



MODUL PERKULIAHAN

Perancangan Basis Data

Pengenalan perancangan
basis data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
01

Kode MK
18033

Disusun Oleh
Tim Dosen

Kompetensi : Mahasiswa dapat mengetahui keuntungan dari penggunaan CASE TOOLS
Mahasiswa dapat membedakan peran dari data administrator dengan database administrator

Pembahasan

Perancangan Basis data

(Database Planning)

1. *Basis Data*

Istilah data bermakna untuk mengetahui fakta-fakta yang dapat direkam dan disimpan pada media komputer. Definisi ini kini berkembang untuk mencerminkan realitas baru. Basis data sekarang digunakan untuk menyimpan objek seperti dokumen, foto, suara, dan video, sebagai tambahan dari data teks dan data numerik. Untuk mencerminkan realitas, kita menggunakan definisi yang diperluas berikut: Data terdiri dari fakta-fakta, hasil-hasil pengujian, grafik, gambar, dan video yang mempunyai arti dalam lingkungan pengguna (Hoffer,2002,p4).

Kita telah mendefinisikan basis data sebagai kumpulan data yang terorganisasi dan saling berhubungan. Terorganisasi maksudnya adalah data yang terstruktur sehingga mudah disimpan, dimanipulasi, dan diambil kembali oleh pengguna.. Saling berhubungan maksudnya adalah data menggambarkan suatu *domain* yang menjadi perhatian sekelompok pengguna dan pengguna-pengguna dapat menggunakan data untuk menjawab pertanyaan yang menjadi perhatian dari *domain* tersebut (Hoffer,2002,p5).

2. *Data dan Informasi*

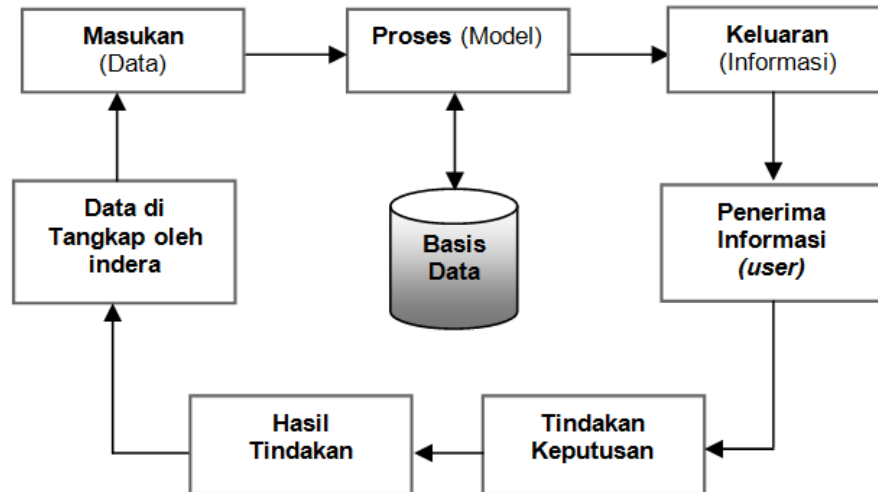
Menurut Turban, Aronson, and Liang (2005) data dan informasi didefinisikan sebagai berikut:

- Data, merupakan sesuatu yang menyangkut barang, kejadian, aktivitas, dan transaksi yang telah tercatat, diklasifikasikan, dan disimpan namun belum memiliki makna. Data dapat berupa nilai numerik, alphanumerik, gambar, dan suara.
- Informasi, adalah data yang telah dikelola dalam bentuk tertentu untuk memberikan makna atau arti bagi penerimanya.

3. *Siklus Informasi*

- Data dan informasi akan saling berkesinambungan sehingga membentuk suatu siklus yang disebut information cycle (siklus informasi).

- Data ditangkap oleh indera kemudian menjadi inputan dalam sebuah model untuk diubah menjadi informasi bagi penerimanya yang nantinya akan membantu pengambilan keputusan dan menjadi sebuah hasil tindakan.



Gambar 0-1 Siklus informasi

4. Pentingnya data dan informasi

- Data dan informasi sebagai sebuah aset penting perusahaan/organisasi.
- Informasi yang benar dapat menjadikan suatu perusahaan/organisasi memperoleh *margin* untuk melakukan aksi.
- Data dan informasi sebagai salah satu parameter kemajuan perusahaan/organisasi (*maturity level*).

5. Sistem Basis Data dan Sistem File

Pada sebuah institusi, data merupakan salah satu hal yang sangat penting. Setiap bagian/divisi dari institusi memiliki data sendiri-sendiri. Tapi setiap bagian pun membutuhkan sebagian data dari bagian yang lain. Hal ini yang biasa dikenal sebagai “*shared data*”. Setiap divisi memiliki aplikasi sendiri-sendiri dalam melakukan manipulasi dan pengambilan data tersebut. Setiap aplikasi memiliki *file-file* dalam sistem operasi yang digunakan untuk menyimpan data-data. Seiring dengan berkembangnya institusi, bertambahnya bagian/divisi, bertambah pula data dan aplikasi yang digunakan. Bertambahnya aplikasi, bertambah pula *file-file* yang dibuat.

Gaya sistem pemrosesan-*file* tersebut menyebabkan setiap data disimpan dalam bentuk *record* dalam berbagai macam *file*, dan diperlukan aplikasi yang berbeda dalam melakukan

pengambilan *record* dari, dan penambahan *record* ke dalam *file*. Hal ini berlaku pada masa sebelum adanya Sistem Basis Data (DBMS).

Menyimpan data dalam bentuk *file* yang berbeda-beda, memiliki kekurangan-kekurangan:

- **Data *redundancy* dan *inconsistency*.**

Dikarenakan programmer yang berbeda membuat *file* dan aplikasi masing-masing, menyebabkan beragam format dan aplikasi yang dibuat. Bahkan, aplikasi pun dibuat menggunakan bahasa pemrograman yang berbeda-beda. Lebih jauh lagi, data atau informasi yang sama bisa terdapat dalam beberapa *file* yang berbeda. Ini yang disebut dengan *redundancy*. *Redundancy* data ini lama kelamaan akan menyebabkan *inconsistency* dari data.

- **Kesulitan dalam pengaksesan data.**

Dikarenakan setiap aplikasi memiliki *file* tersendiri untuk penyimpanan dan pengambilan data, maka jika suatu bagian dari institusi membutuhkan data dari bagian lain, akan menemui kesulitan. Hal ini dikarenakan aplikasi yang dimiliki bagian tersebut, tidak dapat membaca *file* yang terdapat di bagian lain.

- **Isolasi data.**

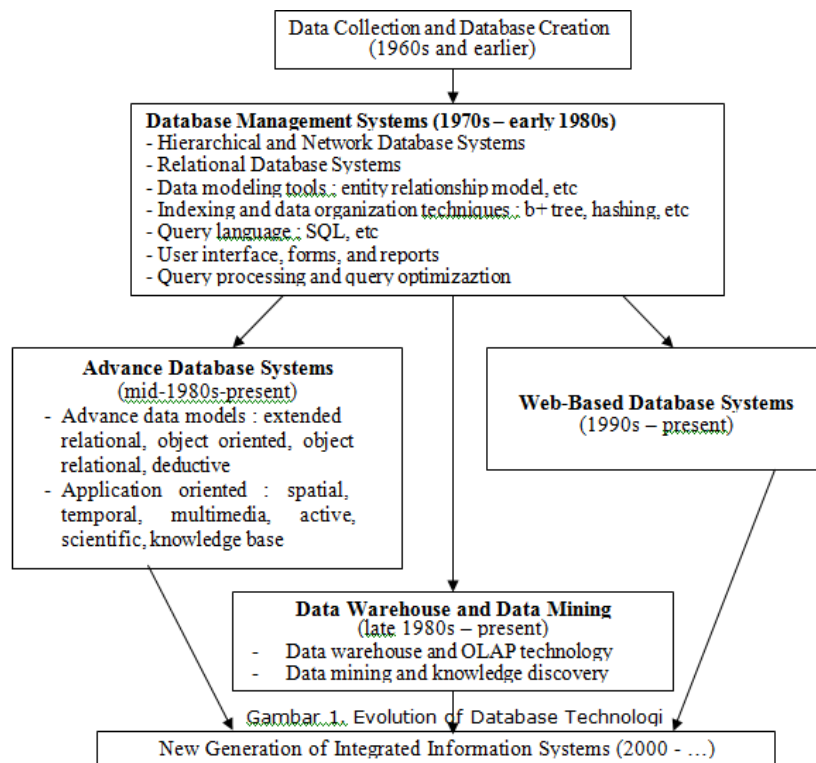
Dikarenakan data tersebar dalam berbagai macam *file*, dan *file* tersebut dalam beragam format, pembuatan aplikasi baru akan terasa sulit ketika harus membaca format dari masing-masing *file* tersebut.

- **Masalah integritas.**

Data yang disimpan harus memenuhi hal yang dinamakan dengan *consistency constraint*. Jika sebuah *constraint* berubah, maka seluruh aplikasi yang digunakan harus mengakomodasinya. Masalah akan muncul, jika *constraint* melibatkan beberapa data dari *file* yang berbeda-beda.

- **Masalah keamanan.**

Tidak semua pengguna dari basis data dapat mengakses semua data. Hal ini akan sulit dilakukan jika menggunakan gaya penyimpanan data dalam *file*.



Gambar 0-2 Perkembangan Database

6. Definisi Basis Data dan Sistem Basis Data (DBMS)

Basis data adalah penyimpanan kumpulan informasi secara sistematis dalam sebuah komputer sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat Lunak yang digunakan untuk mengelola dan memanggil *query* basis data disebut Sistem Manajemen Basis Data (*Database Management System*, DBMS). DBMS memiliki karakteristik sebagai berikut:

- *Software program*
- *Supplements operating sistem*
- *Manages data*
- *Queries data and generates reports*
- *Data security*

Sedangkan sistem adalah sebuah tatanan yang terdiri atas sejumlah komponen fungsional yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses pekerjaan. Sehingga bisa dikatakan bahwa sistem basis data adalah sistem yang terdiri atas kumpulan *file-file* yang saling berhubungan dan dikelola oleh program (DBMS) yang memungkinkan beberapa pemakai dan atau

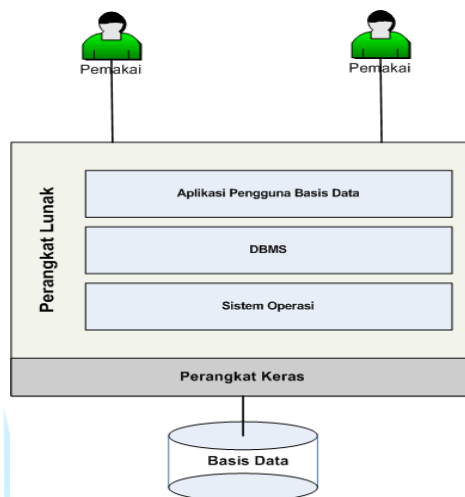
program lain yang memiliki otoritas untuk mengakses dan memanipulasi data tersebut. Kelebihan pemakaian DBMS adalah:

- Data berdiri sendiri (*Data Independence*)
- Pengaksesan data efisien (*Efficient data access*)
- Integritas data dan keamanan terjamin (*Data integrity and security*)
- Administrasi data (*Data administration*)
- Dapat diakses bersamaan (*Concurrent access*)
- *Recovery* saat terjadi kegagalan (*Crash recovery*)
- Mengurangi waktu pembangunan aplikasi (*Reduced application development time*)

7. *Komponen Sistem Basis Data*

Komponen-komponen pada sebuah sistem basis data antara lain:

- Perangkat keras
- Sistem operasi
- Basis data
- DBMS (*Database Management System*)
- Pemakai
- Aplikasi lain



Gambar 0-3 Komponen DBMS

8. Abstraksi Data

Tujuan utama dari sistem basis data adalah untuk menyediakan fasilitas untuk *view* data secara abstrak bagi penggunaanya. Namun bagaimana sistem menyimpan dan mengelola data tersebut, hanya diketahui oleh sistem itu sendiri. Abstraksi data merupakan level dalam bagaimana melihat data dalam sebuah sistem basis data. Berikut ini tiga level abstraksi data:

1. Level fisik

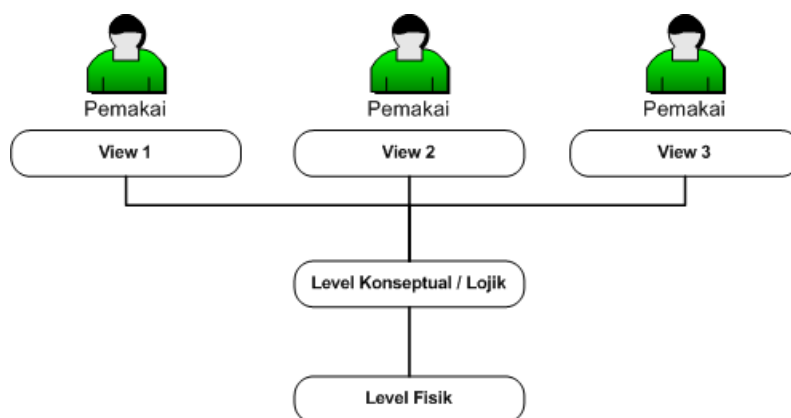
Merupakan level terendah pada abstraksi data yang menunjukkan bagaimana sesungguhnya data disimpan. Pada level ini pemakai melihat data sebagai gabungan dari struktur dan datanya sendiri.

2. Level logik

Merupakan level berikutnya pada abstraksi data, menggambarkan data apa yang disimpan pada basis data dan hubungan apa saja yang ada di antara data tersebut.

3. Level view

Merupakan level tertinggi dari abstraksi data yang hanya menunjukkan sebagian dari basis data. Banyak *user* dalam sistem basis data tidak akan terlibat dengan semua data atau informasi yang ada atau yang disimpan. Para *user* umumnya hanya membutuhkan sebagian data atau informasi dalam basis data yang kemunculannya di mata *user* diatur oleh aplikasi *end user*.



Gambar 0-4 Abstraksi Data.

9. Model Basis Data

- *Hierarchical*

Memiliki struktur pohon dimana *field* hanya memiliki satu buah induk (*parent*), masing-masing *parent* memiliki banyak *child* (anak). Model ini memiliki kecepatan yang baik.

- *Network*

Relationship dibuat menggunakan linked list (*pointer*). Berbeda dengan model *hierarchical* satu anak dapat memiliki beberapa induk. Model ini memiliki fleksibilitas yang tinggi.

- *Relational*

Model ini direpresentasikan dalam tabel dua dimensi, tabel-tabel tersebut memiliki hubungan yang disebut dengan relasi. Model ini memiliki fleksibilitas dan kecepatan yang tinggi.

- *Object oriented*

Object Oriented Database adalah sebuah sistem *database* yang menggabungkan semua konsep *object oriented* seperti pewarisan, abstraksi, enkapsulasi, dll. Model ini dapat berinteraksi dengan baik dengan bahasa pemrograman berorientasi objek seperti java dan C++.

10. Perancangan Database

Di dalam suatu organisasi yang besar, sistem database merupakan bagian penting pada sistem informasi, karena di perlukan untuk mengelola sumber informasi pada organisasi tersebut. Untuk mengelola sumber informasi tersebut yang pertama kali di lakukan adalah merancang suatu sistem database agar informasi yang ada pada organisasi tersebut dapat digunakan secara maksimal.

Tujuan Perancangan Database

- Untuk memenuhi kebutuhan akan informasi dari pengguna dan aplikasi
- Menyediakan struktur informasi yang natural dan mudah di mengerti oleh pengguna
- Mendukung kebutuhan pemrosesan dan beberapa obyek kinerja dari suatu sistem database

Berikut ini siklus kehidupan sistem informasi di mana terdapat siklus kehidupan sistem database.

a. Siklus Kehidupan Sistem Informasi (Macro Life Cycle)

Tahapan–tahapan yang ada pada siklus kehidupan sistem informasi yaitu :

- 1) Analisa Kelayakan
Tahapan ini memfokuskan pada penganalisaan areal aplikasi yang unggul , mengidentifikasi pengumpulan informasi dan penyebarannya, mempelajari keuntungan dan kerugian , penentuan kompleksitas data dan proses, dan menentukan prioritas aplikasi yang akan digunakan.
- 2) Analisa dan Pengumpulan Kebutuhan Pengguna
Kebutuhan–kebutuhan yang detail dikumpulkan dengan berinteraksi pada sekelompok pemakai atau pemakai individu. Mengidentifikasi masalah yang ada dan kebutuhan-butuhan, ketergantungan antar aplikasi, komunikasi dan prosedur laporan.
- 3) Perancangan
Perancangan terbagi menjadi dua yaitu : perancangan sistem database dan sistem aplikasi
- 4) Implementasi
Mengimplementasikan sistem informasi dengan database yang ada
- 5) Pengujian dan Validasi
Pengujian dan validasi sistem database dengan kriteria kinerja yang diinginkan oleh pengguna.
- 6) Pengoperasian dan Perawatan
Pengoperasian sistem setelah di validasi disertai dengan pengawasan dan perawatan sistem

b. Siklus Kehidupan Aplikasi Database (Micro Life Cycle)

Tahapan yang ada pada siklus kehidupan aplikasi database yaitu :

1. Pendefinisian Sistem

Pendefinisian ruang lingkup dari sistem database, pengguna dan aplikasinya.

2. Perancangan Database

Perancangan database secara logika dan fisik pada suatu sistem database sesuai dengan sistem manajemen database yang diinginkan.

3. Implementasi Database

Pendefinisian database secara konseptual, eksternal dan internal, pembuatan file–file database yang kosong serta implementasi aplikasi software.

4. Pengambilan dan Konversi Data

Database ditempatkan dengan baik, sehingga jika ingin memanggil data secara langsung ataupun merubah file–file yang ada dapat di tempatkan kembali sesuai dengan format sistem databasenya.

5. Konversi Aplikasi

Software-software aplikasi dari sistem database sebelumnya di konversikan ke dalam sistem database yang baru

6. Pengujian dan Validasi

Sistem yang baru telah di test dan di uji kinerja nya

7. Pengoperasian

Pengoperasian database sistem dan aplikasinya

8. Pengawasan dan Pemeliharaan

Pengawasan dan pemeliharaan sistem database dan aplikasi software

11. CASE TOOL

Secara umum seorang software engineer maupun engineer dari disiplin ilmu yang lain dalam membangun/mengembangkan suatu produk, memiliki karakteristik sebagai berikut:

1. Mengetahui manfaat tools yang dapat membantu dalam membangun / mengembangkan suatu produk.
2. Mampu mengorganisasikan tools yang memungkinkan untuk bekerja cepat dan efisien.
3. Memiliki pengetahuan teknik membangun/mengembangkan produk serta handal dalam menggunakan tools untuk membantu pekerjaannya.

Dalam software engineering telah dikenal banyak tools (computer-base system) yang dikenal dengan Computer-Aided Software Engineering (CASE). CASE merupakan suatu teknik yang digunakan untuk membantu satu atau beberapa fase dalam life-cycle software, termasuk

fase analisis, desain, implementasi dan maintenance dari software tersebut. Manfaat CASE tools untuk software engineer dijabarkan sebagai berikut:

- 1.CASE tools memperbesar kemungkinan otomatisasi pada setiap fase life-cycle software.
- 2.CASE tools sangat membantu dalam meningkatkan kualitas design model suatu software sebelum software itu dibangun/dikembangkan, baik itu untuk software yang dibangun dalam simple maupun complex environment.

