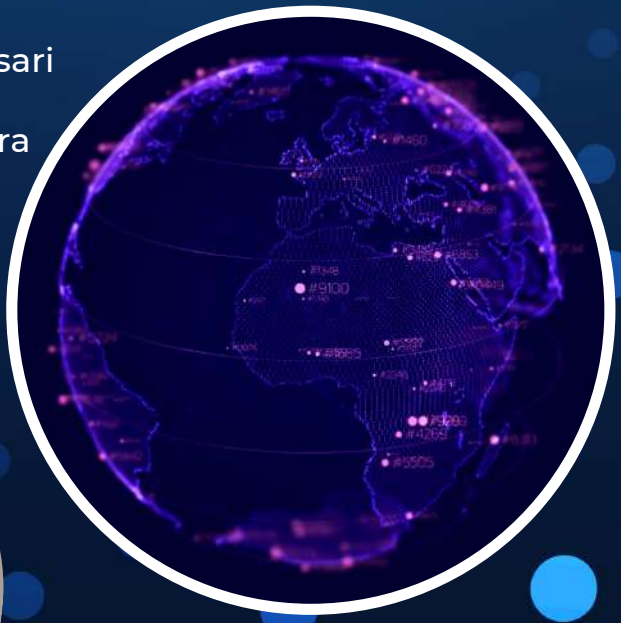
ISBN 978-623-8004-53-9

SISTEM BASIS DATA

Penulis :

- Jamaludin
- Khairunnisa samosir
- Wahyuddin S
- Elmi Devia
- Leo Willyanto Santoso
- Yuniansyah
- Junaidi
- Sri Rezeki Candra Nursari
- Noor Azizah
- Muhamad Hadi Saputra



SISTEM BASIS DATA

JAMALUDIN
KHAIRUNNISA SAMOSIR
WAHYUDDIN S
ELMI DEVIA
LEO WILLYANTO SANTOSO
YUNIANSYAH
JUNAIDI
SRI REZEKI CANDRA NURSARI
NOOR AZIZAH
MUHAMAD HADI SAPUTRA

PT GLOBAL EKSEKUTIF TEKNOLOGI

Sistem Basis Data

Penulis:
Jamaludin
Khairunnisa samosir
Wahyuddin S
Elmi Devia
Leo Willyanto Santoso
Yuniansyah
Junaidi
Sri Rezeki Candra Nursari
Noor Azizah
Muhamad Hadi Saputra

ISBN: 978-623-8004-53-9

Editor: Mila Sari, M.Si.
Penyunting : Yuliatri Novita, M.Hum.
Desain Sampul dan Tata Letak : Handri Maika Saputra, S.ST.

Penerbit : PT GLOBAL EKSEKUTIF TEKNOLOGI
Anggota IKAPI No. 033/SBA/2022

Redaksi :
Jl. Pasir Sebelah No. 30 RT 002 RW 001
Kelurahan Pasie Nan Tigo Kecamatan Koto Tangah
Padang Sumatera Barat
Website : www.globaleksekutifteknologi.co.id
Email : globaleksekutifteknologi@gmail.com

Cetakan pertama, Oktober 2022

KATA PENGANTAR

Alhamdulillah, segala puji selalu Kami panjatkan kepada Allah SWT atas ridho-Nya sehingga penulis mampu menyelesaikan buku berjudul "Sistem Basis Data". Keberhasilan buku ini tentu tidak akan terwujud tanpa adanya dukungan dan bantuan dari berbagai pihak. Ucapan terima kasih penulis sampaikan kepada berbagai pihak yang selalu mendukung dan memberikan do'a terbaik dalam penerbitan buku ini.

Buku ini tidak luput dari kekurangan dan kesalahan. Jika pembaca menemukan kesalahan apapun, penulis mohon maaf setulusnya. Selalu ada kesempatan untuk memperbaiki setiap kesalahan, dukungan berupa kritik & saran akan selalu penulis terima dengan tangan terbuka.

Padang, Oktober 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 1
KONSEP BASIS DATA

Oleh Jamaludin

1.1 Pendahuluan

Informasi merupakan data yang telah diolah sehingga berguna bagi penerimanya. Pemahaman mengenai sistem pengolahan data yang didukung dengan komputer perlu dipahami sehingga pemanfaatan informasi di dalam sebuah organisasi bisa lebih efektif dan efisien. Perangkat lunak membutuhkan ruang untuk menyimpan dan memanfaatkan data tersebut oleh pengguna komputer. Untuk itu diperlukan dibutuhkan software yang digunakan untuk mengelola dan dibisa dipanggi kembali dari basis data tersebut saat dibutuhkan. Softwaretersebut dikenal dengan nama *Data Base Management System, (DBMS*) atau sistem manajemen basis data.

Penerapan basis data banyak ditemukan di sekitar kita. Seperti penerapan teknologi informasi yang memanfaatkan basis data adalah penggunaan ATM (Anjungan Tunai Mandiri), dimana kita bisa mengambil uang di mana saja dan kapan saja. Di dalam basis data tersebut juga tersimpan data seperti nomor rekening, user dan password serta saldo tabungan. Penerapan basis data juga ditemukan pada perpustakaan, penjualan di swalayan, perhotelan dan lain sebagainya. Sebagai gambaran, dengan aplikasi web took buku kita bisa melihat buku-buku yang kita inginkan hanya memasukkan kata kunci tertentu. Kemudian aplikasi tersebut akan menyesuaikan dengan basis data yang ada, kemudian bisa

menampilkan judul-judul buku beserta atribut lainnya seperti ISBN, nama pengarang, tahun, kota ke layar computer (Kadir, 2002).

1.2 Definisi Basis Data

Basis data atau *database*, berasal dari kata yaitu basis dan data. Basis dapat diartikan sebagai markas atau gudang, tempat berkumpul, data merupakan wujud dari dunia nyata yang mewakili suatu objek seperti manusia (siswa, mahasiswa, pegawai, pelanggan), barang, hewan peristiwa, konsep, keadaan, dan sebagainya yang dinyatakan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya. Sehingga dapat disimpulkan bahwa basis data adalah kumpulan informasi yang disimpan didalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data tersebut disebut sistem manajemen basis data (*Data Base Management System*).(Magdalena, 2020)

Basis data adalah kumpulan file-file yang saling berhubunga yang biasa ditunjukan dengan kunci dari tiap file yang ada. Satu basis data menunjukkan kumpulan data yang dipakai dalam satu lingkup informasi. Dalam satu file terdapat record-record yang sejenis, sama besar, sama bentuk, merupakan satu kumpulan entitas yang seragam. Satu record terdiri dari field-field yang saling berhubungan untuk menunjukan bahwa field tersebut dalam satu pengertian yang lengkap dan direkam dalam satu record. Suatu sistem manajemen basis data berisi satu koleksi data yang saling berhubungan dan satu set program untuk mengakses data tersebut. Jadi sistem manajemen basis data dan set program pengelola yang berfungsi untuk membaca data, menambah data, menghapus data dan mengambil data (Fathansyah, 2009).

1.3 Tujuan dan Manfaat Basis Data

Prinsip kerja dan tujuan basis data sama dengan prinsip kerja dan tujuan dari lemari arsip. Prinsip utamanya adalah pengaturan data atau arsip, dan tujuan utamanya adalah kemudahan dan kecepatan dalam pengambilan kembali dari data atau arsip tersebut. Tujuan utama dari basis data adalah untuk mengatur data sehingga diperoleh kemudahan, kecepatan dan ketepatan dalam memanggil kembali data yang diinginkan.(Simarmata & Paryudi, 2011)

Manfaat dari basis data bagi penggunanya adalah (Mustofa, 2022)

1. Kemudahan dan kecepatan
   Sistem basis data memberikan sebuah kemudahan untuk memilih data menjadi satu kelompok secara berurutan dengan cepat. Sistem tersebut juga dapat menghasilkan pencarian data yang diperlukan bisa ditemukan secara cepat.
2. Multi-user
   Basis data memberikan kemudahan akses bagi satu atau beberapa pengguna dalam waktu yang bersamaan. Sehingga kinerja perangkat dan jaringan dapat mempermudah melalui *multi-user* sebab penyimpanan hanya memuat satu unit yang bisa diakses secara bersamaan.
3. Keamanan data
   Keamanan data merupakan hal yang penting bagi layanan sistem basis data. Sistem basis data akan mengamankan data yang ada dalam system tersebut. Melalui perangkat yang diberi password, menyebabkan hanya pihak yang diijinkan untuk mengakses data tersebut.
4. Penghematan biaya perangkat

Satu basis data terpusat bagi para perusahaan besar untuk menghemat biaya perangkat komputer. Hal itu menjadikan perusahaan tidak lagi membutuhkan hard disk di setiap computer pada tempat yang berbeda. Melalui koneksi jaringan internet, cabang perusahaan di daerah terpencil pun dapat melakukan akses data yang ada pada pusat.

5. Kontrol data terpusat
   Satu server terpusat untuk melakukan penyimpanan data, memudahkan banyak pengguna di cabang untuk mengakses data tersebut. Sebagai contoh adalah kantor perusahaan tidak perlu membuat suatu data di tiap divisinya. Setiap divisi dapat mengumpulkan data khusus melalui satu server yang telah ditentukan sehingga mempermuda atas untuk melihat laporan.

Menurut Fathansyah (2009), manfaat basis data banyak digunakan untuk mengatasi masalah-masalah berikut :

1) Data yang tidak stabil (*redudansi*)
   Terjadinya beberapa bagian data mengalami penggandaan pada file-file yang berbeda, karena waktu yang cukup panjang. Penyimpanan data yang berulang-ulang di beberapa file juga dapat mengakibatkan inkonsistensi (tidak konsisten).
2) Kesulitan pengaksesan data
   Timbulnya kesulitan untuk mengakses data secara bersama. Solusi untuk mengatasi permasalahan tersebut dengan menggunakan sistem manajemen basis data yang mengambil data secara langsung dengan bahasa yang mudah digunakan.
3) Isolasi data untuk standarisasi
   Sulitnya dalam menulis program aplikasi untuk mengambil dan menyimpan data, jika data terdistribusi dalam beberapa file dalam bentuk format yang berbeda, maka diperlukan

standarisasi data dalam satu format yang sama sehingga mudah membuat program aplikasinya.

4) Masalah keamanan (*security)*
   Keamanan data dapat diatur melalui program yang dibuat oleh pemrogramer dengan memanfaatkan fasilitas keamanan dari system operasi. Dengan system basis data maka pembagian hak akses sesuai dengan tingkatannya. Misalnya data mengenai gaji pegawai hanya boleh dibuka oleh bagian keuangan dan personalia, dan tidak boleh diakses oleh departemen lain.
5) Masalah integrasi (kesatuan)
   Manajemen basis data berisi file data yang saling berelasi, masalah utama adalah bagaimana hubungan antara file tersebut terjadi. Meskipun diketahui bahwa file B berhubungan dengan file A, namun secara teknis maka ada file kunci yang menghubungkan kedua file tersebut.
6) Masalah data independence (kebebasan data)
   Perintah-perintah dalam sistem manajemen basis data mudah digunakan, seperti ketika ingin melihat data cukuplah dengan perintah USE, hendak menambah data cukup dengan APPEND, ini berarti perubahan apapun dalam basis data, semua perintah akan mengalami kestabilan tanpa perlu ada yang diubah.

1.4 Jenis dan Model Basis Data

Menurut Zaenal Mustofa (2022), jenis-jenis basis data beserta fungsinya bisa dibedakan sebagai berikut :

1. *Operational database*
   Operational database memiliki fungsi sebagai suatu tempat untuk mengelola data dinamis secara langsung dan real-time. Hal ini menyebabkan para penggunanya bisa melihat,

memanggil, dan merubah data, dengan cara menambah atau mengubah, ataupun menghapus data secara langsung.
Contoh : JSON (*JavaScript Object Notation*), XML (*Extensible Markup Language*)

2. *Database warehouse*
   Database warehouse yaitu repository sentral data yang terpadu yang berasal dari satu atau lebih dari satu sumber yang berbeda. Database tersebut juga mempunyai tempat untuk menyimpan data terbaru. Serta history satu tempat yang telah dipakai untuk membuat sebuah laporan analisis.
   Contoh : Microsoft SQL Server
3. *Distributed database*
   Distributed database merupakan suatu basis data dengan perangkat penyimpanannya yang terpasang pada sebuah perangkat komputer yang berbeda. Komunikasi sistem ini terdistribusi melalui suatu situs yang tergabung dan tidak mempunyai sebuah komponen fisik.
   Contoh : Microsoft Access (Office)
4. *Relational database*
   Relational database atau basis data relasional merupakan basis data yang mengorganisir berdasarkan pada model data relasi. Banyak sekali software yang mengatur dan memelihara basis data melalui hubungan setiap data, seperti: My SQL, PostgreSQL, MariaDB, MongoDB, Oracle Database, SAP HANA, IBM Db2, MemSQL, Interbase, Firebird.

Menurut Setiyowati dan Sri Siswanti (2008), model basis data adalah suatu data yang terintegrasi dalam menggambarkan hubungan (*relationships*) antar data dan batasan-batasan (*constraint*) data dalam suatu sistem basis data. Model data yang

paling umum, berdasarkan pada bagaimana hubungan antar record dalam database (*Record Based Data Models*), terdapat tiga jenis, yaitu:

a. Model basis data hirarki (*hierarchical database model*)
b. Model basis data jaringan (*network database model*)
c. Model basis data relasi (*relational database model*)

Model basis data yang paling banyak digunakan saat ini adalah model basis data relasi, karena mampu menyelesaikan berbagai permasalahan dalam sistem basis data. Sementara model basis data hirarki dan jaringan merupakan model database yang tidak banyak lagi digunakan saat ini, karena adanya berbagai kelemahan dan hanya cocok untuk struktur hirarki dan jaringan saja.

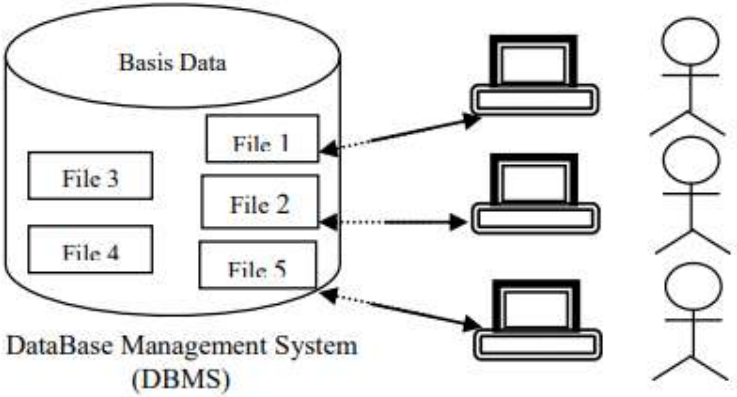
1.5 Sistem Basis Data

Sistem basis data berbeda dengan basis data pada umumnya. Sistem basis data merupakan cakupan yang lebih luas bila dibanding dengan basis data. Sistem basis data memuat sekumpulan basis data dalam suatu sistem yang mungkin tidak ada hubungan satu sama lain, tetapi secara keseluruhan mempunyai relasi sebagai sebuah sistem dengan didukung oleh komponen lainnya (Sutanta, 2004).

Sistem basis data adalah kumpulan dari program aplikasi yang berinteraksi dengan basis data bersama dengan *Database Management System* (DBMS) dan basis data itu sendiri (Connolly & Begg, 2010). Menurut Abdul Kadir (2002), DBMS merupakan perangkat lunak yang dirancang khusus untuk memudahkan pengelolaan basis data. Salah satu jenis DBMS yang populer dewasa ini digunakan RDBMS *(Relational Data Base Management System)*

yang menggunakan model basis data relasional atau dalam bentuk tabel-tabel yang saling berelasi.

Menurut Tatyantoro Andrasto (2013), sebuah sistem basis data merupakan sistem yang terdiri atas kumpulan file (tabel) yang saling berelasi dan sekumpulan program DBMS yang memungkinkan beberapa user dan/atau program lain untuk mengakses dan memanipulasi file-file (tabel-tabel) tersebut, seperti terlihat pada gambar 1.1



Gambar 1.1. Sistem Manajemen Basis Data(Andrasto, 2013)

Menurut Setyowati (2008), sistem basis data terdiri dari 4 komponen pokok, yaitu:

1. Basis data (*database*),
   Basis data dengan ciri-ciri:
   a. Data disimpan secara terintegrasi (*integrated*)

Terintegrasi maksudnya adalah data terkumpul dari berbagai macam file yang disusun dengan cara menghilangkan bagian-bagian yang data yang berulang (*redundansi*)

b. Data dapat dipakai secara bersama-sama (*shared*)
   Shared yaitu masing-masing bagian dari basis data dapat diakses oleh user dalam waktu yang bersamaan, untuk aplikasi yang berbeda.

2. Perangkat keras (*hardware*)
   Terdiri dari semua perangkat keras komputer yang digunakan untuk pengelolaan sistem basis data tersebut.
   Adapun perangkat keras yang terdapat dalam sebuah sistem basis data, terdiri dari :
   a. Komputer (satu perangkat untuk sistem *stand-alone* atau beberapa computer untuk sistem jaringan).
   b. Perangkat untuk menyimpan data permanen (*harddisk*).
   c. Perangkat komunikasi (untuk sistem jaringan).

3. Perangkat lunak (*software*)
   Berfungsi sebagai perantara (*interface*) antara user dengan data fisik pada basis data, dapat berupa:
   a. *Database Management System* (DBMS)
   b. Program-program aplikasi

Menurut Suwanto Raharjo (2012), basis data dapat dibuat dan diolah menggunakan program komputer atau perangkat lunak. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) database disebut *Database Management System* (DBMS). Menurut level dari softwarenya, terbagi menjadi dua level software yaitu *High Level*

Software dan *Low Level Software*. Yang termasuk di dalam *High Level Software*, antara lain seperti : MySQL, Microsoft Access, Microsoft SQL Server, Oracle, Sybase, Interbase, XBase, Firebird, , PostgreSQL, dBase III, Paradox, FoxPro, Visual FoxPro, Arago, Force, Recital, dbFast, dbXL, Quicksilver, Clipper, FlagShip, Harbour, Visual dBase, dan Lotus Smart Suite Approach. Sedangkan yang termasuk di dalam *Low Level Software* antara lain Btrieve dan Tsunami Record Manager.(Setyowati, 2008)

4. Pemakai (*user*)
   Adalah pengguna basis data yang berinteraksi secara tidak langsung dengan basis data melalui program aplikasi basis data (DBMS)
   User terbagi menjadi 3 klasifikasi:
   a. *Database Administrator* (DBA), yang membuat basis data dan mengontrol akses ke basis data.
   b. *Programmer*, yang membuat aplikasi basis data yang digunakan oleh DBA dan pemakai akhir.
   c. Pemakai akhir (*end user*), yang melakukan penambahan, penghapusan, pengubahan, dan pengaksesan data

      Ada 3 jenis data pada sistem basis data, yaitu:
      a. Data operasional dari suatu organisasi, berupa data yang disimpan di dalam basis data
      b. Data masukan (input data), data dari luar sistem yang dimasukan melalui peralatan input (keyboard) yang dapat merubah data operasional
      c. Data keluaran (output data), berupa laporan melalui peralatan output sebagai hasil dari dalam sistem yang mengakses data operasional

DAFTAR PUSTAKA

Andrasto, T. (2013). Pengembangan Sistem Database Hasil Penelitian dan Pengabdian Kepada Masyarakat Dosen Unnes. *Jurnal Teknik Elektro*, *5*(2), 64–68.

Connolly, T., & Begg, C. (2010). *Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition*. Pearson Education. https://www.pearson.com/us/higher-education/program/Connolly-Database-Systems-A-Practical-Approach-to-Design-Implementation-and-Management-6th-Edition/PGM116956.html

Fathansyah. (2009). *Buku Teks Komputer Basis Data Edisi Ke-4*. Informatika.

