplikasi ini dihubungkan dengan sistem basis data MySql sehingga aplikasi ini bisa mendapatkan seluruh basis data pada aplikasi tersebut. *User* memilih salah satu basis data kemudian sistem mengambil atribut-atribut serta isi dari basis data tersebut. Dari masukkan tersebut kemudian sistem menjalankan prosedur *closure* dan diproses menjadi basis data yang baru berbentuk 3NF. Berikut adalah langkah-langkahnya sebagai penjelasan:

a. *User* memilih salah satu basis data yang yang terdapat pada aplikasi kemudian sistem mengambil nama tabel, atribut-atribut beserta isinya dan tipe data. Perhatikan tabel 3.1.

Tabel 3.1 Tabel Penduduk

kode_pend	nama_pend	nama_desa	nama_kecamatan
pd001	M. Habib	Bulu Brangsi	Laren
pd005	Heri Istanto	Godog	Laren
pd007	Arif Firmansyah	karang wungu	Laren
pd008	Ahmad Rizal	Godog	Laren
pd021	Agus Salim	Payaman	Solokuro
pd050	Miftakhul Huda	Banyubang	Solokuro
pd056	Fahrur Rozi	Takeran	Solokuro
pd057	Suhartin	Takeran	Solokuro

b. Sistem melakukan pencarian ketergantungan fungsional dengan membandingkan seluruh data dari tiap attribut. Perbandingan dilakukan dengan cara membandingkan tiap nilai dari dua jenis atribut yang berbeda dengan asumsi pembuatan kombinasi perbandingan bahwa tiap atribut muncul satu kali dan urutan diperhatikan. Misalkan salah satu hasil kombinasi dari sistem adalah perbandingan antara attribut *kode_pend* terhadap *nama_pend.* Dari kedua atribut tersebut sistem meminta memberikan data

yang berbeda (*distinct*) pada MySql dan membandingankan attribut tersebut dari seluruh data yang dimiliki, perhatikan tabel 3.2. Dari tabel tersebut dapat dilihat bahwa setiap nilai pada attribut *nama_pend* bergantung pada nilai atribut *kode_pend* hal ini dapat dilihat bahwa tiap nilai pada atribut *nama_pend* pasti memiliki nilai yang sama pada atribut *kode_pend*, jika ada nilai pada attribut *kode_pend* yang sama kemudian memiliki nilai yang berbeda pada attribu *nama_pend* maka *nama_pend* tidak bergantung pada *kode_pend* atau ditulis dengan *kode_pend* ↛ *nama_pend*. Proses ini diulang sesuai dengan jumlah kombinasi sehingga didapatkan ketergantungan fungsional, perhatikan pada tabel 3.3.

Tabel 3.2 Tabel *Distinct kode_pend* dan *nama_pend*

kode_pend	nama_pend
pd001	M. Habib
pd005	Heri Istanto
pd007	Arif Firmansyah
pd008	Ahmad Rizal
pd021	Agus Salim
pd050	Miftakhul Huda
pd056	Fahrur Rozi
pd057	Suhartin

Tabel 3.3 Tabel Ketergantungan Fungsional

No	Ketergantungan Fungsional
1	*kode_pend → nama_pend; nama_desa; nama_kecamatan*
2	*nama_pend → kode_pend; nama_desa; nama_kecamatan*
3	*nama_desa → nama_kecamatan*

c. Ketergantungan fungsional yang terdapat pada tabel 3.3 kemudian diolah menjadi matrik DM (*dependency matrix*) sesuai dengan ketentuan yang sudah dijelaskan pada BAB II. Dari ketentuan tersebut dapat disimpulkan bahwa matrik DG merupakan matrik n x m. n adalah jumlah kunci penentu

sedangkan m adalah jumlah kunci sederhana. Dari ketergantungan fungsional tersebut dapat digunakan sebagai acuan untuk memberikan nilai pada matrik DM, sebagai contoh, *kode_pend → nama_pend, nama_desa, nama_kecamatan* berdasarkan *inference rule* dapat diartikan menjadi *kode_pend → kode_pend, kode_pend → nama_pend, kode_pend → nama_desa, kode_pend →nama_kecamatan.* Ketika kunci penentu bertemu dengan kunci sederhana yang merukan bagian dari kunci penentu maka matrik diberi nilai 2 sedangkan jika kunci penentu bertemu dengan kunci sederhana maka matrik diberi nilai 1 selain itu diberi nilai 0. Proses tersebut diulang sebanyak ketergantungan fungsional yang dimiliki, perhatikan tabel 3.4. Kemudian pada matrik tersebut dilakukan pencarian ketergantungan fungsional yang berkebalikan, misalnya A → B dan B →A. Pada tabel 3.4 dapat dilihat bahwa *kode_pend → nama_pend* dan *nama_pend → kode_pend* merupakan ketergantungan fungsional yang berkebalikan dan memiliki pola yang berlawanan pada kunci penentu sehingga seluruh nilai pada baris *nama_pend* diganti dengan 0, perhatikan tabel 3.5.

Tabel 3.4 Tabel Matrik DM

	kode_pend	nama_pend	nama_desa	nama_kecamatan
kode_pend	2	1	1	1
nama_pend	1	2	1	1
nama_desa	0	0	2	1

Tabel 3.5 Tabel Hasil Pembetulan Matrik DM

	kode_pend	nama_pend	nama_desa	nama_kecamatan
kode_pend	2	1	1	1
nama_pend	0	0	0	0
nama_desa	0	0	2	1

d. Matrik DG (*directed graph matrix)* merupakan matrik ketergantungan antar kunci penentu yang berbentuk n x n, n adalah jumlah kunci penentu. Nilai dari matrik ini didapatkan dari nilai matrik DM, jika dari matrik tersebut terdapat nilai 0 maka paka pada matrik DG diberi nilai -1, selain itu diberi nilai 1. Tabel 3.5 pada bari dan kolom pertama memiliki nilai bukan 0 sehingga pada matrik DG pada baris dan kolom pertama diberi nilai 1, sedangakan pada baris kedua dan kolom pertama matrik DM memiliki nilai 0 sehingga pada matrik DG diberi nilai -1, perhatikan pada tabel 3.6.

Tabel 3.6 Tabel Matrik DG

	kode_pend	nama_pend	nama_desa
kode_pend	1	1	1
nama_pend	-1	-1	-1
nama_desa	-1	-1	1

e. Kunci transitif digunakan untuk menampilkan ketergantungan fungsional yang lain dengan menggunakan sifat transitifitas pada *inference role*. Proses ini dilakukan pada matrik DG akan tetapi pada matrik ini tidak ditemukan implikasi dari sifat transitif.

f. Dependensi *closure* digunakan untuk mengolah matrik DM berdasarkan sifat transitif ketergantungan fungsional yang dapat memunculkan ketergantungan fungsional yang lain. Pada awalnya sistem melakukan pencarian nilai 1 pada matrik DG dengan ketentuan tidak boleh berada index baris dan kolom yang sama, misalkan ditemukan pada baris pertama dan kolom ketiga (1,3). Kemudian sistem melanjutkan pencarian nilai 1 matrik DM pada baris ketiga, menggunakan index kolom matrik DG yang ditemukan, jika ditemukan nilai 1 pada matrik tersebut misalkan (3,4) maka nilai matrik DM pada baris

pertama, baris yang sama pada matrik DG, dan kolom keempat, kolom yang sama pada matrik DM, diberikan nilai baru yaitu nama kolom matrik DG yang ditemukan. Perhatikan tabel 3.7.

Tabel 3.7 Tabel Dependensi *Closure*

	kode_pend	nama_pend	nama_desa	nama_kecamatan
kode_pend	2	1	1	nama_desa
nama_pend	0	0	0	0
nama_desa	0	0	2	1

g. *Circular* dependensi digunakan untuk mengembalikan ketergantungan fungsional yang telah memudar. Proses ini dilakukan dengan cara membandingkan matrik DM hasil dari proses dependensi closure, perhatikan tabel 3.7 dengan matrik DM yang sebelumnya, perhatikan tabel 3.5. sistem akan melakukan pencarian mulai dari baris dan kolom pertama sampai pada baris dan kolom terakhir. Apabila sistem menemukan nilai yang berubah maka dilanjutkan pencarian nilai 1 pada kolom yang ditemukan sebelumya, jika tidak ditemukan maka matrik yang nilainya telah berubah tersebut diberi nilai 1. Pada proses ini sistem tidak melakukan perubahan pada matrik DM.

h. Sistem melanjutkan pencarian dependensi parsial pada matrik DM dengan ketentuan kunci sederhana bergantung pada kunci penentu yang merupakan bagian dari kunci penentu yang lain. Jika dalam pencarian ini ditemukan dependensi parsial maka sistem menyimpan index baris dalam bentuk array.