Ada banyak tools yang mendukung pembangunan/pengembangan suatu software.

Agar tidak membingungkan, CASE tools dibagi menjadi beberapa kategori:

- 1.Information engineering-supporting products. Ada beberapa proses dari life-cycle, yang dihasilkan dari rencana strategis dari perusahaan dan yang menyediakan suatu repository untuk membuat dan memelihara enterprise models, data models dan process models.
- 2.Structured diagramming-supporting products. Produk ini sangat mendukung dalam memodelkan data flow, control flow dan entity flow.
- 3.Structured development aids-providing products. Merupakan produk yang cocok digunakan oleh sistem analis, karena didukung oleh suatu proses terstruktur sehingga penganalisaan lebih cepat dan akurat.
- 4.Application-code-generating products. Produk ini mampu menghasilkan application-code untuk tujuan tertentu yang telah ditetapkan oleh designer.

CASE tools diklasifikasikan sebagai berikut:

- 1.Upper CASE. CASE tools yang didesain untuk mendukung perencanaan, identifikasi, dan seleksi proyek (permulaan dari perencanaan proyek), tepatnya pada fase analisis dan desain dari suatu system development life cycle (SDLC).Tools yang termasuk kelas ini adalah jenis Diagramming tools, Form and report generators, dan Analysis tools. Contoh CASE tools: Cradle, PRO-IV Workbench, ProK it*WO RKBENCH.
- 2.Lower CASE. CASE tools yang didesain untuk mendukung tahap implementasi dan maintenance dari SDLC. Tools yang termasuk kelas ini adalah jenis Code

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Tahapan Perancangan Basis Data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
02

Kode MK
18033

Disusun Oleh
Tim Dosen

Kompetensi

Mahasiswa dapat menentukan sumber data apa yang dikumpulkan dari setiap tahap pengembangan basisdata dan mengidentifikasi dokumentasi yang dihasilkan dari setiap tahap pengumpulan fakta

Pembahasan

Tahapan Perancangan Basis Data

- ▶ Database Planning
- ▶ System Definition
- ▶ Requirement Collection and Analysis
- ▶ Database Design
- ▶ DBMS SELECTION
- ▶ APPLICATION DESIGN
- ▶ Prototyping
- ▶ Implementation
- ▶ Data conversion and loading
- ▶ Testing
- ▶ Operational maintenance

Perencanaan database mencakup :

- ▶ Cara pengumpulan data
- ▶ Format data
- ▶ Dokumentasi yang di perlukan
- ▶ Cara membuat desain
- ▶ Implementasi

Strategi Sistem Informasi

1. Identifikasi rencana dan sasaran dari organisasi, termasuk mengenai system informasi yang dibutuhkan.
2. Evaluasi system informasi yang ada untuk menetapkan kelebihan dan kekurangan yang di miliki oleh system tersebut
3. Penaksiran kesempatan teknik informatika yang mungkin memberikan keuntungan kompetitif.

Definisi Sistem

- ▶ Untuk mendeskripsikan batasan dan ruang lingkup aplikasi database serta sudut pandang user yang utama
- ▶ Mengidentifikasi user view membantu untuk memastikan agar tidak ada pengguna database yang terlupakan
- ▶ User view dapat mengembangkan aplikasi database yang rumit
- ▶ User view juga dapat menguraikan aplikasi menjadi sub-sub bagian yang lebih sederhana

CONTOH PENGGUNAAN SYSTEM DEFINITION



Macam-macam pendekatan yang digunakan :

1. Centralized approach → kebutuhan untuk tiap pengguna di buat ke dalam satu set of requirement dan model data global yang nantinya diperlukan dalam pembuatan database.
2. View integration approach → kebutuhan untuk tiap user view di buat dalam model data yang terpisah. Model data yang menggambarkan single user view disebut model data local, disusun dalam bentuk diagram dan dokumentasi yang mendeskripsikan kebutuhan user view database.
3. Gabungan antara kedua pendekatan tersebut.

Requirement collection and analysis :

- Merupakan proses mengumpulkan dan menganalisis informasi tentang organisasi yang akan didukung oleh aplikasi database
- Informasi tersebut di analisis untuk mengidentifikasi kebutuhan user terhadap aplikasi database yang baru.

Database design adalah proses membuat desain yang akan mendukung operasional dan tujuan perusahaan. Tujuan database design adalah :

1. menggambarkan relasi data antara data yang dibutuhkan oleh aplikasi dan user view
2. menyediakan model data yang mendukung seluruh transaksi yang diperlukan
3. menspesifikasikan desain dengan struktur yang sesuai dengan kebutuhan system

Pengertian Data Modelling :

- Untuk memahami arti atau semantic dari data
- Untuk memudahkan komunikasi mengenai informasi yang dibutuhkan
- Membuat model data membutuhkan jawaban dan pertanyaan tentang entities, relationship dan attributes

Kegiatan memilih dbms yang akan digunakan dalam pembuatan database

- Berikut langkah-langkah dalam pemilihan dbms :
 1. definisikan waktu untuk melakukan studi referensi
 2. catat dua atau tiga produk yang akan dievaluasi untuk digunakan
 3. evaluasi produk tersebut
 4. rekomendasikan produk yang di pilih dan buat laporan yang mendukungnya

Application Design merupakan perancangan user interface dan program aplikasi yang menggunakan dan melakukan proses terhadap database. Ada 2 aktivitas penting yang ada didalamnya yaitu :

1. Transaction design → bertujuan untuk mendefinisikan dan mendokumentasikan karakteristik transaksi berlevel tinggi yang dibutuhkan dalam database
2. User interface design

Prototyping fungsinya membuat model kerja suatu aplikasi database. Tujuan utamanya yaitu :

1. Mengidentifikasi fitur system yang sedang berjalan
2. Memberikan perbaikan atau penambahan fitur baru
3. Mengklarifikasi kebutuhan user

Mengevaluasi kelayakan dan kemungkinan apa yang terjadi pada design system.

Implementation merupakan realisasi fisik dari database dan aplikasi design. Implementation database dicapai dengan menggunakan :

- DDLI untuk membuat skema database dan database files yang kosong
- DDL untuk membuat user view yang diinginkan

Data conversion dan loading yaitu pemindahan data yang ada ke dalam database yang baru dan menkonversikan aplikasi yang ada agar dapat menggunakan database yang baru.

Testing ialah suatu proses eksekusi program aplikasi dengan tujuan untuk menemukan kesalahan dengan scenario test yang di rencanakan dan data yang sesungguhnya. Pengujian hanya akan terlihat jika terjadi kesalahan pada software.

Operational maintenance ialah suatu proses pengawasan dan pemeliharaan system setelah instalasi, mencakup :

1. Pengawasan kinerja system, jika kinerja menurun maka memerlukan perbaikan atau pengaturan ulang database
2. Pemeliharaan dan pembaruan aplikasi database

Penggabungan kebutuhan baru ke dalam aplikasi database

Daftar Pustaka

David M. Kroenke, “Dasar-Dasar, Desain dan Implementasi : Database processing”, Jilid 2, Penerbit Erlangga.



MODUL PERKULIAHAN

Perancangan Basis Data

Enhanced Entity Relationship

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
03

Kode MK
18033

Disusun Oleh
Tim Dosen

Kompetensi : Mahasiswa dapat memodelkan data dengan menggunakan konsep EER

Pembahasan

Enhanced Entity Relationship

Merupakan Model Entity Relationship yang didukung dengan konsep semantik tambahan. Dimana konsep semantik terdiri dari :

1. Subclass
2. Superclass

dan konsep-konsep yang berhubungan yaitu

1. Specialization (Hierarchy & Lattice)
2. Generalization

Konsep lainnya yang termasuk dalam model EER yaitu

1. Attribute Inheritance
2. Shared Subclass
3. Category

Entity yaitu obyek yang dapat dibedakan dalam dunia nyata. Entity set adalah kumpulan dari entity yang sejenis

Entity set dapat berupa :

- Obyek secara fisik : Rumah, Kendaraan, Peralatan
- Obyek secara konsep : Pekerjaan , Perusahaan, Rencana.

Relationship Yaitu hubungan yang terjadi antara satu atau lebih entity.

Relationship set adalah kumpulan relationship yang sejenis.

Atribut yaitu karakteristik dari entity atau relationship, yang menyediakan penjelasan detail tentang entity atau relationship tersebut.

Jenis-jenis Atribut :

- Key
- Atribut Simple
- Atribut Multivalued
- Atribut Composite

- Atribut Derivatif

Cardinality yaitu menjelaskan batasan jumlah keterhubungan satu entity dengan entity lainnya.

Subclass merepresentasikan entity yang sama dengan superclass, namun memiliki peran spesifik tertentu. Entity dalam subclass merupakan anggota superclass, namun tidak sebaliknya.

Superclass adalah entitas yang merupakan induk dari subclass-subclassnya.

Superclass/Subclass Relationship adalah relationship antara sebuah superclass dengan salah satu subclassnya.

Specialization yaitu proses mendefinisikan himpunan subclass-subclass dari sebuah entity type (Superclass). Dilakukan berdasarkan karakteristik tertentu yang dapat membedakan entity pada Superclass. Suatu Superclass dapat memiliki beberapa spesialisasi berdasarkan karakteristik yang berbeda.

Generalization yaitu proses pendefinisian subclass-subclass yang disatukan menjadi entitas superclass tunggal berdasarkan karakteristik umum. Contoh : Subclass **Mobil** dan **Truk** dapat digeneralisasikan menjadi Superclass **KENDARAAN** berdasarkan atribut umum seperti Kd_Kend, Harga, No_Lisensi.

Disjoint Constraint yang menerangkan bahwa subclass-subclass dari spesialisasi saling disjoint, artinya entity merupakan anggota dari salah satu subclass. Disjoint Constraint direpresentasikan dengan lambang “d” yang berarti *disjoint*. Contoh : entity dari spesialisasi tipe pekerjaan dari PEGAWAI merupakan anggota dari subclass PEGAWAI TETAP atau PEGAWAI HONORER.

Non Disjoint Constraint yang menerangkan bahwa subclass-subclass dari spesialisasi tidak saling disjoint, artinya entity mungkin anggota lebih dari satu subclass. Non-Disjoint digambarkan dengan lambang “o” yang berarti overlapping. Contoh : entity dari spesialisasi tipe barang merupakan anggota dari subclass BARANG PABRIK juga anggota dari subclass BARANG TERJUAL.

Total Specialization Constraint

- Constraint yang menerangkan bahwa setiap entity didalam superclass harus merupakan anggota dari salah satu subclass.
- Contoh : entity PEGAWAI harus termasuk subclass dari PEGAWAI TETAP atau PEGAWAI HONORER

Partial Specialization Constraint

- Constraint yang menerangkan bahwa setiap entity didalam superclass dapat merupakan anggota dari subclass-subclass yang didefinisikan. Contoh dari PEGAWAI dapat merupakan anggota dari subclass MANAGER, TEKNISI atau SALES.

Specialization di bagi menjadi 2 jenis yaitu :

Specialization Hierarchy

Spesialisasi bertingkat dimana setiap subclass berpartisipasi didalam satu kelas / subclass relationship.

Specialization Lattice

Spesialisasi bertingkat dimana suatu subclass dapat berpartisipasi didalam beberapa kelas / subclass relationship.

► Shared-Subclass

- Subclass yang mempunyai lebih dari satu superclass.

Contoh : subclass ASISTEN PELATIH mempunyai dua superclass yang tipenya sama yaitu SALES & PELATIH.

Category ialah proses pendefinisian suatu subclass (disebut kategori) yang memiliki lebih dari satu superclass yang berbeda. Contoh : Kategori **PEMILIK** yang merupakan Subclass dari gabungan **Orang**, **Bank** dan **Perusahaan**. Kategori **KENDARAAN-TERDAFTAR** yang merupakan Subclass dari gabungan **Mobil** dan **Truk**.

Daftar Pustaka

David M. Kroenke, “Dasar-Dasar, Desain dan Implementasi : Database processing”, Jilid 2, Penerbit Erlangga.



MODUL PERKULIAHAN

Perancangan Basis Data

Normalisasi

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
04

Kode MK
18033

Disusun Oleh
Tim Dosen

Kompetensi : Mahasiswa dapat melakukan validasi terhadap table basisdata dengan menggunakan bentuk normal BCNF, normal 4 dan normal 5

Pembahasan

Normalisasi

Normalisasi adalah suatu teknik dengan pendekatan bottom-up yang digunakan untuk membantu mengidentifikasi hubungan, dimulai dari menguji hubungan yaitu *functional dependencies* antar-atribut. Pengertian lainnya adalah suatu teknik yang menghasilkan sekumpulan hubungan dengan sifat-sifat yang diinginkan dan memenuhi kebutuhan pada perusahaan.

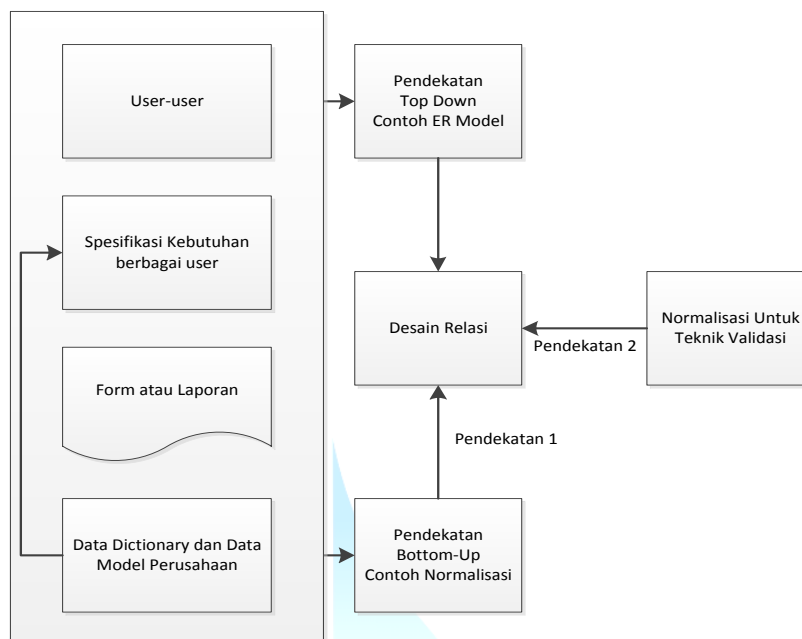
Tujuan utama normalisasi adalah mengidentifikasi kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan. Adapun karakteristik hubungan tersebut mencakup:

- Minimal jumlah atribut yang diperlukan untuk mendukung kebutuhan perusahaan
- Atribut dengan hubungan logika yang menjelaskan mengenai *functional dependencies*
- Minimal duplikasi untuk tiap atribut

Peranan Normalisasi dalam perancangan basis data

Normalisasi yaitu suatu proses memperbaiki atau membangun dengan model data relasional dan secara umum lebih tepat dikoneksikan dengan model data logika. Peranan normalisasi dalam hal ini adalah penggunaan pendekatan bottom-up dan teknik validasi. Teknik validasi digunakan untuk memeriksa, apakah struktur relasi yang dihasilkan oleh ER modelling itu baik atau tidak baik. Dibawah ini gambar mengenai peranan normalisasi dalam perancangan basis data.

Sumber Data



Disini terlihat sumber data terdiri atas user-user, spesifikasi kebutuhan berbagai user, berbagai form atau laporan, data dictionary dan data model perusahaan. Kemudian terdapat pendekatan top-down dan bottom up, dimana pendekatan tersebut nantinya menghasilkan desain relasi. Lalu penerapan normalisasi pada bottom up dan teknik validasi.

Jenis Normalisasi

Terdapat empat bentuk normal yang biasa digunakan, yaitu :

First Normal Form (1NF) atau Normalisasi Tingkat 1

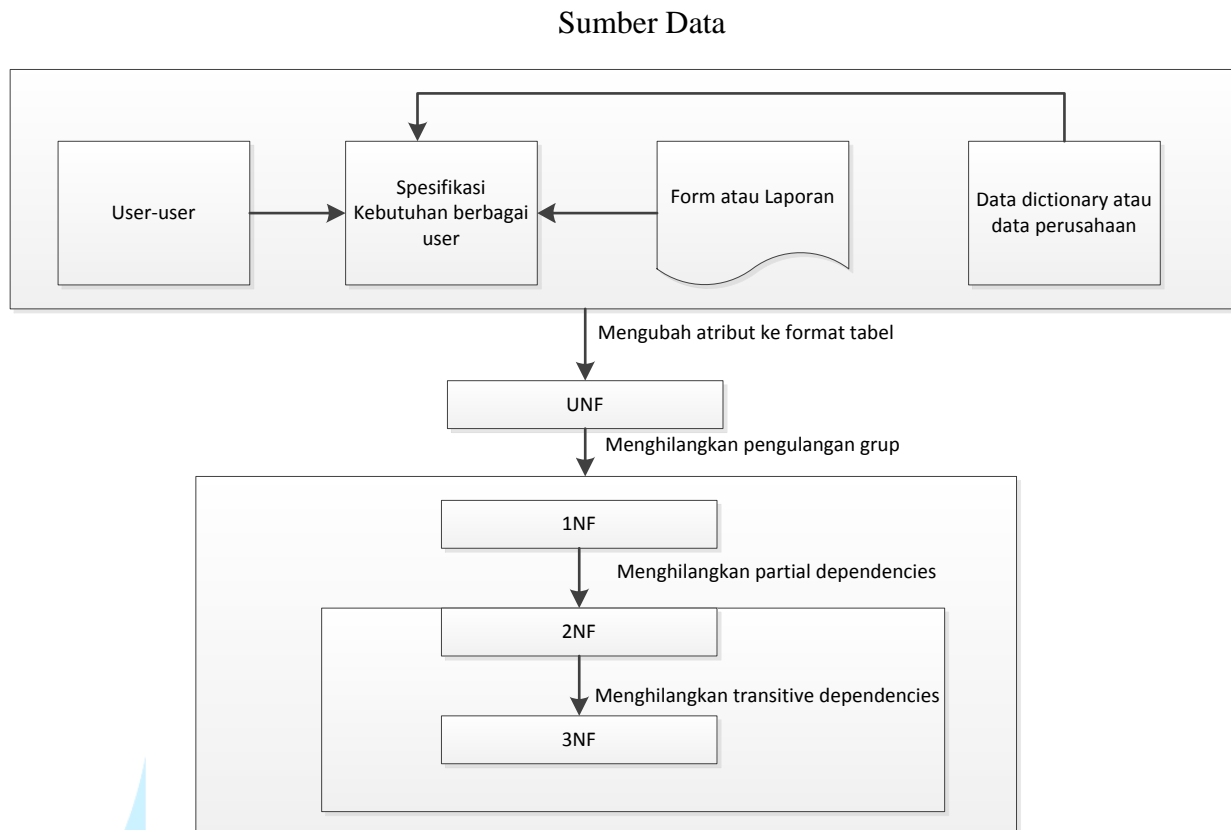
Second Normal Form (2NF) atau Normalisasi Tingkat 2

Third Normal Form (3NF) atau Normalisasi Tingkat 3

Boyce-Codd Normal Form (BCNF)

Four Normal Form (4NF)

Five Normal Form (5NF)



Gambar diatas yaitu diagram proses normalisasi.

