Penulis:

Ardhin Primadewi, S. Si, M.TI
R. Arri Widyanto, S.Kom., M.T

Editor:

Pristi Sukmasetya, S.Komp., M.Kom

Teori dan Praktik Basis Data

"Mengkaji Basis Data dari Sudut Pandang Praktisi"

UNIMMA PRESS
Universitas Muhammadiyah Magelang

Teori dan praktik Basis Data

"Mengkaji basis data dari sudut pandang praktisi"

Penulis:

Ardhin Primadewi, S. Si, M. TI

R. Arri Widyanto, S.Kom., M.T.

Editor:

Pristi Sukmasetya, S.Komp., M. Kom

UNIMMA PRESS
UNIVERSITAS MUHAMMADIYAH MAGELANG

UU No 28 tahun 2014 tentang Hak Cipta

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

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

Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;

Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;

Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan pengumuman sebagai bahan ajar; dan

Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).

Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf *key*, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp. 500.000.000,00 (lima ratus juta rupiah).

Teori dan praktik Basis Data

"Mengkaji basis data dari sudut pandang praktisi"

ISBN: 978-623-7261-69-8

Penulis:
Ardhin Primadewi, S. Si, M. TI
Arri Widyanto, S.Kom., M.T.

Editor:
Pristi Sukmasetya, S.Komp., M. Kom

Penerbit:
UNIMMA PRESS
Gedung Rektorat Lt. 3 Kampus 2 Universitas Muhammadiyah Magelang
Jalan Mayjend Bambang Soegeng km.05, Mertoyudan, Magelang 56172
Telp. (0293) 326945
E-Mail: unimmapress@ummgl.ac.id

Cetakan I, April 2022

Kata Pengantar

Guna meningkatkan pemahaman mahasiswa terhadap basis data sebagai salah satu mata kuliah dalam kerangka kurikulum Teknik Informatika. Maka kami menulis buku yang berjudul "Teori dan Praktik Basis Data".

Meskipun bertujuan meningkatkan pemahaman mahasiswa terhadap basis data yang merupakan mata kuliah di kerangka kurikulum Teknik informatika di Universitas Muhammadiyah Magelang, namun buku ini juga dapat digunakan sebagai referensi bagi mereka yang sedang berkecimpung dalam dunia sistem informasi. Buku ini memberikan wawasan basis data dalam sebuah sistem informasi, dan wawasan menyeluruh terkait normalisasi.

Buku ini menitik beratkan pada pembahasan keterampilan melakukan normalisasi pada sebuah studi kasus real dilanjutkan dengan pembuatan Enhanced Entity Relationship dan mengimplementasikannya pada server *database*.

Dari buku ini diharapkan pembaca mampu mengetahui konsep basis data, mengetahui pembuatan Enhanced Entity Relationship Diagram dan script Data Definition Language.

Kami mengucapkan terimakasih kepada semua pihak yang telah memberikan dukungan dan bantuan dalam pembuatan buku ini. Kami menyadari, buku ini masih banyak kekurangan dalam segi substansi maupun penyajiannya. Untuk itu, kami mengharapkan saran dari para pembaca. Semoga buku ini bermanfaat.

Magelang, April 2022

Penulis

Daftar Isi

Daftar Gambar

Daftar Tabel

BAB 1

PENDAHULUAN

1.1. Pengantar Pengembangan *Database*

Basis data merupakan hal yang sangat vital bagi banyak organisasi dan perusahaan. Hal ini terjadi karena tuntutan organisasi atau tuntutan bisnis yang harus cepat dalam menyediakan data. Pemrosesan basis data ini menjadi alat yang diperlukan untuk oleh berbagai institusi, organisasi dan perusahaan untuk mempercepat memperoleh informasi dan untuk meningkatkan pelayanan. Kebutuhan akan basis data ini menjadi hal yang krusial bagi semua sektor dalam institusi, organisasi maupun perusahaan mulai dari pengguna tingkat rendah sampai tingkat manajerial membutuhkan basis data. Contohnya : kasir dengan cepat bisa menghitung harga belanjaan pelanggan yang menggunakan troli dengan menggunakan sistem Point of Sale (POS) bahkan langsung bisa menghitung jumlah diskon. Petugas perpustakaan bisa langsung memberikan informasi buku yang sedang dipinjam anggotanya dan bisa tahu kapan akan dikembalikan. Mahasiswa bisa tahu nilai mata kuliahnya dan jumlah SKS yang sudah diambil sampai saat ini. Manajer pemasaran bisa tahu produk yang paling laris dan produk yang tidak laku dijual. Semua aktivitas tersebut memanfaatkan basis data.