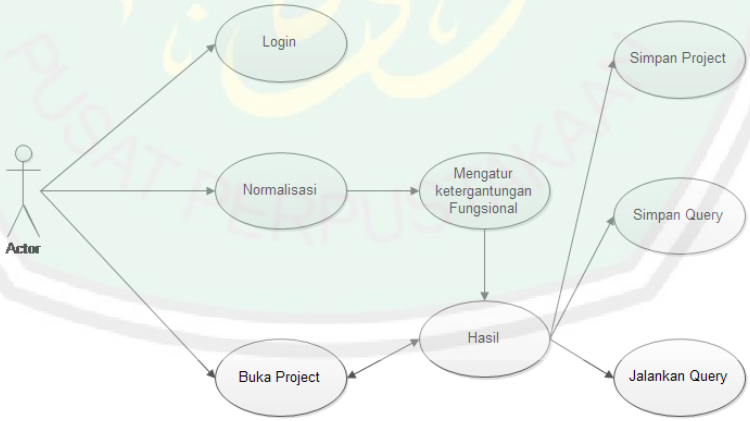
i. Proses pemecahan tabel dilakukan berdasarkan index baris yang telah disimpan dalam array. Pada awalnya dilakukan penyimpanan attribut beserta kuncinya pada array tabel baru kemudian dilakukan pemecahan pada baris sesuai dengan index yang telah ditemukan dependensi parsial.

j. Langkah yang selanjutnya yaitu pencarian dependensi transitif dengan ketentuan kunci sederhana bergantung pada kunci penentu yang bukan bagian dari kunci penentu yang lain. Dari proses ini, sistem menyimpan index baris dari dependensi transitif yang telah ditemukan kemudian dilanjutkan dengan pemecahan tabel.

k. Dari tabel-tabel baru yang sudah disimpan dalam bentuk array, sistem mengubahnya menjadi *query* yang sudah terdapat *constraint* antar tabel sehingga tidak perlu lagi menghubungkan tabel secara manual. Pada aplikasi ini *user* dapat menjalankan kode *query* pada sistem MySql.

3.2.2 Desain Use Case Diagram

Berikut ini adalah desain *use case* pada perancangan aplikasi mengkontruksi ulang basis data secara otomatis hingga berbentuk 3NF.



Gambar 3.2 Desain Use Case Diagram

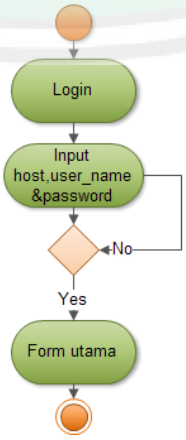
Dari gambar 3.2 dapat dijelaskan bahwa Pertama kali pekerjaan atau aktifitas (*use case*) yang dilakukan adalah aktifitas login, jika Login berhasil maka aktifitas Normalisasi dan Buka Project akan aktif dan bisa digunakan oleh aktor. Pada Aktifitas normalisasi, aktor memilih tabel yang ada pada aplikasi kemudian sistem

menampilkan hasil dari pencarian ketergantungan fungsional kemudian aktor dapat mengolah ketergantungan fungsional tersebut atau langsung melanjutkan proses sehingga aktor langsung mendapatkan hasil pada aktifitas Hasil. Aktifitas Buka Project digunakan ketika aktor ingin membuka kembali project yang telah dibuat sebelumnya, aktor akan langsung mendapatkan hasil normalisasi pada aktifitas Hasil. Setelah aktifitas Hasil, aktor memiliki tiga aktifitas yang lain yaitu, Simpan Project digunakan untuk menyimpan project, Simpan Query digunakan untuk menyimpan query dari proses normalisasi, dan Jalankan Query digunakan untuk menjalankan kode query pada mesin MySql.

3.2.3 Desain Activity Diagram

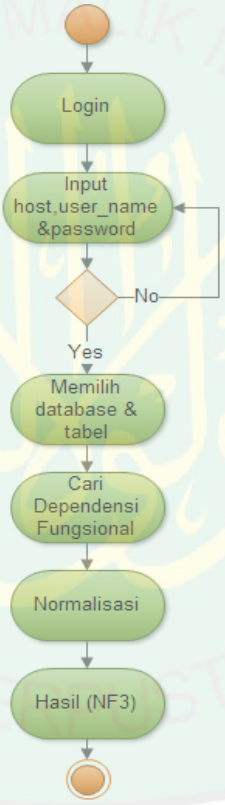
Pada aplikasi ini aktifitas yang pertama kali dilakukan adalah aktifitas login dikarenakan sistem harus terhubung pada MySql agar aplikasi dapat menerima maupun mengolah data dan struktur basis datanya.

Pada gambar 3.3 memperlihatkan aktifitas login, yang dimulai dari memasukkan *host name, user name* dan *password* sampai sistem bisa memunculkan form utama.



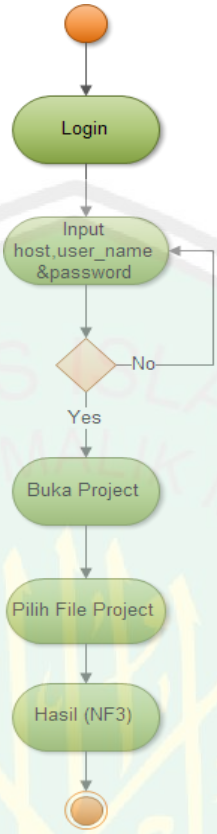
Gambar 3.3 *Activity Diagram* (Aktifitas Login)

Aktifitas yang selanjutnya adalah proses normalisasi basis data, aktor memilih salah satu basis data yang ada pada sistem MySql dan basis data yang dipilih diharus meiliki nilai, bukan *null value*. Dari data tersebut sistem mencari ketergantungan fungsional kemudian dilanjutkan dengan proses normalisasi sehingga aktor mendapatkan desain basis data baru yang sudah berbentuk 3NF. Untuk lebih jelasnya perhatikan gambar 3.4 sebagai berikut.



Gambar 3.4 *Activity Diagram* (Aktifitas Normalisasi)

Selanjutnya adalah aktifitas Buka Project yang digunakan untuk membuka project yang sebelumnya disimpan kemudian aktor bisa langsung mendapatkan hasil dari proses normalisasi basis data yang sudah berbentuk 3NF tanpa melalui proses yang panjang. Untuk lebih jelasnya perhatikan gambar 3.5 sebagai berikut.



Gambar 3.5 *Activity Diagram* (Aktifitas Buka Project)

Aktifitas-aktifitas yang ada pada tiap *use case* tentu tidak semua digambarkan dengan keseluruhan disebabkan oleh fungsi diagram aktifitas hanya sebagai pelengkap untuk mempermudah pemahaman sistem, dan tidak berpengaruh terhadap hasil *generate code* nantinya.

3.2.4 Desain Sequence Diagram

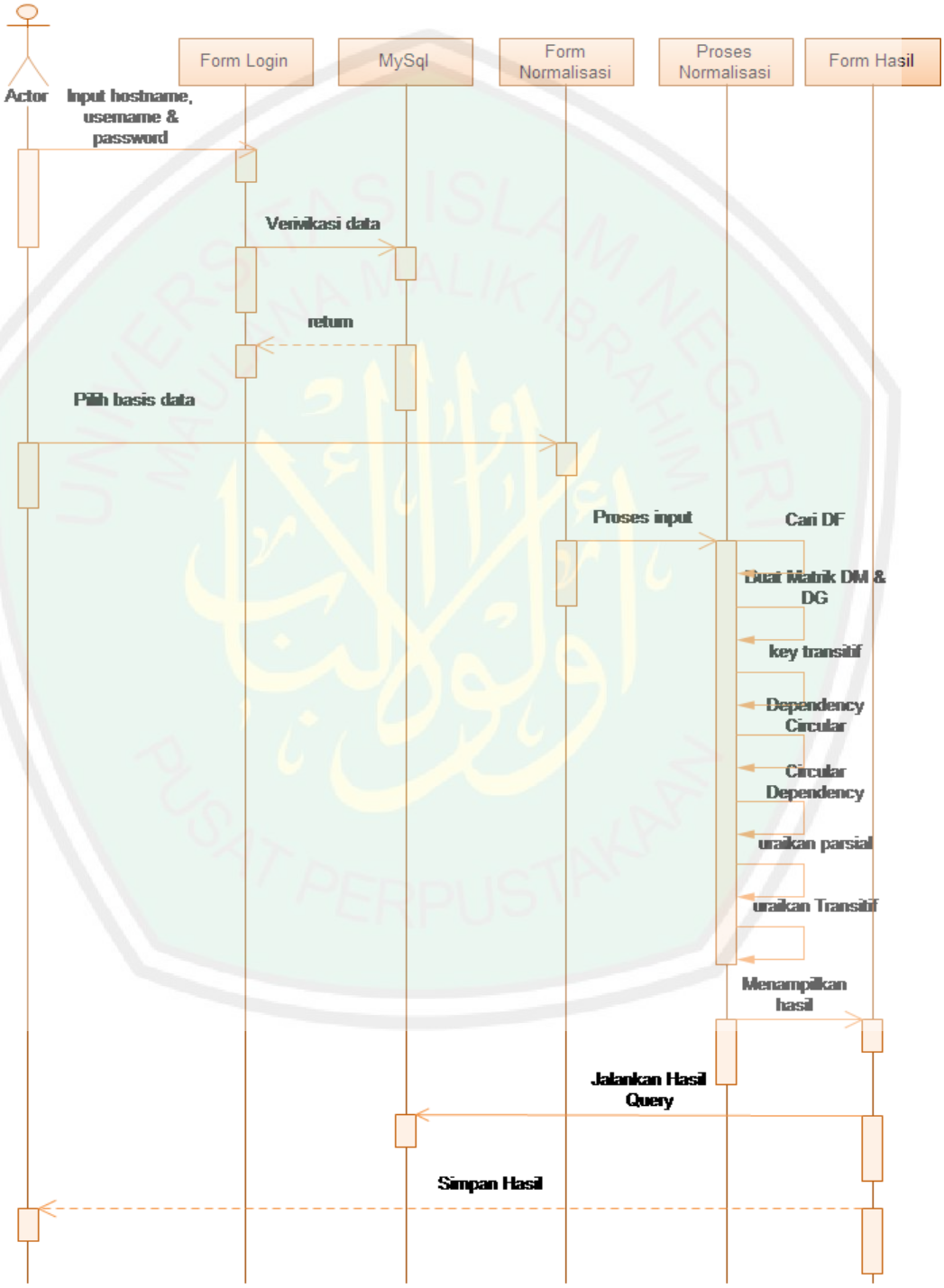
Dalam perancangan sistem mengkontruksi ulang basis data hingga berbentuk 3NF, pertama kali adalah alur melakukan login, karena semua aktor diharuskan untuk melakukan login untuk bisa mengakses pada sistem basis data MySql dengan cara memasukkan *host name, user name,* dan *password.* Kemudian untuk melakukan normalisasi basis data aktor diharuskan untuk memasukkan basis data