Kadir, A. (2002). *Penuntun praktis belajar database menggunakan microsoft access*. Andi. https://lib.ui.ac.id/detail.jsp?id=26556

Magdalena, T. (2020). Konsep Basis Data. *researchgate*. https://www.researchgate.net/publication/341177989_TUGAS_KONSEP_BASIS_DATA_RELASIONAL

Mustofa, Z. (2022). Apa Itu Database? Jenis, Fungsi Dan Manfaatnya. *Universitas STEKOM*. http://teknik-informatika-s1.stekom.ac.id/informasi/baca/Apa-itu-Database-Jenis-fungsi-dan-manfaatnya/8de094f78c9551d2eee97e371a249bd714dc83c0

Raharjo, S. (2012). Constraint Basis Data Sebagai Fondasi yang Kuat Dalam Pengembangan Sistem Informasi. *Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST)*, 347–352.

Setyowati. (2008). *Perancangan Basis Data*. Penerbit Andi.

Simarmata, J., & Paryudi, I. (2011). *Basis Data*. Andi.

Sutanta, E. (2004). *Sistem basis data*. Graha Ilmu. https://opac.perpusnas.go.id/DetailOpac.aspx?id=167921

BAB 2
PROTOKOL JARINGAN KOMPUTER

Oleh Khairunnisa Samosir

2.1 Pendahuluan

Protokol dalam dunia komputer adalah aturan atau ketentuan agar satu atau lebih device dapat saling berkomunikasi. Sedangkan Protokol Jaringan Komputer adalah aturan agar device satu dengan device yang lain dapat saling berkomunikasi sesuai system jaringan komputer yang ada. Macam protokol jaringan komputer yang sering kita jumpai adalah IPv4 dan DHCP, serta dalam komunikasi internet kita bertemu dengan bermacam-macam protokol, semisal HTTP dan POP3, tapi sebenarnya masih banyak protokol lainnya, mari kita pelajari lebih lanjut.

Protokol jaringan adalah sebuah aturan yang sudah ditetapkan untuk dapat memudahkan transmisi data antarperangkat yang berbeda-beda di dalam satu jaringan yang sama. Pada dasarnya, protokol jaringan akan memudahkan semua perangkat dapat berkomunikasi terlepas dari perbedaan internal yang dimiliki, struktur, dan desainnya. Dan protokol jaringan merupakan alasan kuat mengapa Anda dapat berkomunikasi bahkan dengan orang-orang di seluruh dunia.

Selain itu, bukan hanya spesialis IT saja yang bisa menggunakan protokol jaringan, semua orang yang mengakses internet setiap saat juga pasti menggunakannya tanpa mereka sadari. Tanpa adanya protokol jaringan, LAN maupun WAN tidak akan dapat berfungsi. Bisa disimpulkan bahwa pada era teknologi

seperti saat ini, protokol jaringan memegang peran utama dalam berkomunikasi secara digital.

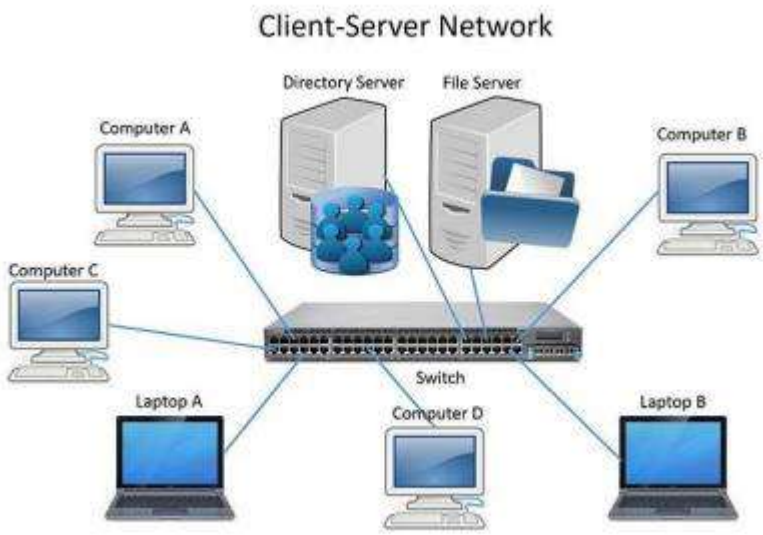
2.2 Uraian Materi

2.2.1 Mendefenisikan Jaringan Komputer

Jaringan Komputer dalam arti luas adalah sekumpulan dari bebebrapa komputer yang tersambung dan saling terhubung sehingga dapat saling berbagi informasi dan berkomunikasi antara satu perangkat dengan perangkat lainnya. Banyak manfaat dari jaringan komputer pada aktivitas individu contohnya seperti berkomunikasi menggunakan video, pesan instan, email serta Berbagi perangkat seperti pencetak, pemindai, dan mesin fotokopi, berbagi berkas, Berbagi perangkat lunak dan program operasi pada sistem jarak jauh, Memperbolehkan pengguna jaringan mengakses dan memelihara informasi dengan mudah, dan masih banyak lagi.

Zymon Machajewski menuturkan bahwa jaringan komputer adalah perangkat komputer atau device yang telah terkoneksi bersama antara komputer satu dengan beberapa komputer yang lain yang bertujuan untuk dapat berbagi sumber daya. Salah satu sumber daya yang pada saat ini yang sering kali diterapkan pada jaringan komputer adalah internet. Jaringan komputer dapat memiliki desain yang berbeda, diantaranya adalah jaringan peer to peer dan client server.

Jaringan client server memiliki server induk untuk penyimpanan data untuk dapat diakses oleh device klien yang terkoneksi pada satu netowork atau jaringan. Lalu Jaringan peer to peer cenderung memiliki perangkat yang mendukung fungsi yang sama yang paling banyak dimanfaatkan di rumah, sementara jaringan klien server lebih banyak digunakan untuk melakukan suatu bisnis.



Gambar 2.1 Jaringan Clien Server

Komputer-komputer yang terkoneksi pada sebuah jaringan tidak mesti memiliki jenis yang sama. Hal ini tentu beda dari pengkonsepan jaringan di bidang ilmu kehidupan, di mana sekumpulan sel membentuk sebuah jaringan harus memiliki fungsi dengan jenis yang sama. Dalam jaringan komputer, komputer yang terhubung bisa memiliki tipe, sistem operasi program atau aplikasi yang berbeda-beda antara satu dengan lainnya, tetapi antara komputer tersebut harus menggunakan aturan komunikasi atau yang sering kita sebut dengan protokol yang sama, tujuan dari penggunaan protokol yang sama ialah agar perangkat komputer mampu bertukar data maupun informasi antara satu komputer dengan perangkat komputer lainnya, saat ini standar protokol internasional adalah *Transmission Control Protocol* atau sering

disebut TCP/IP. Protokol adalah sekelompok peraturan untuk mengatur komunikasi data di internet.

Jenis komputer yang berbeda ataupun perbedaan pada operation system pada jaringan komputer tidak jadi masalah karena setiap komputer tersebut di hubungkan ke internet menggunakan TCP/IP sehingga memiliki bahasa yang sama dalam fungsinya. Sehingga semua komputer yang memakai protokol TCP/IP yang terkoneksi ke jaringan internet dapat berhubungan dengan komputer mana pun selama komputer tersebut terhubung denganinternet.

Gambar 2.2 Jaringan Komputer

2.2.2 Mendeskripsikan manfaat dan tujuan dari jaringan komputer

Banyak tujuan yang dapat diterapkan pada jaringan komputer, hal tersebut sangat bermanfaat di era digital pada saat ini, beberapa tujuan serta manfaat dari jaringan komputer diantaranya:

1. Saling berbagi sumber daya dan peralatan
   Hal ini dapat kita lihat di mana komputer-komputer yang terkoneksi pada sebuah jaringan komputer mempunyai keunggulan untuk saling memanfaatkan alat-alat seperti printer dan aplikasi bersama, seperti aplikasi toko online, database serta sistem informasi secara bersamaan hal ini tentunya sangat menguntungkan karena dapat meningkatkan efektivitas suatu peralatan dan penghematan biayayang dikeluarkan.
2. Integrasi data
   Dengan adanya jaringan komputer, mengintegrasikan data-data komputer satu ke komputer yang lainnya yang tersambung pada suatu jaringan sangatlah dimungkinkan.
3. Komunikasi
   Dengan jaringan komputer kita dapat melakukan komunikasi terhadap user komputer lainya melalui berbagai cara, seperti e-mail , video call dan lain-lain. Jelas hal ini sangat kita butuhkan pada saat ini di mana kita memerlukan pertukaran informasi yang cepat.
4. Pemeliharaan dan Pengembangan
   Pemeliharaan dan Pengembangan pada alat-alat dapat kita lakukan lebih mudah dan dengan anggaran yang lebih hemat karena teknologi jaringan komputer.
   Contohnya, pengguna tidak harus mempunyai banyak printer untuk masingmasing komputer yang kita miliki supaya dapat melakukan pencetakan dari semua komputer

yang kita miliki. Kita hanya perlu memiliki 1 buah printer dan kita dapat membuat sejumlah komputer menggunakan printer tersebut secara bersama-sama.

5. Keamanan (*Security*)
   Dalam hal keamanan, jaringan komputer memudahkan kita terutama soal proses memberikan perlindungan terhadap data. Walaupun pada dasarnya dalam sebuah jaringan suatu komputer bisa mengakses komputer yang lain, kita juga bisa membatasi akses orang lain terhadap data data tadi sesuai keinginan kita. Kita pun dapat melakukan pengamanan secara terpusat pada semua komputer yang tersambung pada jaringan sehingga dapat lebih menghemat waktu dan tenaga.
6. Informasi Terkini dan Sumber Daya Lebih Efisien
   Penggunaan suatu sarana secara bersama-sama dapat meningkatkan perkembangan dan efektivitas dari sumber daya tersebut, selain itu kita akan mendapat keuntungan lainya yaitu informasi yang kita akses merupakan informasi yang baru karena saat ada perubahan datas akan dapat tersampaikan oleh setiap orang yang menggunakan sumber daya tersebut.

2.3 Tipe Jaringan Komputer

a) Jaringan Personal Area Network (PAN)

Jaringan Personal Area Network (PAN) adalah sebuah jaringan yang berfungsi untuk berkomunikasi antara komputer dengan perangkat-perangkat lainnya seperti handphone, speaker, printer, scanner dan masih banyak lagi. PAN dikendalikan menggunakan otoritas pribadi, PAN memakai teknologi Wireless Application Protocol (WAP) dan

Bluetooth. Port yang digunakan untuk jaringan PAN antara lain adalah USB dan Firewire.

Gambar 2.3 Jaringan PAN

Teknologi dari Personal Area Network salah satunya adalah Bluetooth yang dikenal pula menggunakan nama piconet, bisa menghubungkan lebih kurang 8 perangkat menggunakan jaringan berbasis klien server. Perangkat Bluetooth yang pertama dianggap sebagai master, sedangkan seluruh perangkat-perangkat lainnya yang terhubung dan berkomunikasi dengan master adalah slave. Jarak jangkauan piconet biasanya hanya 10 meter, tapi jika perangkat diatur sedemikian rupa menjadi scatternet, jangkauannya bisa mencapai hingga 100 meter.

b) Jaringan Wireless Personal Area Network (WPAN)

Jaringan Wireless Personal Area Network (WPAN) merupakan jaringan area pribadi yang eksklusif dan berpusat

di sekitar perangkat yang terhubung menggunakan media nirkabel. Umumnya, jangkauan dari jaringan ini lebih kurang 10 meter.

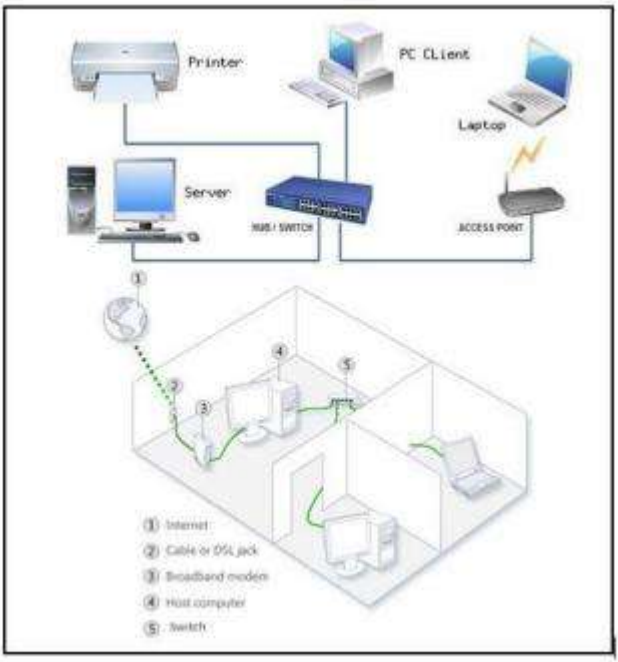
Gambar 2.4 Jaringan WPAN

c) Jaringan Local Area Network (LAN)

Jaringan Local Area Network (LAN) adalah jaringan yang menghubungkan dua perangkat komputer personal atau lebih pada cakupan yang relatif dekat, misalnya dalam satu gedung, sekolah, kantor, warnet, lab, dan lain sebagainya.

Karakteristik jaringan LAN :

- ❖ Memiliki kecepatan data yang semakin tinggi.
- ❖ Mencakup wilayah geografis yang kecil
- ❖ Tidak membutuhkan jalur telekomunikasi yang disewa dari operator Telekomunikasi



Gambar 2.5 Jaringan LAN Antar Ruangan

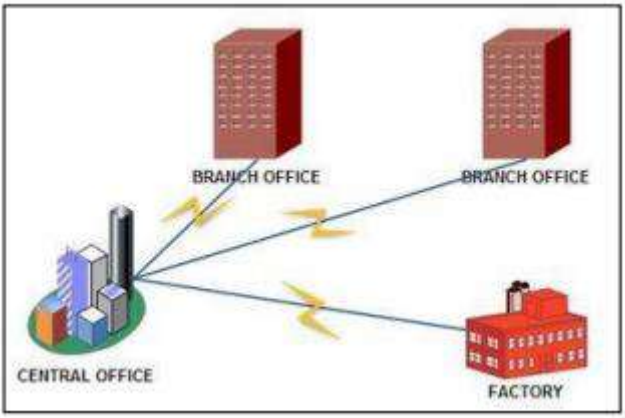
d) Jaringan Metropolitan Area Network (MAN)

Jaringan Metropolitan Area Network (MAN) adalah jaringan yang luasnya meliputi satu kota serta beberapa wilayah di sekitarnya. Contohnya adalah jaringan ponsel (telepon seluler), sistem telepon rumah, dan jaringan relay beberapa Internet Service Provider (ISP).

Karakteristik MAN :

- Mencakup daerah dengan kisaran luas 5 sampai 50 kilometer. Banyak MAN melingkupi daerah sekitar perkotaan.

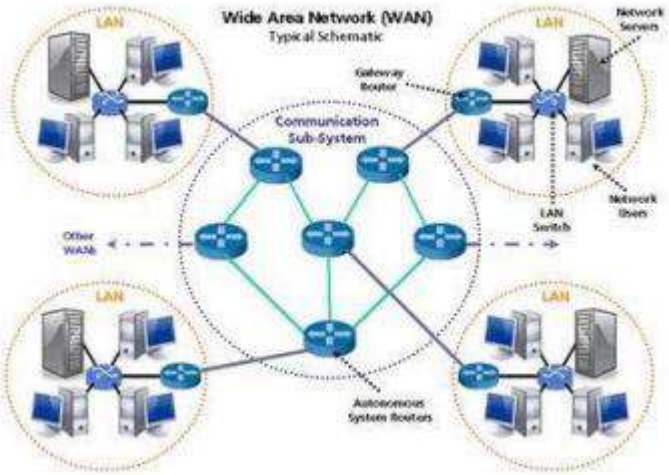
- Seperti halnya WAN, sebuah MAN biasanya tidak dipunyai oleh satu organisasi. MAN, komunikasi link, serta peralatannya, umumnya dimiliki oleh suatu konsorsium pengguna atau bisa juga penyedia layanan jaringan yang menjual pelayanan pada pengguna.
- MAN sering difungsikan sebagai jaringan dengan kecepatan data yang tinggi untuk berbagi sumber daya dalam satu wilayah. Sering kali juga MAN digunakan untuk menyediakan koneksi bersama jaringan lain dengan menggunakan link ke WAN.



Gambar 2.6 Jaringan MAN

e) Jaringan Wide Area Network (WAN)

Jaringan Wide Area Network (WAN) adalah jaringan komputer yang mencakup area yang besar sebagai contoh yaitu jaringan komputer antar wilayah, kota atau bahkan negara, atau dapat didefinisikan juga sebagai jaringan komputer yang membutuhkan router dan saluran komunikasi publik.



Gambar 2. 7 Jaringan WAN

Tabel 2.1 Cakupan Area jaringan Komputer

Jarak (m)	*Network*	Contoh Area
1 - 10	PAN	Ruangan
10 - 1000	LAN	Gedung
10.000 - 100.000	MAN	Kota
100.000 - 1.000.000	WAN	Negara

DAFTAR PUSTAKA

Kurose, J. F., & Ross, K. W. (2017). Computer Networking: A Top-down Approach. Pearson.

Lowe, D. (2018). Networking All-In-One. Hoboken: John Wiley & Sons, Inc.

Sukaridhoto, S. (2014). Buku Jaringan Komputer I. Surabaya: Politeknik Elektronika Negeri Surabaya (PENS).

BAB 3
MODEL DATA RELASIONAL DAN BATASAN BASIS DATA RELASIONAL

Oleh Wahyuddin S

3.1 Pengertian Model Data Relasional

Model Relasional adalah sekumpulan hubungan antar database. Hubungan yang tidak lain adalah tabel nilai. Setiap baris tabel mewakili sekumpulan nilai data terkait. Baris dalam tabel ini mewakili entitas atau hubungan aktual. Nama tabel dan nama kolom sangat membantu untuk menafsirkan arti nilai di setiap baris. Data diwakili sebagai seperangkat relasi. Dalam model relasional, data disimpan sebagai tabel. Namun, penyimpanan fisik data tidak tergantung pada cara data diatur secara logis.

3.2 Konsep Model Relasional

1. Atribut: Setiap kolom dalam Tabel. Atribut adalah properti yang mendefinisikan suatu relasi. misalnya, Student_Rollno, NAME, dll.
2. Tabel: Dalam model relasional, hubungan disimpan dalam bentuk tabel. Itu disimpan bersama dengan entitasnya. Tabel memiliki dua properti baris dan kolom. Baris mewakili record dan kolom mewakili atribut.
3. Tuple: Tidak lain adalah satu baris tabel yang berisi satu record.

4. Skema hubungan: Skema hubungan mewakili nama hubungan dengan atributnya.
5. Derajat: Jumlah atribut dalam hubungan disebut derajat hubungan.
6. Kardinalitas: Jumlah total baris yang ada dalam tabel.
7. Kolom: Kolom mewakili sekumpulan nilai untuk atribut tertentu.
8. Contoh relasi adalah kumpulan tupel berhingga dalam sistem RDBMS. Instance hubungan tidak pernah memiliki tupel duplikat.
9. Kunci hubungan: Setiap baris memiliki satu, dua atau lebih atribut yang disebut kunci hubungan.
10. Domain atribut: Setiap atribut memiliki beberapa nilai dan ruang lingkup yang telah ditentukan sebelumnya yang dikenal sebagai domain atribut (Iskandar, 2017)

3.3 Perkembangan Model Basis Data Relasional

Ada beberapa upaya untuk sampai pada implementasi sempurna dari model basis data relasional yang awalnya didefinisikan oleh Edgar F. Codd. Perkembangan terbaru dari model objek-relasional, berdasarkan asumsi bahwa setiap fakta dapat dinyatakan dalam satu atau lebih hubungan biner. Model ini digunakan pada Object Role Modeling (ORM), RDF/Notation 3 (N3) (Rahman & Maghfiroh, 2019).

3.4 Aspek Penting Basis Data Relasional

Berbagai aspek penting basis data relasional: Structured Query Languange, Integrasi Data, Transaksi, Kepatuhan ACID (ACID atau *Atomic, Consistent, Isolated, dan Durable*)

3.4.1 SQL (*Structured Query Language*)

SQL atau *Structured Query Language* adalah antarmuka utama untuk berkomunikasi dengan database relasional. SQL menjadi standar dari American National Standards Institute (ANSI) pada tahun 1986. SQL Standar ANSI didukung oleh semua mesin basis data relasional utama, dan beberapa mesin ini juga memiliki ekstensi ke ANSI SQL untuk mendukung fungsionalitas khusus mesin itu. SQL digunakan untuk menambah, memperbarui atau menghapus baris data, mengambil subset data untuk pemrosesan transaksi dan aplikasi analitik, dan untuk mengelola semua aspek database (Setiyadi, 2020).