Kita bisa membayangkan seandainya semua kegiatan tersebut dilakukan secara manual hanya mengandalkan catatan-catatan di

buku atau di kertas, tentunya rawan terjadi kesalahan dan akibat dari kesalahan dalam setiap transaksi bisa menimbulkan kerugian. Alasan-alasan tersebut mengakibatkan mulai ditinggalkan cara-cara manual untuk memproses data dan beralih *key* basis data yang terkomputerisasi.

1.2. Gambaran Penerapan *Database*

Penerapan basis data saat ini sangat beragam untuk berbagai keperluan seperti kegiatan akademik, toko online, tiketing, perbankan, ensiklopedia dan lain sebagainya. Beberapa contoh penerapan *database* adalah sebagai berikut :

Akademik

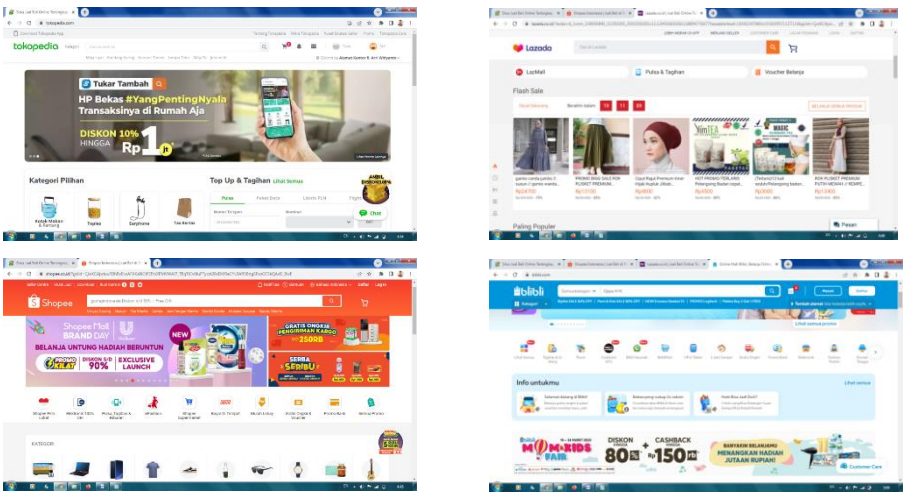
Secara umum *database* akademik digunakan untuk mengelola data-data akademik, diantaranya data mahasiswa, registrasi mahasiswa baru, registrasi mahasiswa lama, data dosen, data mata kuliah, data ruang kuliah dan penjadwalan. Pengelolaan *database* akademik ini menggunakan Sistem Informasi Akademik. Data mahasiswa yang disimpan dalam basis data ini berupa NPM, Nama Mahasiswa, Tempat dan Tanggal lahir, alamat, asal sekolah nama orang tua dan data lain yang yang relevan . Data dosen yang disimpan terdiri dari ID Dosen, Nama Dosen, Tempat Tanggal lahir, Alamat Dosen, pendidikan dan informasi lain yang relevan. Data mata kuliah terdiri Kode Mata Kuliah, Nama Mata Kuliah, Semester, SKS, Prasyarat dan deskripsi mata kuliah. Selain itu pada *database* akademik juga terdapat relasi antar data, diantaranya : Dosen mengajar mata kuliah, mahasiswa mengambil mata kuliah, Mata kuliah memiliki

jadwal, mata kuliah menggunakan ruang. Salah satu contoh sistem informasi akademik dapat dilihat gambar 1.

Gambar 1 Sistem Informasi akademik

Toko Online

Beberapa tahun terakhir, perkembangan penjualan secara online semakin meningkat. Hal ini menyebabkan jasa paket juga meningkat, tetapi membuat beberapa toko bahkan swalayan yang menutup gerai-gerainya. Beberapa platform toko online yang populer diantaranya Tokopedia, Bukalapak, Shopee, Lazada dapat dilihat pada gambar 2.