(berisi tabel, attribut, dan panjang attribut) dan ketergantungan fungsional. Dari masukkan data tersebut kemudian dibentuk menjadi dua matrik yaitu matrik DM dan DG, kemudian dilanjutkan dengan proses *key transitif* yang bertujuan untuk memunculkan ketergantungan antar kunci penentu yang lain pada matrik DG. Setalah itu proses *dependency closure* yang digunakan untuk menampilkan seluruh kemungkinan ketergantungan fungsional dari matrik DM berdasarkan pada matrik DG. Kemudian dilanjutkan pada proses *circular dependency* yang berguna untuk mengecek ketergantungan fungsional dari matrik DM yang semestinya memudar atau tidak.

Setelah proses tersebut dilakukan pencarian ketergantungan parsial pada matrik DM, jika ditemukan maka matrik DM diuraikan menjadi tabel-tabel baru. Proses ini diulang sampai tidak ditemukan kembali ketergantungan parsial pada matrik tersebut. Kemudian dilanjutkan dengan pencarian ketergantungan transiif pada matrik DM, jika ditemukan maka ketergantungan transitif tersebut diuraikan dari matrik DM menjadi tabel-ralasi baru. Proses tersebut diulang sampai tidak ditemukannya ketergantungan transitif. Jika sudah tidak ditemukan ketergantungan transitif maka akan dilanjutkan proses pembuatan query dari tabel yang dihasilkan dari proses normalisasi dan ditampilkan pada form hasil normalisasi. Untuk lebih jelasnya perhatikan gambar 3.6.

Query yang dihasilkan dari proses normalisasi ini dapat langsung dijalankan pada sistem MySql sehingga aktor langsung memperoleh basis data baru pada sistem MySql. Pada aplikasi ini, aktor diberikan fasilitas utuk menyimpan dan membuka project hal ini dikarenakan untuk kemudahan aktor jika suatu saat ingin

memodifikasi ketergantungan fungsional dan mendapatkan hasil secara cepat tanpa melalui proses dari awal.



Gambar 3.6 *Sequence Diagram* (Proses Normalisasi)

3.3 Perancangan Aplikasi

Pada perancangan aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF ini akan dijelaskan mengenai rancangan aplikasi yang akan dikerjakan serta fitur-fitur yang akan dipakai pada aplikasi tersebut. Aplikasi ini memerlukan inputan berupa tabel dan ketergantungan fungsional setelah itu sistem akan memproses inputan tersebut menjadi kode query yang baru yang di dalamnya terdapat tabel-tabel dari proses normalisasi. Query yang dihasilkan dapat dijalankan langsung dalam sistem MySql atau disimpan berupa file sehingga pengguna bisa menggunakan query ini pada aplikasi MySql GUI lain seperti PhpMyAdmin, SqlYog, dan lain-lain. Selain itu aplikasi ini mampu digunakan untuk membuka atau menyimpan project sehingga pengguna akan mendapatkan kemudahan jika ingin memodifikasi dan langsung mendapatkan hasil tanpa melalui proses dari awal.

3.3.1 Perancangan User Interface

Aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF ini dibuat sebagai alat yang dapat digunakan untuk menormalisasi basis data yang mampu menghasilkan desain basis data baru berbentuk 3NF. Desain interface menu-menu nantinya akan diaplikasikan sebagai tampilan pada halam aplikasi. Adapun fasilitas dan struktur menu terdiri dari:

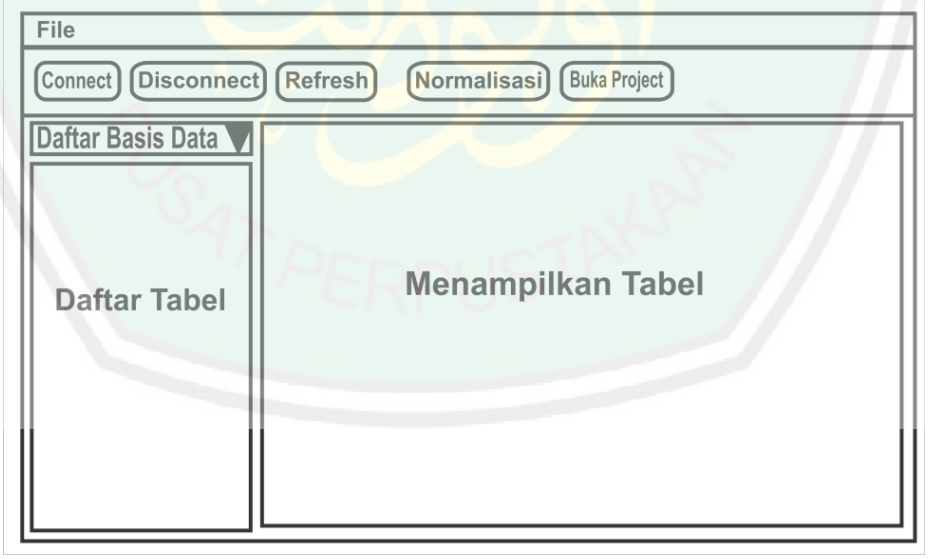
a. Desain Menu Connect/ Login

Menu login merupakan menu yang pertama kali muncul dan digunakan. Jika login berhasil makan menu-menu pada tampilan utama akan aktif dan aplikasi akan terhubung dengan sistem MySql, berikut desain menu login :

Gambar 3.7 Gambar Form Login

b. Desain Halaman Utama

Desain halaman utama terdapat menu connect, disconnect, refresh, buka project dan normalisasi. Selain itu halaman ini dapat menampilkan basis data, tabel dan isi tabel dari MySql. Berikut desain halaman utama:

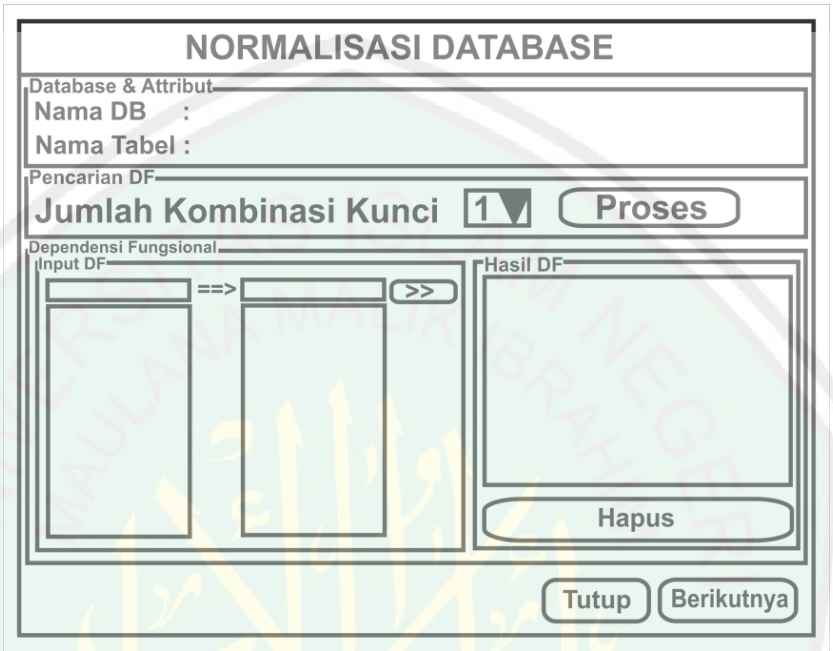


Gambar 3.8 Gambar Halaman Utama

c. Desain Halaman Normalisasi

Desain halaman DB lama merupakan halaman yang digunakan dalam proses normalisai yang memiliki form basis data, form pengisian