3.4.2 Integrasi Data

Integritas data adalah keseluruhan integritas, akurasi dan konsistensi data. Database relasional menggunakan sejumlah batasan untuk menegakkan integritas data dalam database. Ini termasuk Kunci utama, Kunci Asing, batasan 'Not NULL', batasan 'Unique', batasan 'Default' dan batasan 'Check'. Batasan integritas ini membantu menegakkan aturan bisnis pada data dalam tabel untuk memastikan keakuratan dan keandalan data. Selain itu, sebagian besar database relasi juga memungkinkan kode khusus untuk ditanam di pemicu yang mengeksekusi berdasarkan tindakan pada database (Putri, 2021).

3.4.3 Transaksi

Transaksi database terdiri dari satu atau lebih pernyataan SQL yang dieksekusi sebagai urutan operasi yang membentuk satu unit kerja logis. Transaksi menjadi proposisi "semua atau tidak sama sekali", artinya seluruh transaksi harus diselesaikan sebagai satu kesatuan dan ditulis ke database, atau tidak ada komponen transaksi yang dapat terjadi. Dalam terminologi database

relasional, transaksi menghasilkan COMMIT atau ROLLBACK. Setiap transaksi diperlakukan secara konsisten dan andal secara independen dari transaksi lainnya (Fitri, 2013).

3.2.2 ACID (*Atomic, Consistent, Isolated* dan *Durable*)

Semua transaksi basis data harus sesuai dengan ACID atau atomik, konsisten, terisolasi, dan tahan lama untuk memastikan integritas data. Atomicity mengharuskan seluruh transaksi berhasil, atau jika beberapa transaksi gagal, seluruh transaksi tidak valid. Konsistensi mensyaratkan bahwa data yang ditulis ke database sebagai bagian dari transaksi harus sesuai dengan semua aturan dan batasan yang ditentukan, termasuk batasan, level, dan pemicu. Isolasi sangat penting untuk mencapai kontrol konkurensi dan memastikan setiap transaksi independen dari dirinya sendiri. Kegigihan mengharuskan setiap perubahan yang dibuat ke database menjadi permanen setelah transaksi selesai dengan sukses (Joefrie & Kalatiku, 2012).

3.5 Pemanfaatan Database Relasional

Ada beberapa ketidaksepahaman terhadap definisi atas "relasional" dari DBMS. Definisi yang paling populer dari sebuah RDBMS sering kali dianggap kurang tepat; beberapa kalangan berargumentasi bahwa penyajian data sebagai kumpulan baris dan kolom sudah cukup memenuhi syarat untuk dikatakan sebagai sebuah RDBMS.

Pandangan seperti ini, yang banyak diterima oleh parateoretis dan kalangan-kalangan lainnya yang memegang teguh prinsip-prinsip Codd, tentunya akan mendiskualifikasikan banyak sistem basis data yang ada saat ini "tidak murni relasional". Dalam kenyataannya, sistem basisdata yang menggunakan SQL (Structured Query Language)untuk mengakses dan memodifikasi

data tidak bisa dikatakan sebagai RDBMS menurutdefinisi ini. Sementara itu, para pendukung atas sistem basis data yang ada menyebutkansebuah sistem basisdata yang menerapkan hanya beberapa dari hukum-hukum Codd tersebut disebut sebagai Sistem Manajemen Basisdata Semi - Relasional / Pseudo - Relational Database Management Systems (PRDBMS). Untuk sistem manajemen basis data yang sepenuhnyamenerapkan hukum-hukum Codd tersebut selanjutnya disebut sebagai Sistem Manajemen Basis data Murni-Relasional/Trully Relational Database Management Systems (TRDBMS). Saat ini, hampir seluruh RDBMS yang ada menerapkan SQL sebagai bahasa query namunjuga menyediakan dan mengimplementasi beberapa alternatif lainnya. Alpora Dataphor adalah RDBMS yang tersedia secara komersial yang mengikuti secara penuh ke dua belas hukum-hukum Codd tersebut, dan kedua kelompok mengenalnya sebagai RDBMS(Gat, 2015).

3.6 Tipe-tipe Database Relasional

Untuk menyimpan ataupun mengambil data dari basis data kita memerlukan perangkatlunak yang sering disebut dengan DBMS (system managemen basis data). Adapun tipedatabase ada sekurang-kurangnya 12 tipe, yaitu antara lain (Prasetya, 2015):

1. Analytical database, yang merupakan database untuk menyimpan informasi dan datayang diambil dari operasional dan eksternal database. Database ini terdiri dari data daninformasi yang diringkas dan paling banyak dibutuhkan oleh suatu organisasi manajemenmaupun End-user lainnya.
2. Operational database ialah database yang menyimpan data secara rinci yang dibutuhkanuntuk mendukung operasi dari seluruh organisasi. Biasa juga disebut dengan SADB (subject-area databases), transaksi, dan

produksi database. Contohnya seperti: database inventaris, database pribadi, database pelanggan, akuntansi database.

3. Distributed database merupakan kelompok kerja lokal database dan departemen dikantor-kantor dan lokasi kerja yang lainnya. Dalam database ini terdapat dua segmen yaituuser database dan operasional yang datanya digunakan dan duhasilkan hanya pada penggunasitus itu sendiri.
4. Data warehouse yaitu sebuah data warehouse yang menyimpan data dari tahun-tahunsebelumnya hingga saat ini. Data warehouse merupakan sumber utama data yang telahterintegrasi sehingga bisa digunakan dan dimanfaatkan oleh para pengguna seluruh organisasiyang profesional. Perkembangan yang terjadi akhir ini dari data warehouse ialahdipergunakan sebagai Shared nothing architecture untuk mendukung dan memfasilitasiekstrem scalling.
5. End-user database yang terdiri dari file-file data yang dikembangkan dari end-userdalam workstation mereka. Contohnya berbagai koleksi dokumen dalam word processing spreadsheet maupun download file.
6. Real time database merupakan sebuah sistem pengolahan yang dirancang dalammenangani beban kerja suatu negara yang bisa berubah-ubah, mengandung data terusmenerus dan sebagian tidak berpengaruh terhadap waktu. database ini bermanfaat bagi orang-orang hukum, akuntansi, perbankan, multi media dan analisis dta yang ilmiah.
7. Document oriented database yang merupakan salah satu program komputer yangdirangkai untuk sebuah aplikasi yang berorientasi pada dokumen. Sistem ini dapat

diterapkansebagai lapisan diatas database relasional maupun objek database.

8. In memory database. Database ini bergantung pada memori untuk penyimpanan datadalam sebuah komputer.
9. Navigational database. Dalam navigasi database ini, queries menjumpai benda bagiyang mengikuti referensi dari objek tertentu.
10. Hypermedia database on the web merupakan sekumpulan halaman multimedia yangsaling berkaitan dalam sebuah situs web, yang terdiri dari home page, dan hyperlink darimultimedia seperti gambar, teks, grafik audio dll.
11. External database. Adapun database tipe ini menyediakan akses ke eksternal, data milikpribadi online – tersedia untuk biaya pada pengguna akhir ataupun organisasi dari layanankomersial.
12. Relational database. Dari tahun 2009 rational database merupakan standar komputasi bisnis, dan database yang paling umum digunakan pada saat ini (Rahman & Maghfiroh, 2019).

3.7 Kelebihan Menggunakan Basis Data Relasional

Terdapat beberapa kelebihan menggunakan basis data relasional:

1. Kesederhanaan: Model data relasional di DBMS lebih sederhana daripada model hierarki dan jaringan.
2. Independensi Struktural: Basis data relasional hanya berkaitan dengan data dan bukan dengan struktur. Ini dapat meningkatkan kinerja model.

3. Mudah digunakan: Model Relasional di DBMS mudah karena tabel yang terdiri dari baris dan kolom cukup alami dan mudah dipahami.
4. Kemampuan kueri : Ini memungkinkan bahasa kueri tingkat tinggi seperti SQL untuk menghindari navigasi database yang kompleks.
5. Independensi Data: Struktur database Relasional dapat diubah tanpa harus mengubah aplikasi apa pun.
6. Scalable: Mengenai jumlah rekaman, atau baris, dan jumlah bidang, database harus diperbesar untuk meningkatkan kegunaannya (Hanum, Haekal, & Prasetio, 2020).

3.8 Kekurangan Menggunakan Basis Data Relasional

Terdapat beberapa kekurangan menggunakan basis data relasional:

1. Beberapa database relasional memiliki batasan panjang bidang yang tidak dapat dilampaui.
2. Database relasional terkadang dapat menjadi kompleks seiring dengan bertambahnya jumlah data, dan hubungan antar bagian data menjadi lebih rumit.
3. Sistem database relasional yang kompleks dapat mengarah ke database yang terisolasi di mana informasi tidak dapat dibagikan dari satu sistem ke sistem lainnya(Putra, Buana, Putri, & Buana, 2020).

DAFTAR PUSTAKA

Fitri, M. O. (2013). Trend Pengguna NoSQL untuk Basis Data Non Relasional. *Teknosains*, *7 Nomor 1*, 120–127.

Gat. (2015). Perancangan Basis Data Perputakaan Sekolah dengan Menerapkan Model Data Relasional. *Citec Journal*, *2*(4), 305–3015.

Hanum, B., Haekal, J., & Prasetio, D. E. (2020). The Analysis of Implementation of Enterprise Resource Planning in the Warehouse Division of Trading and Service Companies, Indonesia. *International Journal of Engineering Research and Advanced Technology*, *06*(07), 37–50. https://doi.org/10.31695/ijerat.2020.3621

Iskandar, R. (2017). Desain Basis Data Relasional Dinas Kesehatan Kota Sabang. *Journal of Information Systems for Public Health*, *2*(3), 29–38. Retrieved from https://journal.ugm.ac.id/jisph/article/view/17098

Joefrie, Y. Y., & Kalatiku, P. P. (2012). Desain Basis Data Sistem Informasi Akademik Di Fakultas Teknik Universitas Tadulako. *Jurnal IImiahForistek*, *2*(21), 190–194.

Prasetya, W. S. (2015). Perancangan Model Basis Data Relasional Dengan Metode Database Life Cycle. *Prosiding Seminar Nasional Informatika 2015*, 91–98.

Putra, Y. M., Buana, U. M., Putri, R. J., & Buana, U. M. (2020). *Sistem Informasi Akuntansi Pengaplikasian Dan Implementasi Konsep Basis Data Relasional Pada Sistem Pendapatan Dan Pengeluaran*. (June), 1.

Putri, N. I. (2021). Keamanan Basis Data Berdasarkan Teori Himpunan. *J-SIKA| Jurnal Sistem Informasi Karya Anak Bangsa*, *3*(02), 45–52.

Rahman, M. A., & Maghfiroh, L. R. (2019). *Relational and Nonrelational Database Study on BPS-Statistics Indonesia Data in Facing Era of Big Data*. 25–30.

Setiyadi, D. (2020). Aljabar Relational dan Implementasi kedalam Bahasa Query dalam Perancangan Database Relational. *Jurnal Kajian Ilmiah (JKI)*, *20*(2), 213–224.

BAB 4
ALJABAR RELASIONAL

Oleh Elmi Devia

4.1 Pendahuluan

Bahasa *Query* merupakan bahasa yang termasuk dalam kategori bahasa tingkat tinggi (*high level language*) yang digunakan *user* untuk memperoleh informasi dari basis data. Bahasa *Query* dibagi menjadi dua jenis, yaitu bahasa prosedural dan bahasa non-prosedural. Pada sebuah bahasa prosedural, *user* meminta sistem untuk melakukan serangkaian operasi terhadap basis data dalam rangka mendapatkan informasi yang diinginkan. Namun dalam bahasa non-prosedural, *user* menunjukkan informasi yang diinginkan tanpa menyatakan suatu cara atau prosedur tertentu untuk memperoleh informasi tersebut.

4.2 Aljabar Relasional

Aljabar Relasional adalah salah satu dari dua bahasa *query* formal yang terkait dengan model relasional. *Query* dalam aljabar disusun dengan menggunakan satu kumpulan operator, *property* pokok adalah setiap operator dalam aljabar menerima (satu atau dua) contoh relasi sebagai *argument* dan menghasilkan contoh relasi. *Property* ini memudahkan operator *compose* membentuk satu *query* kompleks ekspresi aljabar relasional secara berulang ditentukan untuk menjadi suatu relasi, satu operator aljabar *unary* berlaku untuk ekspresi

tunggal, atau operator operator aljabar biner berlaku untuk dua ekspresi.

Tiap aljabar relasional menggambarkan prosedur langkah-langkah untuk menghitung jawaban yang diinginkan, berdasarkan urutan penerapan operator di dalam *query*. Sifat aljabar yang produral membuat kita dapat melihat ekspresi aljabar sebagai satu rencana untuk mengevaluasi sebuah query, dan sistem basis data relasional sebenarnya menggunakan ekspresi aljabar untuk menyajikan rencana evaluasi *query* (Ramakrishnan, 2004).

4.3 Operasi-operasi Aljabar Relasional

Bahasa *Query* yang didasarkan pada operasi-operasi dalam aljabar relasional merupakan Bahasa *Query* yang prosedural. Bahasa ini memiliki sejumlah operasi yang memanfaatkan suatu atau beberapa tabel atau relasi basis data sebagai masukan dan menghasilkan sebuah tabel atau beberapa relasi basis data yang baru sebagai keluarannya. Sejumlah operasi dasar yang dikenal dalam aljabar relasional, yaitu *select*, *project*, *cartesian-product*, *union*, *set-difference*, dan *rename*. Dan operasi tambahan, yaitu *set intersection, theta-join/, natural join, outer join, dan division.*

4.3.1 *Select* (σ)

Operasi ini digunakan untuk mengambil sejumlah baris data yang memenuhi predikat yang diberikan. Predikat mengacu pada kondisi yang ingin dipenuhi dalam operasi seleksi. Sintaks yang digunakan untuk menyatakan operasi ini adalah:

$$\sigma_P(R) = \{t | t \in R, P(t)\}$$

dimana *P* adalah predikat pada atribut-atribut di *R* yang dapat dihubungkan dengan ∧ (**and**), ∨ (**or**), dan ¬(**not**). Bentuk umum dari predikat seleksi adalah:

< atribut > *op* < atribut >,

atau

< atribut > *op* < konstanta >,

dengan *op* adalah salah satu dari =, /=, >, ≥, <, dan ≤.

Contoh

- Query: Tampilkan daftar karyawan yang tempat lahirnya di 'Bekasi'.
- Aljabar relasional:

  σ Tmp_lhr='Jakarta' (KARYAWAN)
- Tabel/relasi asal:

  KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

- Hasil:

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12042	Rina	Jakarta	12/09/1981	P

4.3.2 *Project* (Π)

Operasi ini dapat menentukan atribut-atribut data dari sebuah tabel atau hasil Query yang akan ditampilkan. Dengan kata lain, operasi ini merupakan relasi antar beberapa kolom yang diperoleh dengan menghapus kolom yang tidak terdaftar. Sintaks yang digunakan untuk menyatakan operasi ini adalah:

Π colum1,...,column (tabel)

Contoh

- Query: Tampilkan Nik, Nama, Jns_Kel dari relasi Karyawan.
- Aljabar relasional:

  Π NIK,Nama,Jns_Kel (KARYAWAN)
- Tabel/relasi asal:

  KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

- Hasil:

NIK	Nama	Jns_Kel
12034	Amir	L
12037	Tono	L
12042	Rina	P
12043	Faisal	L

4.3.3 Cartesian-product (X)

Operasi ini dapat menggabungkan data dari dua buah tabel atau hasil Query. Simbol yang digunakan untuk menyatakan operasi ini adalah "×" dan sintaks yang digunakan untuk operasi ini adalah

R X S = {(x,y) | x R dan y S}

Contoh

- Query: Tampilkan NIK, Nama (dari relasi Karyawan), Kode_Pek, Nama_Pek, Unit (dari relasi Pekerjaan).
- Aljabar relasional:

  Π NIK,Nama,Kode_Pek,Nama_Pek,Unit (KARYAWAN X PEKERJAAN)
- Tabel/relasi asal:

KARYAWAN

NIK	Nama
12034	Amir
12037	Tono

PEKERJAAN

Kd_Pek	Nm_Pek	Unit
102	Administrasi	Produksi
103	Operator	Produksi
201	Kasir	Pemasaran

- Hasil:

NIK	Nama	Kd_Pek	Nm_Pek	Unit
12034	Amir	102	Administrasi	Produksi
12034	Amir	103	Operator	Produksi

NIK	Nama	Kd_Pek	Nm_Pek	Unit
12034	Amir	201	Kasir	Pemasaran
12037	Tono	102	Administrasi	Produksi
12037	Tono	103	Operator	Produksi
12037	Tono	201	Kasir	Pemasaran

4.3.4 *Union* ( )

Operasi ini dapat menggabungkan data dari dua kelompok baris data *(row)* yang sejenis (memiliki hasil *project* yang sama). Notasi untuk operasi ini adalah

R U S = {x | x R atau x S}

Untuk semua $R \cup S$ harus memenuhi:

1. *R* dan *S* harus memiliki atribut yang sama
2. daerah asal atribut harus cocok tipenya. Misalnya, kolom kedua dari *R* harus mempunyai tipe yang sama dengan kolom kedua dari *S*.

Contoh

- Query: Tampilkan Kd_Pek, NIK (dari relasi R) Union dari Kd_Pek,NIK (dari relasi S)
- Aljabar relasional:

  Π Kd_Pek,NIK (R) v Kd_Pek,NIK (S)
- Tabel/relasi asal:

  Relasi *R* menyatakan keadaan sistem informasi karyawan dalam tahap RIWAYAT_PEK sebelumnya.

  R

Kd_Pek	NIK
102	12034
103	12034
103	12037

Relasi *S* menyatakan keadaan sistem informasi karyawan dalam tahap RIWAYAT_PEK terakhir.

S

Kd_Pek	NIK
102	12037
103	12037

- Hasil:

Kd_Pek	NIK
102	12034
102	12037
103	12034
103	12037

4.3.5 Set-difference (−)

Operasi ini merupakan kebalikan dari operasi *union*, yaitu pengurangan data di tabel atau hasil *project* pertama oleh data di tabel atau hasil *project* kedua. Simbol dari operasi ini adalah:

R - S = {x | x ∈ R atau x ∈ S}

Untuk semua $R - S$ harus memenuhi:

1. *R* dan *S* harus memiliki atribut yang sama
2. daerah asal atribut harus cocok tipenya. Misalnya, kolom kedua dari *R* harus mempunyai tipe yang sama dengan kolom kedua dari *S*.

Contoh

- Query: Tampilkan Kd_Pek, NIK (dari relasi R) Union dari Kd_Pek,NIK (dari relasi S)
- Aljabar relasional:

Π Kd_Pek,NIK (R) - Kd_Pek,NIK (S)

- Tabel/relasi asal:
  Relasi *R* menyatakan keadaan sistem informasi karyawan dalam tahap RIWAYAT_PEK sebelumnya.
  R

Kd_Pek	NIK
102	12034
103	12034
103	12037

  Relasi *S* menyatakan keadaan sistem informasi karyawan dalam tahap RIWAYAT_PEK terakhir.
  S

Kd_Pek	NIK
102	12037
103	12037

- Hasil:

Kd_Pek	NIK
102	12034
103	12034

4.3.6 *Rename* ($\rho_X(E)$)

Rename (ρ), adalah operasi untuk menyalin tabel lama kedalam tabel yang baru.
Sintaks yang digunakan dalam operasi rename ini adalah sebagai berikut :

ρ [nama_tabel] (tabel_lama)

Contoh

- Query: Salinlah table baru dengan nama KARYAWAN_NEW dari table KARYAWAN, dimana Jns_Kel adalah 'L'.
- Aljabar relasional:

  ρ KARYAWAN_NEW (σ Jns_Kel='L') (KARYAWAN))
- Tabel/relasi asal:

  KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

- Hasil:

  KARYAWAN_NEW

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12043	Faisal	Padang	10/12/1985	L

4.3.7 Set-intersection (∩)

Set-intersection termasuk kedalam operator tambahan, karena operator ini dapat diderivikasi dari operator dasar seperti berikut :

A ∩ B = A - (A – B), atau A ∩ B = B - (B –A)

Operasi ini merupakan operasi binary, yang digunakan untuk membentuk sebuah relasi baru dengan tuple yang berasal dari kedua relasi yang dihubungkan.