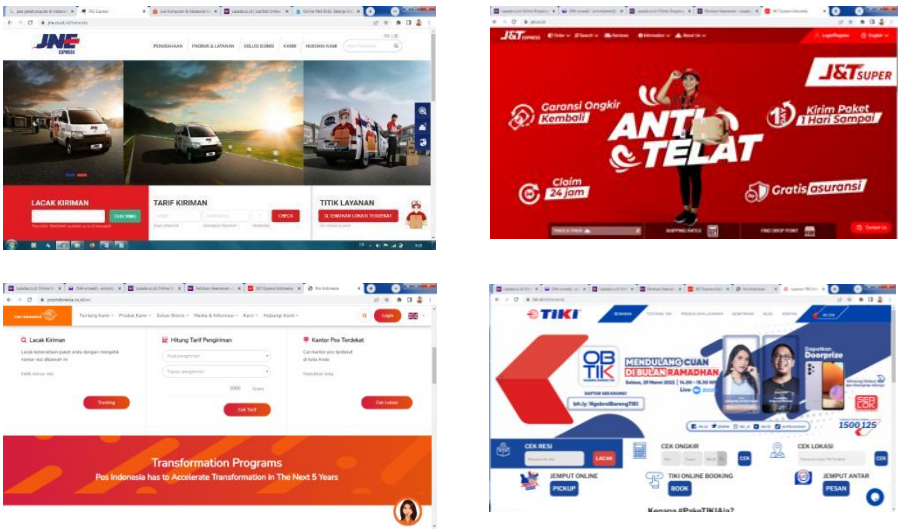
Gambar 2 Toko online di Indonesia

Secara umum toko online menyediakan katalog, pemesanan dan pelacakan kiriman. Pelanggan sebelum belanja harus melakukan registrasi terlebih dahulu. Belanja di toko online memiliki beberapa kemudahan, diantaranya bisa mencari barang dengan harga murah dibandingkan dengan toko lainnya dan pemrosesan transaksi sampai barang dikirim kerumah antara 3-5 hari untuk luar kota. *Database* toko online secara umum digunakan untuk menyimpan data-data yang berupa tabel-tabel :

1. Katalog yang terdiri dari nama barang, harga, diskon, warna, ukuran, kuantitas.
2. Pelanggan yang terdiri dari nomor telepon, nama lengkap, kata sandi, tanggal lahir, jenis kelamin, alamat, kota.
3. Keranjang belanja yang terdiri dari nama toko, nama barang, harga, jumlah, kode voucher.

Jasa Paket

Semakin maraknya toko online, jasa paket juga semakin berkembang dengan berbagai layanan seperti kiriman sehari dan pelacakan paket sehingga pengguna bisa mengetahui posisi paketnya. Selain itu, pengguna juga bisa memperkirakan biaya kirimannya. Beberapa jasa paket yang populer diantaranya adalah JNE, J&T Express, POS Indonesia, TIKI, SiCepat, Pahala Express, Indah Logistik terlihat pada gambar 3

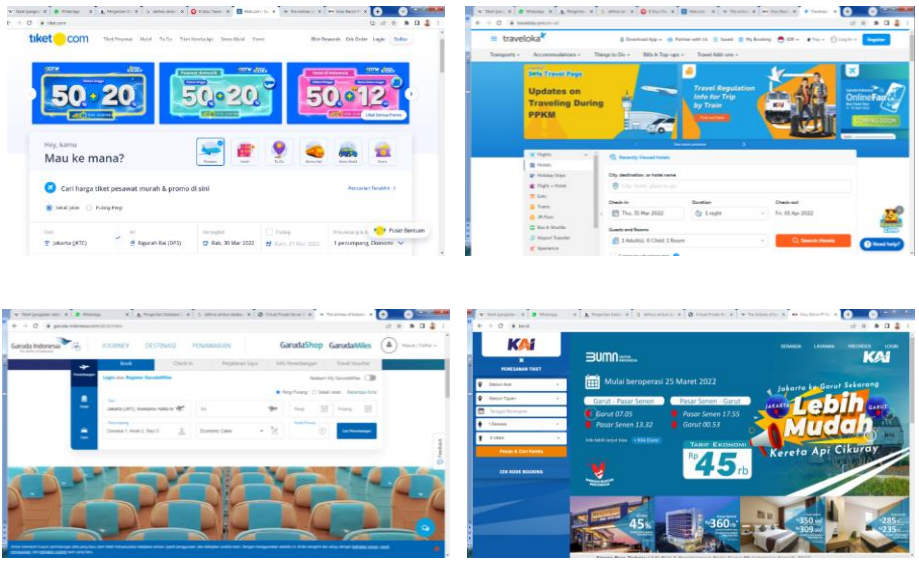


Gambar 3 Contoh Jasa Paket

Penggunaan *database* pada jasa paket ini untuk menyimpan data pelanggan yang terdiri dari ID Pelanggan, Nama Pelanggan, Alamat Pelanggan, No Telepon. Tarif pengiriman : Kode Kota, Kode Kelas, Kode Tarif, Kilo Pertama, Kilo *key* Dua, Rp. Kualifikasi paket : Kode Barang, Kode Kelas, Jenis Kiriman, Keterangan.