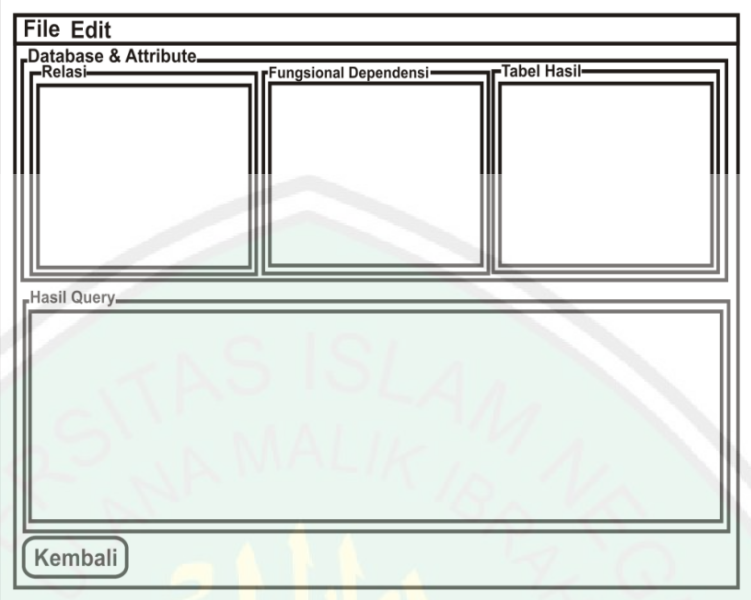
ketergantungan fungsional dan form hasil pengisian yang secara otomatis akan muncul. Berikut desain halaman normalisasi:



Gambar 3.9 Gambar Halaman Normalisasi

d. Desain Halaman Hasil

Desain halaman hasil merupakan halaman yang digunakan untuk menampilkan hasil normalisasi. Halaman ini memiliki form tabel basis data, form ketergantungan fungsional, form hasil basis data dari proses normalisasi dan form yang menampilkan query dari form hasil basis data. Selain itu halaman ini terdapat menu buka project, simpan project, simpan query, jalankan query (pada mesin MySql) dan kembali (pada halaman normalisasi). Berikut desain halaman hasil:



Gambar 3.10 Gambar Halaman Hasil

e. Desain Halaman Pesan Kesalahan

Desain halaman pesan kesalahan terdapat form detail pesan kesalahan yang digunakan untuk menampilkan segala pesan kesalahan yang terjadi pada aplikasi ini. Berikut desain halaman DB lama:

Gambar 3.11 Gambar Halaman Pesan Kesalahan

BAB IV

HASIL DAN PEMBAHASAN

Pada Bab ini akan membahas tentang hasil program yang telah dibangun dengan lingkungan uji coba yang telah ditentukan, hasil program berupa aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF berbasis dekstop. Selanjutnya akan dibahas fitur pada program serta cara pembuatannya.

4.1 Lingkungan Uji Coba

Pada subbab ini dijelaskan mengenai lingkungan uji coba yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam uji coba antara lain adalah :

Tabel 4.1 Lingkungan Uji Coba

Perangkat Keras	Prosesor : Intel dual core 2,2 Ghz Memori : 1,5 GB Piranti Masukan: - Mouse - Keyboard
Perangkat Lunak	Sistem Operasi : Microsoft Windows 7 Ultimate 32bit Perangkat Pengembang : Netbeans IDE 7.0.1 , Java (jdk-6u20), appserv-win32-2.5.10 (MySQL), dan MySql Connector java 5.1.5. Perangkat penunjang : SQLyog Ultimate 9.0.1.1 dan Microsoft Access 2010

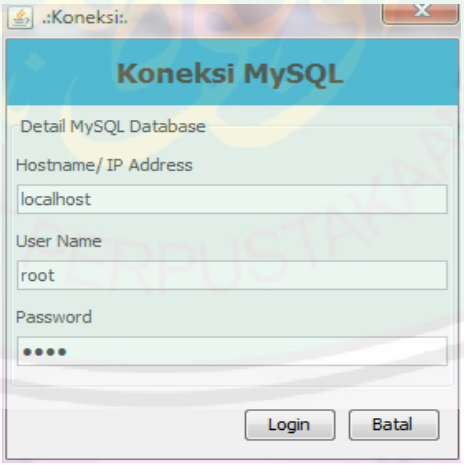
Pada tabel 4.1 dapat dilihat bahwa aplikasi dibuat menggunakan bahasa pemrograman Java dengan Netbeans sebagai editornya dan Appserv sebagai MySql server. Pengujian dilakukan pada basis data anggota Ikatan Mahasiswa Muhammadiyah UIN Malang kemudian dari hasil pengujian dari aplikasi ini dibandingkan dengan hasil dari menu *analyze table* Microsoft Access berdasarkan materi uji yang sama.

4.2 Implementasi Interface

Aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF merupakan aplikasi yang dirancang dan dibangun untuk membantu dan memudahkan perancang atau basis data administrator dalam merancang basis data. Aplikasi ini nantinya dihubungan dengan sistem MySql sehingga pengguna akan dimudahkan dalam memilih basis data yang dinormalisasi selain itu pengguna dapat langsung menjalankan kode query yang dihasilkan dari aplikasi ini. Berikut adalah tampilan aplikasi.

a. Tampilan Login

Pada tampilan login, *user* diminta untuk memasukkan *host name*, *user name*, dan *password*agar aplikasi ini dapat terhubung dengan sistem MySql, jika masukkan benar maka tampilan halaman utama akan aktif.



Gambar 4.1 Tampilan Login

b. Tampilan Utama

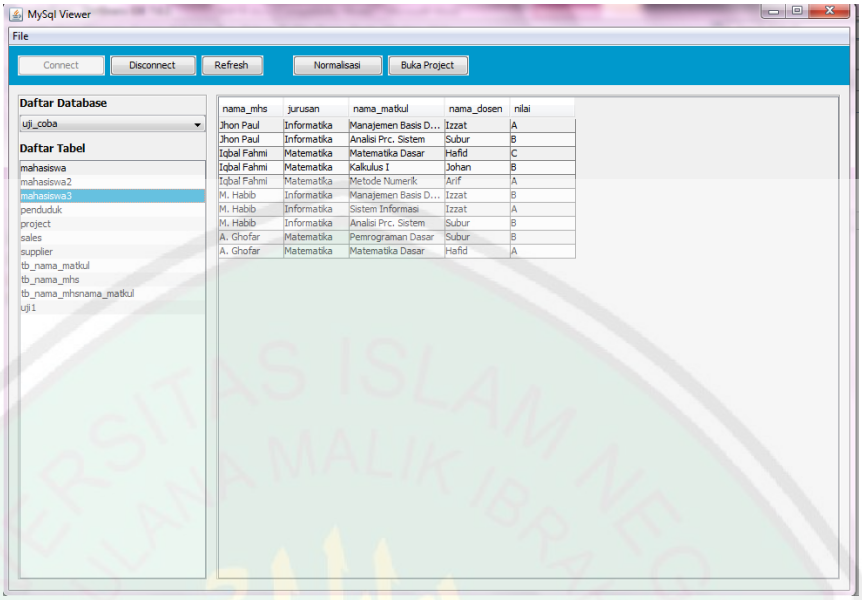
Pada tampilan utama ini *user* diberikan fasilitas untuk melihat seluruh isi basis data dimulai dari nama basis data, tabel, attribut dan isi dari attribut. Selain itu pada halaman ini terdapat fitrur-fitur penting antara lain,

1) Connect digunakan untuk menghubungkan aplikasi dengan sistem MySql dan mengaktifkan fitur-fitur pada aplikasi ini.
2) Disconnect digunakan untuk memutuskan hubugan aplikasi dengan sistem MySql dan fitur-fitur pada aplikasi menjadi aktif.
3) Refresh digunakan menyegarkan kembali data-data dari sistem MySql.
4) DB Lama digunakan untuk melakukan proses normalisasi yang menggunakan basis data yang ada pada sitem MySql. Aplikasi akan mengambil data yang dipilih oleh *user* berupa nama basis data, tabel, attribut dan panjang attribut.
5) DB Baru digunakan untuk melakukan proses normalisasi dengan memasukkan basis data baru yang belum terdapat pada sistem MySql.
6) Buka Project digunakan untuk membuka Project yang telah disimpan dan *user* bisa langsung mendapatkan hasil dari proses ini.
7) Keluar digunakan untuk menutup aplikasi ini.