Contoh

- Query: Tampilkan NIK (dari relasi KARYAWAN) Set-intersection dari NIK (dari relasi RIWAYAT_PEK)
- Aljabar relasional:
  Π NIK (KARYAWAN) ∩ NIK (RIWAYAT_PEK)
- Tabel/relasi asal:
  KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

  RIWAYAT_PEK

NIK	Kd_Pek	Masa_Pek
12034	102	1
12034	103	2
12042	102	3
12042	201	2
12042	301	1
12043	103	1

- Hasil:

NIK
12034
12042
12043

4.3.8 Theta-join (θ)

Theta-join adalah operasi yang bertujuan untuk membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi yang mungkin dari relasi-relasi dengan kondisi/kriteria tertentu.

Contoh

- Query: Tampilkan Nm_Pek yang pernah dikerjakan oleh karyawan dengan NIK='12042' dengan Kd_Pek pada relasi PEKERJAAN sama dengan pada relasi RIWAYAT_PEK.
- Aljabar relasional:
  Π Nm_Pek (δ NIK='12042' ∧ PEKERJAAN.Kd_Pek = RIWAYAT_PEK.Kd_Pek) (PEKERJAAN X RIWAYAT_PEK)
- Tabel/relasi asal:

PEKERJAAN

Kd_Pek	Nm_Pek	Unit
102	Administrasi	Produksi
103	Operator	Produksi
201	Kasir	Pemasaran
301	Sekretaris	Pemasaran

RIWAYAT_PEK

NIK	Kd_Pek	Masa_Pek
12034	102	1
12034	103	2
12042	102	3
12042	201	2
12042	301	1
12043	103	1

- Hasil:

Nm_Pek
Administrasi
Kasir
Sekretaris

4.3.9 Natural-join (∞)

Natural-join adalah operasi yang bertujuan untuk membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi yang mungkin dari relasi-relasi dengan syarat kedua relasi memiliki satu atau lebih atribut yang sama.

Contoh

- Query: Tampilkan Nama karyawan yang pernah bekerja dengan kd_Pek='102'
- Aljabar relasional:

  Π NIK,Nama Kd_Pek='102' (KARYAWAN x RIWAYAT_PEK)
- Tabel/relasi asal:

  KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

  RIWAYAT_PEK

NIK	Kd_Pek	Masa_Pek
12034	102	1
12034	103	2
12042	102	3
12042	201	2
12042	301	1
12043	103	1

- Hasil:

Nama
Amir
Rina

4.3.10 Outer-join (⟕)

Outer-join adalah operasi untuk menggabungkan operasi selection dan cartesian-product dengan suatu kriteria pada kolom yang sama.

Contoh

- Query : Tampilkan NIK,Nama (dari relasi KARYAWAN) dan Masa_Pek (dari relasi RIWAYAT_PEK) dengan outer join, artinya adalah pada kolom NIK,Nama pada relasi KARYAWAN akan ditampilkan walaupun KARYAWAN tersebut tidak mempunyai riwayat pekerjaan.
- Aljabar relasional:
  $\Pi_{\mathbf{NIK,Nama}}(\mathbf{KARYAWAN}) \text{ ⟕ } \Pi_{\mathbf{Masa\_Pek}}(\mathbf{RIWAYAT\_PEK})$
- Tabel/relasi asal:

KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

RIWAYAT_PEKERJAAN

NIK	Kd_Pek	Masa_Pek
12034	102	1
12034	103	2
12042	102	3
12042	201	2
12042	301	1
12043	103	1

- Hasil:

NIK	Nama	Masa_Pek
12034	Amir	1
12034	Amir	2
12037	Tono	<Null>
12042	Rina	3
12042	Rina	2
12042	Rina	1
12043	Faisal	1

4.2.11 Division (÷)

Division adalah operasi yang banyak digunakan dalam query yang mencakup frase "setiap" atau "untuk semua", operasi ini juga merupakan pembagian atas tuple – tuple dari dua relasi.

Contoh

- Query : Tampilkan NIK,Kd_Pek,Masa_Pek (dari relasi RIWAYAT_PEK) dan NIK (dari relasi KARYAWAN) dimana karyawan yang tempat lahirnya = 'Jakarta' dan lakukan dIvision pada kedua relasi tersebut.
- Aljabar relasional:

  Π NIK,Kd_Pek,Masa_Pek (RIWAYAT_PEK) ÷
  (Π NIK (Tmp_Lhr='Jakarta' (KARYAWAN)))
- Tabel/relasi asal:

  KARYAWAN

NIK	Nama	Tmp_Lhr	Tgl_Lhr	Jns_Kel
12034	Amir	Jakarta	03/12/1980	L
12037	Tono	Bandung	23/07/1982	L
12042	Rina	Jakarta	12/09/1981	P
12043	Faisal	Padang	10/12/1985	L

RIWAYAT_PEK

NIK	Kd_Pek	Masa_Pek
12034	102	1
12034	103	2
12042	102	3
12042	201	2
12042	301	1
12043	103	1

- Hasil:

NIK	Kd_Pek	Masa_Pek
12034	102	1
12034	103	2
12042	102	3
12042	201	2
12042	301	1

DAFTAR PUSTAKA

Rames Elmasri (2003 'Fundamentals of Database System'. Department of Computer Science and Engineering. University of Texas at Arlington. Pearson – Addison Weasley.

Ramakrishnan, Reghu dan Johannes gehrke. (2004 'Sistem Manajemen Database'. Yogyakarta : Andi and McGraw-Hill Education.

BAB 5
MANIPULASI DATA MENGGUNAKAN SQL

Oleh Leo Willyanto Santoso

5.1 Pendahuluan

Kita dapat melakukan manipulasi data dalam sebuah *database* dengan bahasa SQL. Perintah SQL untuk memanipulasi data terbagi ke dalam tiga jenis, yaitu insert, update, dan delete. Berikut Tabel Customers yang akan kita gunakan dalam Bab ini.

Kode Customer	Nama	Alamat	Kota	KodePos
1	Andi	Jl. Senopati 20	Jakarta	10217
2	Budi	Jl. Mawar 31	Bandung	22176
3	Citra	Jl. Sudirman 17	Surabaya	62351
4	Doni	Jl. Kusuma II/23	Semarang	47230

5.2 Insert

Insert merupakan perintah yang digunakan untuk memasukkan sebuah record baru ke dalam sebuah tabel *database*. Kita dapat menuliskan perintah Insert dengan 2 cara, yaitu:

a) Tentukan nama kolom dan nilai yang akan disisipkan
   INSERT INTO *nama_tabel (column1, column2, ...)*
   VALUES *(nilai1, nilai2, ...);*

Contohnya:
INSERT INTO Customers (KodeCustomer, Nama, Alamat, Kota, KodePos)
VALUES (5, 'Endang', 'Jl. Delima 15', 'Surakarta', '40106');

Maka, Ketika kita melihat isi table Customers kita, isinya akan seperti berikut ini

Kode Customer	Nama	Alamat	Kota	KodePos
1	Andi	Jl. Senopati 20	Jakarta	10217
2	Budi	Jl. Mawar 31	Bandung	22176
3	Citra	Jl. Sudirman 17	Surabaya	62351
4	Doni	Jl. Kusuma II/23	Semarang	47230
5	Endang	Jl. Delima 15	Surakarta	40106

b) Jika kita akan menambahkan nilai untuk semua kolom pada tabel, kita tidak perlu menentukan nama kolom dalam query SQL. Namun, pastikan urutan nilai dalam urutan yang sama dengan urutan kolom dalam tabel. Sintaks INSERT INTO adalah sebagai berikut:
INSERT INTO *nama_tabel*
VALUES *(nilai1, nilai2, nilai3 ...);*

Contohnya:
INSERT INTO Customers
VALUES (6, 'Faisal', 'Jl. Pisang 27', 'Surakarta', '40106');
Maka, Ketika kita melihat isi tabel Customers kita, isinya akan seperti berikut ini.

Kode Customer	Nama	Alamat	Kota	KodePos
1	Andi	Jl. Senopati 20	Jakarta	10217
2	Budi	Jl. Mawar 31	Bandung	22176
3	Citra	Jl. Sudirman 17	Surabaya	62351
4	Doni	Jl. Kusuma II/23	Semarang	47230
5	Endang	Jl. Delima 15	Surakarta	40106
6	Faisal	Jl. Pisang 27	Surakarta	40106

5.3 Update

Update digunakan ketika kita akan memodifikasi isi dari sebuah tabel. Contoh penggunaannya adalah jika terjadi kesalahan atau perubahan data ketika memasukkan sebuah record, kita tidak perlu menghapusnya. Tetapi kita dapat memperbaiki menggunakan perintah Update ini. Berikut sintaks untuk perintah Update.

```
UPDATE nama_tabel
SET column1 = nilai1, column2 = nilai2, ...
WHERE kondisi;
```

Berikut contoh perintah Update

```
UPDATE Customers
SET Nama = 'Anton', kota = 'Bekasi'
WHERE kodeCustomer = 1;
```

Kode Customer	**Nama**	**Alamat**	**Kota**	**KodePos**
1	Anton	Jl. Senopati 20	Bekasi	10217
2	Budi	Jl. Mawar 31	Bandung	22176
3	Citra	Jl. Sudirman 17	Surabaya	62351
4	Doni	Jl. Kusuma II/23	Semarang	47230
5	Endang	Jl. Delima 15	Surakarta	40106
6	Faisal	Jl. Pisang 27	Surakarta	40106

Kita harus berhati-hati, jika lupa menuliskan WHERE, maka semua data dalam tabel akan terupdate. Berikut contoh sintaks perintah Update, dimana akan mengupdate semua isi data dalam tabel.

```
UPDATE Customers
SET Nama = 'Anton';
```

Kode Customer	Nama	Alamat	Kota	KodePos
1	Anton	Jl. Senopati 20	Bekasi	10217
2	Anton	Jl. Mawar 31	Bandung	22176
3	Anton	Jl. Sudirman 17	Surabaya	62351
4	Anton	Jl. Kusuma II/23	Semarang	47230
5	Anton	Jl. Delima 15	Surakarta	40106
6	Anton	Jl. Pisang 27	Surakarta	40106

5.4 Delete

Delete digunakan Ketika kita akan menghapus data yang ada pada sebuah tabel. Sintaks dari perintah delete adalah:
DELETE FROM nama_tabel WHERE kondisi;

Berikut ini adalah contoh perintah Delete ketika kita akan menghapus customer dengan nama Faisal.
DELETE FROM Customers WHERE nama = 'Faisal';
Maka, kondisi dari table Customers akan seperti berikut ini:

Kode Customer	Nama	Alamat	Kota	KodePos
1	Andi	Jl. Senopati 20	Jakarta	10217
2	Budi	Jl. Mawar 31	Bandung	22176
3	Citra	Jl. Sudirman 17	Surabaya	62351
4	Doni	Jl. Kusuma II/23	Semarang	47230
5	Endang	Jl. Delima 15	Surakarta	40106

Kita juga bisa menghapus semua baris dalam tabel tanpa menghapus tabel. Ini berarti bahwa struktur tabel, atribut, dan indeks akan tetap utuh. Sintaks dari perintah delete ini adalah:
DELETE FROM nama_tabel;

Berikut contoh perintah jika kita ingin menghapus semua isi dari table Customers.
DELETE FROM Customers;

DAFTAR PUSTAKA

Elmasri, R. A., & Navathe, S. B. 2017. Fundamentals of Database Systems, Global Edition. Pearson Education Limited.

BAB 6
DEFINISI DATA MENGGUNAKAN SQL

Oleh Yuniansyah

6.1 Pendahuluan

Bahasa *Query* atau biasa dikenal dengan *Structured Query Language* (SQL) adalah suatu Bahasa (*language*) yang digunakan untuk mengakses database (basis data), khususnya untuk database relational. SQL dikenalkan pertama kali di IBM pada tahun 1970, yang pada awalnya mengacu pada artikel yang ditulis oleh Jhonny Oracle yang merupakan seorang peneliti di IBM. Pada artikel ini dibahas untuk mengembangkan suatu Bahasa standar yang dapat digunakan pada semua database. Istilah ini kemudian banyak di kenal dengan nama "*SEQUEL*" yang merupakan singkatan dari *Structured English Query Language*, dan disingkat menjadi SQL

Proses standarisasi SQL selesai pada tahun 1986, dan terdapat perbaikan-perbaikan pada tahun 1989. Selanjutnya pada tahun 1992 muncul versi SQL92 dan diperbaruhi Kembali pada tahun 1999 dengan nama SQL99, tetapi sampai saat ini yang banyak digunakan adalh SQL92 dan hamper semua *Database Management System* (DBMS) yang dapat menggunakan SQL

Secara umum SQL dibagi menjadi tiga jenis perintah, yaitu pertama ***data definition language*** yang digunakan untuk pendefinisian database, tabel, index, function, procedure maupun trigger. Kedua ***data manipulation language*** yang digunakan manipulasi (menambah, menghapus, melakukan perubahan pada record) serta pengolahan data lainnya pada tabel dan ketiga adalah

data control language yang digunakan untuk melakukan control atau hak akses yang berhubungan dengan database.

6.2 Data Definition Language

Pada SQL perintah yang pertama digunakan adalah *Data Definition Language* (DDL) yaitu perintah yang digunakan untuk membuat database dan membuat tabel, procdure, function dan lainnya. Selain membuat database dan membuat tabel perintah DDL juga dapat digunakan untuk merubah nama database atau tabel, merubah struktur field pada tabel, menambahkan key (kunci) pada tabel, serta menghapus database atau tabel

Pada DDL terbagi menjadi lima jenis perintah, yaitu :

1. **Create**, yaitu suatu perintah yang dapat kita gunakan untuk membuat suatu database baru, selain itu perintah create dapat digunakan untuk membuat index, function, procedure, table, maupun trigger yang baru,.
2. **Drop**, yaitu suatu perintah yang dapat kita gunakan untuk menghapus database, tabel atau indek yang ada
3. **Rename**, yaitu suatu perintah yang dapat kita gunakan untuk merubah nama database atau tabel yang ada
4. **Alter**, yaitu suatu perintah yang dapat kita gunakan untuk melakukan perubahan struktur tabel dengan menambah atau menghapus kolom yang ada. Perubahan juga bisa digunakan untuk mengubah nama tabel atau nama field yang sudah ada.
5. **Show,** yaitu suatu perintah yang digunakan untuk melihat Database atau tabel yang ada.

Pada pembahasan disini penulis akan membahasa satu-persatu perintah *Data Definition Language* beserta penerapan nya pada salah satu *Database Management System* (DBMS). Software DBMS yang akan penulis gunakan adalah My SQL yang berjalan pada web server XAMPP. Pada pembahasan disini penulis menggunakan

contoh Database sederhana yaitu Database Perputaskaan yang digunakan untuk mendata proses peminjaman buku oleh anggota di perpustakaan.

6.2.1. Perintah Create

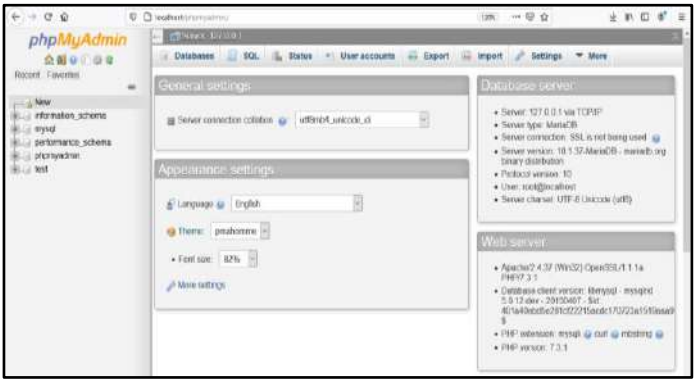
Perintah create digunakan untuk membuat database, table, procedure, function atau yang lainnya pada suatu database. Seperti telah dibahas di atas, kita akan membuat database perpustakaan menggunakan perintah SQL. Form penulisan SQL untuk membuat database adalah sebagai berikut :

CREATE DATABASE nama_database;

Jadi untuk membuat database perpustakaan kode SQL nya adalah :

CREATE DATABASE dbperpustakaan;

Pada contoh disini kita menggunakan My SQL pada XAMPP untuk membuat database, table dan perintah lainnya pada Data Definition Language. Untuk mengaktifkan My SQL, aktifkan control Panel XAMPP kemudian buka browser dan ketikkan localhost/phpmyadmin, maka akan tampil jendela phpMyAdmin seperti terlihat pada gambar 8.1 berikut ini



Gambar 6.1. Jendela phpMyAdmin (My SQL)

Untuk membuat database menggunakan perintah SQL, kita bisa melakukan dengan cara aktifkan tab SQL pada phpMyAdmin, maka akan tampil jendela SQL Editor.Selanjutnya kita dapat menuliskan perintah untuk membuat database. Perintah untuk membuat database perpustakaan adalah sebagai berikut

CREATE DATABASE dbperpustakaan;

Gambar 6.2 Perintah SQL Untuk Membuat Database

Setelah selesai menuliskan perintah SQL di kita klik button Go yang terletak di bawah jendela, maka database dbperpustakaan akan dibuat oleh My SQL. Perintah create juga dapat digunakan untuk membuat table pada database yang telah kita buat, selain itu juga kita dapat menambahkan atau membuat indek pada table yang kita buat. Format penulisan untuk membuat tabel dengan perintah SQL adalah :

```
CREATE TABLE nama_tabel
(       field1 tipe(panjang),
        field2 tipe(panjang),
            ...
            Field n tipe(panjang),
            PRIMARY KEY (field_key)
);
```

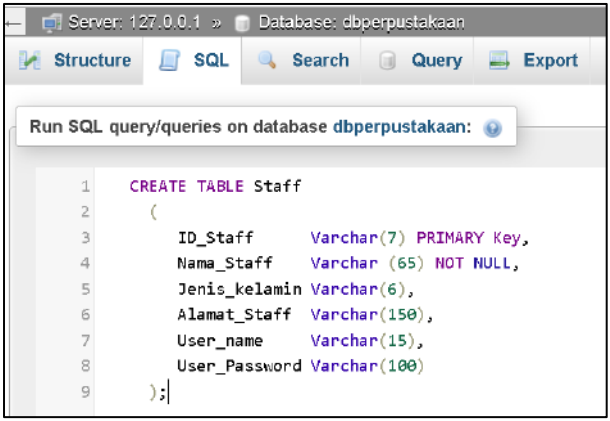
Untuk penambahan primary key bisa di bagian bawah, atau langsung di sisi kanan filednya, seperti berikut ini :

field1 tipe(panjang) primary key,

Sebagai contoh kita akan membuat tabel Staff perpustakaan, maka perintah SQL ny adalah sebagai berikut :

```
CREATE TABLE Staff
    (
        ID_Staff        Varchar(7) PRIMARY Key,
        Nama_Staff      Varchar (65) NOT NULL,
        Jenis_kelamin   Varchar(6),
        Alamat_Staff    Varchar(150),
        User_name       Varchar(15),
        User_Password   Varchar(100)
    );
```

Perintah pada My SQL dapat dilihat pada gambar 8.3 dibawah ini



Gambar 6.3. Perintah SQL Untuk Membuat Tabel

6.2.2. Perintah Show

Perintah Show digunakan untuk menampilkan database atau tabel yang ada. Format penulisan perintah show adalah sebagai berikut :

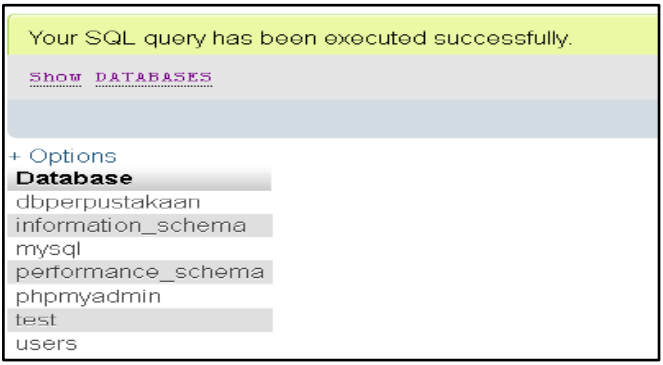
Show Database

Atau

Show TABLES

Sebagai contoh untuk melihat database yang telah dibuat, kita dapat menggunakan perintah Show. Caranya aktifkan pilihan SQL, pada SQL Editor ketikkan perintah :

SHOW DATABASES



Gambar 6.4. Hasil Perintah Show Database Pada SQL

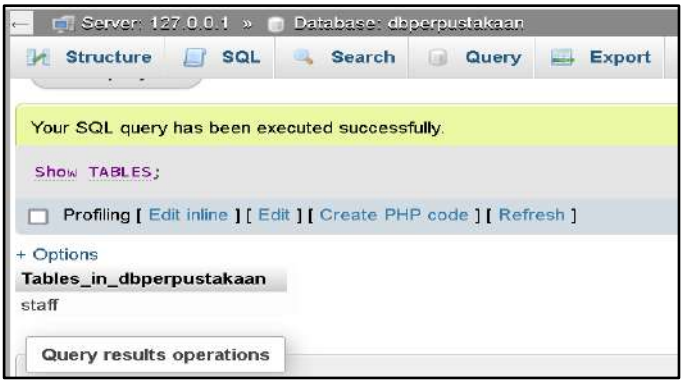
Perintah di atas digunakan untuk menampilkan Database yang ada atau yang telah dibuat sebelumnya, untuk perintah menampilkan tabel yang telah dibuat, pertama kali kita piliha database nya dan ketik

Show Tables

Seperti contoh pada My SQL untuk menampilkan tabel pada Database dbperpustakaan dapat dilihat pada gambar 8.5 berikut ini

Gambar 6.5. Perintah Show Tables Pada SQL

Setelah menuliskan perintah SQL, klik Go. maka akan tampil tabel yang pernah dibuat pada database dbperpustakaan ini. Hasil perintah show tables ini dapat dilihat pada gambar 8.6 berikut ini :



Gambar 6.6. Hasil Perintah Show Tables Pada SQL

6.2.3. Perintah Alter

Alter, yaitu suatu perintah yang dapat kita gunakan untuk mengubah struktur tabel yang sebelumnya sudah ada. Perubahan bisa jadi mengubah nama tabel, nama field, penambahan atau penghapusan field yang ada.