Ticketing

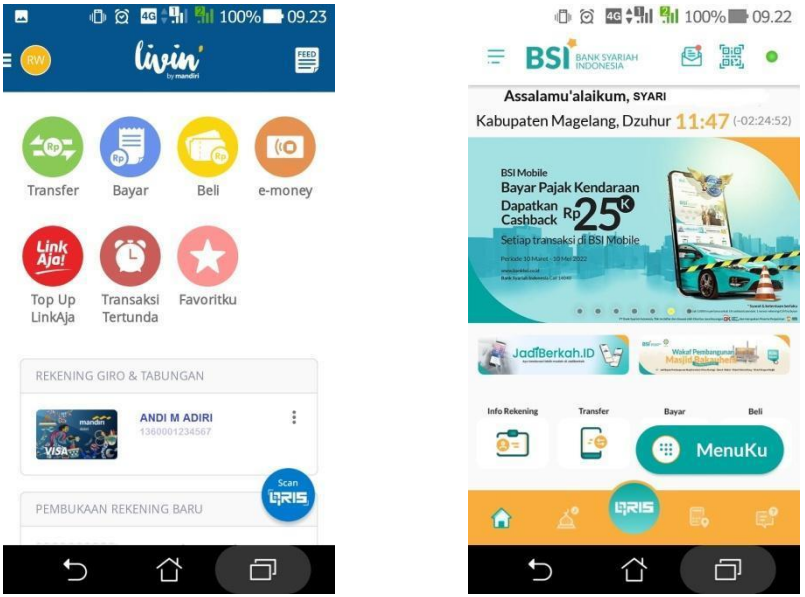
Pemesanan tiket, juga merupakan salah satu contoh penerapan *database*. Data-data yang disimpan dalam *database* tiket diantaranya Data Penumpang sperti NIK, Sapaan, Nama Depan dan Tengah, Nama Belakang, Tanggal Lahir, Kode Telepon Negara, No telepon, Nomor HP. Berikut gambar 4 yang menampilkan beberapa website pemesakan tiket.



Gambar 4 Contoh Pemesanan Tiket

Perbankan

Database di perbankan digunakan untuk mengelola data nasabah, data transaksi perbankan seperti tabungan, penarikan, peminjaman dan pembayaran angsuran. Sistem yang digunakan dari waktu *key* waktu selalu berubah menjadi semakin praktis. Nasabah bisa berinteraksi dengan *database* perbankan melalui internet banking, ATM dan M-Banking. Saat ini layanan internet banking dari beberapa bank sudah mulai tidak diaktifkan digantikan aplikasi M-Banking yang diinstal di handphone. Gambar 5 merupakan contoh mobile banking.



Gambar 5 Contoh Mobile banking

1.3. Pengenalan Basis Data

Data merupakan suatu fakta tentang benda, kejadian, aktivitas, dan transaksi, yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai. Data bisa berupa teks grafik, citra, suara, atau bahkan video. Data – data yang berdiri sendiri tidak akan memberikan sebuah makna sehingga perlu adanya pengelolaan data agar menjadi informasi.

Informasi adalah data yang sudah diolah sesuai kebutuhan sehingga memberikan informasi. Dari informasi tersebut maka akan mendapatkan *insight* yang akan dalam mendukung sebuah keputusan organisasi. Informasi yang telah bermakna apabila dikumpulkan akan membentuk suatu basis data. Basis data (*database*) merupakan

Kumpulan data yang merupakan kumpulan informasi mengenai fakta-fakta yang disimpan dalam komputer secara sistematik. Basis data akan menjadi kumpulan informasi terstruktur apabila dituangkan kedalam suatu sistem. Sehingga mampu mendukung pengelolaan serta kegiatan dalam organisasi. (Andaru, 2018)

Sistem yang menggunakan *database* dirancang untuk mengelola banyak informasi. Data-data ini perlu diolah melalui analisis tertentu sehingga berguna dalam pengambilan keputusan. Basis data sangat erat kaitannya dengan kehidupan sehari-hari, yaitu data perusahaan, data bank, universitas, dan lain-lain.