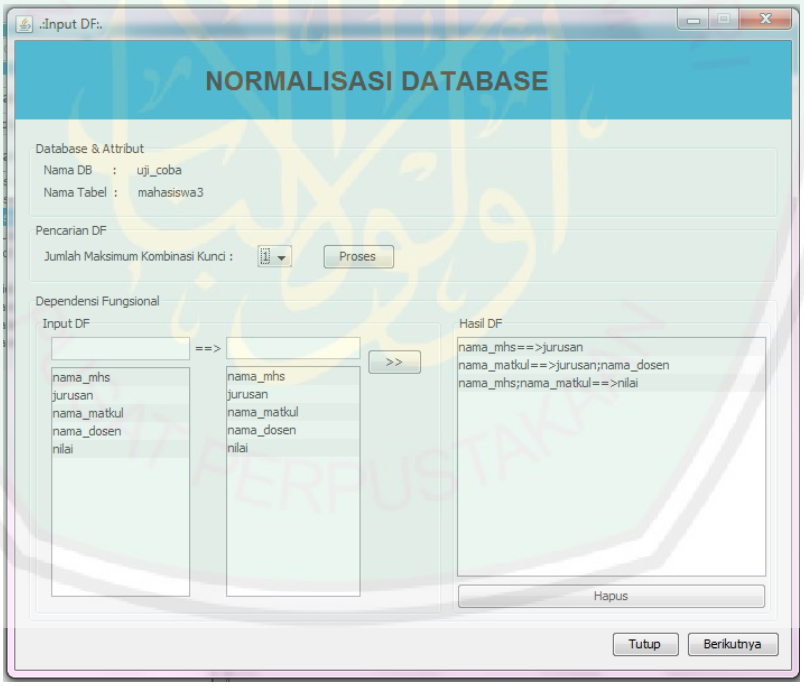
Perhatikan pada gamabar 4.2 sebagai berikut.

c. Tampilan Normalisasi

Pada tampilan normalisasi, *user* diminta untuk memilih jumlah kombinasi kunci kemudian sistem menampilkan ketergantungan fungsional. *User* juga dapat mengolah hasil DF berupa menambahkan atau mengubah ketergantungan fungsional. Perhatikan pada gamabar 4.3 sebagai berikut.



Gambar 4.2 Tampilan Utama



Gambar 4.3 Tampilan Normalisasi

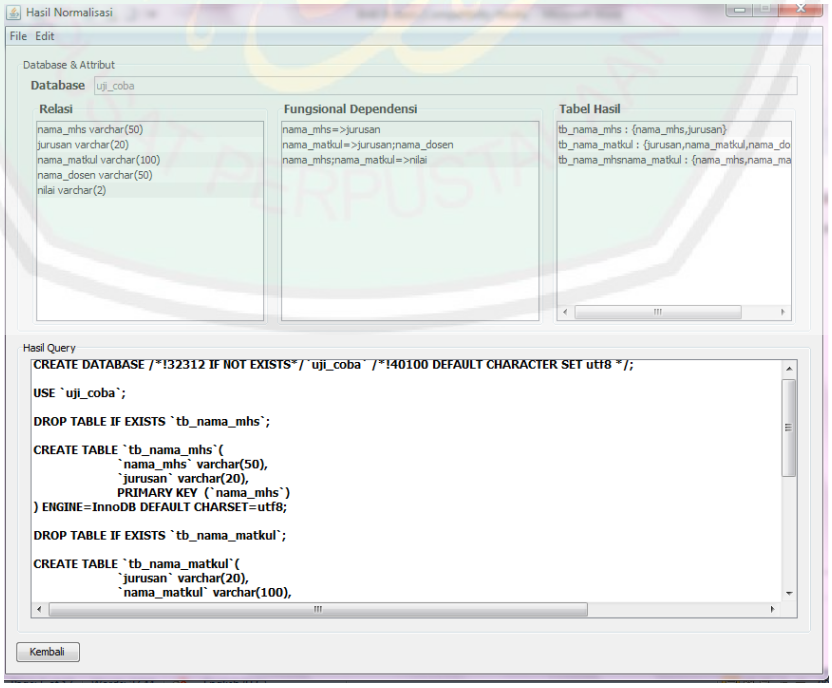
d. Tampilan Hasil

Pada tampilan hasil ini, sistem melakukan proses normalisasi berdasarkan masukkan sebelumnya berupa tabel dan ketergantungan fungsional. Dari proses tersebut dihasilkan tabel-tabel yang menjadi basis data baru kemudian

sistem menampilkan berupa code query MySql. Tampilan ini terdapat fitur-fitur yaitu,

1) Simpan Project digunakan untuk menyimpan Project yang telah dibuat. Di dalamnya tersimpan nama basis data, tabel, attribut, panjang attribut dan ketergantungan fungsional.
2) Buka Project digunkan untuk membuka Project yang telah dibuat.
3) Simpan Query digunakan untuk menyimpan kode query dari hasil normalisasi.
4) Jalankan Query digunakan untuk menjalankan kode query yang dihasilkan pada sistema MySql.
5) Kembali digunakan untuk kembali ke tampilan sebelumnya.
6) Keluar digunakan untuk menutup tampilan hasil.

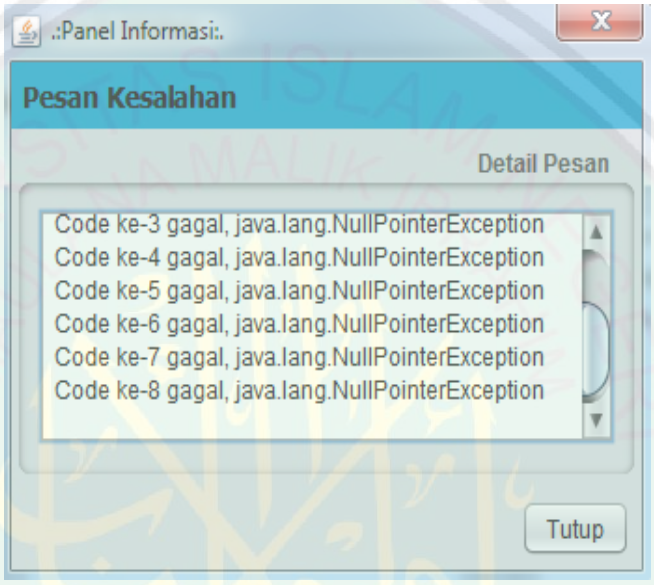
Perhatikan pada gamabar 4.4 sebagai berikut.



Gambar 4.4 Tampilan Hasil

e. Tampilan Pesan Kesalahan

Pada tampilan pesan kesalahan ini digunakan untuk menampilkan segala pesan kesalahan yang terjadi dalam proses normalisasi. Perhatikan pada gamabar 4.5.



Gambar 4.5 Tampilan Pesan Kesalahan

4.3 Implementasi Sistem

Proses normalisasi dimulai dengan memasukkan tabel dari basis data berisi attribut-atribut. kemudian dilakukan proses normalisasi hingga mendapatkan tabel-tabel dari basis data yang sudah berbentuk 3NF. Untuk mendapatkan tabel-tabel yang berbentuk 3NF ini sistem harus melewati beberpa tahapan.

Tahapan-tahapan tersebut adalah (1) Pembuatan matrik DG dan DM, (2) ketergantungan transitif kunci penentu, (3) *dependency closure*, (4) menggantikan ketergantungan transitif dengan ketergantungan langsung yang asli atau dikenal dengan *circular dependency*, (5) menguraikan ketergantungan parsial, (6) menguraikan ketergantungan transitif.

4.3.1 Pembuatan Matrik DG dan DM

Pembuatan matrik DG dan DM merupakan proses yang dilakukan untuk inisialisasi matrik DG dan DM berdasarkan data yang dimasukkan berupa tabel dan ketergantungan fungsional. Attribut-atribut dari tabel ini nantinya akan disebut sebagai kunci sederhana sedangkan ketergantungan fungsional terbagi menjadi dua yaitu kunci penentu (sisi kiri) dan dependen (sisi kanan).

Pertama kali Matrik-matrik ini akan dideklarasikan dalam bentuk array dua dimensi. Matrik DG memiliki panjang baris berdasarkan panjang tabel sedangkan panjang kolom berdasarkan kunci penentu dan matrik DM memiliki panjang baris dan kolom berdasarkan kunci penentu.

Proses yang selanjutnya yaitu pemberian nilai pada matrik DG berdasarkan ketergantungan fungsional. Tiap-tiap ketergantungan fungsional memiliki kunci penentu dan dependen. Jika kunci penentu bertemu dengan kunci sederhana yang merupakan himpunan bagian dari kunci penentu maka matrik diset 2, jika kunci penentu bertemu dengan kunci sederhana yang merupakan himpunan bagian dari dependen maka matrik DG diset 1 dan jika kunci penentu tidak bertemu dengan keduanya maka diset 0, perhatikan gambar 4.6.