Format penulisan-penulisan untuk perintah alter adalah sebagai berikut :

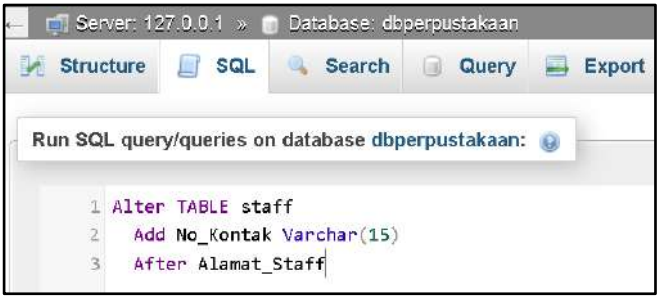
```
ALTER  TABLE tbl_name

  | ADD [COLUMN] col_name column_definition
      [FIRST | AFTER col_name ]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD {INDEX|KEY} [index_name]
      [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
      [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
      UNIQUE [INDEX|KEY] [index_name]
      [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
```

Terlihat pada sintak perintah-perintah di atas, perintah alter dapat digunakan untuk menambah colum, menambah index, menambah constraint. Perintah alter juga dapat digunakan untuk menghapus atau merubah nama field maupun nama tabel.

Sebagai contoh jika kita ingin menambahkan field kontak pada tabel staff yang telah dibuat pada bagian terdahulu, maka perintah nya adalah sebagai berikut :

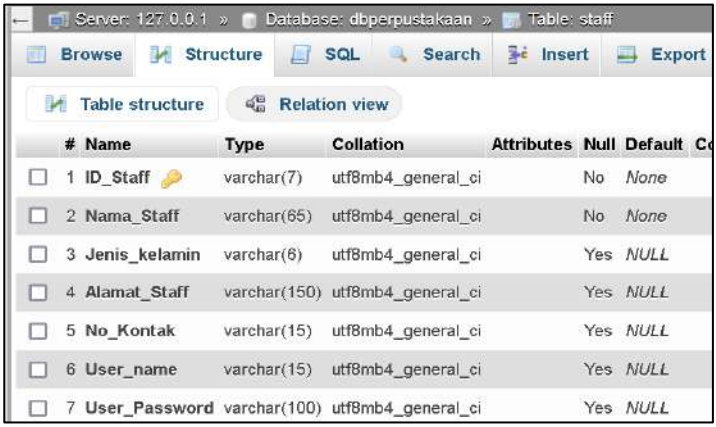
Alter TABLE staff
No_Kontak Varchar(15)
After Alamat_Staff

Tampilan penambahan colum No_kontak menggunakan perintah SQL pada My SQL dapat dilihat pada gambar berikut ini.



Gambar 6.7. Perintah Alter SQL Pada My SQL

Setelah di eksekusi, maka struktur tabel staff akan terdapat kolom baru, yaitu field No_kontak yang akan berada pada posisi setelah field Alamat Staff, Struktur field tabel staff setelah perintah alter dapat dilihat pada gambar 6.8. berikut ini :



Gambar 6.8. Hasil Perintah Alter SQL Pada My SQL

6.2.4. Perintah Rename

Rename, yaitu suatu perintah yang dapat kita gunakan untuk merubah nama database atau tabel yang ada. Pada perubahan tertentu perintah rename dapat digunakan bersamaan dengan perintah Alter.

Format penulisan perintah Rename adalah senagai berikut :

RENAME TABLE [Nama table lama] to [Nama Table Baru]

Sebagai contoh :

RENAME TABLE staff to staff_perpustakaan

Perintah di atas digunakan untuk mengganti nama tabel staff menjadi tabel staff_perpustakaan.

6.2.5. Perintah Drop

Perintah Drop digunakan untuk menghapus tabel, database, function, index, procedure ataupun trigger.

Format penulisan perintah Drop adalah sebagai berikut :

Menghapus Database :

DROP DATABASE nama_database

Contoh : DROP DATABASE FBPercetakan

Menghapus Table :

DROP TABLE nama_table

Contoh : DROP TABLE Barang

DAFTAR PUSTAKA

Didik Setiyadi, 2020, Sistem Basis Data dan MySQL, Mitra Wacana Media – Jakarta.

Handra Jatnika, 2013, Pengantar Sistem Basis Data (Memahami Konsep Dasar dan Tuntunan Praktis Perancangan Database),Andi Offset – Yogyakarta.

James R Groff, Paul N Weinberg, 2019, SQL : The Complete Reference, Osborne/McGraw-Hill, Berkeley – California U.S.A.

Muhammad Fikry, 2019, Basis D**ata, Unimal Press, Kampus Bukit** Indah Lhokseumawe.

Yudi Priyadi, 2021, Kolaborasi SQL dan ERD Dalam Implementasi Database, Andi Publisher – Yogyakarta.

BAB 7
DESAIN BASIS DATA LOGIKA UNTUK MODEL DATA RELASIONAL

Oleh Junaidi

7.1 Pendahuluan

Perancangan (desain) merupakan suatu hal yang sangat penting dalam pembuatan basis data. Permasalahan yang dihadapi pada waktu perancangan adalah bagaimana basis data yang akan dibangun ini dapat memenuhi kebutuhan saat ini dan masa yang akan datang. Untuk itu diperlukan perancangan basis data baik secara fisik maupun secara konseptualnya.

Perancangan konseptual akan menunjukkan entity dan relasinya berdasarkan proses yang diinginkan oleh organsisasinya. Untuk menentukan entity dan relasinya perlu dilakukan analisis data tentang informasi yang ada dalam spesifikasi di masa yang akan datang. Metodologi perancangan basis data adalah kumpulan teknik terorganisasi untuk pembuatan rancangan basis data. Teknik terorganisasi ini merupakan kumpulan tahap-tahapan yang memiliki aturan-aturan terurut.

7.2 Desain Basis Data

Desain Basis Data adalah kumpulan proses yang memfasilitasi perancangan, pengembangan, implementasi, dan pemeliharaan sistem manajemen data perusahaan. Basis data yang dirancang dengan benar mudah dipelihara, meningkatkan

konsistensi data, dan hemat biaya dalam hal ruang penyimpanan disk. Desainer database memutuskan bagaimana elemen data berkorelasi dan data apa yang harus disimpan.

Tujuan utama perancangan basis data adalah untuk menghasilkan model desain logis dan fisik dari sistem basis data yang diusulkan. Model logis berkonsentrasi pada persyaratan data dan data yang akan disimpan terlepas dari pertimbangan fisik. Itu tidak peduli dengan bagaimana data akan disimpan atau di mana akan disimpan secara fisik. Model desain data fisik melibatkan penerjemahan desain logis dari database ke media fisik menggunakan sumber daya perangkat keras dan sistem perangkat lunak seperti sistem manajemen basis data (DBMS).

Desain basis data merupakan hal yang penting karena bisa membantu menghasilkan sistem database:

1. Untuk memenuhi persyaratan para pengguna
2. Memiliki sistem database dengan performa tinggi

Desain basis data sangat penting untuk membuat sistem database **performa tinggi.**

7.3 Tahapan Desain Basis Data

Tahapan desain basis data terdiri dari beberapa tahap, yaitu:

Tahap 1 : Pengumpulan data dan analisa

Proses identifikasi dan analisa kebutuhan-kebutuhan data disebut pengumpulan data dan analisa.

Untuk menentukan kebutuhan-kebutuhan suatu sistem database, pertama-tama harus mengenal bagian-bagian lain dari sistem informasi yang akan berinteraksi dengan sistem database, termasuk para pemakai yang ada dan para pemakai yang baru serta aplikasi-aplikasinya. Kebutuhan-kebutuhan

dari para pemakai dan aplikasi-aplikasi inilah yang kemudian dikumpulkan dan dianalisa.

Aktifitas-aktifitas pengumpulan data dan analisa

1. Menentukan kelompok pemakai dan bidang-bidang aplikasinya.
2. Peninjauan dokumentasi yang ada.
3. Analisa lingkungan operasi dan pemrosesan data.
4. Daftar pertanyaan dan wawancara.

Tahap 2 : Perancangan database konseptual

Tujuan dari fase ini adalah menghasilkan c*onceptual 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.

Aktifitas paralel perancangan database secara konseptual

- Perancangan skema konseptual :
  menguji kebutuhan-kebutuhan data dari suatu database yang merupakan hasil dari fase 1, dan menghasilkan sebuah *conceptual database schema* pada DBMS independent model data tingkat tinggi seperti EER (*enhanced entity relationship*) model.
- Perancangan transaksi :
  menguji aplikasi-aplikasi database dimana kebutuhan-kebutuhannya telah dianalisa pada fase 1, dan menghasilkan perincian transaksi-transaksi ini.

Tahap 3 : Pemilihan DBMS

Pemilihan database ditentukan oleh beberapa faktor, diantaranya:

- Struktur data

Jika data yang disimpan dalam database mengikuti struktur hirarki, maka suatu jenis hirarki dari DBMS harus dipikirkan.

- Personal yang telah terbiasa dengan suatu sistem

Jika staf programmer dalam suatu organisasi sudah terbiasa dengan suatu DBMS, maka hal ini dapat mengurangi biaya latihan dan waktu belajar.

- Tersedianya layanan penjual

Keberadaan fasilitas pelayanan penjual sangat dibutuhkan untuk membantu memecahkan beberapa masalah sistem.

- Teknik

Keberadaan DBMS dalam menjalankan tugasnya seperti jenis-jenis DBMS (relational, network, hierarchical, dll), struktur penyimpanan, dan jalur akses yang mendukung DBMS, pemakai, dll.

Tahap 4 : Perancangan database secara logika (pemetaan model data)

Fase selanjutnya dari perancangan database adalah membuat sebuah skema konseptual dan skema eksternal pada model data dari DBMS yang terpilih. Fase ini dilakukan oleh pemetaan skema konseptual dan skema eksternal yang dihasilkan pada fase 2. Pada fase ini, skema konseptual ditransformasikan dari model data tingkat tinggi yang digunakan pada fase 2 ke dalam model data dari DBMS yang dipilih pada fase 3.

Pemetaan diproses dalam 2 tingkat:

- Pemetaan system-independent

Pemetaan ke dalam model data DBMS dengan tidak mempertimbangkan karakteristik atau hal-hal yang khusus yang berlaku pada implementasi DBMS dari
model data tersebut.

- Penyesuaian skema ke DBMS yang spesifik

mengatur skema yang dihasilkan pada langkah 1 untuk disesuaikan pada implementasi yang khusus di masa yang akan datang dari suatu model data yang
digunakan pada DBMS yang dipilih.

Tahap 5 : Perancangan database fisik

Perancangan database secara fisik merupakan proses pemilihan struktur-struktur penyimpanan dan jalur-jalur akses pada file-file database untuk mencapai penampilan yang terbaik pada bermacam-macam aplikasi. Selama fase ini, dirancang spesifikasi-spesifikasi untuk database yang disimpan yang berhubungan dengan struktur-struktur penyimpanan fisik, penempatan record dan jalur akses.

- Response time

Waktu yang telah berlalu dari suatu transaksi database yang diajukan Untuk menjalankan suatu tanggapan. Pengaruh utama pada response time adalah di bawah pengawasan DBMS yaitu : waktu akses database untuk data item yang ditunjuk oleh suatu transaksi. Response time juga dipengaruhi oleh beberapa faktor yang tidak berada di bawah pengawasan DBMS, seperti penjadwalan sistem operasi atau penundaan komunikasi.

- Space Utility

Jumlah ruang penyimpanan yang digunakan oleh file-file database dan struktur-struktur jalur akses.

- Transaction throughput

Rata-rata jumlah transaksi yang dapat diproses per menit oleh sistem database, dan merupakan parameter kritis dari sistem transaksi (misal, digunakan pada pemesanan tempat di pesawat, bank, dll). Hasil dari fase ini adalah penentuan awal dari struktur penyimpanan dan jalur akses untuk file-file database.

Tahap 6 : Implementasi sistem database

- Setelah perancangan secara logika dan secara fisik lengkap, kita dapat melaksanakan sistem database.
- Perintah-perintah dalam DDL dan SDL (storage definition language) dari DBMS yang dipilih, dihimpun dan digunakan untuk membuat skema database dan file-file database (yang kosong) kemudian database tsb dimuat (disatukan) dengan datanya.
- Jika data harus dirubah dari sistem komputer sebelumnya, perubahan-perubahan yang rutin mungkin diperlukan untuk format ulang datanya yang kemudian dimasukkan ke database yang baru.
- Transaksi-transaksi database sekarang harus dilaksanakan oleh para programmmer aplikasi.

Didalam merancang basis data, dapat dilakukan dengan berbasis konseptual, logikal, fisikal dan perpaduan diantara ketiganya.

7.4 Desain Basis Data Logika

Tahapan yang dilakukan saat merancang basis data logikal adalah:

- Menurunkan *Relationship* untuk data model logikal

Pada tahap ini, *Relationship* yang ada pada data model konseptual diturunkan menggunakan DBDL untuk *relational database*. Dengan menggunakan DBDL, kita memberikan nama dari relasi, diikuti dengan *simple attribute* yang dibungkus dengan tanda kurung. Lalu mengidentifikasi *primary key* dan *alternate key* serta *foreign key* jika ada. Untuk menentukan *foreign key,* harus diketahui antara " *parent"* dan *"child" Entity.*

Lalu mendeskripsikan bagaimana relasi diturunkan untuk struktur yang terjadi pada data model konseptual:

- ***Strong Entity types*** Untuk tiap *strong Entity type,* buat sebuah relasi yang terdiri dari semua *simple attribute* pada *Entity* tersebut.
- ***Weak Entity types*** Untuk tiap *weak Entity type,* buat sebuah relasi yang terdiri dari semua *simple attribute* dari *Entity* tersebut. *Primary key* dari *Entity* ini diturunkan sebagian atau sepenuhnya dari tiap *Entity* pemilik sehingga identifikasi dari *primary key* tidak dapat dilakukan hingga setelah semua *Relationship* dengan *Entity* pemilik telah dipetakan.
- ***One-to-many binary Relationship types*** Untuk relasi 1:* *binary Relationship, Entity* yang berada di "*one side"* menjadi *parent,* dan yang berada di *"many side"* menjadi *child.* Pada *child,* terdapat *attribute primary key* dari *parent,* yang dijadikan sebagai *foreign key.*
- ***One-to-one binary Relationship types*** Dalam menentukan representasi 1:1, yang dilihat bukanlah *cardinality*, melainkan *participation*.
- *Mandatory participation* pada kedua sisi dari 1:1 *Relationship* Menggabungkan kedua *Entity* kedalam satu relasi. Salah satu *primary key* dijadikan *primary key* dari relasi baru, dan yang satunya dijadikan *alternate key.*
- *Mandatory participation* pada satu sisi 1:1 *Relationship* Pada sisi yang *mandatory,* dijadikan *child,* sedangkan pada sisi yang *optional,* dijadikan *parent.*
- *Optional participation* pada kedua sisi 1:1 *Relationship* Untuk kasus ini, penentuan *parent* dan *child* tidak diharuskan kecuali didapatkan hubungan yang dapat membantu keputusan.

- ***One-to-one recursive Relationships types*** Untuk 1:1 *recursive Relationship,* tetap menggunakan *Entity* yang sama untuk relasi tersebut, namun ditambahkan satu *attribute* baru yang sebenarnya sama dengan *primary key* , namun diberi nama lain dan dijadikan *foreign key.*
- ***Many-to-many binary Relationship types*** Untuk *: Relationship type,* buat satu relasi baru yang merepresentasikan *Relationship* dan memasukkan *attribute* yang menjadi bagian dari *Relationship.* Lalu, memasukkan *primary key* dari kedua *Entity* untuk dijadikan *foreign keys.*
- ***Multi-valued Attributes*** Merupakan sebuah atribut yang menyimpan banyak nilai- nilai untuk setiap kejadian dari sebuah tipe entitas.

- Validasi relasi dengan normalisasi

Relasi yang telah dibuat dibandingkan dengan hasil normalisasi. Normalisasi bertujuan untuk memastikan bahwa set relasi memiliki jumlah *attribute* minimal yang dibutuhkan untuk menunjang kebutuhan dari perusahaan. Normalisasi melalui beberapa proses untuk memeriksa penggabungan dari *attribute- attribute* pada relasi dengan langkah-langkah 1NF, 2NF, dan 3NF.

- Validasi relasi dengan transaksi pengguna

Tujuan dari langkah ini adalah untuk memvalidasi model data logikal untuk memastikan bahwa model data mampu mendukung transaksi-transaksi yang dibutuhkan, seperti yang ada pada *user's requirements* .

- Memeriksa *integrity constraint*

Integrity constaint adalah *constraint* yang kita harapkan dapat melindungi basis data dari perubahan yang membuat basis

data menjadi tidak lengkap, tidak akurat, dan tidak konsisten. Tipe-tipe *integrity constraint* antara lain:

- ***Required data*** Beberapa *attribute* harus selalu memiliki nilai yang valid, dengan kata lain, *attribute* tersebut tidak dizinkan untuk bernilai *null.*
- ***Attribute domain constraints*** Tiap *attribute* memiliki domain, yaitu suatu kumpulan nilai yang diperbolehkan.
- ***Multiplicity*** *Multiplicity* merupakan *constraint* yang berada pada *Relationship* diantara data dalam basis data.
- ***Entity integrity*** *Primary key* dari suatu *Entity* tidak boleh bernilai *null.*
- ***Referential integrity*** Sebuah *foreign key* menghubungkan tiap *tuple* pada relasi *child* ke *tuple* pada relasi *parent* mengandung nilai *candidate key* yang sama. Maksud dari *refrential integrity* adalah jika pada *foreign key* mengandung nilai, maka nilai tersebut harus merujuk ke *tuple* yang ada pada *parent.* Terdapat dua permasalahan mengenai *foreign key.* Permasalahan pertama adalah menentukan apakah nilai *null* diperbolehkan pada *foreign key.* Masalah kedua adalah bagaimana cara memastikan *referential integrity* . Untuk melakukannya, tentukan *existence constraints* yang mendefinisikan kondisi dimana suatu *candidate key* atau *foreign key* dapat di- *insert, update,* atau *delete.* Ada beberapa strategi yang dapat dipertimbangkan:
    a. *NO ACTION* : Menahan penghapusan dari relasi *parent* jika ada *child tuple* yang terhubung.
    b. *CASCADE* : Saat *parent tuple* dihapus, secara otomatis menghapus tiap *child tuples* yang terhubung. Jika *child tuple* tersebut juga memiliki *child tuple* yang terhubung, maka ia juga akan ikut terhapus.

c. *SET NULL* : Saat *parent tuple* dihapus, nilai *foreign key* pada semua *child tuple* yang terhubung akan secara otomatis berubah menjadi *null.*
d. *SET DEFAULT* : Saat *parent tuple* dihapus, nilai *foreign key* pada semua *child tuple* yang terhubung akan secara otomatis berubah menjadi nilai *default.*
e. *NO CHECK* : Saat *parent tuple* dihapus, tidak melakukan apa-apa.