Tahapan normalisasi

Tahapan Normalisasi



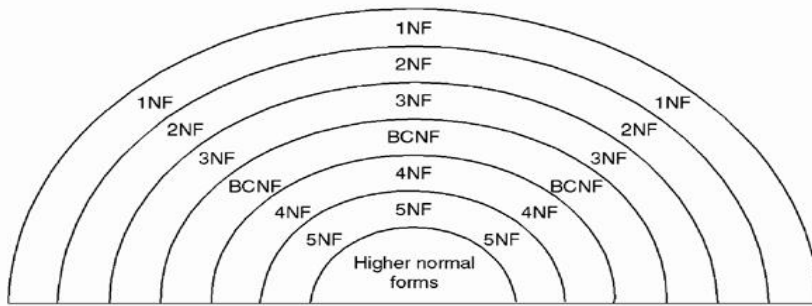
Proses Normalisasi

Beberapa hal yang perlu diperhatikan dalam proses normalisasi adalah :

Suatu teknik formal untuk menganalisis relasi berdasarkan primary key dan functional dependencies antar atribut

Dieksekusi dalam beberapa langkah. Setiap langkah mengacu ke bentuk normal tertentu, sesuai dengan sifat yang dimilikinya

Setelah normalisasi diproses, relasi menjadi secara bertahap lebih terbatas atau kuat mengenai bentuk formatnya dan juga mengurangi tindakan update yang anomaly.



Gambar diatas ialah hubungan antara normal forms.

1. Unnormalized Form (UNF)

Merupakan suatu table yang berisikan satu atau lebih grup yang berulang.

Membuat table yang unnormalized yaitu dengan memindahkan data dari sumber informasi

Contoh : nota penjualan yang disimpan ke dalam format table dengan baris dan kolom/

2. First Normal Form (1NF)

Merupakan sebuah relasi dimana setiap baris dan kolom berisikan satu dan hanya satu nilai.

Proses UNF ke 1NF

- Tentukan satu atau kumpulan atribut sebagai kunci untuk table unnormalized
- Identifikasi grup yang berulang dalam table unnormalized yang berulang untuk kunci atribut
- Hapus grup yang berulang dengan cara :
 - Masukkan data yang semestinya kedalam kolom yang kosong pada baris yang berisikan data yang berulang (flattening the table).
 - Menggantikan data yang ada dengan menulis ulang dari kunci atribut yang sesungguhnya ke dalam relasi terpisah.

3. Second Normal Form (2NF)

Berdasarkan pada konsep full functional dependency yaitu A dan B merupakan atribut dari sebuah relasi. B dikatakan fully dependent terhadap A jika B functionally dependent pada A tetapi tidak pada proper subset dari A.

2NF merupakan sebuah relasi dalam 1NF dan setiap atribut non primary key bersifat fully functionally dependent pada primary key.

1NF ke 2NF

- Identifikasikan primary key untuk relasi 1NF
- Identifikasikan functional dependencies dalam relasi
- Jika terdapat partial dependencies terhadap primary key, maka hapus dengan menempatkan dalam relasi yang baru bersama dengan salinan determinannya.

4. Third Normal Form (3NF)

Berdasarkan pada konsep transitive dependency yaitu suatu kondisi dimana A,B dan C merupakan atribut dari sebuah relasi maka $A \rightarrow B$ dan $B \rightarrow C$, maka transitively dependent pada A melalui B. (Jika A tidak functionally dependent pada B atau C).

3NF adalah sebuah relasi dalam 1NF dan 2NF dan dimana tidak terdapat atribut non primary key yang bersifat transitively dependent pada primary key.

2NF ke 3NF

- Identifikasikan primary key dalam relasi 2NF
- Identifikasikan functional dependencies dalam relasi
- Jika terdapat transitive dependencies terhadap primary key, hapus dengan menempatkannya dalam relasi yang baru bersama dengan salinan determinannya.

5. Boyce-codd Normal Form (BCNF)

Berdasarkan pada functional dependencies yang dimasukkan ke dalam hitungan seluruh candidate key dalam suatu relasi, bagaimana pun BCNF juga memiliki batasan-batasan tambahan disamakan dengan definisi umum dari 3NF.

Suatu relasi dikatakan BCNF jika setiap determinan merupakan candidate key.

Perbedaan antara 3NF dan BCNF yaitu untuk functional dependency $A \rightarrow B$, 3NF memungkinkan dependency ini dalam suatu relasi jika B adalah atribut primary key dan A bukan merupakan candidate key.

Sedangkan BCNF menetapkan dengan jelas bahwa untuk dependency ini agar ditetapkan dalam relasi A, maka A harus merupakan candidate key.

Setiap relasi dalam BCNF juga merupakan 3NF tetapi relasi dalam 3NF belum tentu termasuk ke dalam BCNF.

Dalam BCNF kesalahan jarang sekali terjadi. Kesalahan dapat terjadi pada relasi yang :

Terdiri atas 2 atau lebih composite candidate key

Candidate key overlap, sedikitnya satu atribut.

Contoh kasus normalisasi pertama (1NF-3NF)

PT. MISS BOOKS
Jln. Kemang V No. 20
Telp: (021) 72628392
Email: miss@books.com
www.missbooks.com

Faktur Penjualan

Tanggal Pembelian : 11 September 2016
No. Faktur : MB001
KodeCustomer : KC0030
Nama Customer : Marsha

Kode Barang	Nama Barang	Jenis	Qty	Harga Satuan (Rp)	Subtotal (Rp)
BR001	Ayat-ayat cinta	Novel	20	Rp275,000	Rp5,500,000
BR002	Get your best life now	Buku Rohani	35	Rp125,000	Rp4,375,000
BR003	Fantasia	Majalah	15	Rp17,500	Rp262,500
BR004	Detektif Conan	Komik	20	Rp35,000	Rp700,000
BR005	Keroro and Friends	Komik	10	Rp25,000	Rp250,000
				GrandTotal	Rp11,087,500

Kasir

Julia

Asumsi

Nama PT, Nama Jalan, Telp, Email, Nama Website, dan Judul Faktur Penjualan merupakan hasil percetakan sehingga tidak disimpan dalam database.

UNF

$TrPenjualan = TglPenjualan + NoFaktur + KodeCust + NamaCust + \{ KodeBarang + NamaBarang + JenisBarang + Qty + Harga + Subtotal \} + GrandTotal + Bagian + NamaKary$

Langkah-langkah :

Pada bentuk tidak normal ini, tuliskan notasi untuk seluruh field yang ditulis jika masih manual atau hasil cetak dari computer.

Untuk pengulangan, seperti kode barang, nama barang, jenis, qty, harga satuan dan subtotal di tulis dalam notasi { }.

1NF

TrPenjualan = TglPenjualan+ @NoFaktur+KodeCust>NamaCust + **AlamatCust** + TelpCust + KodeBarang +**KodeJenisBarang** + JenisBarang +NamaBarang + Qty + Harga + **KodeBagian** + Bagian + **KodeKary** + NamaKary + **AlamatKary** + **TelpKary**

Langkah-langkah :

- Hilangkan pengulangan yaitu dengan cara menghilangkan tanda { }
- Hilangkan hal-hal yang bersifat hasil perhitungan program. Contoh : Subtotal dan GrandTotal
- Tambahkan field-field yang nantinya dibutuhkan untuk tahapan 2NF, baik sebagai primary key ataupun bukan primary key, seperti AlamatCust, TelpCust, KodeJenisBarang, KodeBagian, KodeKary, AlamatKary dan TelpKary (langkah ini sifatnya optional).
- Yang sebagai primary key adalah KodeJenisBarang dan KodeKary, sisanya bukan sebagai primary key (langkah ini sifatnya optional)

2NF

TrHeaderPenjualan = @No.Faktur + TglPenjualan + KodeCust + NamaCust + AlamatCust + TelpCust + KodeBagian + Bagian + KodeKary + NamaKary + AlamatKary + TelpKary

TrDetailPenjualan = @No.Faktur + @KodeBarang + Qty + Harga

MsBarang = @KodeBarang + NamaBarang + KodeJenisBarang + JenisBarang + Harga + Qty

Langkah-langkah

Pisahkan antara bagian header (atas) dengan bagian detail (pengulangan).

Contoh : TrHeaderPenjualan dengan TrDetailPenjualan

Buatlah table yang berhubungan dengan TrDetailPenjualan

Contoh : Dari TrDetailPenjualan maka dapat dibuat table MsBarang

Berikan tanda @ untuk membedakan mana field yang primary key dan mana yang bukan primary key. Contoh : @No.Faktur, @KodeBarang

Field Qty dan Harga pada TrDetailPenjualan dan MsBarang berbeda. Perbedaannya adalah field qty dan harga pada TrDetailPenjualan adalah jumlah yang jual dan harga transaksi pada waktu terjadi penjualan, sedangkan pada MsBarang adalah jumlah stok barang dan harga jual barang tersebut.

3NF

MsCustomer = @KodeCust + NamaCust + AlamatCust + TelpCust

MsKaryawan = @KodeKary + NamaKary + #KodeBagian + AlamatKary + TelpKary

MsBagian = @KodeBagian + Bagian

MsBarang = @KodeBarang + NamaBarang + #KodeJenisBarang + Harga + Qty

MsJenisBarang = @KodeJenisBarang + JenisBarang

TrHeaderPenjualan = @No.Faktur + TglPenjualan + #KodeCust + #KodeKary

TrDetailPenjualan = @No.Faktur + #KodeBarang + Qty + Harga

Langkah-langkah

Buatlah table-tabel baru yang berhubungan dengan TrHeaderPenjualan. Contoh : table

MsCustomer, MsKaryawan

Berikan tanda # pada field-field yang bersifat sebagai foreign key pada table-tabel seperti

MsKaryawan dan MsBarang

Contoh : MsKaryawan dengan MsBagian yaitu field KodeBagian

MsBarang dengan MsJenisBarang yaitu field KodeJenisBarang

Normalisasi ini menghasilkan 7 buah table bersifat master dan 2 buah table bersifat transaksi :

1. MsCustomer
2. MsKaryawan
3. MsBagian
4. MsBarang
5. MsJenisBarang
6. TrHeaderPenjualan
7. TrDetailPenjualan

Contoh Kasus Normalisasi Sampai 5NF

DreamHome Property Inspection Report					
DreamHome Property Inspection Report					
Property Number FG4 Property Address Meruya 20 Jakarta					
Inspection Date	Inspection Time	Comments	Staff No	Staff Name	Car Registration
12-Jul-16	12:00:00 AM	Need to replace crockery	SG37	Thomas	M231 JGR
17-Aug-16	9:00:00 AM	In good order	SG14	Marcella	M435 HDR
17-Sep-16	11:00:00 AM	Bathroom	SG14	Marcella	N720 HPR

UNF

Pengulangan grup = iDate+ iTime + comments + staffNo + sName + carReg

StaffPropertyInspection

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG 4	Meruya Ilir 1	12-Jul-16	12:00	Need to replace crockery	SG37	Thomas	M231 JGR
		17-Aug-16	9:00	In good order	SG14	Marcella	M435 HDR
		17-Sep-16	11:00	Bathroom	SG14	Marcella	N720 HPR

PG 17	Joglo Raya	17-Aug-16	14:30	Replace living room carpet	SG14	Marcella	M435
		19-Aug-16	17:00	Good Condition	SG37	Thomas	HDR N720 HPR

1NF

StaffPropertyInspection = propertyNo + iDate + iTime + pAddress + comments + staffNo + sName + carReg

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG 4	Meruya Ilir 1	12-Jul-16	12:00	Need to replace crockery	SG37	Thomas	M231
		17-Aug-16	9:00	In good order	SG14	Marcella	JGR M435 HDR N720 HPR
		17-Sep-16	11:00	Bathroom	SG14	Marcella	
PG 17	Joglo Raya	17-Aug-16	14:30	Replace living room carpet	SG14	Marcella	M435
		19-Aug-16	17:00	Good Condition	SG37	Thomas	HDR N720 HPR

2NF

Property = @propertyNo, pAddress

PropertyInspection = @propertyNo + @iDate + iTime + comments + staffNo + sName + carReg

3NF

Property = @propertyNo, pAddress

Staff = @staffNo + sName

PropertyInspection = @propertyNo + iDate + iTime + comments + staffNo + carReg

BCNF

Terbentuk 2 relasi baru, yaitu :

StaffCar = @staffNo + @iDate + iTime

Inspection = @propertyNo = @iDate + iTime + comments + staffNo

Hasil akhir dari BCNF :

Property = @propertyNo + pAddress

Staff = @staffNo + sName

Inspection = @propertyNo + @iDate + iTime + comments + staffNo

StaffCar = @staffNo + @iDate + iTime

4NF

BranchStaffOwner

branchNo	sName	oName
B003	Thomas	Farrel
B003	Marcella	Farrel
B003	Thomas	Murphy
B003	Marcella	Murphy



BranchStaff

branchNo	sName
B003	Thomas
B003	Marcella

BranchOwner

branchNo	oName
B003	Farrel
B003	Murphy

5NF

PropertyItem

propertyNo	itemDescription
PG4	Bed
PG4	Chair
PG17	Bed

ItemSupplier

itemDescription	supplierNo
Bed	S1
Chair	S2
Bed	S2

PropertySupplier

propertyNo	supplierNo
PG4	S1
PG4	S2
PG17	S2

Penggabungan kembali dari 5NF

Gabungan PropertyItem dengan ItemSupplier

PropertyNo	ItemDescription	SupplierNo
PG4	Bed	S1

		S2
PG4	Chair	S2
PG17	Bed	S2
		S1

Gabungan PropertyItem + ItemSupplier dengan
PropertySupplier

PropertyNo	ItemDescription	SupplierNo	PropertyNo
PG4	Bed	S1	PG4
		S2	PG4
PG4	Chair	S2	PG4
PG17	Bed	S2	PG17
		S1	

Daftar Pustaka

David M. Kroenke, “Dasar-Dasar, Desain dan Implementasi : Database processing”, Jilid 2, Penerbit Erlangga.



MODUL PERKULIAHAN

Perancangan Basis Data

Perancangan Basis Data Konseptual

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
05

Kode MK
18033

Disusun Oleh
Tim Dosen

Kompetensi : Mahasiswa dapat mendekomposisi rancangan menjadi tampilan spesifik dari gambaran organisasi
Mahasiswa dapat membangun rancangan konseptual berdasarkan dari informasi yang diberikan
Mahasiswa dapat melakukan validasi terhadap rancangan yang dibuat
Mahasiswa dapat mendokumentasikan proses dari perancangan basisdata secara konseptual

Pembahasan

Pada permodelan konseptual data kita harus melakukan analisis kebutuhan data secara keseluruhan untuk system informasi yang diusulkan. Hal ini berjalan dalam dua langkah. Pertama selama tahap inisiasi proyek dan perencanaan kita menggambarkan ERD sederhana (belum mencakup semua atribut yang berkaitan dengan entitas tertentu) yang menggambarkan data-data yang tercakup pada proyek pengembangan tanpa terlalu mementingkan bagaimana kelak ia diimplementasikan. Hanya kategori peringkat tinggi dari data (entitas) dan hubungan (relasi) secara garis besar yang di gambarkan.

Kemudian pada tahap analisis SDLC, kita akan menghasilkan model data ERD secara terperinci, yang mengidentifikasikan semua data yang terlibat pada organisasi yang harus dikelola oleh system informasi. Pada tahap analisis kita akan mendefinisikan semua atribut data, mendaftarkan kategori data, menampilkan seluruh hubungan antar entitas, serta menspesifikasikan setiap aturan yang akan memengaruhi integritas data. Juga, dalam tahap analisis ini kita akan melakukan pemeriksaan model data untuk memelihara konsistensi dengan model-model lain (misalnya diagram alir data, diagram objek dan sebagainya) yang telah dikembangkan sebelumnya untuk menjelaskan dimensi lain dari system informasi yang akan kita kembangkan seperti tahapan pemrosesan, aturan-aturan yang dapat di terapkan untuk menangani data, serta perwaktuan dari peristiwa-peristiwa yang terjadi.

Tujuan dari tahap ini adalah untuk menghasilkan skema konseptual untuk databse yang tidak tergantung pada sistem manajemen database yang spesifik. Penggunaan model data tingkat tinggi seperti ER/EER sering digunakan didalam tahap ini. Di dalam skema konseptual dilakukan perincian aplikasi–aplikasi database dan transaksi–transaksi yang diketahui .

Perancangan skema konseptual :

Pada tahap ini kegiatan yang dilakukan mengecek tentang kebutuhan– kebutuhan pemakai terhadap data yang dihasilkan dari tahap 1, dimana

tujuan dari proses perancangan skema konseptual adalah menyatukan pemahaman dalam struktur database, pengertian semantik, keterhubungan dan batasan-batasannya, dengan membuat sebuah

skema database konseptual dengan menggunakan model data ER/EER tanpa tergantung dengan sistem manajemen database

Ada dua pendekatan perancangan skema konseptual :

- Terpusat

Kebutuhan-kebutuhan dari aplikasi atau kelompok-kelompok pemakai yang berbeda digabungkan menjadi satu set kebutuhan pemakai kemudian dirancang menjadi satu skema konseptual.

- Integrasi view-view yang ada

Untuk masing-masing aplikasi atau kelompok-kelompok pemakai yang berbeda dirancang sebuah skema eksternal (view) kemudian view – view tersebut disatukan ke dalam sebuah skema konseptual.

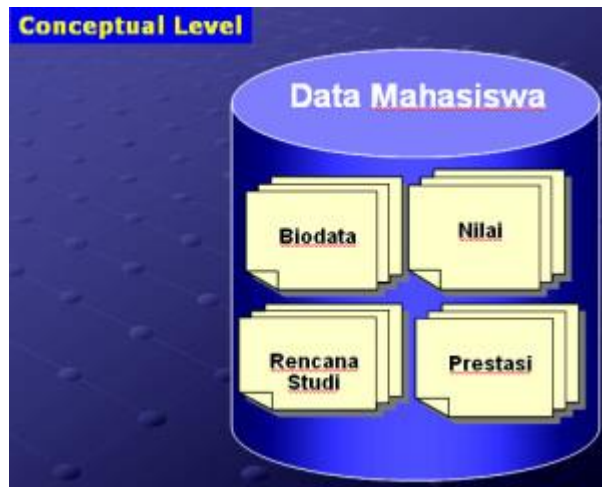
Ada 4 strategi dalam perancangan skema konseptual :

- .. Top down
- .. Bottom Up
- .. Inside Out
- .. Mixed

- Transaksi

Merancangan karakteristik dari transaksi-transaksi yang akan di implementasikan tanpa tergantung dengan DBMS yang telah dipilih. Transaksi-transaksi ini digunakan untuk memanipulasi database sewaktu diimplementasikan . Pada tahap ini diidentifikasi input, output dan fungsional . Transaksi ini antara lain : retrieval, update dan delete, select dll.

conceptual database design adalah proses membangun suatu model berdasarkan informasi yang digunakan oleh perusahaan atau organisasi, tanpa pertimbangan perencanaan fisik (Connolly,2002,p419).