Kemudian proses yang selanjutnya Pemberian nilai pada matrik DM berdasarkan pada nilai matrik DG yang mana nilai-nilai tersebut didapatkan dari penggabung kunci-kunci sederhana sesuai dengan kunci penentu. Jika pada kolom-kolom yang digabungkan ditemukan nilai 0 maka pada matrik DM diberikan nilai -1 dan jika tidak maka diberikan nilai 1, perhatikan gambar 4.7.

```
private void buatDg(){
for (int i = 0; i < node1.length; i++) {

        node1Sem=new ArrayList();
        StringTokenizer st=new StringTokenizer(node1[i],";");
while (st.hasMoreTokens())
        {  node1Sem.add(st.nextToken());}
        dtSem1=new String[node1Sem.size()];
node1Sem.toArray(dtSem1);

        node2Sem=new ArrayList();
        StringTokenizer st2=new StringTokenizer(node2[i],";");
while (st2.hasMoreTokens())
        {  node2Sem.add(st2.nextToken());}
        dtsem2=new String[node2Sem.size()];
node2Sem.toArray(dtsem2);
for (int j = 0; j < table.length; j++) {
        //mengeset nilai 2
if (dtSem1.length==1) {
if (node1[i].hashCode()==table[j].hashCode())
                dg[i][j]="2";
}else if(dtSem1.length>1) {
for (int k = 0; k < dtSem1.length; k++) {
if (dtSem1[k].hashCode()==table[j].hashCode())
                  dg[i][j]="2";
}}
            //mengeset nilai 1
if (dtsem2.length==1) {
if (node2[i].hashCode()==table[j].hashCode()) {
                dg[i][j]="1";
              }
            }else if(dtsem2.length>1) {
for (int k = 0; k < dtsem2.length; k++) {
if (dtsem2[k].hashCode()==table[j].hashCode())
                  dg[i][j]="1";
            }}
            //mengeset nilai 0
if (dg[i][j]==null)
             dg[i][j]="0";
   }}}
```

Gambar 4.6 *Source Code* Pemberian Nilai Matrik DG

```
private void buatDm(){
for (int i = 0; i < dm.length; i++) {
for (int j = 0; j < dm[0].length; j++) {
        ArrayList indexSem=new ArrayList();
        String indexdm[];
        node1Sem=new ArrayList();
        StringTokenizer st=new StringTokenizer(node1[j],";");
while (st.hasMoreTokens()) {


node1Sem.add(st.nextToken());}
        dtSem1=new String[node1Sem.size()];
node1Sem.toArray(dtSem1);
                        for (int k = 0; k < table.length; k++) {
for (int l = 0; l <dtSem1.length; l++) {
if (table[k].hashCode()==dtSem1[l].hashCode())
indexSem.add(String.valueOf(k));
        }}
indexdm=new String[indexSem.size()];
indexSem.toArray(indexdm);

        String isiDm="1";
        Mulai:
for (int k = 0; k < indexdm.length; k++) {
if (dm[i][Integer.parseInt(indexdm[k])].toString()=="0") {
isiDm="-1";
break Mulai;
                        }}
dm[i][j]=isiDm;
}}}
```

Gambar 4.7 *Source Code* Pemberian Nilai Matrik DM

4.3.2 Ketergantungan Transitif Kunci Penentu

Ketergantungan transitif kunci penentu ini digunakan untuk memunculkan kunci-kunci penentu yang lain berdaarkan pada kunci penentu yang telah ada. Proses ini dilakukan atas dasar sifat transifitas yang dimiliki oleh ketergantungan fungsional.

```
private void buatKeyTransitif(){
for (int i = 0; i < dm.length; i++) {
for (int j = 0; j < dm[0].length; j++) {
if (dm[i][j]=="-1") {
for (int k = 0; k < dm[i].length; k++) {
if (dm[i][k]=="1"&&dm[k][j]=="1")
dm[i][j]="1";
 } } } } }
```

Gambar 4.8 *Source Code key transitif*

Pada gambar 4.9 dapat dilihat bahwa untuk mendapatkan ketergantungan antar kunci penentu yang lain dilakukan proses pencarian nilai -1 pada matrik DM setelah itu dilakukan pencarian pada seluruh baris dan kolom yang sama dengan matrik yang ditemukan nilai -1, jika ditemukan nilai 1 pada baris dan kolom matrik DM maka matrik tersebut diisi nilai 1.

4.3.3 *Dependency Closure*

Dependency closure digunakan untuk pencarian semua kumungkinan ketergantungan fungsional pada matrik DG berdasarkan matrik DM. Proses ini mampu menampilkan ketergantungan fungsional yang tidak lepas dari sifat-sifat pada *inference rule.*

```
private void buatClosure(){
   for (int i = 0; i < dm.length; i++)
{
    for (int j = 0; j < dm[0].length; j++) {
        if (i!=j&&dm[i][j]!="-1")
{
            for (int k = 0; k < dg[j].length; k++) {
              if (dg[j][k]!="0"&&dg[j][k].toString()!="2") {
                  if (dg[i][k]!="2")
                     dg[i][k]=node1[j];
 } } } } } }
```

Gambar 4.9 *Source Code Dependency Closure*

Pada gambar 4.9 dapat dilihat bahwa pertama kali yang dicari pada matrik DM adalah nilai -1 yang menunjukkan bahwa kunci-kunci penentu ini tidak memiliki ketergantungan fungsional kemudian dilanjutkan pencarian pada matrik DG pada kolom yang sama yang ditemukan kunci penentu tersebut. Pencarian ini dimulai pada kolom pertama sampai terakhir. Jika ditemukan nilai selain 0 dan 2 maka matrik DG diberi nialai baru pada index yang ditemukan.

4.3.4 *Circular Dependency*

Circular dependency merupakan proses yang dilakukan untuk memastikan ketergantungan fungsional yang memudar akibat dari proses *dependency closure* apabila ditemukan maka nilai pada matrik DG diganti 1. Proses ini juga mampu untuk memudarkan ketergantungan fungsional yang semestinya memudar.

```
private void buatCircular(){
    for ( int i=0; i<dg.length; i++){
        for(int j=0; j<dg[i].length; j++){
            if(dg[i][j]!="0" && dg[i][j]!="1" && dg[i][j]!="2"){
                if (FindOne(j, dg.length)==1 && dg2[i][j]=="1")
                    dg[i][j]="1";

    } } } }
```

Gambar 4.10 *Source Code Circular dependency*

4.3.5 MenguraikanKetergantungan Parsial

Menguraikan ketergantungan parsial merupakan proses yang dilakukan untuk menghilangkan seluruh ketergantungan parsial. Pertama kali yang dilakukan adalah mencari seluruh ketergantungan parsial pada matrik DG kemudian akan dilakukan proses pengurain ketergantungan parsial menjadi tabel-tabel baru.

```
private void pecahParsial(){
   hsl_nodeSem=node1;
dmSem=dm;
   tb_sem=table;

   ArrayList par = new ArrayList();
for (int i = 0; i < hsl_nodeSem.length; i++) {
if (cariParsial(i))
par.add(i);
   }

if (par.isEmpty()==false) {
int[] ps=new int[par.size()];
for (int i = 0; i < par.size(); i++) {
ps[i]=Integer.parseInt(par.get(i).toString());
       }

          for (int i = 0; i < ps.length; i++) {
pecahTabel(dmSem, hsl_nodeSem, tb_sem, ps[i]);
for (int j = i; j < ps.length; j++) {
ps[j]=ps[j]-1;
          }
       }
   }
}
```

Gambar 4.11 *Source Code* Pecah Parsial

4.3.6 Menguraikan Ketergantungan Transitif

Menguraikan ketergantungan transitif digunakan untuk menghilangkan ketergantungan-ketergantungan transitif yang ada pada matrik DG. Proses ini dilakukan melalui baris pertama sampai yang terakhir, jika ditemukan maka dilanjutkan proses penguraian menjadi tabel-tabel baru.

```
private void pecahTransitif(){
      ArrayList trans = new ArrayList();
for (int i = 0; i < hsl_nodeSem.length; i++) {
if (cariTransitif(i)) {
trans.add(i);
          }
       }

int[] ts=new int[trans.size()];
for (int i = 0; i < trans.size(); i++) {
ts[i]=Integer.parseInt(trans.get(i).toString());
       }

for (int i = 0; i < ts.length; i++) {
pecahTabel(dmSem, hsl_nodeSem, tb_sem, ts[i]);
for (int j = i; j < ts.length; j++) {
ts[j]=ts[j]-1;
          }
       }
   }
```

Gambar 4.12 *Source Code* Pecah Transitif

4.4 Pengujian dan Analisa

Tahap pengujian ini dilakukan untuk mengetahui kemampuan aplikasi dalam mengkontruksi ulang basis data hingga dalam bentuk 3NF. Proses pengujian ini dilakukan sesuai dengan batasan masalah yang ada pada bab ii. Proses pengujian dilakukan dengan pembandingan hasil aplikasi yang dibuat dengan hasil dari aplikasi Microsoft Access. Basis data beserta data yang diuji menggunakan *sample* dari basis data anggota Ikatan Mahasiswa Muhammadiyah UIN Malang.

Pertama kali dialakukan pengujian pada aplikasi mengkontruksi ulang basis data hingga berbentu 3NF. Sebelum melakukan proses normalisasi dilakukan pembuatan tabel beserta isinya pada sistem MySql. Kemudian menjalankan aplikasi dilanjutkan dengan memilih tabel anggota yang telah dibuat sebelumnya.

Sistem pada aplikasi ini secara otomatis mampu menampilkan ketergantungan fungsional tabel, setelah itu menjalankan perintah normalisasi pada *button* "Berikutnya" maka secara otomatis akan mendapatkan basis data baru.