Penggunaan *database* disekitar kita

Basis data juga dikenal sebagai *database.* Kehidupan modern tidak bisa terlepas dari adanya peran *database.* Hampir semua aspek kehidupan sudah mulai menerapkan *database* seperti bisnis, rumah sakit, pendidikan, kantor pemerintah dan lain - lain. Penggunaan *database* bertujuan untuk mempercepat dalam mendapatkan informasi dan untuk meningkatkan pelayanan. Dengan penerapan *database*, maka proses yang semula manual mulai migrasi *key* sistem dengan menggunakan *database*

Jenis *Database*

Database adalah kumpulan dari data yang saling berelasi atau terhubung. *Database* memiliki bermacam-macam tipe yaitu *tradisional database, multimedia database* dan *GIS database. Tradisional database* merupakan sebuah data yang terdiri atas teks dan angka data gambar.

Sedangkan *multimedia database* merupakan kumpulan data berupa video dan suara. Terakhir *GIS database* adalah kumpulan data berupa data lokasi dan geografis.

Manfaat *Database*

Penggunaan *database* memiliki manfaat kepada keberlangsungan sebuah sistem aplikasi. Sistem *database* memberikan sebuah kemudahan diantara lain kecepatan dan kemudahan *(speed)* efisiensi ruang penyimpanan (*space),* keakuratan *(accuracy),* ketersediaan *(availability),* kelengkapan *(completeness),* keamanan *(security),* data dapat dipakai secara bersama *(shareability).*

Kecepatan dan Kemudahan *(Speed)*. Dalam pengolahan data sistem *database* mempermudah serta melakukan pemroses data secara cepat. Kecepatan tersebut juga berdasarkan suatu jenis *database* yang digunakan. Sehingga perlu adanya pemahaman terhadap data agar dalam pembuatan *database*nya diperlakukan secara tepat.

Efisiensi Ruang Penyimpanan (*Space).* Penggunaan *database* yang menghubungkan beberapa data dan membentuk suatu relasi akan mengurangi data yang redundan atau data yang diulang.

Keakuratan *(Accuracy). Database* akan lebih terjaga keaturatannya karena memperlakukan data sesuai aturan dan ketentuan tertentu, keunikan data, tipe data dan domain data.

Ketersediaan *(Availability).* Data yang ada pada *database* dapat dipisahkan berdasarkan penggunaannya. Data yang sudah tidak

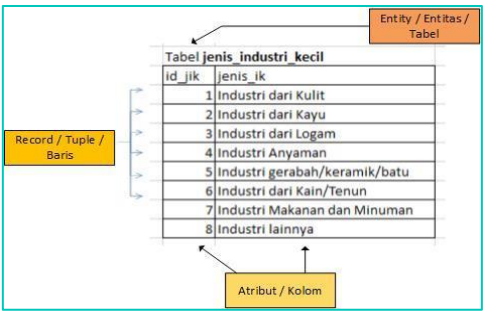
digunakan maka dapat dikelompokan serta dipindahkan atau dihapuskan untuk menghemat ruang penyimpanan. Ditambah teknologi yang mendukung maka data data tersebut akan menjadi lebih mudah diakses dari mana saja.

Kelengkapan *(Completeness).* Seiring dengan berjalannya waktu maka kebutuhan user akan bertambah. Dalam pengembangannya pasti akan terjadi perubahan-perubahan yang mengharuskan adapnya pemetaan baru. *Database* mampu memetakan data sehingga dalam pengolahannya menjadi lebih mudah untuk dibaca serta dipahami.

Keamanan *(Security). Database* yang sudah menjadi sistem informasi akan dilengkapi dengan sebuah enskripsi-enskripsi data yang akan meningkatkan keamanan serta meminimalisir kebocoran data kepada orang luar yang tidak memiliki hak akses didalamnya.

Data dapat dipakai secara bersama *(Shareability).* Data akan disimpan dan saling terhubung satu sama lain melalui sebuah jaringan baik antar individu atau antar kelompok. Sehingga memudahkan dalam pemakaiannya yang tidak perlu menyalurkan antara satu persatu *key* antar bagian.

Istilah dalam Relational *Database*



Gambar 6. Contoh penggunaan record, entitas, atribut.

Database terdapat beberapa istilah – istilah yang perlu dipahami meliputi, entitas, record dan atribut seperti pada gambar 6. Berikut penjelasan mengenai istilah – istilah tersebut:

1. **Entity / entitas / Tabel**

   Entitas merupakan segala sesuatu yang kita simpan sebagai informasi. Entitas mewakili objek yang ada di dunia nyata (fakta). Simbol entitas dapat dilihat pada gambar 7.