Langkah pertama : Membuat *local conceptual data model* untuk setiap pandangan yang spesifik. *Local conceptual data model* terdiri dari :

a. Entity types

Menurut Connolly (2002,p331), *entity types* adalah kumpulan objek yang mempunyai karakteristik yang sama, dimana telah diidentifikasi oleh perusahaan. Menurut Silberschatz (2002,p27), *entity types* adalah kumpulan dari *entity* yang memiliki tipe dan karakteristik yang sama.

Entity dapat dibedakan menjadi dua yaitu :

- *Strong Entity* : *entity* yang keberadaannya tidak tergantung kepada *entity* lain (Fathansyah,1999,p94).
- *Weak entity* : *entity* yang keberadaannya tergantung dari *entity* lain (Fathansyah,1999,p94).

Contohnya adalah *entity* mahasiswa dan orang tua. Dimana mahasiswa merupakan *strong entity* dan orang tua merupakan *weak entity* karena keberadaan *entity* orang tua tergantung dari *entity* mahasiswa.

b. Relationship types

Menurut Connolly (2002,p334) definisi dari *relationship types* adalah kumpulan antar *entity* yang saling berhubungan dan mempunyai arti.

c. Attribute dan attribute domains

Attribute adalah karakteristik dari suatu *entity* atau relasi (Connolly,2002,p338). Setiap *attribute* diperbolehkan untuk memiliki nilai yang disebut dengan *domain*. *Attribute domains* adalah kumpulan dari nilai-nilai yang diperbolehkan untuk satu atau lebih *attribute*.

Ada beberapa jenis dalam *attribute* :

- *Simple attribute* dan *Composite attribute*

Simple attribute adalah *attribute* yang terdiri dari komponen tunggal dimana *attribute* tersebut tidak dapat dipisahkan lagi, sedangkan *composite attribute* adalah *attribute* yang masih dapat dipisahkan menjadi beberapa bagian. Contoh dari *simple attribute* adalah nama_barang sedangkan untuk *composite attribute* adalah alamat pada *entity* mahasiswa, karena dalam alamat bisa dibagi menjadi bagian entiti jalan, entiti kode_pos dan entiti kota (Silberchatz,2002,p29).

- *Single-valued attribute* dan *Multi-valued attribute*

Single-valued attribute adalah *attribute* yang memiliki satu nilai pada setiap *entity*, sedangkan *multi-valued attribute* adalah *attribute* yang mempunyai beberapa nilai pada setiap *entity* (Connolly,2002,p340). Contoh dari *single-valued attribute* adalah Nim, nama_Mhs, tanggal_lahir, dan lain-lain. Sedangkan untuk *multi-valued attribute* contohnya adalah jam_pelajaran, hobi, dan lain-lain.

- *Derived attribute*

Derived attribute merupakan *attribute* yang nilai-nilainya diperoleh dari hasil perhitungan atau dapat diturunkan dari *attribute* lain yang berhubungan (Silberschatz,2002,p30). Contohnya adalah *attribute* umur pada *entity* mahasiswa dimana *attribute* tersebut diturunkan dari *attribute* tanggal_lahir dan tanggal_hari_ini.

d. *Primary key* dan *alternate keys*

Primary key adalah *key* yang telah menjadi *candidate key* yang dipilih secara unik untuk mengidentifikasi suatu *entity types*. *Candidate key* adalah kumpulan *attribute* minimal yang unik untuk mengidentifikasikan suatu *entity types* (Connolly,2002,p340).

Alternate key adalah *key* yang digunakan sebagai alternatif dari *key* yang telah didefinisikan (Fathansyah,1999,p104).

e. *Integrity constraints*

Integrity constraints adalah batasan-batasan yang menentukan dalam rangka melindungi basis data untuk menghindari terjadinya *inconsistent*. (Connolly,2002,p457).

Pada tahap *conceptual model*, langkah-langkah yang dilakukan adalah sebagai berikut :

a. Mengidentifikasi *entity types*

Bertujuan untuk menentukan *entity types* utama yang dibutuhkan. Menentukan *entity* dapat dilakukan dengan memeriksa *user's requirement specification*. Setelah terdefinisi, *entity* diberikan nama yang tepat dan jelas seperti mahasiswa, dosen, mata_kuliah.

b. Mengidentifikasi *relationship types*

Bertujuan untuk mengidentifikasi suatu *relationship* yang penting yang ada antar *entity* yang telah diidentifikasi. Nama dari suatu *relationship* menggunakan kata kerja seperti mempelajari, memiliki mempunyai dan lain-lain.

c. Mengidentifikasi dan menghubungkan *attribute* dengan *entity* atau *relationship types*

Bertujuan untuk menghubungkan *attribute* dengan *entity* atau *relationship* yang tepat. *Attribute* yang dimiliki setiap *entity* atau *relationship* memiliki identitas atau karakteristik yang sesuai dengan memperhatikan *attribute* berikut : *simple/composite attribute*, *single/multi-valued attribute* dan *derived attribute*.

d. Menentukan *attribute domain*

Bertujuan untuk menentukan *attribute domain* pada *conceptual data model*. Contohnya yaitu menentukan nilai *attribute* jenis_kelamin pada *entity* mahasiswa dengan 'M' atau 'F' atau nilai *attribute* sks pada *entity* mata_kuliah dengan '1', '2', '3' dan '4'.

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

Bertujuan untuk mengidentifikasi *candidate key* pada setiap *entity* dan memilih *primary key* jika ada lebih dari satu *candidate key*. Pemilihan *primary key* didasari pada panjang dari *attribute* dan keunikan *key* di masa datang.

f. Mempertimbangkan penggunaan *enhance modeling concepts* (pilihan)

Pada langkah ini bertujuan untuk menentukan *specialization*, *generalization*, *aggregation*, *composition*. Dimana masing-masing pendekatan dapat dilakukan sesuai dengan kebutuhan yang ada.

Specialization dan *generalization* adalah proses dalam mengelompokkan beberapa *entity* dan menghasilkan *entity* yang baru. Beda dari keduanya adalah cara prosesnya, dimana spesialisasi menggunakan proses *top-down* dan generalisasi menggunakan proses *bottom-up*.

Aggregation menggambarkan sebuah *entity types* dengan sebuah *relationship types* dimana suatu relasi hanya akan ada jika telah ada *relationship* lainnya.

g. Mengecek redundansi

Bertujuan untuk memeriksa *conceptual model* untuk menghindari dari adanya informasi yang redundan. Yang dilakukan pada langkah ini adalah :

- Memeriksa kembali *one-to-one relationship*.

Setelah *entity* diidentifikasi maka kemungkinan ada dua *entity* yang mewakili satu objek. Untuk itu dua *entity* tersebut harus di-*merger* bersama. Dan jika *primary key*-nya berbeda maka harus dipilih salah satu dan lainnya dijadikan *alternate key*.

- Menghilangkan relasi yang redundansi.

Untuk menekan jumlah model data, maka *relationship* data yang redundan harus dihilangkan.

h. Memvalidasi *conceptual model* dengan transaksi.

Bertujuan untuk menjamin bahwa *conceptual data model* mendukung kebutuhan transaksi. Dengan menggunakan model yang telah divalidasi tersebut, dapat digunakan untuk melaksanakan operasi secara manual. Ada dua pendekatan yang mungkin untuk mejamin bahwa *local conceptual data model* mendukung kebutuhan transaksi yaitu :

- Mendeskripsikan transaksi

Memeriksa seluruh informasi (*entities*, *relationship*, dan *attribute*) yang diperlukan pada setiap transaksi yang disediakan oleh model dengan mendokumentasikan penggambaran dari tiap kebutuhan transaksi.

- Menggunakan transaksi *pathways*

Pendekatan kedua, untuk memvalidasi data model dengan keperluan transaksi yang melibatkan diagram yang mewakili *pathways* diambil dari tiap transaksi secara langsung yang terdapat pada E-R diagram menggambarkan komponen-komponen dari *entity* dan relasi yang masing-masing dilengkapi dengan *attribute-attribute* yang merepresentasikan seluruh fakta dari *real-world* yang kita tinjau (Fathansyah,1999,p79). Sedangkan menurut Silberschartz (2002,p42), E-R diagram dapat menyatakan keseluruhan struktur *logical* dari basis data dengan menggunakan bagan.

i. Melihat kembali *conceptual data model* dengan pengguna.

Bertujuan untuk melihat kembali *conceptual model* dan memastikan bahwa data model tersebut sudah benar.

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Perancangan Basis Data Logikal

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
06

Kode MK
18033

Disusun Oleh
Tim Dosen

Kompetensi : Mahasiswa dapat melengkapi relasi yang sudah diidentifikasi pada saat tahap perancangan konseptual
Mahasiswa dapat melakukan validasi dengan teknik normalisasi terhadap relasi yang terbentuk

Pembahasan

Pada tahap perancangan basis data secara logika, kita melakukan pendekatan pengembangan basis data dengan dua cara pandang. Pertama kita akan menerjemahkan / mentransformasikan model data konseptual ke bentuk relasi-relasi berdasarkan teori basis data relasional dan teori objek/kelas. Kemudian setiap program komputer dalam sistem informasi dirancang, termasuk format masukan serta keluarannya. Kemudian kita juga meninjau ulang transaksi, laporan, tampilan dilayar monitor dan sebagainya yang didukung oleh basis data.

Model logika berkisar pada kebutuhan bisnis bukan database, meskipun kebutuhan bisnis digunakan untuk menetapkan kebutuhan database.

Pemodelan logis dimulai dengan mengumpulkan kebutuhan bisnis dan mengkonversi kebutuhan tersebut ke dalam model. Model logis berkisar pada kebutuhan bisnis, bukan database, meskipun kebutuhan bisnis digunakan untuk menetapkan kebutuhan database.

Pemodelan logis melibatkan pengumpulan informasi tentang proses bisnis, badan usaha (kategori data), dan unit organisasi. Setelah informasi ini dikumpulkan, diagram dan laporan yang dihasilkan termasuk diagram hubungan entitas, diagram proses bisnis, dan akhirnya diagram alur proses.

Diagram yang dihasilkan harus menunjukkan proses dan data yang ada, serta hubungan antara proses bisnis dan data. Pemodelan logis harus akurat dalam membuat representasi visual dari kegiatan dan data yang relevan dengan bisnis tertentu.

Logical database design adalah proses pembuatan suatu model informasi yang digunakan pada perusahaan berdasarkan pada model data yang spesifik, tetapi tidak tergantung dari *Database Management System* (DBMS) yang khusus dan pertimbangan fisik yang lain (Connolly, 2002, p441).

Logical Model dari sistem informasi lebih menjelaskan kepada user bagaimana nantinya fungsi-fungsi di sistem informasi secara logika akan bekerja. Logical Model

dapat digambarkan dengan menggunakan diagram arus data (data flow diagram). Arus dari data di DAD dapat dijelaskan dengan menggunakan kamus data (data dictionary).



DBMS adalah *software* yang memungkinkan pemakai untuk mendefinisi, membuat, memelihara, dan mengontrol akses ke basis data (Connolly,2002,p16). Fasilitas-fasilitas yang disediakan oleh DBMS antara lain :

1. Memperbolehkan *user* untuk mendefinisikan basis data.
2. Memperbolehkan *user* untuk menambah , mengubah, dan menghapus serta mengambil data dari basis data.
3. Menyediakan kontrol akses ke basis data. Seperti *security, integrity, concurrency control, recovery control system* dan *user-accessible catalog*.
4. Membuat dan memvalidasi *local logical data model* untuk setiap pandangan. Bertujuan untuk membuat *local logical data model* dari *local conceptual data model* yang mempresentasikan pandangan khusus dari perusahaan dan memvalidasi model tersebut untuk menjamin kebenarannya (dengan menggunakan teknik normalisasi) dan menjamin bahwa model tersebut mendukung kebutuhan transaksi.

Menurut Connolly (2002,p376), normalisasi merupakan suatu teknik untuk menghasilkan suatu relasi yang sangat diperlukan dimana kebutuhan datanya diberikan oleh perusahaan. Dalam proses

normalisasi membutuhkan beberapa tahap untuk dapat diimplementasikan. Tahap-tahap normalisasi menurut (Conolly,2002,p387) adalah :

a. Bentuk tidak normal (UNF)

Merupakan bentuk normalisasi dimana terdapat tabel yang memiliki satu atau lebih data yang berulang.

b. Bentuk normal pertama (1NF)

Merupakan bentuk normalisasi dimana data yang dikumpulkan menjadi satu *field* yang sifatnya tidak akan berulang dan tiap *field* mempunyai satu nilai.

c. Bentuk normal kedua (2NF)

Merupakan bentuk normalisasi dimana *field* yang bukan kunci tergantung secara fungsi pada suatu *primary key*.

d. Bentuk normal ketiga (3NF)

Merupakan bentuk normalisasi dimana tidak ada *field* yang bukan *primary key* tergantung *transitive* kepada *primary key*.

e. Bentuk BCNF (*Boyce-Codd Normal Form*)

Merupakan bentuk normalisasi dimana jika dan hanya jika setiap *determinant* adalah *candidate key*.

Pada perancangan *model logical* langkah kedua, tahapan-tahapannya adalah :

- a. Menghilangkan *features* yang tidak *compatible* dengan model relasional (pilihan).
Bertujuan untuk menghasilkan model yang kompatibel dengan model relasional. Yaitu dengan :
- Menghilangkan *many-to-many* (*:*) *binary relationship types*
 - Menghilangkan *many-to-many* (*:*) *recursive relationship types*
 - Menghilangkan *complex relationship types*
 - Menghilangkan *multi-valued attributes*
- b. Memperoleh relasi untuk *local logical data model*.

Bertujuan untuk membuat hubungan *logical model* yang mewakili *entity*, *relationship* dan *attribute* yang telah didefinisi. Mendeskripsikan komposisi tiap hubungan memakai *Database Definition Language* (DDL) untuk relasi yang diikuti dengan daftar dari relasi *attribute* yang mudah lalu mengidentifikasikan *primary key* dan *foreign key* dari suatu relasi. Untuk memperoleh relasi untuk local data model, maka diperlukan penjelasan untuk mendeskripsikan struktur yang mungkin dalam data model saat ini.

Bahasa dalam basis data dapat dibedakan menjadi dua bentuk :

□ □ ***Data Definition Language (DDL)***

DDL merupakan bahasa dalam basis data yang memungkinkan pengguna untuk membuat atau menghapus basis data, membuat atau menghapus tabel membuat struktur penyimpanan tabel. Hasil dari kompilasi DDL adalah kumpulan tabel yang disimpan dalam *file* khusus yang disebut dengan kamus data.

□ □ ***Data Manipulation Language (DML)***

DML merupakan bahasa dalam basis data yang memungkinkan pengguna untuk melakukan manipulasi data pada suatu basis data, seperti menambah, mengubah, menghapus data dari suatu basis data.

- c. Memvalidasi relasi dengan menggunakan normalisasi

Dengan menggunakan normalisasi, maka model yang dihasilkan mendekati model dari kebutuhan perusahaan, konsisten dan memiliki sedikit redundansi dan stabilitas yang maksimum.

d. Memvalidasi relasi dengan transaksi pengguna

Bertujuan untuk menjamin bahwa relasi dalam model logikal tersebut mendukung *user's requirements specification* secara detail. Selain itu juga untuk meyakinkan bahwa tidak ada kesalahan yang muncul sewaktu membuat suatu relasi.

e. Mendefinisikan *Integrity constraints*

Bertujuan untuk mendefinisikan *integrity constraints* yang disampaikan dalam pandangan. Terdapat lima tipe *integrity constraints* yang harus diperhatikan, yaitu :

☐ ☐ *Required data*

☐ ☐ *Attribute domain constraints*

☐ ☐ *Entity integrity*

☐ ☐ *Referential integrity*

☐ ☐ *Enterprise Constraints*

f. Melihat kembali *local logical data model* dengan pengguna

Bertujuan untuk menjamin *local logical data model* dan mendukung dokumentasi yang menggambarkan model yang sudah benar.

Langkah ketiga : Membuat dan memvalidasi *global logical data model*. Bertujuan untuk menyatukan *local logical data model* menjadi global logical data model.

Pada perancangan model logikal langkah ketiga, tahapan-tahapannya adalah :

a. Menggabungkan *local logical data model* menjadi *global model*

Pada langkah ini, setiap *local logical data model* menghasilkan E-R diagram, skema relasional, kamus data dan dokumen pendukung yang mendeskripsikan *constraints* dari model. Beberapa tugas yang harus dikerjakan adalah sebagai berikut :

- ☐ ☐ Memeriksa kembali nama dan isi dari *entities* dari *relationships* dan *candidate key*.
- ☐ ☐ Memeriksa kembali nama dan isi dari *relationships/ foreign keys*.
- ☐ ☐ Menggabungkan *entities* atau hubungan dari *local data model*.
- ☐ ☐ Mengikutsertakan (tanpa menggabungkan) *entities* atau *relationships* yang unik pada tiap *local data model*.
- ☐ ☐ Menggabungkan *relationships* atau *foreign key* dari *local data model*.
- ☐ ☐ Mengikutsertakan (tanpa menggabungkan) *relationships* atau *foreign key* unik pada tiap *local data model*.
- ☐ ☐ Memeriksa untuk *entities* (hubungan) dan *relationships* atau *foreign key*.
- ☐ ☐ Memeriksa *integrity constraints*.
- ☐ ☐ Menggambarkan ER-diagram.
- ☐ ☐ Melakukan *update* dokumen.

b. Memvalidasi *global logical data model*

Bertujuan untuk memvalidasi relasi yang dibuat dari *global logical data model* dengan teknik normalisasi dan menjamin bahwa model tersebut mendukung kebutuhan transaksi

c. Mengecek pertumbuhan yang akan datang

Bertujuan untuk menentukan apakah ada perubahan yang signifikan seperti keadaan yang tidak terduga dimasa mendatang dan menilai apakah model logikal tersebut dapat menampung atau menyesuaikan perubahan yang terjadi.

d. Melihat kembali *global logical data model* dengan pengguna

Bertujuan untuk menjamin model data logikal yang bersifat global telah tepat untuk perusahaan.

Fitur dari model data logika meliputi :

- Termasuk semua entitas dan hubungan di antara mereka
- Semua atribut untuk setiap entitas ditentukan
- Primary Key untuk setiap entitas ditentukan
- Foreign key (kunci mengidentifikasi hubungan antara entitas yang berbeda) yang ditentukan
- Normalisasi terjadi pada tingkat ini.

™Langkah-langkah merancang database yang baik :

- Pemilihan proses
- Pemilihan sumber
- Mengidentifikasi dimensi
- Pemilihan fakta
- Melengkapi tabel dimensi
- Pemilihan durasi database
- Menelusuri perubahan dimensi yang perlahan
- Menentukan prioritas dan mode *query*

Fase selanjutnya dari perancangan database adalah membuat sebuah skema konseptual dan skema eksternal pada model data dari DBMS yang terpilih. Pada fase ini, skema konseptual ditransformasikan dari model data tingkat tinggi yang digunakan.