- ***General constraints*** Terkahir, pertimbangkan *constraint* yang diketahui sebagai *general contraints.* Perbaharuan pada *Entity* mungkin dikendalikan oleh *constraint* yang mengatur transaksi pada dunia nyata.

- Melakukan *review* model data logikal dengan *user*

Data model logikal seharusnya sudah lengkap dan terdokumentasi secara penuh. Namun, untuk mengkonfirmasi kebenarannya, pengguna diminta untuk meninjau data model logikal untuk memastikan bahwa mereka menyatakan kalau model tersebut benar. Jika pengguna tidak puas dengan model tersebut, maka perlu dilakukan pengulangan pada beberapa langkah sebelumnya.

- Mempertimbangkan perkembangan di masa depan

Suatu model data logikal, seharusnya dapat memenuhi perkembangan data di masa depan.

- Pemilihan *Database Management System*

DAFTAR PUSTAKA

Connolly, T. M., Begg, C. E. 2010. Database Systems: A Practical Approach to Design, Implementation, and Management. Boston : Fifth Edition, Pearson Education.

Gat. 2015. Perancangan Basis Data Perputakaan Sekolah dengan Menerapkan Model Data Relasional. Citec J., vol. 2, no. 4, pp. 304–315.

BAB 8
NORMALITAS DATA

Oleh Sri Rezeki Candra Nursari

8.1 Pendahuluan

Normalitas data dilakukan untuk menilai sebaran data pada kelompok data dan merupakan proses pembentukan struktur basis data sehingga sebagian besar *ambiguity* dapat dihilangkan.

8.2 Key dan Atribut Deskriptif

Key merupakan satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*record*) dalam tabel secara unik. Jika suatu atribut dijadikan sebuah *key*, maka tidak diperkenankan ada dua atau lebih data dengan nilai yang sama untuk atribut tersebut. Ada tiga macam *key* yang dapat diterapkan pada suatu tabel, yaitu : (Teorey *et al.*, 2011)

1. *Super-key*
2. *Candidate-Key*
3. *Primary-Key*

8.2.1 Superkey

Merupakan satu atau lebih atribut (kumpulan atribut) yang dapat membedakan setiap baris data (*record*) dalam sebuah tabel secara unik.

Contoh :

- Tabel Mahasiswa yang dapat menjadi *super-key*
  - (npm, nm_mhs, alamat_mhs, tgl_lhr)
  - (npm, nm_mhs, alamat_mhs)

- (npm, nm_mhs)
- (nm_mhs), jika dapat menjamin tidak akan ada nilai yang sama untuk atribut ini, tetapi kalau tidak dapat menjamin maka atribut ini tidak dapat dijadikan *super-key*.
- (npm)

8.2.2 Candidate-Key

Merupakan kumpulan minimal yang dapat membedakan setiap baris data (*record*) dalam sebuah tabel secara unik. Ci*ri-ciri Candidate-key* :

- Tidak dapat berisi atribut
- Kumpulan atribut yang telah menjadi *super-key*
- *candidate-key* pasti merupakan *superkey*, *super-key* belum tentu *candidate-key*
- Sebuah tabel dimungkinkan adanya lebih dari satu *candidate-key*
- Apabila *candidate-key* lebih dari satu, maka salah satu *candidate-key* dapat dijadikan sebagai *primary-key*

Contoh :

- Tabel Mahasiswa yang dapat menjadi *candidate-key*
  - (nim)
  - (nm_mhs), apabila tidak ada nilai yang sama

8.2.3 Primary-Key

Apabila ada lebih dari satu yang menjadi *candidate-key*, maka *key* dapat dijadikan *primary-key*. Dasar pemilihan *primary-key* adalah

- *Key* tersebut lebih sering untuk dijadikan sebagai acuan
- *Key* tersebut lebih ringkas
- Jaminan keunikan *key* tersebut lebih baik

8.2.4 Atribut Deskriptif

Merupakan atribut yang bukan menjadi *primary-key*. Macam-macam atribut yaitu :

1. Atribut Sederhana
   Atribut atomik yang tidak dapat dipisahkan lagi.
2. Atribut Komposit
   Atribut yang dapat dipisahkan menjadi sub atribut yang masing-masing memiliki makna dan dapat dikomposisi menjadi atribut-atribut sederhana.
3. Atribut Bernilai Tunggal
   Atribut yang memiliki paling banyak satu nilai setiap *record*
4. Atribut Bernilai Banyak
   Atribut yang dapat memiliki lebih dari satu nilai, tetapi jenisnya sama.
5. Atribut Harus Bernilai
   Sejumlah atribut pada sebuah tabel yang harus berisi data dan tidak boleh kosong.
6. Atribut Turunan
   Atribut yang nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut atau tabel lain yang berhubungan.

8.3 Normalisasi

Normalisasi digunakan untuk pemodelan *bootom-up*. Normalisasi merupakan bentuk atau memecah tabel dalam suatu database dari *record-record* data yang komplek menjadi lebih sederhana. Normalisasi juga merupakan proses pengelompokan elemen data menjadi tabel yang menunjukkan entitas sekaligus relasinya. Teknik memecah tabel yang komplek menjadi sederhana biasa dikenal dengan istilah **dekomposisi.** Tabel dikategorikan normal, jika memenuhi kriteria sebagai berikut :

- Jika ada *dekomposisi* (penguraian) tabel, *dekomposisi*-nya harus dijamin aman
- Terpeliharanya ketergantungan fungsional pada saat perubahan data
- Tidak melanggar *Boyce-Code Normal Form*

Normalisasi dilakukan untuk :

- Mengurangi *redundancy* data
- Meningkatkan waktu yang dibutuhkan dalam menjalan-kan aktifitas
- Meningkatkan penggunaan kapasitas harddisk
- Peningkatan proses *Input*/*Output*
- Meminimasi pengulangan informasi
- Memudahkan identifikasi entiti

Tujuan utama dari normalisasi adalah untuk mencegah (Silberschatz, Korth and Sudarshan, 2011) :

- *Insertion anomaly* (relasi dengan anomali modifikasi)
- *Delete anomaly* (relasi dengan anomali modifikasi)
- *Update anomaly* (relasi dengan anomali modifikasi)

Beberapa bentuk normal adalah sebagai berikut :

1. Nomal Pertama (*First Normal Form* / 1NF)
2. Nomal Kedua (*Second Normal Form* / 2NF)
3. Nomal Ketiga (*Third Normal Form* / 3NF)
4. *Boyce-Codd* (*Boyce-Codd Normal Form* / BCNF)
5. Nomal Keempat (*Fourth Normal Form* / 4NF)
6. Nomal Kelima (*Fifth Normal Form* / 5NF)
7. *Project-Joinx* (*Project-Join Normal Form* / PJNF)

Normalisasi didasarkan pada konsep ketergantungan fungsional, dapat diilustrasikan sebuah atribut A memiliki ketergantungan fungsional (*functional dependent*) pada atribut B

jika dan hanya jika setiap data atau nilai pada atribut A ditentukan oleh data atau nilai atribut B, dimana nilai atribut A memiliki satu pasang data atau nilai pada atribut B. Atribut B disebut sebagai determinan atau penentu. Bentuk notasi sebagai berikut :

R. B → R. A

8.3.1 Bentuk Normal Pertama (*First Normal Form* / 1NF)

Sebuah tabel dikatakan pada bentuk 1NF ketika masing-masing *record* data pada tabel memiliki pasangan satu record data. Suatu relasi memenuhi 1NF jika dan hanya jika setiap atribut dari relasi hanya memiliki nilai tunggal dalam baris *record*. Bentuk 1NF bertujuan untuk menghilangkan ketergantungan parsial.

8.3.2 Bentuk Normal Kedua (*Second Normal Form* / 2NF)

Suatu relasi memenuhi 2NF jika dan hanya jika memenuhi 1NF. Setiap atribut yang bukan kunci utama tergantung secara fungsional terhadap semua atribut kunci dan tidak hanya sebagian atribut kunci. Bentuk 2NF bertujuan untuk menghilangkan ketergantungan transitif.

- Harus berlaku :
  - A, B ➔ C, D, E artinya
    - A, B ➔ C
    - A, B ➔ D
    - A, B ➔ E

1. Non 2NF

- Jika diketahui R = (**A**, **B**, C, D, E)
  - A, B ➔ C, D
  - B ➔ E
    - Atribut E hanya tergantung secara fungsional terhadap B saja dan bukan terhadap A, B

- Harus berlaku :
    - A, B ➔ C, D
        - Atribut E tidak tergantung secara fungsional terhadap A, B
- Menghasil **2NF** apabila R = (**A, B**, C, D, E) dan berlaku A, B ➔ C, D serta B ➔ E, yang berarti R memenuhi 2NF, maka dapat didekomposisi menjadi dua relasi yaitu :
    - R1 = (A, B, C, D)
    - R2 = (B, E)
- Relasi R1 dan R2 secara keseluruhan memenuhi 2NF, karena semua atribut bukan kunci dan tergantung secara fungsional terhadap semua atribut kunci.

Jika suatu relasi memenuhi 1NF dan relasi tersebut memiliki tepat satu atribut yang membentuk kunci utama, maka relasi tersebut memenuhi **2NF.** Sebuah tabel dikatakan pada Bentuk 2NF jika tabel telah normal sesuai standar normal form pertama (1NF), dan setiap atribut *non-key* pada baris-barisnya memiliki ketergantungan fungsional pada semua *key*, bukan hanya pada sebagian *key* saja. Jika suatu tabel atau relasi mempunyai atribut tunggal sebagai *key*-nya, maka relasi tersebut secara otomatis ada pada form normal kedua. Karena *key* hanya memiliki satu atribut, maka secara default setiap atribut *non-key* bergantung pada semua *key*; tidak ada ketergantungan sebagian. 2NF hanya berhubungan dengan relasi atau tabel yang mempunyai *key-key* komposit.

Langkah melakukan normalisasi **2NF** adalah

1. Cari, temukan dan hilangkan atribut yang hanya memiliki ketergantungan pada sebagian *key* tidak pada semua *key*
2. Letakkan atribut tersebut pada tabel yang berbeda

3. Kelompokkan atribut lainnya

8.3.3 Bentuk Normal Ketiga (*Third Normal Form* / 3NF)

Suatu relasi memenuhi 3NF jika dan hanya jika memenuhi 2NF. Setiap atribut yang bukan kunci tergantung secara fungsional terhadap atribut bukan kunci yang lain dalam relasi tersebut. Bentuk 3NF bertujuan untuk menghilangkan anomali hasil dari ketergantungan fungsional.

Contoh : 3NF

- Jika diketahui R = (**A**, **B**, C, D, E)
  - A, B ➔ C, D, E
  - C ➔ D, E
  - Dalam hal ini R **tidak memenuhi** 3NF karena atribut D, E yang bukan kunci tergantung secara fungsional kepada C yang juga bukan atribut kunci.
- Agar relasi R diatas memenuhi 3NF, maka harus dilakukan di dekomposisi menjadi :
  - R1 = (**A**, **B**, C)
  - R2 = (**C**, D, E)
  - Masing-masing R1 dan R2 **memenuhi** 3NF, maka keseluruhan relasi memenuhi 3NF
  - Suatu relasi yang memenuhi 2NF dan hanya memiliki 1 atribut bukan kunci selalu memenuhi 3NF

Sebuah tabel dikatakan bentuk **3NF** jika setiap atribut *non-key* memiliki ketergantungan fungsional pada *primary-key* dan tidak memiliki ketergantungan transitif (ketergantungan atribut *non-key* pada atribut *non-key* yang lainnya)

Langkah melakukan normalisasi **3NF** adalah

1. Cari dan pisahkan atribut yang memiliki ketergantungan fungsional pada atribut yang bukan *non-key* dan letakkan pada tabel yang berbeda
2. Kelompokkan sisa atribut lainnya pada tabel

8.3.4 Bentuk Boyce-Codd (Boyce-Codd Normal Form / BCNF)

Jika semua ketergantungan fungsional A ➔ B, maka A harus merupakan *superkey* pada tabel tersebut. Jika tidak maka tabel tersebut harus di dekomposisi berdasarkan ketergantungan fungsional yang ada, sehingga A menjadi *superkey* dari tabel-tabel hasil dekomposisi. Bentuk BCNF bertujuan untuk menghilangkan ketergantungan multi nilai.

Suatu relasi memenuhi BCNF jika dan hanya jika setiap determinan yang ada pada relasi tersebut adalah *candidate key*. Determinan adalah atribut dimana satu atau lebih atribut lain tergantung secara fungsional.

Contoh : BCNF

- Jika diketahui R = (A, B, C, D, E)
  - A, B ➔ C, D, E
  - C ➔ D, E
  - Dalam hal ini R **tidak memenuhi** BCNF karena ada determinan yang bukan *candidate key* yaitu B, C, sedangkan A, B adalah determinan yang juga merupakan *candidate key.*
    - A, B ➔ C, D, E
    - C ➔ A, B
    - A, B ➔ A, B, C, D, E
    - Jadi A, B adalah *candidate key*
  - Relasi R dapat diubah menjadi BCNF dengan mendekomposisi R menjadi :
    - R1 = (A, B, C)

- R2 = (B, C, D, E)

BCNF dibutuhkan jika 3NF masih terjadi anomali pada perubahan dan penghapusan data. Ciri- ciri yang dilakukan pada BCNF adalah

- Tabel banyak memiliki *candidate key*
- *Candidate key* memiliki sifat komposit
- *Candidate key overlap*

Langkah melakukan normalisasi **BCNF** adalah

1. Cari dan hilangkan *Candidate key* yang overlap
2. Letakkan sebagian *Candidate key* dan atribut yang memiliki ketergantungan fungsional pada tabel yang berbeda
3. Kelompokkan item lainnya ke dalam sebuah tabel

8.3.5 Bentuk Normal Keempat (*Fourth Normal Form* / 4NF)

Suatu relasi memenuhi 4NF jika dan hanya jika memenuhi BCNF dan tidak memiliki ketergantungan multi nilai atribut. Setiap multi dependencies merupakan functional dependencies. Bentuk 4NF bertujuan untuk menghilangkan anomali yang tersisa. Bentuk 4 NF ini tidak mengandung dua atribut atau lebih yang bernilai banyak.

Contoh : 4NF

- Jika diketahui R = (A, B, C)
  - A ➔ B
  - A ➔ C
  - A ➔ B | C

8.3.6 Bentuk Normal Kelima (*Fith Normal Form* - 5NF)

Suatu relasi memenuhi 5NF jika dan hanya jika setiap dependensi gabungan dalam R. 5NF tidak dapat memiliki sebuah *lossless decomposition* dan memenuhi 4NF. Jika 4NF dibentuk berdasarkan *functional dependencies*, sedangkan 5NF dibentuk berdasarkan konsep *join dependencies*.

Contoh : 5NF

- Jika diketahui R = (V, W, Z) memenuhi dependensi gabungan, dari proyeksi A, B, C merupakan subhimpunan dari atribut R. Dependensi gabungan dinyatakan dengan notasi *(A, B, C) dengan
  - A ➔ X, Y
  - B ➔ W, Z
  - C ➔ Z, V

8.3.6 Bentuk Project-Join (Project-Join Normal Form / PJNF)

PJNF merupakan bentuk lain dari 5NF yang berhubungan dengan ketergantungan relasi antar tabel (*Join Dependency*).

DAFTAR PUSTAKA

Silberschatz, A., Korth, H. F. and Sudarshan, S. (2011) *Database System Concepts - 6th. ed.*, *Database*.

Teorey, T. J. *et al.* (2011) 'Database Modeling and Design, Fifth Edition: Logical Design (The Morgan Kaufmann Series in Data Management Systems)', p. 335. Available at: http://library1.nida.ac.th/termpaper6/sd/2554/19755.pdf.

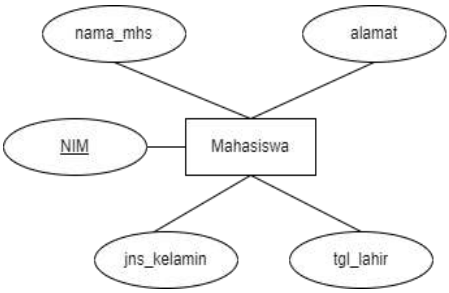
BAB 9
TRANSFORMASI MODEL DATA KE BASIS DATA FISIK

Oleh Noor Azizah

9.1. Transformasi Umum

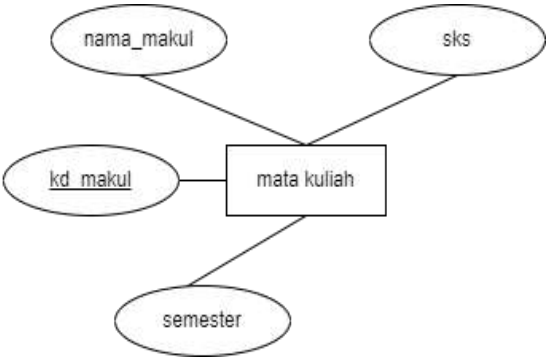
Aturan umum dalam pemetaan model yang telah digambarkan dalam E-R Model menjadi basis data fisik adalah (Fathansyah, 2012) :

1. Setiap himpunan entitas akan diimplementasikan sebagai sebuah tabel. Adapun contohnya sebagai berikut :



Dari diagram ER diatas, berubah menjadi tabel seperti dibawah ini :

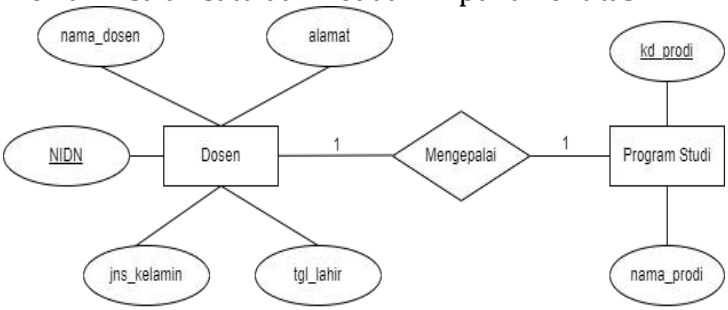
NIM	nama_mhs	alamat	tgl_lahir	jns_kelamin



Dari diagram ER diatas, berubah menjadi tabel seperti dibawah ini :

kd_makul	nama_makul	sks	semester

2. Relasi dengan derajat relasi 1-1, akan direpresentasikan dalam bentuk penambahan atribut-atribut relasi ke tabel yang mewakili salah satu dari kedua himpunan entitas



Dari diagram ER diatas, berubah menjadi tabel seperti dibawah ini :

Tabel Dosen :

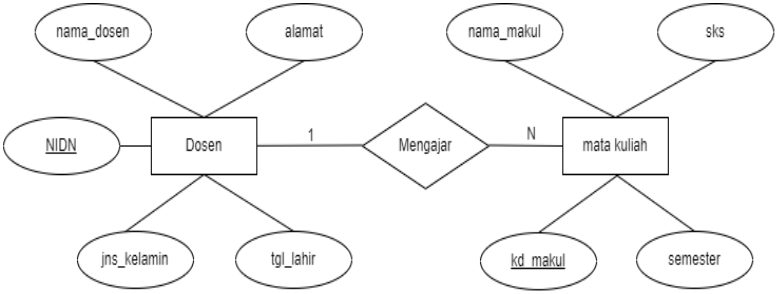
NIDN	nama_dosen	alamat	tgl_lahir	jns_kelamin

Tabel Program Studi :

kd_prodi	nama_prodi	NIDN

Pada setiap relasi 1-1, pasti akan dihadapkan 2 pilihan terkait atribut key mana yang harus dileburkan pada salah satu himpunan entitas. Misalnya, kenapa atribut NIDN yang harus berpindah ke tabel program studi?. Untuk menentukan pilihan yang tepat terkait atribut yang dipindahkan, maka kita harus menentukan derajat relasi minimumnya. Himpunan entitas dosen memiliki derajat relasi minimum 0 karena setiap dosen hanya boleh mengepalai satu program studi (jadi ada dosen yang tidak mengepalai program studi). Sedangkan dari sisi himpunan entitas program studi, program studi hanya boleh dikepalai 1 dosen dan tidak boleh ada program studi yang tidak memiliki kepala. Jadi derajat relasi minimum himpunan entitas program studi adalah 1.