Gambar 7 Simbol Entitas

2. **Record / tuple / baris**

   Kumpulan elemen yang sejenis, saling berkaitan menginformasikan tentang entity secara lengkap. Satu record mewakili satu data atau informasi tentang seseorang misalnya, nomor karyawan, nama karyawan, alamat kota, tanggal masuk.

3. **Atribut / Kolom**

Setiap entitas memiliki atribut yang mampu deskripsi dari entitas. Sebuah atribut dapat diturunkan dari atribut lainnya disebut dengan Composite Attribute. Simbol atribut dapat dilihat pada gambar 8

Gambar 8 Simbol Atribut

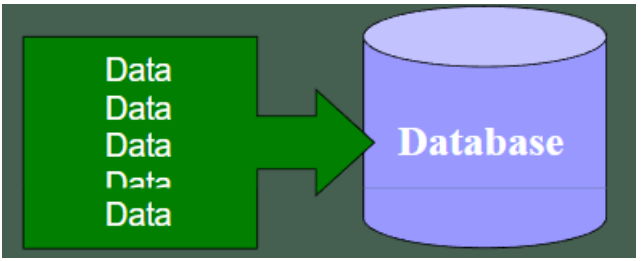
1.4. Konsep Basis Data dan Manajemen Informasi

Mengenal Manajemen Informasi

Seiring berkembangnya revolusi industri 4.0 dengan berbasis Teknologi. Teknologi yang digunakan tergantung pada kemampuan suatu sistem untuk mengelola data dan informasi. Data adalah kunci dalam sebuah Sistem Informasi. Data yang berupa informasi menjadi akan aset dalam perusahaan yang berharga. Seseorang yang mampu memiliki serta mengelolah informasi akan menjadi tokoh yang penting dalam perusahaan.

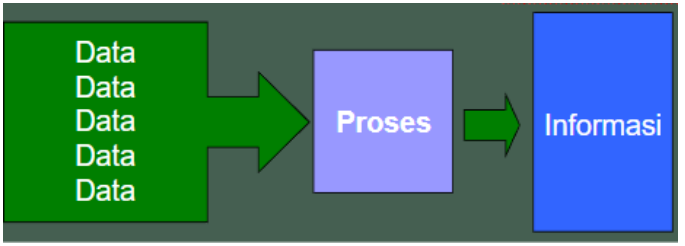
Manajemen Data

Data sebagai sumber informasi perlu dikelola dengan baik. Beberapa data yang dikumpulkan menjadi satu akan menjadi sebuah *database*s. Sebagai visualisasinya dapat dilihat pada gambar 9



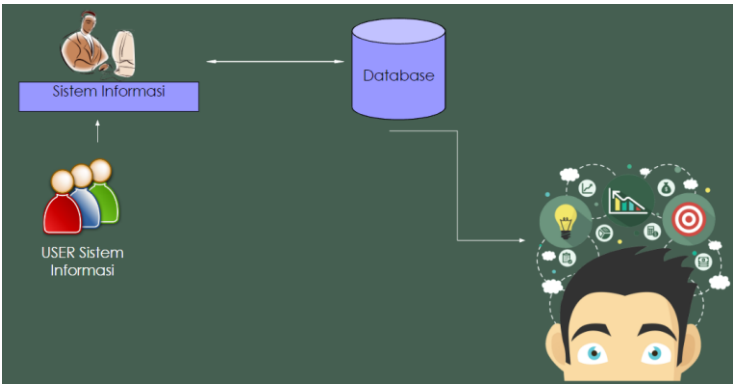
Gambar 9 Visualisasi Data pada *Database*

Pengelolaan *database* berawal dari input dengan memasukan berbagai data-data. Lalu diprosess dengan sebuah algoritma / ilmu statistik / AI / Data Science sehingga menghasilkan sebuah output berupa informasi. Sebagai visualisasinya dapat dilihat pada gambar 10. Pengelolaan data dapat diterapkan dengan menggunakan ***database***



Gambar 10 Visualisasi proses pengelolaan data menjadi *database*

Secara penerapannya, data yang semula berbentuk abstrak pada pikiran manusia terlihat sangat rumit sehingga membutuhkan penyederhaan dengan melakukan pengelolaan menjadi sebuah *database*. Sebuah *database* merupakan salah satu hal utama dalam pembuatan sistem informasi yang nantinya dapat diterima oleh berbagai pengguna sistem tersebut. Sebagai visualisasinya dapat dilihat pada gambar 11.