Pada saat pengisian nilai pada setiap attribut diharuskan untuk lebih teliti dikarenakan sistem akan memberikan hasil yang kurang akurat jika terjadi kesalahan pengisian data. Misalkan dilakukan pengujian pada kolom *fakultas* dan *jurusan*, perhatikan gambar 4.13, mendapatkan hasil ketergantungan fungsional berupa *jurusan* → *fakultas* kemudian dilanjutkan dengan pengisian data pada kolom *fakultas* diisi dengan "sainte" dan *jurusan* diisi dengan "Informatika", perhatikan gambar 4.14. Kemudian dilakukakan pengujian pada basis data tersebut. Pengujian ini tidak mendapatkan hasil ketergantungan fungsional dikarenakan kesalahan pengisian pada kolom *fakultas* padahal semestinya setiap jurusan Informatika memiliki fakultas Saintek.

fakultas	jurusan
Tarbiyah	PGMI
Tarbiyah	PGMI
Tarbiyah	PAI
Tarbiyah	PAI
Saintek	Matematika
Saintek	Matematika
Saintek	Kimia
Saintek	Informatika
Saintek	Informatika
Syariah	Hukum Perdata Islam
Saintek	Fisika

Gambar 4.13 Tabel *fakultas* dan *jurusan*

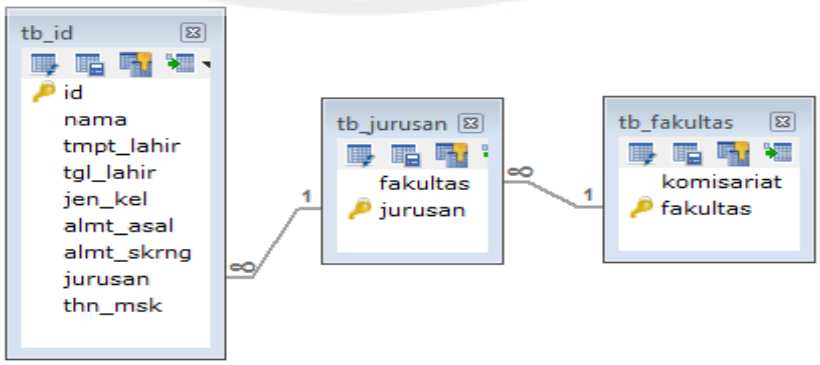
fakultas	jurusan
Tarbiyah	PGMI
Tarbiyah	PGMI
Tarbiyah	PAI
Tarbiyah	PAI
Saintek	Matematika
Saintek	Matematika
Saintek	Kimia
Saintek	Informatika
Saintek	Informatika
Syariah	Hukum Perdata Islam
Saintek	Fisika
Sainte	Informatika

Gambar 4.14 Penambahan Isi Tabel *fakultas* dan *jurusan*

id	nama	tmpt_lahir	tgl_lahir	jen_kel	almt_asal	almt_skrng	komisariat	fakultas	jurusan	thn_msk
1	Nur Zaidah	Lamongan	25121990	P	Jl. Ponpes karanga	-	Revivalis	Saintek	Informatika	2009
2	Izzat Al Hikam	Gresik	14101992	L	Jl. Pangeran dipor	-	Revivalis	Saintek	Informatika	2010
3	Arif Firmansyah E	Lamongan	27051991	L	Karang wungulor -	-	Revivalis	Saintek	Informatika	2010
4	Vivid Rohmaniyah	Lamongan	19101991	P	Karang Wungulor -	-	Revivalis	Saintek	Informatika	2010
5	Syifa'ul Amamah	Lamongan	12011992	P	Jl. Kebun Sari Sol	-	Revivalis	Saintek	Matematika	2010
6	Fahmi Muhammad	Gresik	15101991	L	Jl. Kyai Ageng Are	Sumber Sari	Revivalis	Saintek	Matematika	2010
7	Subur Promono	Ponorogo	26061990	L	Jl. Abdul Hadi 77	Sumber Sari	Revivalis	Saintek	Fisika	2009
8	David Saidi	Gorontalo	04021990	L	Falabisahaya - sar	Perum Perma	Revivalis	Saintek	Kimia	2009
9	Robbi Teo Aziz	Madiun	22051992	L	Jl. Mandu Rt 06b F	-	Pelopor	Tarbiyah	PGMI	2010
10	Askurul Hasin	Probolinggo	18061990	L	Jl. KH Hasan Gengg	-	Pelopor	Tarbiyah	PAI	2010
11	Muhammad Fazrin Dwi K	Gresik	11021992	L	Kedung Kakap Rt 04	-	Pelopor	Syariah	Hukum Perdata	2010
12	Eka Hasanah Wati	Lamongan	15071992	P	Payaman, Solokuro	-	Pelopor	Tarbiyah	PGMI	2010
13	Syahar Banu al-Abqori	Lamongan	25071992	P	Ds. Gedangan, Kec.	-	Pelopor	Tarbiyah	PAI	2010

Gambar 4.15 Tabel Anggota

Hasil pengujian dari aplikasi ini didapatkan ketergantungan fungsional (*id → nama, tmpt_lahir, tgl_lahir, jen_kel, almt_asal, almt_skrng, komisariat, fakultas, jurusan, thn_msk; Nama → id, tmpt_lahir, tgl_lahir, jen_kel, almt_asal, almt_skrng, komisariat, fakultas, jurusan, thn_msk; tgl_lahir → id, nama, tmpt_lahir, jen_kel, almt_asal, almt_skrng, komisariat, fakultas, jurusan, thn_msk; almt_asal → id; nama, tmpt_lahir, tgl_lahir, jen_kel, almt_skrng, komisariat, fakultas, jurusan, thn_msk; fakultas → komisariat; jurusan → komisariat, fakultas*) dan tabel baru berupa *tb_id : (id, nama, tmpt_lahir, tgl_lahir, jen_kel ,almt_asal, almt_skrng, jurusan, thn_msk), tb_fakultas : (komisariat, fakultas}*, dan *tb_jurusan : (fakultas, jurusan)*. Perhatikan visualisasi hasil percobaan dengan bantuan Sqlyog pada gambar 4.16.



Gambar 4.16 Hasil Pengujian Aplikasi

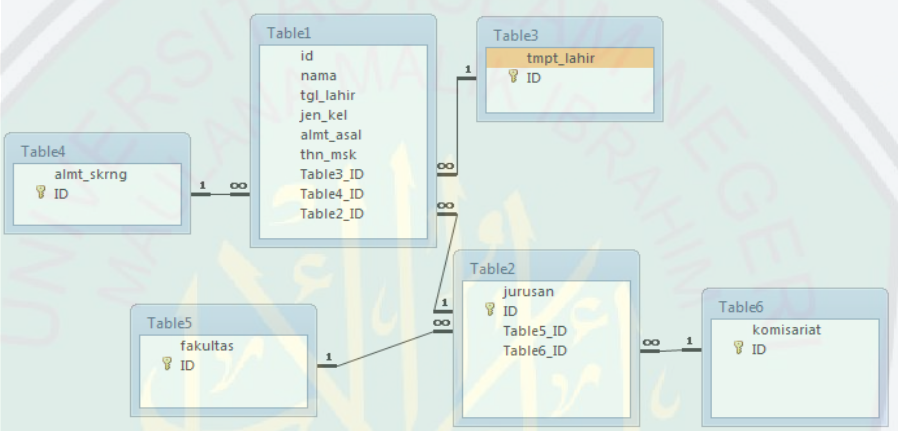
Dari hasil visualisasi pada gambar 4.16 dapat dilihat bahwa basis data yang baru mampu mengurangi anomali *update* dan *delete* pada attribut *fakultas* dan *jurusan* dengan memisahkan menjadi tabel-tabel baru. Sebagi contoh, mengubah nilai Saintek pada attribut *fakultas* pada basis data yang lama maka harus mengubah semua nilai tersebut pada attribut *fakultas*, jika tidak dilakukan maka basis data menjadi tidak konsisten sedangkan pada basis data yang baru tinggal mengubah nilai tersebut pada *tb_fakultas*. Penghapusan data pada basis data baru mampu mengurangi resiko kehilangan semua data. Sebagai contoh, menghapus salah satu identitas anggota tidak akan berpengaruh pada *tb_jurusan* dan *tb_fakultas* sehingga informasi jurusan dan fakultas tidak ikut terhapus.

Pengujian yang selanjutnya yaitu pada aplikasi Microsoft Access dengan cara membuat tabel dan memasukkan isinya sesuai pada gambar 4.15, perhatikan gambar 4.17 kemudian memilih menu *analyze table*. Perhatikan visualisasi hasil percobaan tersebut pada gambar 4.18.