3. Relasi dengan derajat relasi 1-N, akan direpresentasikan dalam bentuk pencantuman atribut key dari himpunan entitas yang berderajat 1 ke himpunan entitas yang berderajat N



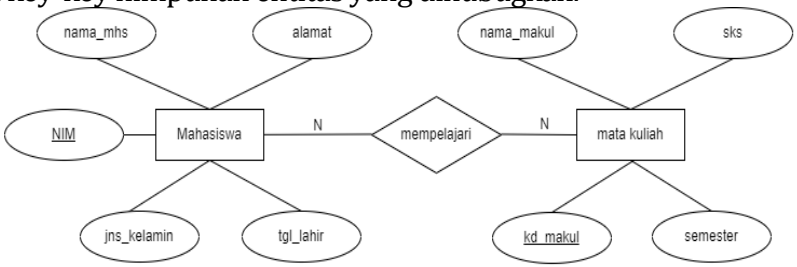
Tabel Dosen :

NIDN	nama_dosen	alamat	tgl_lahir	jns_kelamin

Tabel Mata Kuliah :

kd_makul	nama_makul	sks	semester	NIDN

4. Relasi dengan derajat N-N, akan direpresentasikan dalam bentuk penambahan tabel baru yang mewakili field yang berasal dai key-key himpunan entitas yang dihubugkan.



Tabel Mahasiswa :

NIM	nama_mhs	alamat	tgl_lahir	jns_kelamin

Tabel Mata Kuliah :

kd_makul	nama_makul	sks	semester

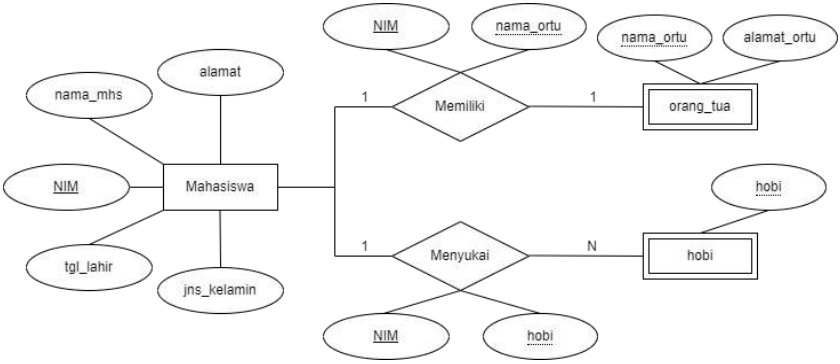
Tabel Nilai (Tabel Mempelajarai) :

NIM	kd_makul	Indeks_nilai

Tabel nilai ini merupakan tabel khusus yang mewakili himpunan relasi

9.2. Penerapan Himpunan Entitas Lemah

Penggunaan himpunan entitas lemah dan sub entitas dalam diagram E-R diterapkan dalam bentuk tabel sebagaimana himpunan entitas kuat. Perbedaannya jika himpunan entitas kuat sudah dapat langsung menjadi sebuah tabel yang utuh dan sempurna, walaupun tanpa melihat relasinya dengan himpunan entitas lain. Sedangkan himpunan entitas lemah dapat ditransformasikan menjadi sebuah tabel dengan menyertakan atribut primary key pada entitas kuat yang berelasi dengannya. Dan atribut tersebut menjadi key dari tabel yang telah terbentuk.



Dari diagram ER diatas, berubah menjadi tabel seperti dibawah ini :

Tabel Mahasiswa :

NIM	nama_mhs	alamat	tgl_lahir	jns_kelamin

Tabel orang_tua :

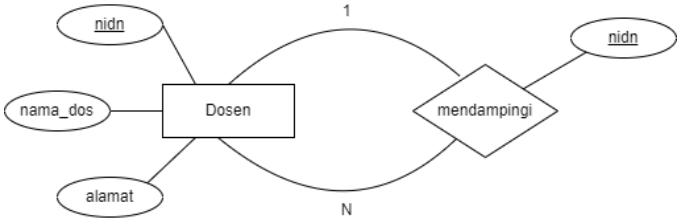
NIM	nama_ortu	alamat_ortu

Tabel hobi :

NIM	hobi

9.3. Penerapan Relasi Tunggal (*Unary Relation*)

Penerapan relasi tunggal (*unary relation*) dari/ke himpunan entitas yang sama dalam diagram E-R tergantung pada derajat relasinya. Untuk relasi satu ke banyak (1-N) dapat diimplementasikan melalui penggunaan field key sebanyak dua kali tapi untuk fungsi yang berbeda. Contohnya, jika adasebuah entitas yang memiliki 2 atribut x dan y dengan atribut x sebagai field key nya, maka relasi tunggal terhadap himpunan tersebut diimplemnetasikan dengan menambhakan atribut x kedalam tabel namun dengan nama yang berbeda karena penamaan field key harus unik.

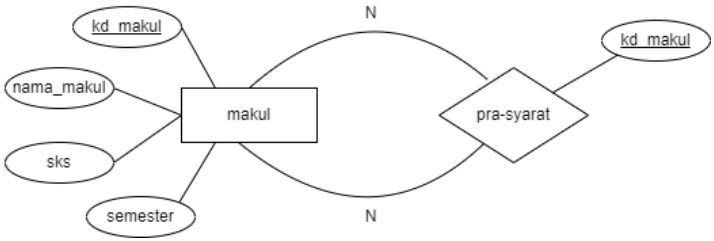


Dari diagram ER diatas, berubah menjadi tabel seperti dibawah ini :

Tabel Dosen :

NIDN	nama_dos	alamat	NIDN_pend

Sedangkan untuk relasi banyak ke banyak (N-N) akan diimpkementasikan melalui pembentukan tabel baru yang merepresentasikan relasi tersebut. Tabel baru ini mendapatkan field dari semua atribut relasi (jika ada) dan atribut key dari entitasnya.



Dari diagram ER diatas, berubah menjadi tabel seperti dibawah ini :
Tabel Makul :

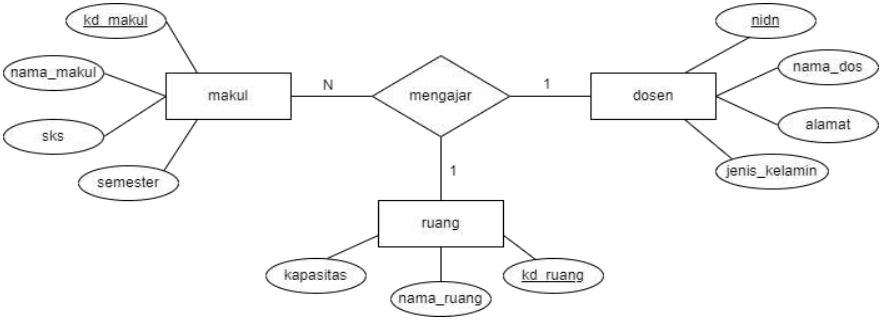
Kd_makul	nama_makul	sks	semester

Tabel Pra-syarat Makul :

Kd_makul	kd_makul_prasyarat

9.4. Penerapan Relasi Multi Entitas (N-*ary Relation*)

Secara umum, relasi multi entitas yang menghubungkan lebih dari 2 entitas (N), akan diimplementasikan menjadi tabel khusus dan entitas-entitas yang terlibat dan himpunan relasi tetap direpresentasikan dalam tabel-tabel terpisah. Namun, jika pada relasi yang menghubungkan N entitas memiliki derajat relasi satu ke banyak (1-N), maka relasi tersebut tidak perlu diwujudkan sebagai tabel khusus dan atribut-atributnya cukup dilekatkan pada entitas tersebut.



Pada diagram E-R diatas, derajat relasi parsial di antara setiap pasang himpunan entitas yang ada, adalah sebagai berikut :

1. Pada relasi mengajar, setiap mata kuliah dapat diajarkan oleh seorang dosen, dan setiap dosen dapat mengajar banyak mata kuliah. Maka derajat relasi dari himpunan entitas dosen-makul adalah 1-N (satu ke banyak)
2. Pada relasi mengajar, setiap mata kuliah hanya dapat diselenggarakan di sebuah ruang yang telah ditentukan dan setiap ruang pada saat yang berbeda dapat digunakan untuk beberapa mata kuliah, maka derajat relasi himpunan entitas ruang-makul adalah 1-N (satu ke banyak)
3. Pada relasi mengajar, setiap ruangan dapat digunakan banyak dosen (untuk mengajar banyak mata kuliah) dan setiap dosen dapat menggunakan banyak ruangan karena mengajar lebih dari satu mata kuliah. Maka derajar relasi himpunan entitas dosen-ruang adalah N-N (banyak ke banyak)

Dari hasil pengamatan tersebut, kita dapat menyetujui bahwa semua derajat relasi antara entitas dosen atau ruang dengan entitas makul selalu 1-N, dengan demikian relasi mengajar tidak perlu diimplementasikan ke tabel khusus, tetapi atribut-atributnya

dilekatkan pada entitas makul. Berikut tabel yang terbentuk dari hubungan relasi tersebut :

Tabel Dosen :

nidn	nama_dos	alamat	jenis_kelamin

Tabel Ruang

kd_ruang	nama_ruang	kapasitass

Tabel Makul :

kd_makul	nama_makul	sks	semester	nidn	kode_ruang	waktu

Catatan : 3 buah field diatas (nidn, kode_ruang, waktu) merupakan field yang mewakili relasi mengajar

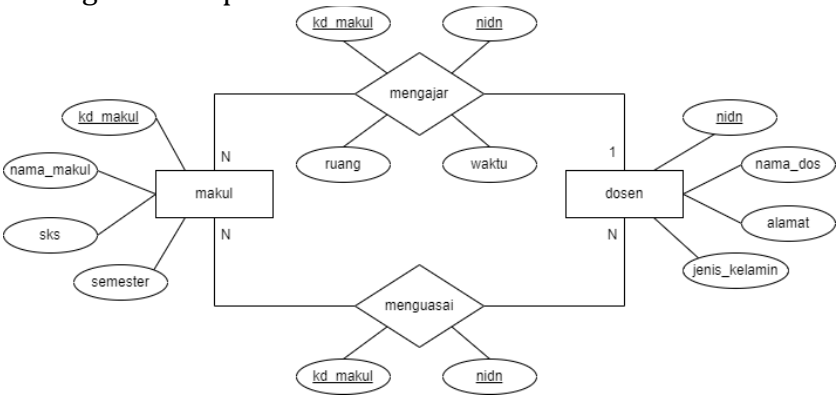
Jika ternyata di kemudian hari, suatu mata kuliah (dengan jumlah sks yang besar) dapat dilaksanakan lebih dari satu kali dalam seminggu, maka dimungkinkan dapat diselenggarakan di ruang yang berbeda untuk satu mata kuliah, maka derajar relas antara himpunan entitas ruang dan makul bukan lagi 1-N namun menjadi N-N. Jika kenyataan ini harus diakomodasi, maka tabel makul tetap sebagaimana semula (dengan 4 field yaitu kode_makul, nama_makul, sks, semester) dan relasi diatas harus diimplementasikan sebagai sebuah tabel khusus seperti berikut :

Tabel mengajar / jadwal

kode_makul	nidn	kode_ruang	waktu

9.5. Penerapan Relasi Ganda (*Redundant Relation*)

Implementasi dari penerapan relasi ganda diantara 2 himpunan entitas adalah dengan melihat derjat relasi pada masing-masing relasi tanpa terikat satu sama lain.



Pada diagram E-R diatas dapat kita lihat bahwa derajat relasi mengajar adalah 1-N, maka field nidn pada tabel dosen akan ditambahkan pada tabel makul. Kemudian untuk relasi menguasai, derajat relasinya N-N maka akan diimplementasikan pada tabel khusus. Yaitu tabel menguasai.

Tabel Dosen :

NIDN	nama_dos	alamat	jenis_kelamin

Tabel Mata Kuliah :

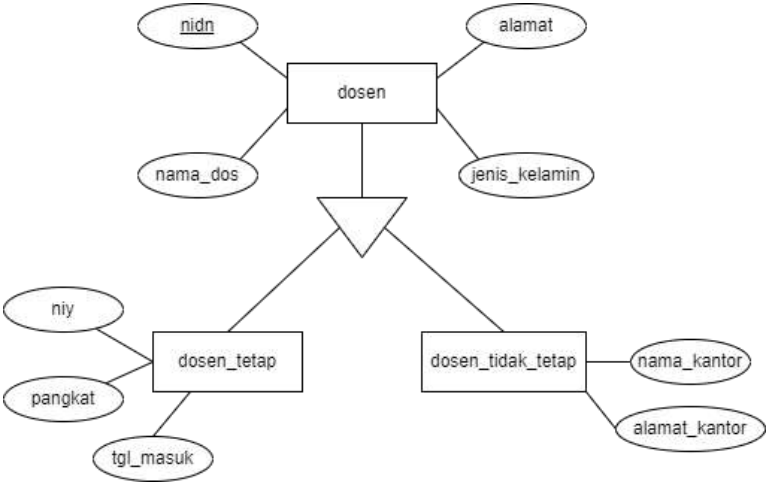
kd_makul	nama_makul	sks	semester	nidn

Tabel Menguasai

NIDN	Kode_makul

9.6. Penerapan Relasi Spesialisasi dan Generalisasi

Spesialisasi terhadap sebuah himpunan entitas akan menghasilkan sejumlah entitas baru. Satu himpunan entitas kuat akan menjadi acuan bagi himpunan entitas lainnya (entitas lemah). Maka pengimplementasiannya, masing-masing akan berubah menjadi tabel dan untuk tabel dari entitas lemah, harus menyertakan atribut key dari entitas kuat.



Tabel Dosen :

nidn	nama_dos	alamat	jenis_kelamin

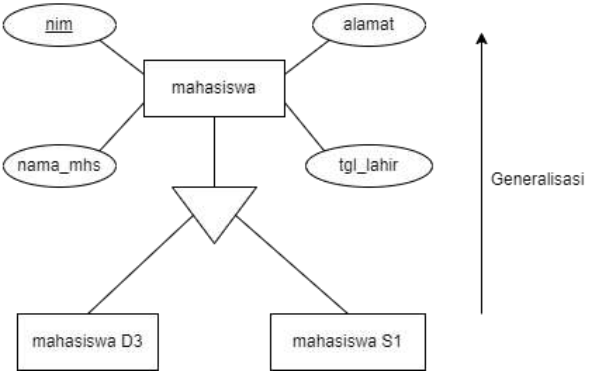
Tabel Dosen Tetap :

nidn	niy	pangkat	tgl_masuk

Tabel Dosen Tidak Tetap :

nidn	nama_kantor	alamat_kantor

Generalisasi dilakukan dengan mengabaikan perbedaan beberapa himpunan entitas yang memang memiliki banyak kesamaan. Berlawanan dengan spesialisasi, pada tahap implementasi generalisasi justru akan menyusutkan jumlah himpunan entitas menjadi sebuah tabel saja. Untuk tetap mengakomodasi adanya perbedaan itu, maka ditabel tersebut ditambahkan sebuat atribut yang nantinya akan diisi dengan kode khusus yang menyatakan perbedaan tersebut.



Tabel Mahasiswa :

nim	nama_mhs	alamat	tgl_lahir	prodi

Catatan : field prodi merupakan field tambahan untuk mengakomodasi perbedaan kelompok entitas.

DAFTAR PUSTAKA

Fathansyah. (2012). *Basis Data* (2nd ed.). Informatika Bandung.

BAB 10
QUERY PADA MY SQL

Oleh Muhamad Hadi Saputra

10.1 Pendahuluan

Perintah atau biasa yang disebut query pada mySQL memiliki arti yaitu sebuah intruksi dengan tujuan untuk mengelola database beserta tabel didalam database my SQL. Dalam mempelajari Query mySQL, ini merupakan hal dasar yang sangat penting dalam mempelajari ilmu sistem basis data. Query memiliki arti yang sering disebut dengan MySQL *(Structured Query Language).* Dimana mempunyai arti yaitu sebagai bahasa yang bertujuan untuk mengakses sebuah data di dalam sistem basis data relasional. Query mempunyai sebuah kelebihan dalam mengatur data – data yang dapat ditampilkan sesuai keinginan user. Query juga dapat saling berinteraksi dimana bahasa query tersebut merupakan bahasa standar yang digunakan dalam mengelola banyak database modern seperti mysql, sql server, oracle dimana menggunakan bahasa SQL.

Query MySQL terdapat 3 jenis secara umum antara lain:

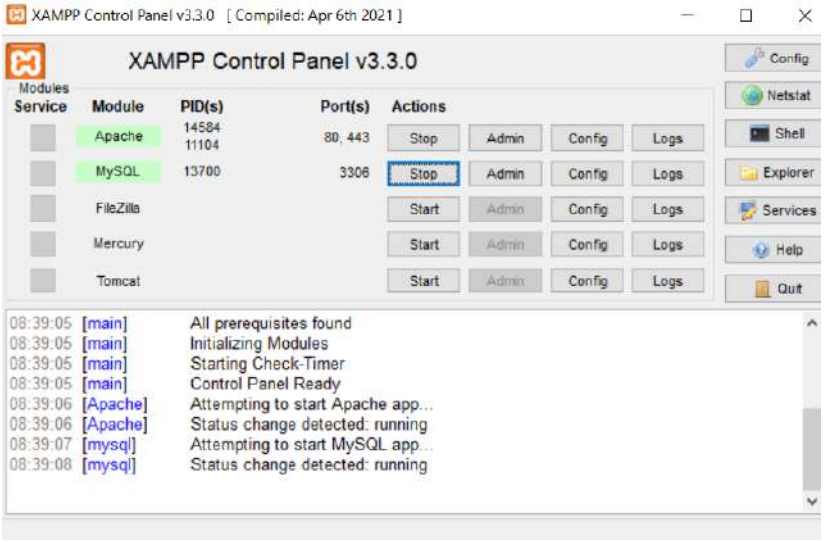
1. Query DDL (*Data Definition Language)*
2. Query DML (*Data Manipulation Language)*
3. Query DCL (*Data Control Language)*

Ketiga query tersebut memiliki sebuah manfaat dimana untuk memberikan pengertian terhadap objek yang ada di sebuah sistem basis data seperti dalam pembuatan tabel, memanipulasi database, hingga dapat mengontrol database.