Gambar 11 visualisasi data abstrak menjadi data informasi

Berdasarkan hal tersebut maka perlu adanya perubahan dari pengolahan data secara tradisional atau manual *key* pengolahan data menggunakan *database*. Pengolahan data secara tradisional atau manual memiliki kelemahan yang mampu teratasi dengan menggunakan *database* seperti, dalam pencatatan data secara manual memiliki resiko duplikasi data yang tinggi, data menjadi sulit diakses karena data hanya akan diolah oleh satu orang tertentu (*single user)*, data memiliki tingkat keamanan yang rendah karena tidak dapat terenskripsi, data tidak terintegrasi satu sama lain dan akan terisolasi pada bidang bidang tertentu, data akan dikelola pada setiap bidang dan memiliki kemungkinginan bahwa data akan berdiri sendiri atau tidak terikat satu sama lain.

Database mampu mengatasi redundansi atau ketidakkonsistenan suatu data pada bidang tertentu dengan data pada bidang lain. Dapat diartikan bahwa data nantinya akan saling terhubung dan terintegrasi kedalam sebuah sistem informasi. Apabila data sudah terealisasikan menjadi sebuah sistem informasi maka data akan semakin mudah

untuk diakses oleh banyak pengguna di berbagai bidang, keamana data yang lebih ketat dikarenakan adanya enskripsi dari mesin atau sistem sehingga pengelolaannya data terisolasi sebagai wujud sebuah standarisasi.

Aplikasi *Database*

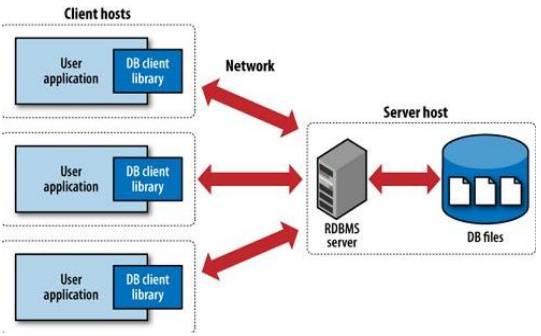
Aplikasi *database* adalah program aplikasi yang digunakan untuk melaksanakan sederet kegiatan yang ditentukan oleh pemakai. Sebuah aplikasi *database* mampu melakukan operasi dasar. Beberapa operasi dasar yang dilakukan oleh aplikasi *database* yaitu menambah data, membaca data, mengubah data dan menghapus data.

Ranah Aplikasi *Database*

Aplikasi *database* memiliki sudut pandang yang berbeda berdasarkan lingkungan pengguna dan pengaksesan sistem. Berdasarkan lingkungannya maka aplikasi *database* terdapat 4 kategori didalamnya. Arah aplikasi *database* berdasarkan lingkungannya selaras dengan topologi suatu jaringan. *Database* yang hanya digunakan pada satu orang atau dengan jangkauan kecil dan penggunaan pribadi disebut *Personal Computer databases.* Kumpulan dari *database*s perorangan yang di jadikan satu pada ruangan tertentu dan saling terhubung satu sama lain dapat disebut dengan *Workgroup databases*.

Selajutnya, merupakan *Department databases* yang berarti kumpulan dari workgroup *database*. Biasanya digunakan antar ruangan dan grub lain. Pada gambar 12, kumpulan data yang

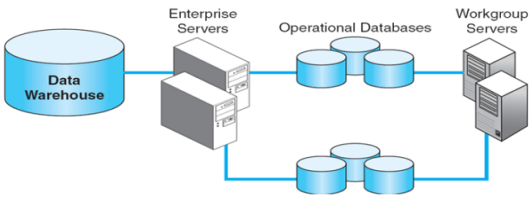
dijadikan satu kedalam *database* diletakan *key* dalam sebuah server host. Untuk mengakses *database* tersebut diperlukan sebuah RDBMS server yang bertujuan agar data dapat diakses oleh *client host* melalui jaringan atau *network.* Pengaksesan *database* tidak hanya dipatasi pada user tertentu, bahkan dapat digunakan oleh berbagai User aplikasi serta mampu singkronisasi *database* dari masing grub kepada *database* yang ada diservernya.