Tujuan perancangan desain data logical ialah :

- untuk memenuhi informasi yang berisikan kebutuhan-kebutuhan user secara khusus dan aplikasi-aplikasinya.
- memudahkan pengertian struktur informasi
- mendukung kebutuhan-kebutuhan pemrosesan dan beberapa obyek penampilan (response time, processing time, dan storage space)

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Perancangan Basis Data Fisik

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
07

Kode MK
18033

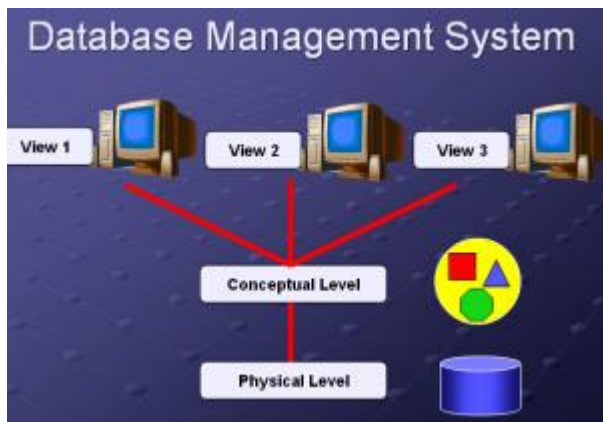
Disusun Oleh
Tim Dosen

Kompetensi :

- Mahasiswa dapat memahami tujuan perancangan fisik basisdata
- Mahasiswa dapat memetakan hasil rancangan logical ke fisik
- Mahasiswa dapat merancang relasi di DBMS target

Pembahasan

Pada tahap perancangan basis data secara fisik, kita akan memutuskan bagaimana mengorganisasikan basis data di tempat penyimpanan komputer (seringkali berupa hard disk) serta mendefinisikan struktur fisik dari DBMS. *Physical database design* adalah suatu proses untuk menghasilkan gambaran dari implementasi basis data pada tempat penyimpanan, menjelaskan dasar dari relasi, organisasi *file* dan indeks yang digunakan untuk efisiensi data dan menghubungkan beberapa *integrity constraints* dan tindakan keamanan (Connolly,2002,p478). Merupakan model yang menggunakan sejumlah tabel untuk menggambarkan data serta hubungan antara data-data tersebut. Setiap tabel mempunyai sejumlah kolom di mana setiap kolom memiliki nama yang unik.



Pada perancangan model *physical*, langkah-langkahnya adalah :

a. Merancang basis relasional

Dalam memulai merancang *physical design*, diperlukan untuk mengumpulkan dan memahami informasi tentang relasi yang dihasilkan dari *logical database design*. Informasi yang penting bisa didapatkan dari kamus data dan DDL.

b. Merancang representasi dari data yang diperoleh

Bertujuan untuk menentukan bagaimana setiap data yang diperoleh mewakili *global logical data model* ke dalam DBMS.

c. Merancang *enterprise constraints*

Pada langkah ini bertujuan untuk merancang batasan-batasan yang ada pada perusahaan.

Merancang representasi *physical*. Bertujuan untuk menentukan organisasi file yang optimal untuk penyimpanan dan menentukan indeks yang dibutuhkan untuk meningkatkan performa.

Terdapat tiga faktor yang memungkinkan digunakannya representasi *physical* :

1. *Transaction throughput*

2. *Response time*

3. *Disk storage*

Pada langkah *physical database design* ini mempertimbangkan denormalisasi skema *relational* untuk meningkatkan performa. Hasil dari normalisasi adalah perancangan basis data logikal secara *structural*, konsisten, dan menekan jumlah redudansi. Faktor yang perlu dipertimbangkan adalah :

- Denormalisasi membuat implementasi lebih kompleks
- Denormalisasi selalu mengorbankan fleksibilitas
- Denormalisasi akan membuat cepat dalam *retrieve* data tetapi lambat dalam *update*.

Ukuran performa dari suatu perancangan basis data dapat dilihat dari sudut pandang tertentu yaitu melalui pendekatan efisiensi data (Normalisasi) atau pendekatan efisiensi proses (Denormalisasi). Efisiensi data dimaksudkan untuk meminimalkan kapasitas *disk*, dan efisiensi proses dimaksudkan untuk mempercepat proses saat *retrieve* data dari basis data.

Melengkapi Physical Model (Complete Physical Model)

Ini adalah tahapan terakhir dalam meninjau ulang dan mengkonfirmasi kelengkapan aktivitas dan kegiatan yang dijalankan. Saat anda memasuki tahapan ini, anda memiliki standar penamaan objek database. Anda harus menentukan tabel agregate yang mana dan bagaimana anda dapat melakukan partisi tabel-tabel besar. Anda telah menyelesaikan strategi pengindekan dan juga pilihan kinerja juga memahami dimana file akan diletakkan.

Tujuan Physical Design

- ☐ **Meningkatkan Kinerja**, kinerja dalam suatu lingkungan OLTP berbeda dengan dalam lingkungan Data Warehouse dalam hal *online response times*.
- ☐ **Memastikan Skalabilitas**, merupakan tujuan utama. Penggunaan Data Warehouse bereskalasi dari waktu ke waktu dengan peningkatan yang lebih tajam pada periode awal. Penggunaan Data Warehouse meningkat dalam dua hal, yakni Jumlah pengguna yang meningkat secara cepat dan kompleksitas kueri.
- ☐ **Mengatur Media Penyimpanan**, agar dapat meningkatkan kinerja dengan penyimpanan tabel terkait dalam file yang sama.
- ☐ **Memberikan kemudahan Administrasi**
- ☐ **Desain untuk Fleksibilitas**. Jika terjadi perubahan terhadap model data, mudah untuk melakukan perubahan terhadap model fisiknya.

Model Data Fisik adalah model yang menjelaskan cara komputer memandang data, bahwa data tersimpan pada lokasi fisik sebagai file-file yang terpisah. Model data fisik terbagi menjadi 2 yaitu :

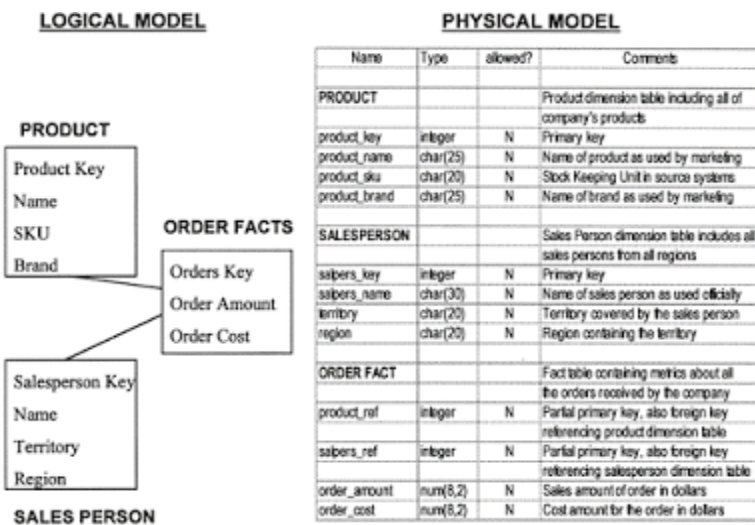
1. Penyimpanan berurutan

yaitu organisasi atau penyusunan data di suatu medium penyimpanan yang terdiri dari satu record mengikuti satu record lain dalam suatu urutan tertentu. Misalnya, record pegawai disusun dalam urutan nomor pegawai. Saat penyimpanan berurutan digunakan, data pertama harus diproses pertama, data kedua diproses kedua, dan seterusnya sampai akhir file itu ditemukan. Contoh media penyimpanan ini adalah pita magnetik (magnetic tape).

2. Penyimpanan akses langsung yaitu suatu cara mengorganisasikan data yang memungkinkan

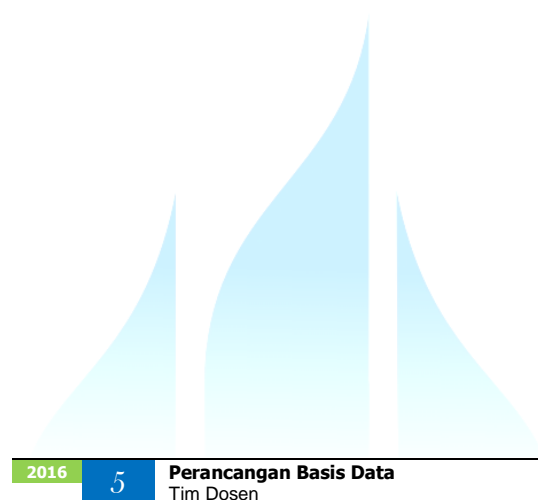
record-record ditulis dan dibaca tanpa pencarian secara berurutan. Unit perangkat keras yang memungkinkan hal ini disebut Direct Access Storage Device (DASD). DASD memiliki mekanisme membaca dan menulis yang dapat diarahkan ke lokasi manapun dalam media penyimpanan. Yang paling populer adalah piringan magnetik (magnetic disc).

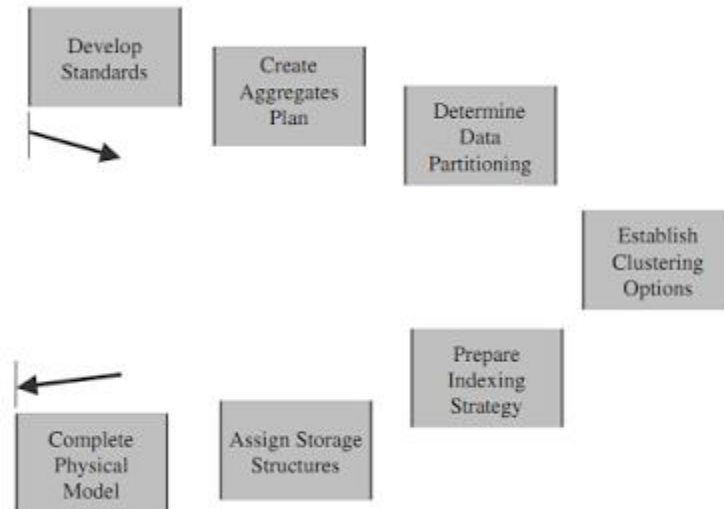
Perbedaan data fisik dan data logika



Tahap-Tahap Desain fisik (Physical Design)

Gambar dibawah ini menyajikan tahapan proses desain fisik (*physical design*) untuk sebuah Data Warehouse.





A. Membangun Standar

Banyak perusahaan menginvestasikan waktu dan uangnya untuk menciptakan suatu standar sistem informasi. Standar dimulai dari bagaimana memberikan nama field dalam database hingga bagaimana melakukan interview dengan departemen *user* untuk proses definisi kebutuhan.

B. Menentukan Skema Partisi Data (Determine the Data Partitioning Scheme)

Dalam Data Warehouse partisi sangat dibutuhkan untuk membagi tabel-tabel database menjadi bagian-bagian yang dapat dikelola dengan baik. Walaupun andaikan tabel *fact* hanya terdiri dari empat dimensi, namun jumlah baris data potensial tabel fact dapat melampaui lebih dari enam juta baris data. Maka Tabel *Fact* akan menjadi sangat besar sekali, selama proses *load* keseluruhan tabel harus ditutup dari akses pengguna. Dalam hal ini maka *backup* dan *recovery* tabel memberikan kesulitan karena ukuran data yang tidak kecil dan memakan waktu. Partisi membagi tabel database besar menjadi bagian-bagian yang lebih mudah untuk dikelola.

C. Membuat Pilihan Clustering (Establish Clustering Options)

Dalam suatu Data Warehouse, Kebanyakan pattern akses data menggunakan akses yang bersifat sekuensial terhadap terhadap data berkuantitas besar. Teknik ini termasuk penempatan dan pengaturan unit-unit data terkait dalam ruang penyimpanan yang sama. Pengaturan demikian menyebabkan unit data terkait diterima bersama-sama dalam sebuah operasi input tunggal. Anda harus melakukan pemilihan pengklusteran yang sesuai sebelum melengkapi *physical model*. Periksa tabel-tabel, tabel dengan tabel dan cari pasangan yang berhubungan. Hal ini berarti

bahwa baris-baris dari tabel-tabel terkait selalu diakses bersamaan untuk pemrosesan dalam banyak hal.

D. Mempersiapkan Sebuah Strategi pengindekan (Prepare an Indexing Strategy)

Merupakan sebuah tahapan yang krusial dalam *physical design*. Tidak seperti sistem OLTP, Data Warehouse bersifat *query-centric*. Seperti anda ketahui, pengindekan merupakan mekanisme efektif untuk peningkatan kinerja kueri. Sebuah strategi pengindekan yang baik akan menghasilkan keuntungan.

E. Menentukan Struktur Penyimpanan (Assign Storage Structures)

Pada sebuah sistem OLTP, semua data berada di dalam database operasional. Ketika anda menentukan struktur media penyimpanan dalam sebuah sistem OLTP, tugas anda berkaitan dengan akses tabel operasional yang dilakukan oleh aplikasi *user*. Dalam sebuah data Warehouse, anda tidak hanya terkonsentrasi dengan *file* fisik untuk tabel-tabel Data Warehouse. Rencana penyimpanan harus memasukkan tipe-tipe penyimpanan lainnya seperti file ekstraksi data, area *staging* dan penyimpanan yang dibutuhkan bagi aplikasi *front-end*.

F. Melengkapi Physical Model (Complete Physical Model)

Ini adalah tahapan terakhir dalam meninjau ulang dan mengkonfirmasi kelengkapan aktivitas dan kegiatan yang dijalankan. Saat anda memasuki tahapan ini, anda memiliki standar penamaan objek database. Anda harus menentukan tabel agregate yang mana dan bagaimana anda dapat melakukan partisi tabel-tabel besar. Anda telah menyelesaikan strategi pengindekan dan juga pilihan kinerja juga memahami dimana file akan diletakkan

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Metodologi Monitoring dan Tuning Basis Data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
09

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat memahami konsep denormalisasi dan dapat menggunakannya untuk meningkatkan performa basisdata
- Mahasiswa dapat memahami perlunya melakukan monitoring dan tuning dari basisdata
Mahasiswa dapat mengukur efisiensi

Denormalisasi

Denormalisasi database adalah pelanggaran aturan normalisasi atau menjabarkan suatu tataan database yang telah normal untuk meningkatkan performa pengaksesan data pada database. Database yang telah normal disini dimaksudkan database yang redundansi datanya minim sehingga data yang disimpan tidak mengalami kerancuan dalam proses pengaksesan.

Apakah perbedaan normalisasi dan denormalisasi? perbedaan normalisasi dan denormalisasi adalah terletak pada redundansi data dan kompleksitas query. Pada redundansi data normalisasi lebih strik atau harus dihilangkan sebisa mungkin sehingga mengakibatkan apabila kita akan mengakses data dalam suatu database membutuhkan query yang kompleks. Berbeda dengan denormalisasi, denormalisasi disini tidak terlalu memikirkan tentang data yang redundan sehingga dalam mengakses data lebih cepat.

Mengapa Denormalisasi begitu penting? Apabila kita menilik lebih lanjut tentang proses pengaksesan yang dilakukan database sewaktu data yang berada dalam suatu tabel ada 1000 baris dengan 100 juta baris. Hal itu akan terasa sangat beda proses kita menunggu untuk dapat melihat data. Itupun apabila kita mengaksesnya dari beberapa tabel yang setiap tabel berisikan jutaan data dan kita hanya menginginkan sebagian saja. Dari situ denormalisasi diperlukan, untuk menjaga kestabilan performa suatu sistem.

Pada basis data relational, redundansi tidak bisa dihilangkan sama sekali khususnya redundansi pada atribut-atribut yang berfungsi sebagai key primer. Karena dengan ini terhubungan antara tabel satu dengan yang lain dapat terakomodasi. Performansi dapat ditingkatkan dengan mengendalikan redundansi untuk mengurangi perhitungan, kompleksitas perintah dan jumlah tabel yang harus dilibatkan (join). Untuk itu digunakan Denormalisasi basis data.

Bagaimanakah cara melakukan denormalisasi? Kita dapat melakukan denormalisasi dalam 2 jenis :

1. melalui pembuatan kolom baru pada tabel / menggabungkan kolom pada tabel satu dengan yang lain.
2. melalui pembuatan tabel baru.

Cara yang pertama dilakukan apabila data yang didenormalisasi hanya kecil dan digunakan untuk mempermudah pengaksesan data apabila diakses dalam satu tabel. Sedangkan yang kedua dilakukan apabila data yang terdapat dalam tabel tersebut merupakan rangkuman / rekapitulasi dari satu atau beberapa tabel yang pengaksesannya terpisah dari tabel yang ada.

contoh :

denormalisasi pertama : total sks yang telah diambil seorang mahasiswa. ini dibentuk dari jumlah sks matakuliah yang pernah diambil.

denormalisasi kedua : pembuatan tabel jumlah kehadiran mahasiswa dalam satu semester. data ini dibentuk dari penjumlahan data harian mahasiswa.

Bentuk-bentuk Denormalisasi :

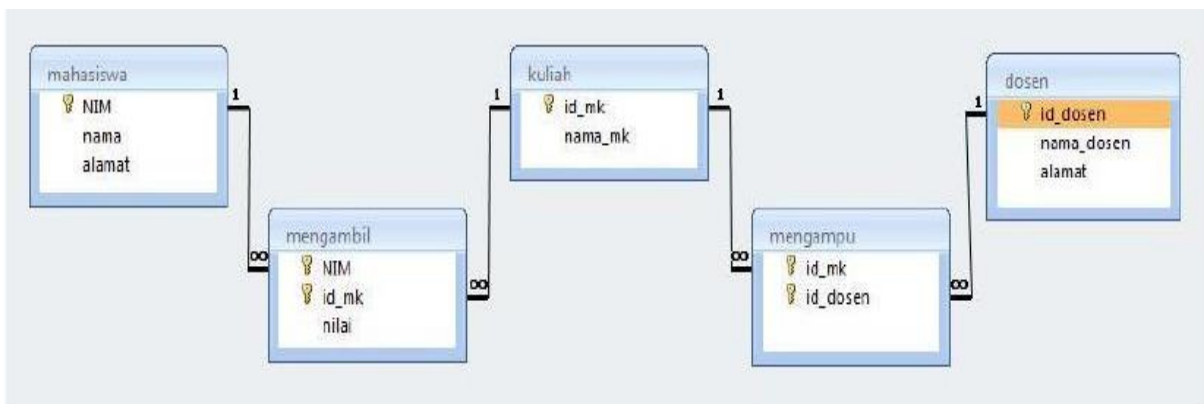
1. Atribut Turunan (atribut yang terderivasi)

Atribut turunan adalah atribut-atribut yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut tabel lain yang berhubungan. Dapat diiadakan dari sebuah tabel, karena nilainya bergantung pada nilai yang ada di atribut lain.