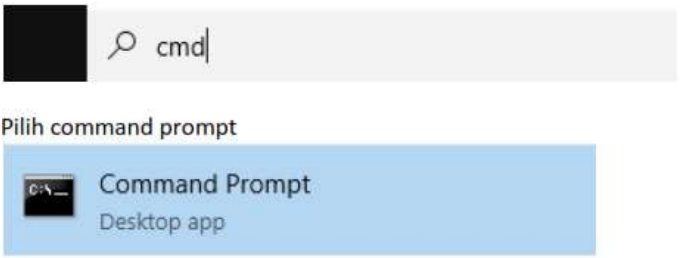
10.2 Dasar Penggunaan

Tahap pertama kali harus sudah menginstall MySQL pada komputer kita. Berikut cara yang dapat diikuti :

a) Intsalasi XAMPP



b) Setelah berhaisl diinstall, tampilan xampp akan seperti gambar diatas,jangan lupa untuk klik start pada MySQL agar nantinya dapat berjalan di cmd.



c) Setelah terbuka jalankan MySQL dengan mengetik

```
C:\Users\muham>cd c:\xampp\mysql\bin
```

d) Lokasi tempat penyimpanan file xampp disesuaikan yang ada pada komputer masing-masing. Setelah itu jalankan MySQL dengan mengetikan perintah :

```
c:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.4.24-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Untuk password dibiarkan default atau bawaan dari awal instalasi nya kosong, jadi langsung klik enter , jika MySQL berjalan maka akan mengeluarkan output seperti gambar diatas.

10.3 Query DDL (*data definition language*)

DDL merupakan suatu proses pembuatan sebuah ruang dari data. Data tersebut akan disimpan pada sebuah field dan field tersebut akan disimpan didalam sebuah tabel yang akan disimpan di database.

a) Menampilkan dan membuat database

Agar seluruh database yang ada di MySQL dapat kita lihat , kita dapat gunakan perintah seperti gambar dibawah ini

SHOW DATABASES;

```
MariaDB [(none)]> show databases;
+--------------------------+
| Database                 |
+--------------------------+
| akademik                 |
| db_akademik              |
| db_formuser              |
| db_hadi                  |
| db_muhamadhadi           |
| db_muhammadrizky         |
| db_toko_buku_hadi        |
| db_toko_buku_mrizky      |
| dbmuhamadhadi_formbiodata |
| dbuts_hadi               |
| information_schema       |
| mysql                    |
| performance_schema       |
| phpmyadmin               |
| test                     |
+--------------------------+
15 rows in set (0.001 sec)
```

b) Menghapus dan mengaktifkan database

Gunakan perintah berikut ini

CREATE DATABASE nama_database;

```
MariaDB [(none)]> CREATE DATABASE dblatihan1;
Query OK, 1 row affected (0.002 sec)
```

c) Mengaktifkan database

Untuk memilih atau mengaktifkan gunakan perintah berikut ini :

USE nama_database;

```
MariaDB [(none)]> USE dblatihan1;
Database changed
MariaDB [dblatihan1]>
```

Bisa dilihat perubahan yang terjadi ketika mengaktifkan database yang dipilih dimana sebelumnya menunjukan none sekarang bisa berganti menjadi dblatihan.

d) Membuat dan menampilkan struktur tabel

Dalam membuat struktur tabel berdasarkan desain yang sudah direncakan , pastikan sebelum membuat tabel kita harus memilih database dengan benar. Untuk membuat tabel gunakan perintah sebagai berikut :

CREATE TABLE nama_tabel (nama_field Tipe data);

```
MariaDB [dblatihan1]> create table tblatihan
    -> (
    -> nama varchar(50)not null,
    -> PRIMARY KEY(nama)
    -> );
Query OK, 0 rows affected (0.128 sec)
```

Untuk menampilkan tabel gunakan perintah

SHOW TABLES;

```
MariaDB [dblatihan1]> show tables;
+----------------------+
| Tables_in_dblatihan1 |
+----------------------+
| tblatihan            |
+----------------------+
1 row in set (0.001 sec)
```

Untuk menampilka struktur tabel gunakan perintah DESC nama_tabel;

```
MariaDB [dblatihan1]> desc tblatihan;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| nama  | varchar(50) | NO   | PRI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.016 sec)
```

e) Menambah kolom dan merubah tipe data

Jika sewaktu-waktu terdapat sebuah kekurangan pada tabel, kita bisa menambahkan kolom lagi dengan perintah berikut

ALTER TABLE nama_table ADD nama_field TIPE DATA yang digunakan;

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan ADD Stock int(10)not null;
Query OK, 0 rows affected (0.032 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| nama  | varchar(50) | NO   | PRI | NULL    |       |
| Stock | int(10)     | NO   |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.010 sec)
```

Untuk merubah tipe data pada field stock gunakan perintah sebagai berikut

ALTER TABLE nama_table MODIFY nama_field TIPE DATAyang digunakan;

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan MODIFY Stock varchar(10)not null;
Query OK, 0 rows affected (0.130 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| nama  | varchar(50) | NO   | PRI | NULL    |       |
| Stock | varchar(10) | NO   |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.071 sec)
```

f) Menghapus dan merubah nama kolom

Untuk menghapus sebuah field/kolom gunakan perintah sebagai berikut

ALTER TABLE nama_tabel DROP COLUMN nama_filed;

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan DROP COLUMN Stock;
Query OK, 0 rows affected (0.014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| nama  | varchar(50) | NO   | PRI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.009 sec)
```

Untuk merubah nama field/kolom gunakan perintah :
ALTER TABLE nama_tabel CHANGE field_lama field_baru TIPE DATA;

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan CHANGE nama nama_barang varchar(50)not null;
Query OK, 0 rows affected (0.014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nama_barang | varchar(50) | NO   | PRI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
1 row in set (0.012 sec)
```

g) Menghapus & Menambahkan primary key

Untuk menghapus primary key gunakan perintah berikut

ALTER TABLE nama_table DROP PRIMARY KEY;

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan DROP PRIMARY KEY;
Query OK, 0 rows affected (0.298 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nama_barang | varchar(50) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
1 row in set (0.024 sec)
```

Untuk menambahkan primary key nya kembali cukup masukkan perintah sebagai berikut

ALTER TABLE nama_table ADD PRIMARY KEY(nama_field);

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan ADD PRIMARY KEY(nama_barang);
Query OK, 0 rows affected (0.088 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nama_barang | varchar(50) | NO   | PRI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
1 row in set (0.010 sec)
```

h) Menampilkan Engine yang di gunakan

Perlu diketahui didalam membuat basis data terdapat 2 mesin yang biasa digunakan yaitu seperti *MYISAM* dan *InnoDB.* Untuk database relasional hanya menggunakan

InnoDB, sedangkan MYISAM hanya untuk menerima *FOREIGN KEY.*

Untuk menampilkan engine yang sedang digunakan menggunakan perintah sebagai berikut :

```
MariaDB [dblatihan1]> SHOW CREATE TABLE tblatihan;
+-----------+-----------------------------------------
-----------------------------+
| Table     | Create Table
                               |
+-----------+-----------------------------------------
-----------------------------+
| tblatihan | CREATE TABLE `tblatihan` (
  `nama_barang` varchar(50) NOT NULL,
  PRIMARY KEY (`nama_barang`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
+-----------+-----------------------------------------
-----------------------------+
1 row in set (0.001 sec)
```

i) Menambahkan kolom setelah kolom

Untuk menambahkan sebuah field setelah field berikutnya yaitu menggunakan perintah

ALTER TABLE nama_table ADD nama_field TIPE DATA AFTER nama_field_lama;

```
MariaDB [dblatihan1]> ALTER TABLE tblatihan ADD jumlah int(10)not null AFTER nama_barang;
Query OK, 0 rows affected (0.042 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [dblatihan1]> desc tblatihan;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nama_barang | varchar(50) | NO   | PRI | NULL    |       |
| jumlah      | int(10)     | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
2 rows in set (0.010 sec)
```

10.4 Query DML (*data manipulation language*)

a) Insert

Insert bertujuan untuk memasukkan data pada semua field tabel yang dibuat. Sebelum melakukan insert periksa terlebih dahulu tabel yang akan ingin dimasukkan datanya. Kemudian masukkan perintah

INSERT INTO nama_table VALUES (kolom, semua kolom);

```
MariaDB [dblatihan1]> INSERT INTO tblatihan VALUES
    -> ('Sabun','2');
Query OK, 1 row affected (0.021 sec)
```

Agar dapat dipastikan apakah data tersebut sudah masuk di table tersebut kita bisa menggunakan perintah sebagai berikut

SELECT*FROM nama_table;

```
MariaDB [dblatihan1]> SELECT*FROM tblatihan;
+-------------+--------+
| nama_barang | jumlah |
+-------------+--------+
| Sabun       |      2 |
+-------------+--------+
1 row in set (0.019 sec)
```

b) Update

Agar bisa memperbaruhi data atau update caranya gunakan perintah berikut ini

UPDATE nama_table SET nama_field=isi_field WHERE nama_field = baris_yang_dirubah;

```
MariaDB [dblatihan1]> UPDATE tblatihan SET nama_barang='Minyak' WHERE nama_barang='Sabun';
Query OK, 1 row affected (0.005 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [dblatihan1]> SELECT*FROM tblatihan;
+-------------+--------+
| nama_barang | jumlah |
+-------------+--------+
| Minyak      |      2 |
+-------------+--------+
1 row in set (0.013 sec)
```

c) Select

Untuk select semua kolom dari table atau view gunakan perintah beirkut ini

SELECT*FROM nama_tabel atau nama_view;

```
MariaDB [dblatihan1]> SELECT*FROM tblatihan;
+--------------+--------+
| nama_barang | jumlah |
+--------------+--------+
| Minyak       |      2 |
+--------------+--------+
1 row in set (0.001 sec)
```

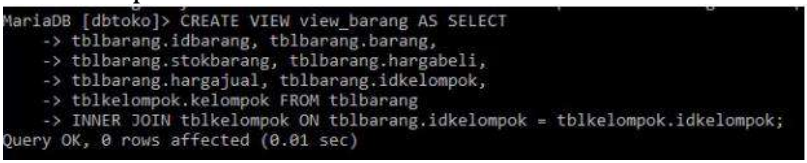
Untuk select sebagaian dimana hanya menampilkan kolom yang dipilih menggunakan perintah berikut ini

SELECT nama_kolom, nama_kolom FROM nama_tabel;

```
MariaDB [dblatihan1]> SELECT nama_barang FROM tblatihan;
+-----------------+
| nama_barang     |
+-----------------+
| Minyak          |
| Pewangi pakaian |
| Rinso           |
| Shampo          |
+-----------------+
4 rows in set (0.050 sec)
```

d) View

View memiliki fungsi untuk menggabungkan sebuah tabel master dan tabel transaksi menjadi tabel baru agar mudah dipahami..



e) Trigger

Trigger adalah suatu perintah dimana letaknya dapat dijalankan disetiap kejadian seperti INSERT, UPDATE, DELETE, FUNCTION, PROCEDURE .AFTER INSERT (setelah INSERT) pada tabel yang dimaksud

- BEFORE INSERT (sebelum INSERT) pada tabel yang dimaksud
- AFTER DELETE (setelah DELETE) pada tabel yang dimaksud
- BEFORE DELETE (Sebelum DELETE) pada tabel yang dimaksud
- AFTER UPDATE (Setelah UPDATE) pada tabel yang dimaksud
- BEFORE UPDATE (Sebelum UPDATE) pada tabel yang dimaksud.

Pembuatan Trigger

Setelah semua konsep , proses, dan pemeriksaan tabel telah dilakukkan sekarang saatnya membuat trigger.

CREATE TRIGGER nama_trigger AFTER INSERT ON nama_table_yang_digunakan FOR EACH ROW perintah_trigger;

Menampilkan Trigger

```
MariaDB [dbtoko]> SHOW TRIGGERS;
```

Menghapus Trigger

```
MariaDB [dbtoko]> DROP TRIGGER kurang_total;
Query OK, 0 rows affected (0.01 sec)
```

f) Delete Record

Untuk menghapus record menggunakan perintah

Sebagian record :

DELETE FROM nama_table WHERE record_yang ingin dihilangkan;

Semua Record :

DELETE FROM nama_tabel;

10.5 Query DCL (*data control language*)

Merupakan perintah yang dapat digunakan untuk menjaga keamanan basis data.

a) **Mengaktifkan database MySQL**
   USE MySQL;
b) **Menampilkan data pada kolom host, user dan password**
   Select HOST, USER, Password, FROM USER;
c) **Membuat User Baru**
   CREATE USER ‚nama_user' IDENTIFIED BY 'Password';
d) **Menghapus user**
   DELETE FROM USER WHERE ='nilai";
e) **Mengganti Password**
   UPDATE USER SET PASSOWRD =('xxx') WHERE USER='root';
f) **Menampilkan semua isi dari tabel user**

SELECT*FROM USER;

g) Membuat Privileges

INSERT INTO USER (HOST, USER, PASSWORD, SELECT_PRIV, INSERT_PRIV, UPDATE_PRIV, DELETE_PRIV) VALUES (‘LOCALHOST‘,‘username‘,‘PASSWORD (‘nilai‘);

h) Perintah Flush

FLUSH PRIVILLEGES;

i) Perintah GRANT

- GRANT hak_akses ON nama_tabel TO pemakai;
- GRANT hak_akses on nama_database. *TO pemakai;
- GRANT ALL PRIVILEGES ON database_name.* TO ‘username‘ INDENTIFIED BY ,‘Password‘.

j) Perintah REVOKE

Menghapus batasan hak akses database dan table

REVOKE hak_akses ON nama_tabel FROM user;

Menghapus semua hak akses

REVOKE ALL PROVILEGES ON nama_database.nama_tabel FROM ADMIN;

DAFTAR PUSTAKA

https://mariadb.org/

Saputro, Haris (2012) Modul Pembelajaran Praktek Basis Data (MySQL)

Solichin, Achmad (2010) MYSQL 5: Dari Pemula Hingga Mahir

Jayanti, Ari, Dewi, Ketut N, Sumiari, Kadek N. 2018. Teori Basis Data. Yogyakarta: Andi.

Widodo, Agus W, Kurnianingtyas D. 2017. NoSistem Basis Data. Malang: UB Press.

BIODATA PENULIS

Jamaludin,M.Kom.
Dosen Politeknik Ganesha Medan

Seorang praktisi dan akademisi yang lahir di Bah Jambi, 11 Januari 1973 memiliki latar belakang sarjana teknik informatika dari Sekolah Tinggi Poliprofesi Medan dan magister komputer dari Universitas Sumatera Utara dengan peminatan komputer. Saat ini bertugas sebagai dosen di Politeknik Ganesha Medan sejak tahun 2013 sampai sekarang. Aktif dalam penelitian dan pengabdian kepada masyarakat untuk merealisasikan kerja dosen dalam Tri Dharma Perguruan Tinggi. Mulai aktif menulis buku sejak September 2019 sampai sekarang. Kemudian aktif juga menulis artikel di media cetak/online mulai sejak September 2020 sampai sekarang. Tema yang digemari dalam penulisan buku adalah komputer, bisnis online, technopreneurship dan pendidikan.

BIODATA PENULIS

Khairunnisa Samosir, S. Kom., M. Kom.
Dosen Program Studi Ilmu Komputer Universitas Graha Nusantara

Penulis lahir di Tanjungbalai pada tanggal 01 Mei 1994. Penulis merupakan dosen tetap di Universitas Graha Nusantara Program Studi Ilmu Komputer. Penulis telah menyelesaikan pendidikan S2 di Magister Teknik Informatika Universitas Putra Indonesia YPTK Padang.

BIODATA PENULIS

Wahyuddin S, S. Kom., M. Kom.
Dosen STMIK AMIKA Soppeng

Wahyuddin S. was born at Malaka-Bone-Sulawesi Selatan in 1992. In 2011 he attended STMIK Dipanegara Makassar and was completed in 2015. He was completed after attending 7 semesters and active on an XPcom (Extreme Programmer Computer) campus organization. He was also active as a lecturer assistant for three semesters and taught several courses on programming. He continued his Master of Information systems at UNIKOM Bandung in 2016 and was completed in April 2019. He worked as a lecturer at a campus (STMIK AMIKA Soppeng) 2019 to present and also a Freelance Web Programmer. Has competence in the field of software engineer, application developer, multimedia, web developer, network security, and data analyst.

BIODATA PENULIS

Elmi Devia, S. Kom., M. Kom.
Staf Dosen Program Studi Sistem Informasi
Fakultas Teknik Universitas Krisnadwipayana

Penulis lahir di Padang Panjang tanggal 26 April 1979. Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik, Universitas Krisnadwipayana. Menyelesaikan pendidikan S1 pada Jurusan Sistem Informasi dan melanjutkan S2 pada Jurusan Teknik Informatika konsentrasi Sistem Informasi. Penulis menekuni bidang Sistem Penunjang Keputusan, Sistem Cerdas, dan Basis Data.

BIODATA PENULIS

Leo Willyanto Santoso
Dosen Program Studi Informatika
Universitas Kristen Petra Surabaya

Penulis merupakan dosen tetap di Program Studi Informatika Universitas Kristen Petra Surabaya. Penulis telah menyelesaikan pendidikan S2 di University of Melbourne, Australia.

BIODATA PENULIS

Yuniansyah
Dosen

Mengenal Ilmu Komputer Pada Tahun 1995 dimana penulis mendapatkan kesempatan melanjutkan pendidikan Strata satu di Universitas Bina Darma Palembang dan selesai pada tahun 1999. Pada saat kuliah juga selama 2 tahun (1997-1999) sempat menjadi Asisten Dosen di Laboratorium komputer Universitas Bina Darma untuk beberapa matakuliah pemrograman.

Pada awal tahun 2002 melanjutkan pendidikan di Program Studi Magister Ilmu Komputer Fakultas Ilmu Matematika dan Ilmu Pengetahuan Alam (F-MIPA) Universitas Gadjah Mada – Yogyakarta dan selesai pada akhir tahun 2013. Selama di Yogyakarta juga penulis berkesempatan menjadi Dosen di salah satu Perguruan Tinggi yang ada di Yogyakarta

Selepas meraih gelar Magister Komputer penulis menjadi dosen di beberapa Perguruan Tinggi swasta dan Universitas Negeri di Kota Palembang. Penulis juga pernah menjadi Dosen di Kota Lubuk Linggau, Batam, dan Duri

Saat ini penulis menjadi dosen praktisi di salah satu perguruan tinggi ternama di Kota Pelembang Penulis ini dapat dihubungi melalui email: yuniansyah.mr@gmail.com serta dan dapat juga melalui WA / Telegram : 0812 3516 8181

BIODATA PENULIS

Ir. Junaidi, M.Kom.
Staf Dosen Program Studi Sistem Informasi
Universitas Krisnadwipayana

Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Teknik, Universitas Krisnadwipayana. Menyelesaikan pendidikan S1 pada Program Studi Manajemen Informatika/Sistem Informasi dan melanjutkan S2 pada Program Studi Teknik Informatika dengan Konsentrasi Teknologi Informasi. Penulis menekuni bidang Sistem Basis Data, Kecerdasan Buatan, Sistem Cerdas, dan Sistem Pendukung Keputusan.

BIODATA PENULIS

Sri Rezeki Candra Nursari
Staf Dosen Program Studi Teknik Informatika Fakultas Teknik
Universitas Pancasila

Penulis lahir di Surabaya – Jawa Timur tanggal 22 Juli 1966 Penulis adalah dosen tetap pada Program Studi Teknik Informatika Fakultas Teknik Universitas Pancasila. Menyelesaikan pendidikan S1 pada Program Studi Teknik Elektro di IKIP PGRI Madiun – Jawa Timur. Melanjutkan S2 Teknik Informatika pada STTI Benarif Jakarta. Penulis menekuni bidang Artificial Intelligence dan Software Engineering.

BIODATA PENULIS

Noor Azizah
Dosen Program Studi Sistem Informasi
Universitas Islam Nahdlatul Ulama Jepara

Penulis lahir di Kudus pada tanggal 07 Januari 1990. Penulis merupakan dosen tetap di Universitas Islam Nahdlatul Ulama Jepara Program Studi Sistem Informasi. Penulis telah menyelesaikan pendidikan S2 di Magister Sistem Informasi Universitas Diponegoro.

BIODATA PENULIS

Muhamad Hadi Saputra, lahir pada 04 Juni 2001 di Palembang, Sumatera Selatan. Sekarang sedang menyelesaikan studinya di Institut Teknologi dan Bisnis Palcomtech di Kota Palembang mengambil jurusan Sistem Informasi. Memiliki hobi yaitu bermain game genre sepak bola, suka mempelajari teknologi. Selama menjadi mahasiswa, penulis aktif dalam mengikuti pelatihan online diantaranya pelatihan Junior Web Developer yang diadakan oleh kominfo. Penulis juga mempunyai pengalaman bekerja sebagai Assisten Quality Control di sebuah percetakan digital printing yang ada di kota Palembang selama kurang lebih 6 bulan. Penulis juga mengikuti kegiatan kemahasiswaan yaitu kegiatan programming.

Penulis dapat dihubungi melalui surat elektronik atau email muhamad.hadisaputraa@gmail.com