Dari hasil visualisasi pada gambar 4.18 menjelaskan bahwa Microsoft Acces lebih mampu mengurangi anomali *update* dan *delete* pada attribut *jurusan*, *komisariat*, *fakultas*, dan *almt_skrng*. Kelebihan dari Microsoft Acces mampu menambahkan attribut ID pada setiap tabel memiliki kode unik sehingga bisa mengurangi kesalahan penulisan. Aplikasi ini masih belum mampu mengkontruksi ulang basis data lebih baik dari Microsoft Acces akan tetapi sudah memenuhi tujuan dari pembuatan aplikasi.

id	nama	tmpt_lahir	tgl_lahir	jen_kel	almt_asal	almt_skrng	komisariat	fakultas	jurusan	thn_msk
1	Nur Zaidah	Lamongan	25121990	P	Jl. Ponpes karan	-	Revivalis	Saintek	Informatika	2009
2	Izzat Al Hikam	Gresik	14101992	L	Jl. Pangeran dipo	-	Revivalis	Saintek	Informatika	2010
3	Arif Firmansyah E	Lamongan	27051991	L	Karang wungulo	-	Revivalis	Saintek	Informatika	2010
4	Vivid Rohmaniya	Lamongan	19101991	P	Karang Wungulc	-	Revivalis	Saintek	Informatika	2010
5	Syifa'ul Amamah	Lamongan	12011992	P	Jl. Kebun Sari So	-	Revivalis	Saintek	Matematika	2010
6	Fahmi Muhamma	Gresik	15101991	L	Jl. Kyai Ageng Ar	Sumber Sari III	Revivalis	Saintek	Matematika	2010
7	Subur Promono	Ponorogo	26061990	L	Jl. Abdul Hadi 77	Sumber Sari III	Revivalis	Saintek	Fisika	2009
8	David Saidi	Gorontalo	04021990	L	Falabisahaya - sa	Perum Permata	Revivalis	Saintek	Kimia	2009
9	Robbi Teo Aziz	Madiun	22051992	L	Jl. Mandu Rt 06b	-	Pelopor	Tarbiyah	PGMI	2010
10	Askurul Hasin	Probolinggo	18061990	L	Jl. KH Hasan Gen	-	Pelopor	Tarbiyah	PAI	2010
11	Muhammad Fazri	Gresik	11021992	L	Kedung Kakap R	-	Pelopor	Syariah	Hukum Perdat	2010
12	Eka Hasanah Wat	Lamongan	15071992	P	Payaman, Soloki	-	Pelopor	Tarbiyah	PGMI	2010
13	Syahar Banu al-A	Lamongan	25071992	P	Ds. Gedangan, K	-	Pelopor	Tarbiyah	PAI	2010

Gambar 4.17 Tabel Anggota Microsoft Access



Gambar 4.18 Hasil Pengujian Microsoft Access

4.5 Kajian Agama Terhadap Hasil Penilitian

Hasil dari penelitian yang telah dilakukan, memperlihatkan bahwa dengan adanya aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF yang nantinya menggantikan proses normalisasi secara manual memberikan kemudahan pada administrator dalam perancangan basis data, dalam Islam mengajarkan tentang memberikan solusi terhadap permasalahan dengan kemudahan dalam setiap urusan. Sebagaimana Allah SWT telah berfirman dalam Al-Qur'an surat Al-Baqarah ayat 185 yang berbunyi:

شَهْرُ رَمَضَانَ ٱلَّذِىٓ أُنزِلَ فِيهِ ٱلْقُرْءَانُ هُدًى لِّلنَّاسِ وَبَيِّنَٰتٍ مِّنَ ٱلْهُدَىٰ وَٱلْفُرْقَانِ
فَمَن شَهِدَ مِنكُمُ ٱلشَّهْرَ فَلْيَصُمْهُ ۖ وَمَن كَانَ مَرِيضًا أَوْ عَلَىٰ سَفَرٍ فَعِدَّةٌ مِّنْ أَيَّامٍ ۚ

أُخَرَ ۗ يُرِيدُ ٱللَّهُ بِكُمُ ٱلْيُسْرَ وَلَا يُرِيدُ بِكُمُ ٱلْعُسْرَ وَلِتُكْمِلُوا۟ ٱلْعِدَّةَ وَلِتُكَبِّرُوا۟ ٱللَّهَ عَلَىٰ مَا هَدَىٰكُمْ وَلَعَلَّكُمْ تَشْكُرُونَ ﴿١٨٥﴾

Artinya : *"(Beberapa hari yang ditentukan itu ialah) bulan Ramadhan, bulan yang di dalamnya diturunkan (permulaan) Al Quran sebagai petunjuk bagi manusia dan penjelasan-penjelasan mengenai petunjuk itu dan pembeda (antara yang hak dan yang bathil). karena itu, Barangsiapa di antara kamu hadir (di negeri tempat tinggalnya) di bulan itu, Maka hendaklah ia berpuasa pada bulan itu, dan Barangsiapa sakit atau dalam perjalanan (lalu ia berbuka), Maka (wajiblah baginya berpuasa), sebanyak hari yang ditinggalkannya itu, pada hari-hari yang lain. Allah menghendaki kemudahan bagimu, dan tidak menghendaki kesukaran bagimu. dan hendaklah kamu mencukupkan bilangannya dan hendaklah kamu mengagungkan Allah atas petunjuk-Nya yang diberikan kepadamu, supaya kamu bersyukur.*" (QS.Al-Baqarah :185).

Pada dasarnya agama Islam itu ringan dan mudah, baik dalam aqidah, akhlak, amal-amal ibadah, perintah dan larangannya. Sehingga pada surat Al-Baqarah ayat 185 memberikan gambaran bahwa sesungguhnya memudahkan tiap-tiap urusan dan menghindari hal-hal yang menyulitkan merupakan kehendak dari Allah SWT. Kemudahan dalam hal ini mendapatkan manfaat yang luar biasa dalam proses perancangan basis data yaitu dapat menghemat waktu, tenaga, dan biaya. Selain itu, kemudahan ini juga bisa bermanfaat untuk mengurangi kesalahan-kesalahan yang terjadi jika dilakukan secara manual sehingga mampu melakukan proses ini dengan efektif dan efisien.

Selain kemudahan, aplikasi ini dibuat untuk untuk menghilangkan redundansi dan anomali yang terjadi pada relasi basis data yang mengakibatkan penggunaan penyimpanan basis data menjadi berlebihan dan tidak konsisten. Penyimpanan basis data tersebut merupakan pemborosan dan mengakibatkan banyak data menjadi sia-sia. Allah SWT telah berfirman dalam Al-Qur'an surat

Al-Kahfi ayat 103-104 yang berbunyi:

قُلۡ هَلۡ نُنَبِّئُكُم بِٱلۡأَخۡسَرِينَ أَعۡمَٰلاً ١٠٣ ٱلَّذِينَ ضَلَّ سَعۡيُهُمۡ فِي ٱلۡحَيَوٰةِ ٱلدُّنۡيَا وَهُمۡ يَحۡسَبُونَ أَنَّهُمۡ يُحۡسِنُونَ صُنۡعًا ١٠٤

Artinya : *"Katakanlah: "Apakah akan Kami beritahukan kepadamu tentang orang-orang yang paling merugi perbuatannya?". Yaitu orang-orang yang telah sia-sia perbuatannya dalam kehidupan dunia ini, sedangkan mereka menyangka bahwa mereka berbuat sebaik-baiknya.*" (QS. Al-Kahfi : 103-104)

Pada surat Al-Kahfi ayat 103-104 menyebutkan dengan jelas bahwa perbuatan yang sia-sia merupakan perbuatan yang sangat merugikan dan benar-benar harus dihindari sehingga mendapatkan relasi basis data yang minim akan anomali dan redundansi dan mengurangi kerugian yang akan didapatkan.

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil pembuatan aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF ini mampu menghasilkan basis data baru yang sudah berbentuk 3NF dalam bentuk *query* sehingga pennggguna tinggal menjalankannya pada sistem MySql. Dengan adanya aplikasi ini diharapkan dapat membantu administrator basis data dikarenakan aplikasi ini mampu memproses dengan cepat dan akan mengurangi kesalahan manusia pada perancangan basis data secara manual.

Keakuratan hasil dari aplikasi ini bergantung pada variasi data yang ada pada basis data, jika pengguna melakukan kesalahan pada saat memasukkannya maka tentunya mendapatkan hasil yang kurang maksimal. Maka dari itu pengguna diharuskan untuk lebih teliti dalam memasukkan data sehingga aplikasi ini mendapatkan hasil yang maksimal.

5.2 Saran

Dalam aplikasi mengkontruksi ulang basis data hingga berbentuk 3NF ini tentunya masih membutuhkan penyempurnaan. Untuk lebih dapat menyempurnakan aplikasi, maka disarankan untuk meningkatkan kemampuan aplikasi sehingga mampu menormalisasi basis data ke dalam bentuk yang lebih tinggi lagi yaitu: BCNF, bentuk normal keempat (4NF), dan bentuk normal kelima (5NF).

DAFTAR PUSTAKA

Connolly, Thomas dan Begg, Carolyn. 2002. *Database Systems: A Practical Approach to Design, Implementation, and Management*, edition 3. Addison Wesley, Massachusetts.

Date, C.J. 2000. *Pengenalan Sistem Basis Data Edisi Ketujuh Jilid 1*. Jakarta: Indeks.

Al Qurthubi. 2008. Tafsir Al Qurthubi. Jakarta: Pustaka Azzam.

Silberschatz, A., Korth, H.F., Sudarshan, S. (2002). *Database System Concepts*, edition 4. McGraw-Hill., Singapore.

Mata-Toledo, *Ramon A. dan Pauline K. Cushman. 2007. Schaum's Outlines Dasar-Dasar Database Relasional*. Jakarta: Erlangga.

Bahmani A, Naghibzadeh M and Bahmani B (2008). *Automatic database normalization and primary key generation*. Niagara Falls Canada IEEE.

Dongare, Y, P.S. Dhabe, S.V. *Deshmukh. RDBNorma: A Semi automated Tool For Relational Database Schema Normalization Up To Third Normal Form*. International Journal of Database Management Systems (IJDMS), Vol. 3, No. 1, Februari 2011.

Ramakrishnan, Raghu dan Johannes Gehrke. 2004. *Sistem Manajemen Database Edisi 3*. Yogyakarta: Andi.