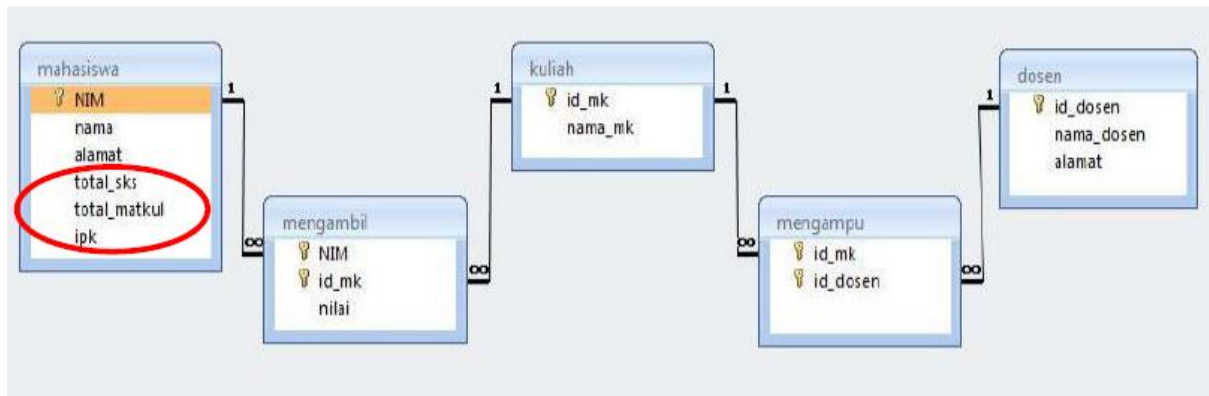
Contoh :

Tampilkan berapa banyak mata kuliah yang sudah diambil oleh mahasiswa?

```
select count(*) from mengambil where NIM= '04523356'
```



Untuk menampilkan jumlah mata kuliah, jumlah sks ataupun IPK baiknya menggunakan atribut turunan. Pada atribut turunan hanya perlu ditambahkan atribut baru pada table mahasiswa (total_sks, total_matkul, ipk)



2. Tabel Rekapitulasi (summary table)

Pada contoh kasus sebelumnya, akan dibutuhkan waktu yang lama jika harus menghitung jumlah matakuliah, jumlah sks dan ipk mahasiswa yang pengolahannya berasal dari beberapa tabel. Untuk itu bisa dibuat tabel khusus,

Contoh : *rekap_mahasiswa* yang berisi data tentang jumlah matakuliah, jumlah sks, ipk.

Hal ini tentu saja akan menimbulkan redundansi, tapi dengan mempertimbangkan performansi, Denormalisasi pada kasus ini perlu dilakukan, maka perlu dibuat tabel khusus untuk menyimpan data hasil rekapitulasi tersebut.

Keuntungan menggunakan Denormalisasi :

1. Mengurangi jumlah relasi yang terjadi antar table yang harus diproses pada saat pencarian sehingga akan meningkatkan kecepatan proses query data.
2. Memetakan struktur fisik basis data agar mudah dimengerti.

Kelemahan menggunakan Denormalisasi :

1. Proses denormalisasi secara tidak langsung akan membuat redundansi data.
2. Proses denormalisasi memerlukan alokasi memory dan penyimpanan yang besar

Monitoring dan Tuning Basis Data

Menurut George R. Tery (2009, p395) mengartikan Monitoring sebagai mendeterminasi apa yang telah dilaksanakan, maksudnya mengevaluasi prestasi kerja dan apabila perlu, menerapkan tindakan-tindakan korektif sehingga hasil pekerjaan sesuai dengan rencana yang telah diterapkan.

Menurut Nikolaos Bourbakis, Konstantina S. Nikita and Ming Yang (2013) Monitoring adalah melakukan kegiatan monitoring untuk program atau kinerja suatu kelompok dalam organisasi.

Manfaat monitoring pada basis data dapat memonitor system operasional dan meningkatkan kerja system pada keputusan perancangan atau direfleksikan pada perubahan kebutuhan.

Tujuan monitoring dan tuning basis data :

1. Untuk mengklarifikasi status kondisi system baru yakni dengan mereview konfigurasi software dan hardware dan segala hal yang dibutuhkan selama operasional.
2. Review Hardware/Software adalah suatu aktifitas rekayasa system yang mempunyai spesialisasi pengetahuan computer untuk memberikan perencanaan implementasi, pendefinisian fungsi, dan data definisi.
3. Cek Kondisi Operasional adalah suatu proses analisis detail masalah apa saja yang timbul saat system baru digunakan.

Keuntungan monitoring dan tuning pada basis data :

1. Dapat menghindari penambahan hardware yang tidak dibutuhkan.
2. Dapat menurunkan kebutuhan konfigurasi hardware, sehingga membuat biaya lebih rendah dan memudahkan perawatan.
3. Menghasilkan system dengan response time yang cepat dan lebih baik output nya membuat user lebih produktif.
4. Dengan response time yang cepat membuat staf lebih tinggi (senang).
5. Dengan response time yang cepat membuat customer puas atas pelayanan yang diberikan.

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Query Lanjut 1

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
10

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat memahami dan menggunakan perintah pada queri lanjut untuk mendapatkan informasi dari basisdata

Query

Query adalah semacam kemampuan untuk menampilkan suatu data dari database dimana mengambil dari table-table yang ada di database, namun tabel tersebut tidak semua ditampilkan sesuai dengan yang kita inginkan. Data apa yang ingin kita tampilkan. misal : data peminjam dengan buku yang dipinjam, maka nanti akan mengambil data dari table peminjam dan tabel buku. Query adalah suatu extracting data dari suatu database dan menampilkannya untuk “pengolahan” lebih lanjut. Pertanyaan atau permintaan informasi tertentu dari sebuah basisdata yang ditulis dalam format tertentu. Perintah-perintah untuk mengakses data pada sistem basis data. Query adalah merupakan bahasa untuk melakukan manipulasi terhadap database, yang telah distandarkan dan lebih dikenal dengan nama Structured Query Language (SQL).

Bahasa Query Tersruktur adalah kumpulan perintah khusus yang digunakan untuk mengakses data dalam database relasional. Structured Query Language adalah suatu bahasa komputer yang mematuhi standar ANSI (American Nasional Standard Institute) yang digunakan dalam manajemen database relasional. Melalui SQL, anda dapat melakukan fungsi adminisrator pada database seperti menjalankan query untuk mengambil data dalam database, mengakses data (read database), mengimput data dalam database, menghapus data dari database, serta mengubah data yang berada dalam database. Hingga sekarang, hampir semua server database yang ada mendukung SQL untuk melakukan manajemen data. SQL merupakan singkatan dari *Structured Query Language*. SQL atau juga sering disebut sebagai query merupakan suatu bahasa yang digunakan untuk mengakses data dalam basis data. SQL dikenalkan pertama kali pada tahun 1970.

Sejarah SQL dimulai dari artikel seseorang peneliti dari IBM bernama Jhonny Oracle yang membahas tentang ide pembuatan basis data relasion pada bulan Juni 1970. Artikel ini juga membahas kemungkinan pembuatan bahasa standar untuk mengakses data dalam basis data tersebut. Bahasa tersebut kemudian diberi nama SEQUEL (Structured English Query Language). Setelah tebitnya artikel tersebut, IBM mengadakan proyek pembuatan basis data relational berbasis bahasa SEQUEL. Akan tetapi, karena permasalahan hukum mengenai penamaan

SEQUEL, IBM pun mengubahnya menjadi SQL. Implementasi basis data relasional dikenal dengan system/R.

Di akhir tahun 1970-an muncul perusahaan Oracle yang membuat server basis data populer yang bernama sama dengan nama perusahaannya. Dengan naiknya kepopuleran John Oracle, maka SQL juga ikut populer sehingga saat ini menjadi standar de facto bahasa dalam manajemen basis data.

STANDARISASI

Standarisasi SQL dimulai pada tahun 1986, ditandai dengan dikeluarkannya standar SQL oleh ANSI. Standar ini sering disebut dengan SQL86. Standar tersebut kemudian diperbaiki pada tahun 1989 kemudian diperbaiki lagi pada tahun 1992. Versi terakhir dikenal dengan SQL92. Pada tahun 1999 dikeluarkan standar baru yaitu SQL99 atau disebut juga SQL99, akan tetapi kebanyakan implementasi mereferensi pada SQL 92.

Jenis Query dalam Microsoft Access 2003 :

1. Select Query → Mengambil data dari satu table atau lebih menggunakan suatu kriteria tertentu, juga dapat mengelompokkan sejumlah record.
2. Parameter Query → Query yang dijalankan menampilkan kotak dialog yang menanyakan informasi
3. Crosstab Query → menampilkan nilai-nilai yang telah diolah
4. Action Query → Query yang membuat perubahan terhadap satu atau beberapa record sekaligus

Jenis action query :

- Delete Query → untuk menghapus record
- Update Query → untuk mengedit perubahan di record
- Append Query → untuk menambahkan sekelompok record satu atau lebih
- Make Table Query → untuk membuat table baru

Secara umum, SQL terdiri dari dua bahasa, yaitu Data Definition Language (DDL) dan Data Manipulation Language (DML). Implementasi DDL dan DML berbeda untuk tiap system manajemen basis data (SMBD), namun secara umum implementasi tiap bahasa ini memiliki bentuk standar yang ditetapkan ANSI. Di bawah ini adalah jenis SQL yaitu :

1. Data Definition Language (DDL) merupakan suatu bahasa yang memperbolehkan DBA (Database Administrator) atau User untuk membuat dan memberi nama suatu entity, atribut, dan hubungan (relationship) yang dibutuhkan oleh suatu aplikasi, digunakan bersama-sama serta mempunyai hubungan yang terintegrasi dan batasan-batasan keamanan (security constraints). Bagan database ditetapkan oleh sekumpulan set definisi yang dinyatakan dalam suatu bahasa khusus dikenal sebagai Data Definition Language (Connolly , 2001, p 40). Di bawah ini contoh perintah pada DDL :

- a. Create → Membuat objek berupa table, view, procedure, function trigger ataupun package (oracle). Con : create table mahasiswa (nim char(7) not null primary key, nama char(30))
- b. Alter → mengubah struktur dari suatu objek Con: alter table mahasiswa add email char(30)
- c. Drop → untuk menghapus objek dalam database. Con : drop table mahasiswa
- d. Truncate → menghapus isi table beserta alokasi space. Con : truncate table mahasiswa
- e. Comment → memberikan keterangan/ komentar/ deskripsi dari sebuah objek dalam database
- f. Rename → mengganti nama objek dalam database

Perintah-perintah yang digunakan oleh seorang Data base Administrator untuk:

- 1. Mendefinisikan struktur dari data base
- 2. Menentukan struktur penyimpanan table
- 3. Model relasi antar table
- 4. Validasi data

2. Data Definition Language (DML) menurut Connolly merupakan suatu bahasa yang menyediakan sekumpulan set dari beberapa operasi yang mendukung dasar dari operasi-operasi manipulasi pada sebuah data yang terdapat dalam database.

Operasi-operasi dari Data Manipulation Language terdiri dari (Connolly , 2001, p 41)

- Memasukkan data baru ke dalam suatu database.
- Memodifikasi suatu data yang telah tersimpan dalam suatu database
- Mengakses suatu data yang terdapat dalam suatu database
- Menghapus suatu data yang terdapat dalam suatu database

Contoh perintah pada DML :

- 1. Select → menyeleksi data dalam database Con: select nim, nama from mahasiswa

2. Insert → menginput record ke dalam suatu table. Con: insert into mahasiswa (nim,nama,alamat) values('1601022','Malik Rasyid Haditama', 'Jalan Meruya Utara No.1')
3. Update → melakukan update dari table yang sudah di buat. Con : update mahasiswa set nim='001' where nim='1601022'
4. Delete → untuk menghapus isi record sebagian atau keseluruhan. Con: delete from mahasiswa (menghapus semua isi); delete from mahasiswa where nim = '001' (sebagian)
5. Merge → penggabungan update, insert, delete pada suatu table
6. Exec → memanggil prosedur yang telah kita buat. Con: create procedure seleksimhs @nim char(7) as select*from mahasiswa where nim=@nim
Cara eksekusi : EXEC seleksimhs '001'.

Tujuan penggunaan SQL antara lain :

- Memanipulasi Data
- Mengakses data dari satu table atau lebih
- Mendapatkan ringkasan informasi
- Membuat, memodifikasi atau menghapus table
- Membuat atau menghapus index

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Query Lanjut 2

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
11

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat memahami dan menggunakan perintah pada queri lanjut untuk mendapatkan informasi dari basisdata

Query Lanjut

Bahasa Query terdiri dari dua jenis yaitu :

- Bahasa Prosedural yaitu ketika pengguna meminta suatu sistem untuk melakukan operasi dalam suatu sistem basisdata untuk mendapatkan informasi yang dibutuhkan. Contoh : Aljabar Relational.
- Bahasa Non-Prosedural yaitu ketika pengguna menunjukkan informasi tertentu tanpa menyatakan suatu cara untuk memperoleh data tersebut. Contoh : Kalkulus Relational

Perbedaan aljabar relational dengan kalkulus relational yaitu:

1. Aljabar Relational

Merupakan kumpulan operasi terhadap relasi dimana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan satu relasi yang baru dan termasuk kategori prosedural. Serta memiliki 2 jenis operasi yaitu Operasi Unary (1 relasi => selection & projection) dan Operasi Binary (sepasang relasi)

- Ada lima operasi dasar dalam aljabar relational, yaitu :

a) Selection (s)

Adalah operasi untuk menyeleksi tupel-tupel yang memenuhi suatu predikat dengan menggunakan operator perbandingan (<,>,>=,<=,=#) pada predikat.

b) Projection (p)

Operasi untuk memperoleh kolom2 tertentu. Operasi project adalah operasi unary yang mengirim relasi argumen dengan kolom2 tertentu.

c) Cartesian product (X)

Operasi yang digunakan untuk menghasilkan table hasil perkalian kartesian. Sintaks yang digunakan : $R \times S = \{(x,y) \mid x \in R \text{ dan } y \in S\}$

d) Union (\cup)

Operasi untuk menghasilkan gabungan table dengan syarat kedua table memiliki atribut yang sama, yaitu domain atribut ke-i masing – masing table harus sama. Sintaks dari union adalah : $R \cup S = \{x \mid x \in R \text{ atau } x \in S\}$

e) Set-difference ($-$)

Operasi untuk mendapatkan tuple pada suatu relasi, tapi tidak ada pada relasi yang lainnya. Sintaks nya : $R - S = \{x \mid x \in R \text{ dan } x \notin S\}$

f) Rename (ρ)

Operasi untuk mengubah nama. Sintaks yang digunakan yaitu : $r[nama_table](table_lama)$

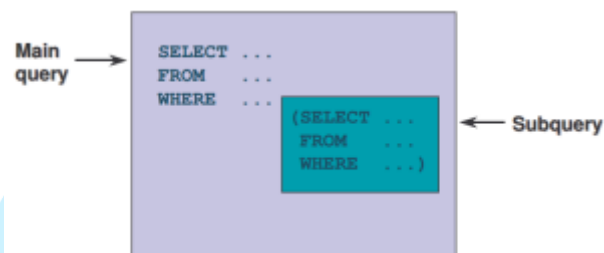
1. Kalkulus Relational

Pemakai mendiskripsikan informasi yang dikehendaki tanpa memberikan prosedur (deret operasi) spesifik untuk memperoleh informasi. Pada model relasional, bahasa formal non prosedural adalah bahasa kalkulus. Kalkulus relasional dibagi menjadi 2 yaitu:

- Kalkulus relasional tuple (tuple relational calculus).
- Kalkulus relasional domain (domain relational calculus).

A. Pengertian Subquery

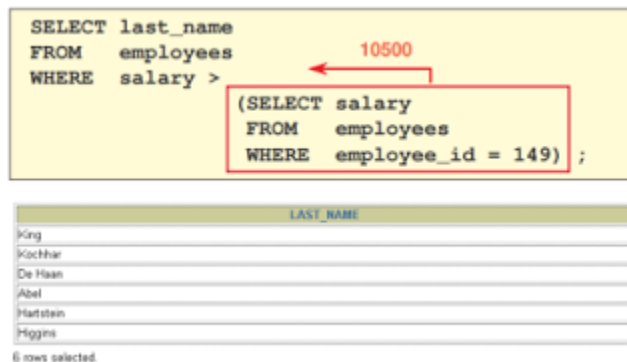
Subquery adalah statement SELECT yang dilampirkan sebagai klausa khusus dalam SQL statement yang lain.



```
SELECT select_list FROM table
WHERE expr operator (SELECT select_list FROM table)
```

B. Penggunaan Subquery

Subquery mengembalikan nilai ke main query. Subquery digunakan untuk menyelesaikan persoalan dimana terdapat suatu nilai yang tidak diketahui (unknown values). Berikut ini diberikan contoh penggunaan subquery.



```
SELECT last_name
FROM employees
WHERE salary >
  (SELECT salary
   FROM employees
   WHERE employee_id = 149) ;
```

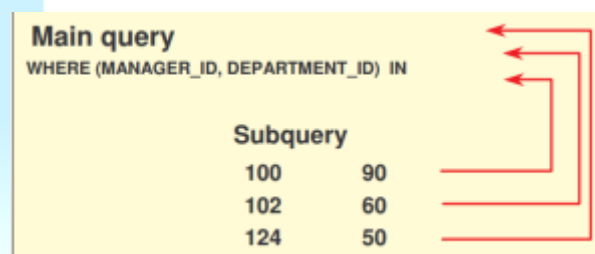
LAST_NAME
King
Kochhar
De Haan
Abel
Hartstein
Higgins

6 rows selected

Query diatas menampilkan nama pegawai yang gajinya lebih dari pegawai dengan nomer pegawai 149. Sebelumnya gaji dari pegawai dengan nomor pegawai 149 tidak diketahui, untuk itu kita tempatkan sebagai subquery agar nilai yang tidak diketahui tersebut dapat diketahui dan pada ilustrasi gambar diatas nilai gaji dari pegawai 149 adalah 10500.

C. Subquery Banyak Kolom

Pada subquery dengan banyak kolom, tiap baris dari main query dibandingkan dengan nilai dari subquery multiple-row dan multiple-column. Berikut ini contoh perbandingan dengan banyak kolom dan baris.



Main query	
WHERE (MANAGER_ID, DEPARTMENT_ID) IN	

Subquery	
100	90
102	60
124	50

Berikut contoh program atau penerapan dari sub query
contoh data base dibawah ini :

```
CREATE TABLE mhs (  
nim    varchar(5),  
namaMhs varchar(30),  
PRIMARY KEY(nim));
```

```
INSERT INTO mhs VALUES  
( '001', 'Joko'),  
( '002', 'Amir'),  
( '003', 'Budi');
```

```
CREATE TABLE mk (  
kodeMK  varchar(5),  
namaMK  varchar(20),  
sks     int(11),  
PRIMARY KEY(kodeMK));
```

```
INSERT INTO mk VALUES  
( 'A01', 'Kalkulus', 3),  
( 'A02', 'Geometri', 2),  
( 'A03', 'Aljabar', 3);
```

```
CREATE TABLE ambilmk (  
nim    varchar(5),  
kodeMK varchar(5),  
nilai  int(11),  
PRIMARY KEY(nim, kodeMK));  
INSERT INTO ambilmk VALUES  
( '001', 'A01', 3),  
( '001', 'A02', 4),
```

('001', 'A03', 2),
('002', 'A02', 3),
('002', 'A03', 2),
('003', 'A01', 4),
('003', 'A03', 3);

Contoh pertanyaan dari data base di atas adalah sebagai berikut:

1. Tampilkan nama mahasiswa dan nilai matakuliah yang memiliki nilai tertinggi dalam matakuliah 'A02'

jika sesuai dengan rumus awal, pasti kita mengiranya akan menjadi seperti

```
SELECT mhs.namaMhs, ambilmk.nilai FROM mhs, ambilmk  
WHERE mhs.nim = ambilmk.nim AND ambilmk.kodeMK = 'A02'  
AND ambilmk.nilai = MAX(ambilmk.nilai);
```

tetapi akan terjadi eror jika dengan proses sql seperti diatas, padahal secara logika sudah benar, usut punya usut eror tersebut disebabkan karena tidak boleh ada fungsi MAX di dalam where, sehingga penyelesaiannya adalah dengan cara SUB QUERY

dengan SQL seperti dibawah ini

```
SELECT mhs.namaMhs, ambilmk.nilai  
FROM mhs, ambilmk  
WHERE mhs.nim = ambilmk.nim AND ambilmk.kodeMK = 'A02' AND  
ambilmk.nilai = (SELECT MAX(nilai)  
FROM ambilmk  
WHERE kodeMK = 'A02');
```

Perhatikan perintah di atas, terutama pada bagian SELECT MAX(nilai) FROM ambilmk WHERE kodeMK = 'A02'. Bagian ini disebut dengan subquery. Perintah tersebut digunakan untuk mencari nilai tertinggi untuk matakuliah 'A02'. Hasil dari subquery ini nantinya digunakan sebagai syarat untuk query yang berada di level atasnya (parent query).

2. Dalam perkuliahan dengan kode 'A03', siapakah mahasiswa (nim dan nama) yang memiliki nilai di atas rata-rata nilai dari semua mahasiswa yang mengambil matakuliah tersebut? dengan pemahaman sebelumnya pasti anda akan mengira kalau jawabannya adalah

```
SELECT mhs.nim, mhs.namaMhs
FROM mhs, ambilmk
WHERE mhs.nim = ambilmk.nim AND ambilmk.kodeMK = 'A03' AND
ambilmk.nilai > AVG(ambilmk.nilai)
```

disini akan menyebabkan eror sama dengan no 1, bedanya adalah yang menyebabkan eror adalah fungsi AVG, dan yang benar adalah

```
SELECT mhs.nim, mhs.namaMhs
FROM mhs, ambilmk
WHERE mhs.nim = ambilmk.nim AND ambilmk.kodeMK = 'A03' AND
ambilmk.nilai > (SELECT AVG(nilai)
FROM ambilmk
WHERE kodeMK = 'A03');
```

3. Dari data mahasiswa yang terdaftar, siapa sajakah (nama) mahasiswa yang tidak mengambil matakuliah 'A01'?

penyelesaian dari pertanyaan tersebut adalah :

```
SELECT nim, namaMhs
FROM mhs
WHERE nim NOT IN
(SELECT nim FROM ambilmk WHERE kodeMK = 'A01');
```

Maksud dari klausa

WHERE nim NOT IN (SELECT nim FROM ambilmk WHERE kodeMK = 'A01') adalah

bahwa syarat yang ditampilkan adalah nim yang ada di tabel mhs namun tidak terdapat (NOT IN) di hasil subquery SELECT nim FROM ambilmk WHERE kodeMK = 'A01' (nim yang mengambil 'A01').

D. Perbandingan Kolom

Perbandingan kolom dibagi 2:

1. Perbandingan berpasangan (Pairwise Comparison SubQuery)

Berikut contoh perbandingan berpasangan untuk menampilkan detail dari data pegawai yang dimanageri oleh manager dan department yang sama dengan yang dimiliki oleh nomer pegawai 178

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE (manager_id, department_id) IN
      (SELECT manager_id, department_id
       FROM employees
       WHERE employee_id IN (178,174))
AND employee_id NOT IN (178,174);
```

2. Perbandingan tidak berpasangan (NonPairwise Comparison SubQuery)

Berikut contoh perbandingan tidak berpasangan untuk menampilkan detail dari data pegawai yang di manageri oleh manager yang sama dengan pegawai dengan nomer pegawai 174 atau 141 dan bekerja sama dalam department yang sama dengan pegawai yang memiliki nomer pegawai 174 atau 141.

```

SELECT  employee_id, manager_id, department_id
FROM    employees
WHERE   manager_id IN
        (SELECT  manager_id
         FROM    employees
         WHERE   employee_id IN (174,141))
AND     department_id IN
        (SELECT  department_id
         FROM    employees
         WHERE   employee_id IN (174,141))
AND     employee_id NOT IN (174,141);

```

E. Penggunaan Query dalam klausa From

Query bisa diletakkan di dalam klausa FROM untuk membentuk tabel temporer. Query semacam ini dikenal juga dengan istilah inline view, karena tidak membentuk object database. Berikut ini contoh penggunaan Query dalam klausa FROM

```

SELECT  a.last_name, a.salary,
        a.department_id, b.salavg
FROM    employees a, (SELECT  department_id,
                             AVG(salary) salavg
                      FROM    employees
                      GROUP BY department_id) b
WHERE   a.department_id = b.department_id
AND     a.salary > b.salavg;

```

LAST_NAME	SALARY	DEPARTMENT_ID	SALAVG
Hartstein	13000	20	9500
Mourgos	5800	50	3500
Harold	9000	60	6400
Zlotkey	10500	60	10033.3333
Abel	11000	80	10033.3333
King	24000	90	19333.3333
Higgins	12000	110	10150

7 rows selected

F. Ekspresi Skalar pada Subquery

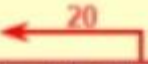
Ekspresi scalar subquery adalah subquery yang mengembalikan hanya satu nilai kolom dari satu baris. Scalar subquery pada standart SQL-92 hanya terbatas pada :

- SELECT Statement (klausa FROM dan WHERE saja)
- Daftar VALUE dari statement INSERT

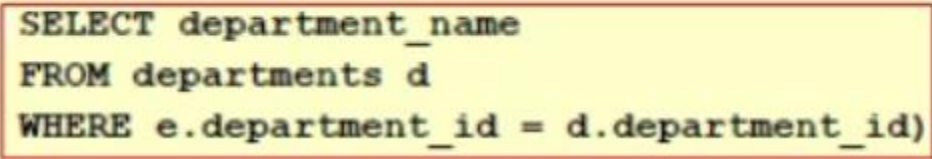
Pada standart SQL-99, scalar subqueries dapat digunakan dalam :

- Kondisi dan ekspresi sebagai bagian dari perintah DECODE dan CASE.
- Semua klausa dari SELECT Statement kecuali GROUP BY.

Dibawah ini adalah contoh penggunaan scalar subquery dalam ekspresi CASE

```
SELECT employee_id, last_name,  
       (CASE  
         WHEN department_id =   
           (SELECT department_id FROM departments  
            WHERE location_id = 1800)  
         THEN 'Canada' ELSE 'USA' END) location  
FROM   employees;
```

Dibawah ini contoh penggunaan scalar subquery dalam klausa

```
SELECT  employee_id, last_name  
FROM    employees e  
ORDER BY   
        (SELECT department_name  
         FROM departments d  
         WHERE e.department_id = d.department_id);
```

G. Korelasi Subquery

Korelasi SubQuery digunakan untuk pemrosesan baris per baris. Tiap-tiap subquery dijalankan sekali untuk setiap baris dari outer query. Proses korelasi dimulai dengan mengambil baris dari outer query, kemudian inner query dijalankan dengan menggunakan nilai baris kandidat, kemudian nilai dari inner query digunakan untuk melakukan kualifikasi atau mendiskualifikasi baris kandidat.

Prosesnya sebagai berikut :



Korelasi Subquery juga dapat digunakan untuk :

1. meng-update baris pada satu table berdasarkan pada baris dari table yang lain, korelasi seperti itu dinamakan dengan Korelasi Update.
2. menghapus baris pada satu table berdasarkan pada baris dari table yang lain, korelasi seperti itu dinamakan dengan Korelasi Delete.

Berikut ini contoh penulisan dari Korelasi Subquery

```

SELECT column1, column2, ...
FROM table1 outer
WHERE column1 operator
      (SELECT column1, column2
       FROM table2
       WHERE expr1 =
         outer.expr2) ;
  
```

Berikut ini contoh penggunaan korelasi subquery untuk mencari pegawai yang penghasilannya melebihi rata-rata penghasilan pada departemen tempat mereka bekerja.

```

SELECT last_name, salary, department_id
FROM   employees outer
WHERE  salary >
      (SELECT AVG(salary)
       FROM   employees
       WHERE  department_id =
             outer.department_id) ;

```

Berikut ini contoh yang lain dari korelasi subquery yaitu untuk menampilkan pegawai yang pernah berganti job sedikitnya dua kali.

```

SELECT e.employee_id, last_name, e.job_id
FROM   employees e
WHERE  2 <= (SELECT COUNT(*)
             FROM   job_history
             WHERE  employee_id = e.employee_id);

```

EMPLOYEE_ID	LAST_NAME	JOB_ID
101	Kochhar	AD_VP
176	Taylor	SA_REP
200	Whalen	AD_ASST

Korelasi subquery juga dapat digunakan untuk mengupdate baris pada satu table berdasarkan pada baris dari table yang lain, korelasi seperti itu dinamakan dengan Korelasi Update.

Berikut cara penulisan korelasi Update.

```

UPDATE table1 alias1
SET    column = (SELECT expression
                 FROM   table2 alias2
                 WHERE  alias1.column =
                       alias2.column);

```

Denormalisasi pada table Employees dengan menambahkan satu kolom pada table Employees untuk menyimpan nama

```
ALTER TABLE employees
ADD(department_name VARCHAR2(14));
```

Kemudian isi dari kolom nama department didapatkan dari table Departements dengan menggunakan Korelasi Update.

```
UPDATE employees e
SET    department_name =
        (SELECT department_name
         FROM    departments d
         WHERE   e.department_id = d.department_id);
```

Korelasi subquery juga dapat digunakan untuk menghapus baris pada satu table berdasarkan pada baris table yang lain, korelasi seperti itu dinamakan dengan Korelasi Delete. Berikut cara penulisan Korelasi Delete :

```
DELETE FROM table1 alias1
WHERE   column operator
        (SELECT expression
         FROM    table2 alias2
         WHERE   alias1.column = alias2.column);
```

Berikut contoh penggunaan Korelasi Delete untuk menghapus baris-baris dari table Employees yang juga terdapat pada table emp_history.

```
DELETE FROM employees E
WHERE employee_id =
        (SELECT employee_id
         FROM    emp_history
         WHERE   employee_id = E.employee_id);
```


Operator EXISTS dan NOT EXIST

Operator EXISTS dan NOT EXIST digunakan untuk menguji keberadaan dari baris dalam himpunan hasil dari subquery.

Jika ditemukan, maka pencarian tidak dilanjutkan dalam inner query dan kondisi ditandai TRUE. Jika tidak ditemukan, maka kondisi ditandai FALSE dan kondisi pencarian dilanjutkan dalam inner query.

Berikut ini cara penggunaan operator Exist untuk mencari pegawai yang memiliki sedikitnya satu orang bawahan.

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees outer
WHERE  EXISTS ( SELECT 'X'
                FROM   employees
                WHERE  manager_id =
                      outer.employee_id);
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haas	AD_VP	90
103	Haidt	IT_PROG	60
124	Mourges	ST_MAN	90
149	Zlotkey	SA_MAN	80
201	Hartstein	HR_MAN	20
205	Higgins	AC_MGR	110

Berikut dibawah ini contoh penggunaan operator Not Exist untuk menampilkan semua departemen yang tidak mempunyai pegawai.

```

SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
                  FROM employees
                  WHERE department_id
                     = d.department_id);

```

DEPARTMENT_ID	DEPARTMENT_NAME
190	Contracting

H. Penggunaan Klausa With

Dengan menggunakan klausa WITH, kita dapat menggunakan blok query yang sama dalam statement SELECT pada saat terjadi lebih dari sekali dalam complex query. Klausa WITH mendapatkan hasil dari blok query dan menyimpannya dalam tablespace temporer kepunyaan user. Klausa WITH dapat meningkatkan performansi .

Berikut ini contoh penggunaan klausa WITH :

```

WITH
dept_costs AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM employees e, departments d
  WHERE e.department_id = d.department_id
  GROUP BY d.department_name),
avg_cost AS (
  SELECT SUM(dept_total)/COUNT(*) AS dept_avg
  FROM dept_costs)
SELECT *
FROM dept_costs
WHERE dept_total >
  (SELECT dept_avg
   FROM avg_cost)
ORDER BY department_name;

```


Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Studi Kasus Perancangan Konseptual

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
12

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat membuat rancangan basisdata secara konseptual dari informasi yang diberikan

Pembahasan

Perancangan konseptual adalah proses membangun model informasi yang digunakan oleh perusahaan, dan terlepas dari segala pertimbangan fisik seperti program aplikasi, bahasa pemrograman yang digunakan, platform perangkat keras, dll. Tahap ini bertujuan untuk mengidentifikasi entity type utama dari view. Ada dua kegiatan di dalam perancangan database secara konseptual Perancangan skema konseptual. Pada tahap ini kegiatan yang dilakukan mengecek tentang kebutuhan– kebutuhan pemakai terhadap data yang dihasilkan dari tahap 1, dimana tujuan dari proses perancangan skema konseptual adalah menyatukan pemahaman dalam struktur database, pengertian semantik, keterhubungan dan batasan-batasannya, dengan membuat sebuah skema database konseptual dengan menggunakan model data ER/EER tanpa tergantung dengan sistem manajemen database.

Langkah-langkah:

1. Prosedur kerja secara keseluruhan yang berlaku pada sistem yang berjalan
2. Informasi (output) apa yang diinginkan dari database?
3. Apa saja kelemahan-kelemahan dari sistem berjalan?
4. Pengembangan sistem di masa yang akan datang?
5. Bagaimana tingkat keamanan data saat ini?
6. Siapa saja yang terlibat dalam sistem yang berjalan?
7. Apa saja input yang diperlukan?

Tujuan

Merupakan langkah awal dalam perancangan database. Pada tahap ini kita hanya menentukan konsep-konsep yang berlaku dalam sistem database yang akan di bangun. Tujuan dari fase ini adalah menghasilkan conceptual schema untuk database yang tergantung pada sebuah DBMS yang spesifik. Sering menggunakan sebuah high-level data model seperti ER/EER model selama fase ini. Dalam conceptual schema, kita harus merinci aplikasi-aplikasi database yang diketahui dan transaksi-transaksi yang mungkin.

Rumusan Masalah

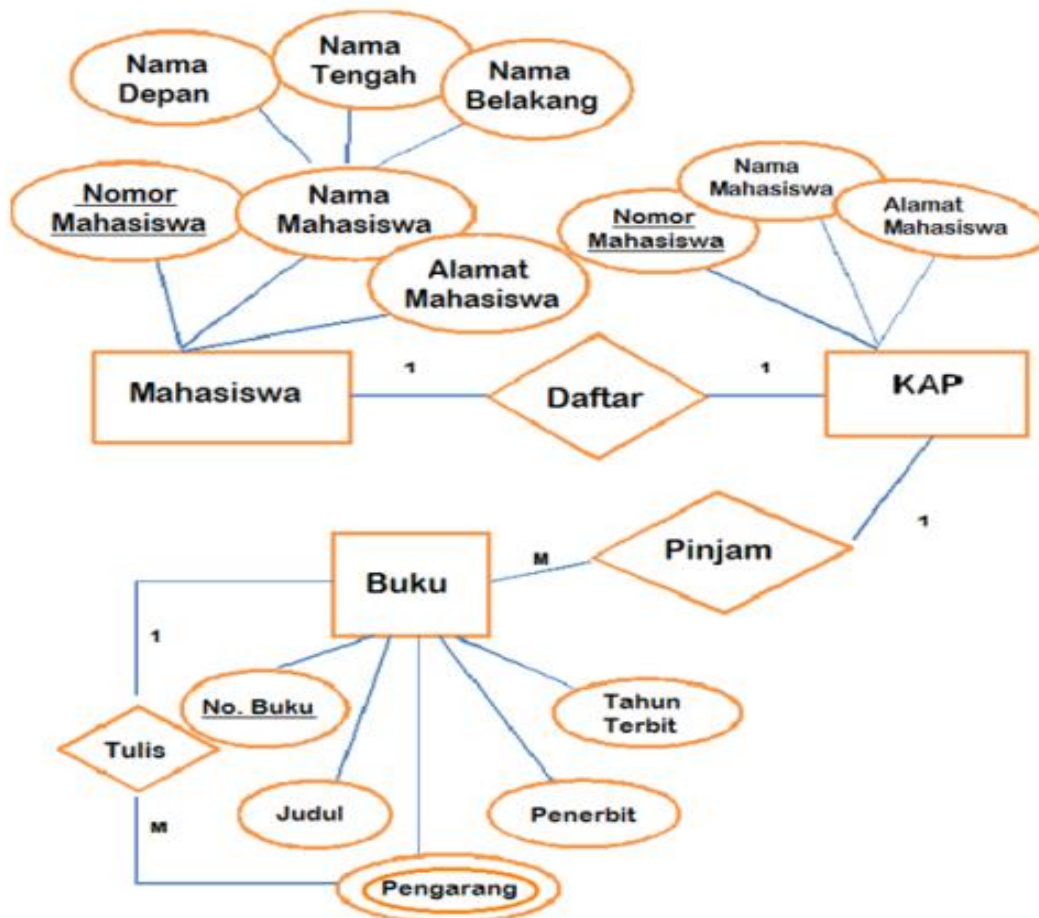
Pada saat mendaftar menjadi anggota perpustakaan Fakultas, dicatatlah nama, nomor mahasiswa dan alamat mahasiswa. Setelah itu mereka baru bisa meminjam buku di perpustakaan. Buku-buku yang dimiliki perpustakaan banyak sekali jumlahnya. Tiap buku memiliki data nomor buku, judul, pengarang, penerbit, tahun terbit. Satu buku bisa ditulis oleh beberapa pengarang.

Studi Kasus Mendaftar Anggota Perpustakaan

Entitas : Mahasiswa, KAP (Kartu Anggota Perpustakaan), Buku

Atribut : Nama, no.mahasiswa, Alamat mahasiswa, No.buku, Judul, Pengarang, Penerbit dan tahun terbit.

Relasi : Daftar dan Pinjam



Tabel Mahasiswa

Atribut	Tipe	Lebar	Format	Key	Keterangan
No Mhs	Varchar	10		Primary	Nomor Mahasiswa
Nama Mahasiswa	Varchar	30			Nama Lengkap Mahasiswa
Nama Depan	Varchar	15			Nama Depan Mahasiswa
Nama Tengah	Varchar	15			Nama tengah Mahasiswa
Nama Belakang	Varchar	15			Nama Belakang Mahasiswa
Alamat Mahasiswa	Varchar	50	Jl, kel, kec, kota/kab		Alamat Mahasiswa

Tabel KAP

Atribut	Type	Lebar	Format	Key	Keterangan
No Mahasiswa	Varchar	10		Foreign	Nomor Mahasiswa
Nama Mahasiswa	Varchar	30		Foreign	Nama Mahasiswa
Alamat Mahasiswa	Varchar	50		Foreign	Alamat Mahasiswa

Tabel Buku

Atribut	Type	Lebar	Format	Key	Keterangan
No Buku	Varchar	10		Primary	Nomor Buku
Judul	Varchar	30			Judul Buku
Pengarang	Varchar	30		Multi	Nama Pengarang Buku
Penerbit	Varchar	30			Nama Penerbit Buku

Tahun terbit	year		YYYY		Tahun diterbitkan buku
--------------	------	--	------	--	------------------------------

Kesimpulan

Ada dua pendekatan perancangan skema konseptual :

- **Terpusat**

Kebutuhan-kebutuhan dari aplikasi atau kelompok-kelompok pemakai yang berbeda digabungkan menjadi satu set kebutuhan pemakai kemudian dirancang menjadi satu skema konseptual.

- **Integrasi**

view-view yang ada Untuk masing-masing aplikasi atau kelompok-kelompok pemakai yang berbeda dirancang sebuah skema eksternal (view) kemudian view – view tersebut disatukan ke dalam sebuah skema konseptual.

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Studi Kasus Perancangan Logical

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
13

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat membuat rancangan basisdata secara logical dari informasi yang diberikan

Pembahasan

Perancangan database secara logika (Transformasi model data)

Transformasi dari skema konseptual dan eksternal (Tahap 2) ke model data sistem manajemen database yang terpilih, ada dua proses yaitu :

- Transformasi yang tidak tergantung pada sistem, pada tahap ini transformasi tidak mempertimbangkan karakteristik yang spesifik atau hal– hal khusus yang akan diaplikasikan pada sistem manajemen database
- Penyesuaian skema ke sistem manajemen database yang spesifik, di lakukan suatu penyesuaian skema yang dihasilkan dari tahap 1 untuk dikonfirmasi pada bentuk implementasi yang spesifik dari suatu model data seperti yang digunakan oleh sistem manajemen database yang terpilih

Hasil dari tahap ini dituliskan dengan perintah DDL ke dalam bahasa sistem manajemen database terpilih. Tapi jika perintah DDL tersebut termasuk dalam parameter–parameter perancangan fisik , maka perintah DDL yang lengkap harus menunggu sampai tahap perancangan database secara fisik telah lengkap

Logical data model merupakan pemodelan dari proses bisnis yang berfokus pada analisis data. *Logical data model* dibangun oleh tiga notasi yaitu entiti, atribut dan relasi. Entiti adalah tempat, obyek, kejadian maupun konsep pada lingkungan *user* dimana diperlukan *maintain* data pada organisasi tersebut. Atribut adalah karakteristik yang dimiliki tiap entiti. Relasi adalah hubungan asosiasi data antar entiti.

Beberapa hal yang perlu diperhatikan dalam pembuatan *logical data model* menurut Moss Larissa :

- a. Memeriksa definisi, semantik dan tipe data pada tiap entiti untuk mencari duplikasi obyek bisniskarena dapat tidak terlihat apabila nama yang digunakan berbeda.
- b. Memastikan tiap data pada entiti bahwa hanya memiliki satu pengenal yang unik (*primary key*), dimana termasuk apabila ada data lama yang dihapus dari *database*.

- c. Menggunakan aturan normalisasi untuk memastikan bahwa sebuah atribut hanya dimiliki oleh satu entiti saja.
- d. Mengadopsi aturan bisnis dengan obyek pada dunia nyata. Aturan bisnis ini memperlihatkan relasi data antar entiti.

Beberapa hal yang perlu diperhatikan dalam pembuatan *logical data model* menurut Elmasri Ramez :

- a. Semantic atribut

Bagaimana menggambarkan relasi yang dapat menggambarkan fakta yang ada.

- a. Memperkecil terjadinya data redundansi

Tujuan dalam pembuatan *database* adalah mengoptimalkan penyimpanan data.

- a. Memperkecil terjadinya nilai null pada data

Null value dapat menyebabkan penyimpanan data yang besar dan dapat terjadi kesalahpahaman dalam mengartikan suatu atribut. Null value dapat diinterpretasikan sebagai :

(1) atribut ini tidak dimiliki oleh data tersebut,

(2) nilai atribut tidak diketahui dan

(3) nilai atribut diketahui tetapi belum dicatat.

Tiap entiti memiliki definisi/semantik yang jelas.

PENERAPAN *LOGICAL DATA MODEL* DALAM CONTOH KASUS

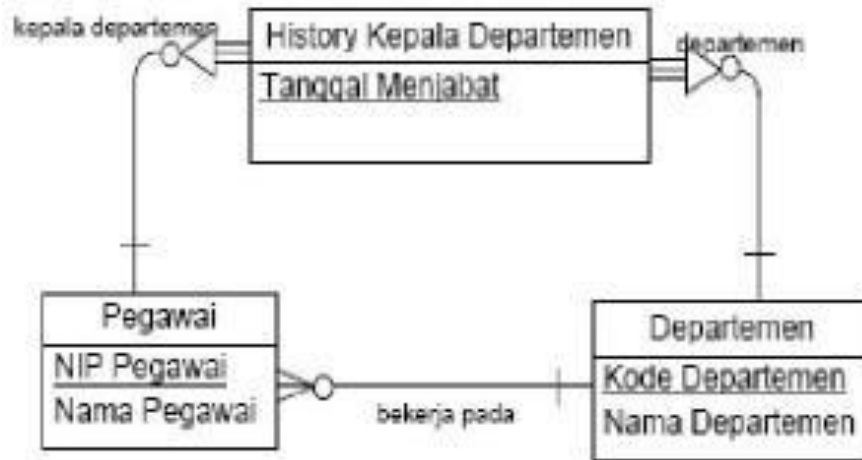
Contoh kasus 1

Gambar 1 menunjukkan relasi kepala departemen antara entiti Pegawai dengan Departemen adalah 1 : 1 yang berarti satu orang pegawai hanya dapat mengepalai satu departemen dan satu departemen hanya boleh dikepalai oleh satu orang pegawai. Dilihat dari kenyataan yang terjadi, relasi tersebut adalah benar karena tidak mungkin pada satu waktu, ada lebih dari satu pegawai yang mengepalai suatu departemen dan begitu pula sebaliknya.



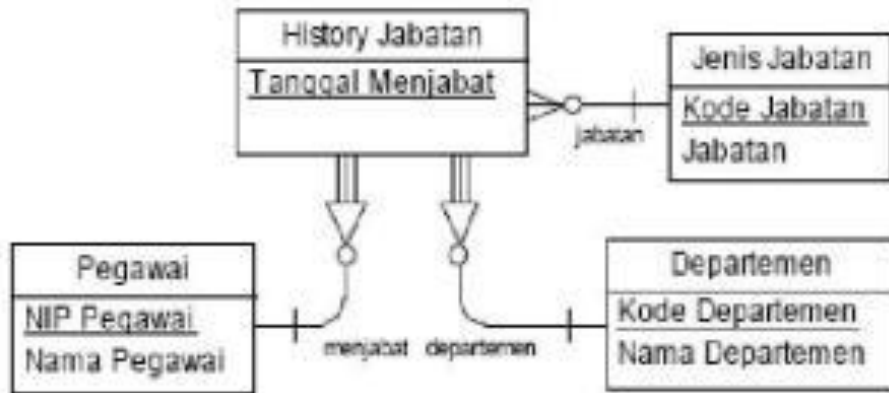
Gambar 1. Relasi entiti Pegawai dan entitiDepartemen

Namun ternyata ketika terjadi pergantian kepala departemen, data kepala departemen yang lama sudah tidak dapat lagi diketahui. Dengan kata lain, *database* tidak menyediakan penyimpanan data masa lampau. Oleh karena itu, desain gambar 1 ditambahkan suatu entiti yang mencatat tanggal seorang pegawai menjabat suatu departemen, sehingga dapat ditelusuri siapa saja yang pernah menjabat menjadi kepala departemen suatu departemen (Gambar 2)



Gambar 2. Penambahan entiti History Kepala Departemen

Apabila ruang lingkup ditambah dengan pencatatan setiap jabatan yang dipegang selama seorang pegawai, maka gambar 2 harus diubah lagi dengan memasukkan informasi pilihan jabatan yang mungkin dijabat seorang pegawai selain kepala departemen. Sebagai seorang pegawai pasti mempunyai sebuah jabatan, sehingga dari entiti history jabatan dapat diketahui departemen yang saat itu menaunginya. Oleh karena itu, relasi antara entiti Pegawai dengan entiti Departemen dapat dihilangkan.



Gambar 3. Perubahan entiti History Jabatan

Moss poin (d) dan Elmasri poin (a) menyatakan bahwa pembuatan model harus disesuaikan dengan fakta. Contoh kasus 1 memperlihatkan bahwa dalam melakukan desain perlu memastikan data apa saja yang dibutuhkan baik sekarang maupun yang akan datang dapat tersedia. Pada gambar 1 terdapat permasalahan yaitu tidak menyediakan penyimpanan data masa lampau. Kemudian yang kedua adalah apakah *database* dapat memenuhi kebutuhan sesuai dengan pertumbuhan *user* sebagaimana yang digambarkan pada gambar 3 yaitu bahwa jenis jabatan dapat semakin beragam, oleh karena itu dibuat sebuah entiti tersendiri.

KESIMPULAN

Berdasarkan dari pembahasan sebelumnya, maka dapat diambil kesimpulan yaitu :

1. Kesalahan yang sering terjadi dalam melakukan desain adalah
 - a. tidak mempersiapkan desain yang dapat digunakan pada perkembangan system di masa yang akan datang,
 - b. pembuatan relasi yang salah,
 - c. pembuatan relasi yang redundansi,
 - d. adanya redundansi data,
 - e. pemilihan primary key untuk suatu tabel dan
 - f. pemilihan tipe data.
2. Dalam mendesain *logical data model* harus memperhatikan *database* yang akan digunakan, proses bisnis pada sistem dan mempersiapkan desain yang dapat digunakan pada perkembangan sistem di masa yang akan datang.

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Studi Kasus Perancangan Fisikal

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
14

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat membuat rancangan basisdata secara fisikal dari informasi yang diberikan

Pembahasan

Sebuah apotik di Jalan Pemuda 83 Semarang – “OBATKU” akan mengimplementasikan aplikasi basis data pada apotiknya. Sebagai apotik yang melayani langsung pelanggan obat, OBATKU melakukan pencatatan terhadap beberapa hal, yaitu :

1. Data Karyawan
2. Data Obat
3. Data Supplier

Setiap pelanggan dapat membeli lebih dari satu jenis obat. Didalam satu transaksi penjualan, seorang pelanggan hanya dilayani oleh seorang karyawan dan seorang karyawan setiap harinya dapat melayani lebih dari satu orang pelanggan dalam beberapa transaksi penjualan.

Data pelanggan secara lengkap seperti ditunjukkan dalam kartu pelanggan yaitu terdiri ID pelanggan, nama, alamat, jenis kelamin, pekerjaan

Data karyawan secara lengkap seperti ditunjukkan dalam kartu tanda pengenal karyawan yaitu terdiri ID karyawan, nama, alamat, kota, status, no tlp.

Data obat secara lengkap seperti ditunjukkan dalam kartu identifikasi obat yaitu terdiri ID obat, nama, jenis, harga, stock, ID supplier.

Sedangkan data supplier seperti ditunjukkan dalam kartu supplier yaitu terdiri ID supplier, nama, alamat, kota, no tlp.

Faktur penjualan yang dipergunakan dalam proses penjualan obat berisi No, tanggal, ID pelanggan, ID karyawan, ID obat, jumlah, total, pajak, total bayar.

Faktur penyuplaian obat yang dipergunakan dalam proses pembelian obat dari supplier obat berisi No, tanggal, ID karyawan, ID supplier, ID obat, jumlah obat, total, pajak, totalbayar.

Di dalam operasionalnya, pemilik OBATKU menginginkan adanya laporan penjualan untuk tiap periode waktu tertentu (misal harian, mingguan atau bulanan).

Solusi :

1. Tabel Data Pelanggan

Pelanggan		
PK	ID_Pelanggan	Varchar (10)
	Nama	Varchar (100)
	Alamat	Varchar (150)
	Jenis_Kelamin	Varchar (2)
	Pekerjaan	Varchar (30)

2. Tabel Data Karyawan

Karyawan		
PK	ID_Karyawan	Varchar (10)
	Nama	Varchar (100)
	Alamat	Varchar (150)
	Kota	Varchar (30)
	Status	Varchar (50)
	No_Tlp	Varchar (15)

3. Tabel Data Obat

Obat		
PK	ID_Obat	Varchar (10)
	Nama	Varchar (100)
	Jenis	Varchar (60)
	Harga	Int
	Stock	Int
	ID_Supplier	Varchar (15)

4. Tabel Data Supplier

Supplier		
PK	ID_Supplier	Varchar (10)
	Nama	Varchar (100)
	Alamat	Varchar (150)
	Kota	Varchar (30)
	No_Tlp	Varchar (15)

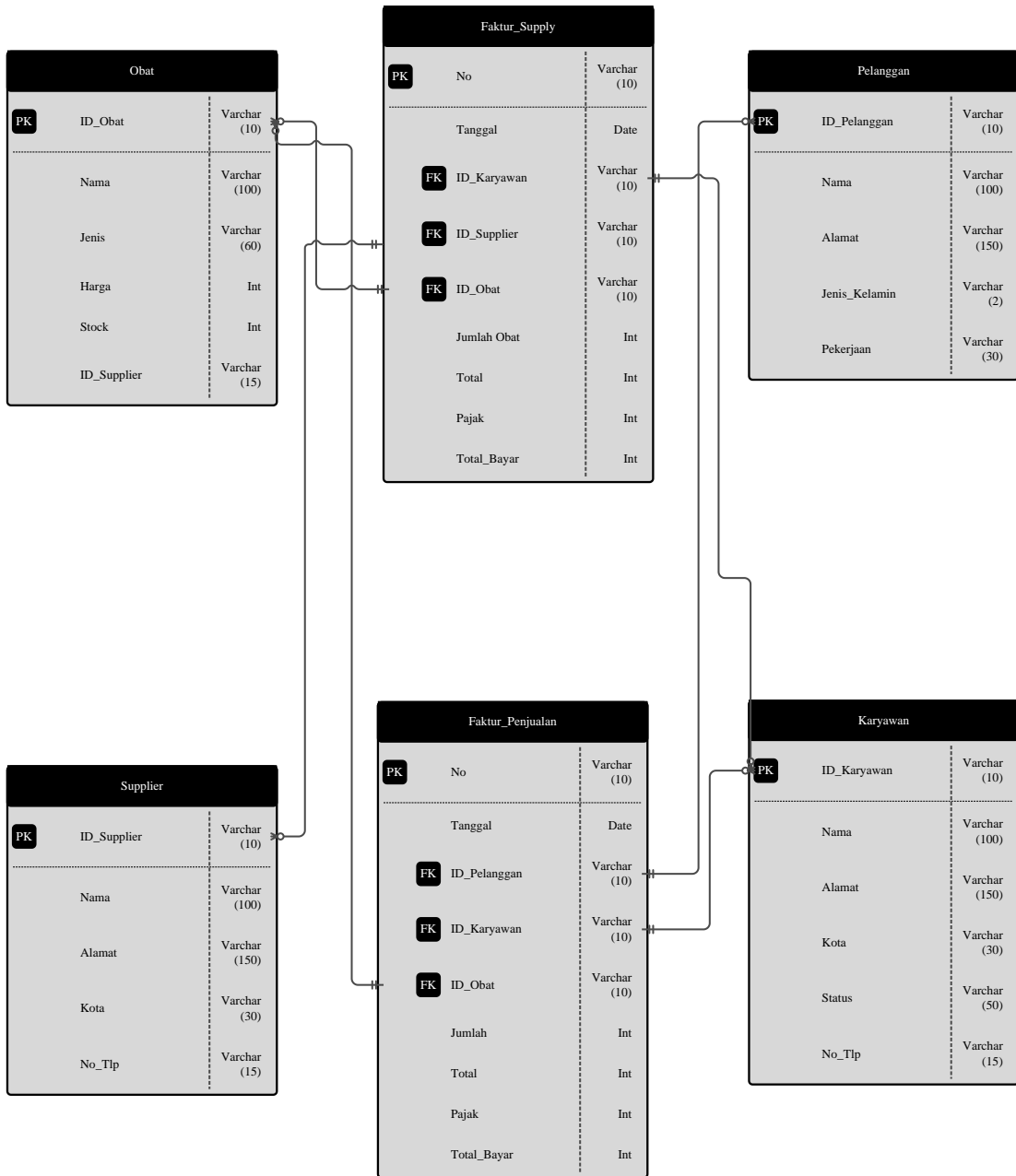
5. Tabel Faktur Penjualan

Faktur_Penjualan		
PK	No	Varchar (10)
	Tanggal	Date
FK	ID_Pelanggan	Varchar (10)
FK	ID_Karyawan	Varchar (10)
FK	ID_Obat	Varchar (10)
	Jumlah	Int
	Total	Int
	Pajak	Int
	Total_Bayar	Int

6. Tabel Faktur Supply

Faktur_Supply		
PK	No	Varchar (10)
	Tanggal	Date
FK	ID_Karyawan	Varchar (10)
FK	ID_Supplier	Varchar (10)
FK	ID_Obat	Varchar (10)
	Jumlah Obat	Int
	Total	Int
	Pajak	Int
	Total_Bayar	Int

Physical Database Design



Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.



MODUL PERKULIAHAN

Perancangan Basis Data

Presentasi Kelompok

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
15

Kode MK
18033

Disusun Oleh
Tim Dosen

- Kompetensi Mahasiswa dapat mempresentasikan hasil pekerjaan kelompok dengan menggunakan teknik-teknik presentasi yang tepat

Pembahasan

Pada pertemuan 15 seluruh mahasiswa di tugaskan untuk presentasi kelompok. Tema dari masing-masing kelompok pun berbeda- beda agar mereka bisa saling berbagi dan belajar dari kelompok lainnya. Tugasnya yaitu membuat studi kasus pada perancangan tertentu dan membuat solusi dari permasalahan tersebut dengan menggunakan perancangan sesuai tema yang di dapat.

Dibawah ini adalah pembagian kelompok beserta tema yang akan di presentasikan oleh kelompok tersebut.

No	Nama Kelompok	Materi
1	M. Rizaldi A. Syarifuddin Fitri Anggri Laela Shaumi	Studi Kasus Perancangan Konseptual
2	M. Abrar Haki Uri Umam	Studi Kasus Perancangan Konseptual
3	Budiman Santoso	Studi Kasus Perancangan Konseptual
4	Herman Yoseph Dedy Syah Putra Indah Leswitriani Dimas Fahrulsyah	Studi Kasus perancangan logical
5	Anjar Rochana Herlisa Dwi Wily H Ahmad Uais	Studi Kasus perancangan logical
6	Bimo Fikri Eka Prasetyawati Fryda Farizha Reynaldi Jatusaputra	Studi Kasus perancangan logical
7	Fitri Dwi P Ayudina Nur Anindyka Elras Dani Nurahman	Studi Kasus perancangan fisik
8	Agus Riandi Himawan M. Insan	Studi Kasus perancangan fisik

	Rofi'I Nofriyah	
	M. Fadhlul	
	Sony Darmawan	
	Devi	
9	Ami	Studi Kasus perancangan fisikal

Daftar Pustaka

Connolly, Thomas., Begg, Carolyn (2005). Database System: A practical Approach to Design, Implementation and management, 4th Ed. Pearson Education, England.