Gambar 12 Visualisasi *Department databases*

Terakhir adalah *database* yang digunakan dalam skala yang besar atau pada perusahaan yaitu *enterprise databases. Database* yang digunakan pada skala perusahaan biasanya menggunakan suatu data *warehouse* atau gudang data yang terdiri atas kumpulan serta berbagai data dari seluruh bidang, baik data yang digunakan atau data yang tidak digunakan. Keunggulannya adalah data yang tidak digunakan pada *workgroup server* akan disimpan pada data *warehouse* dan diakses melalui *enterprice servers*. Sisi lain maka ruang penyimpasnan yang seharusnya untuk data tidak terpakai dapat digunakan untuk data baru yang digunakan. Namun, ketika perusahaan membutuhkan data lama misal 30 tahun terakhir maka dapat melihat rekap data yang

disimpan pada data *warehouse.* Sebagai visualisasinya dapat dilihat pada gambar 13.

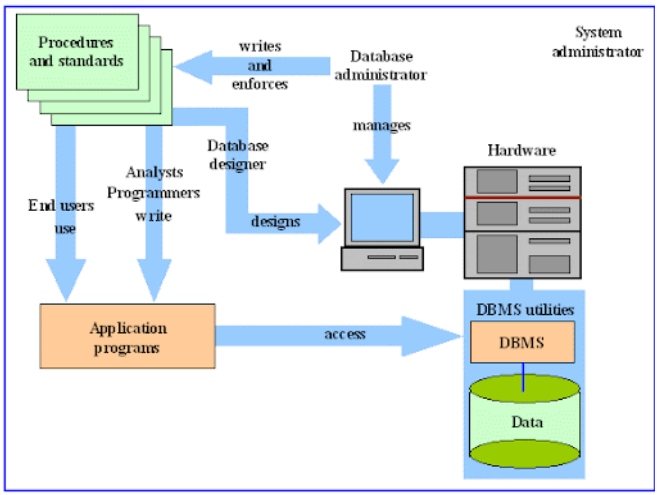


Gambar 13. Visualisasi *workgroup server*

Seperti yang kita tahu bahwa aplikasi tidak hanya berbasis kepada web saja. Terdapat beberapa macam aplikasi yang perlu penyesuaian penggunaan *database*-nya agar lebih maksimal serta tidak membuang resourch yang ada. Ranah aplikasi berdasarkan akses aplikasinya seperti *Web based database application* lebih cenderung menggunakan pemrograman yang bersifat *server-side* seperti Microsoft SQL Server. Selanjutnya adalah *Desktop based database application* atau aplikasi berbasis desktop yang biasanya menggunakan Oracle, MariaDB. Terakhir adalah *Mobile based database application* atau aplikasi dengan berbasis mobile yang lebih baik menggunakan SQLlite dan PostgreSQL.

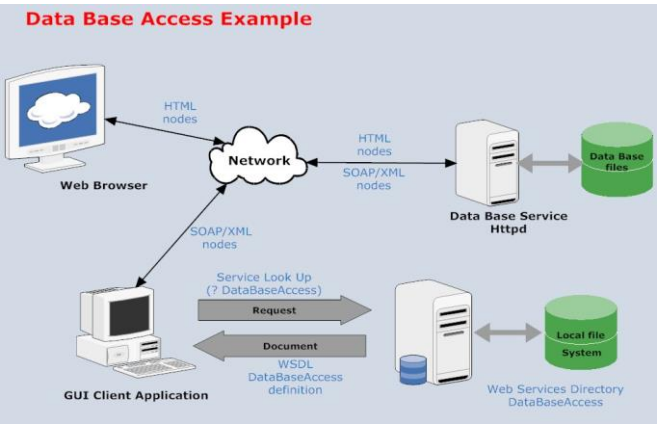
Dalam sebuah sistem pasti memiliki protocol dan standar yang perlu diperhatikan. Tidak semua user bersinggungan dengan *database*. Seorang *end user* dan dan *analysts* program write membuat aplikasi programnya yang membutuhkan sebuah informasi. Informasi yang dibutuhkan terkandung dalam sebuah *database* yang dapat diakses melalui DBMS. Sedangkan *database* designer melakukan pemetaan terhadap data yang ada menjadi sebuah *database* melalui sebuah

computer. *Database* akan disimpan pada server dan berkaitan dengan data satu dan lainnya. Selanjutnya dalam pengelolaan *database* yang telah dipetakan adalah peran dari *database administrator.* Seorang *database administrator* selain mampu mengelola *database* juga dapat melakukan perubahan pada prosedur dan standar sebuah sistem untuk menyesuaikan kebutuhan dari client atau pengguna. Sehingga dalam pembuatan suatu system, *database* merupakan sebagian kecil yang penting untuk dipelajari sebagai pondasi sistem. Sebagai visualisasinya dapat dilihat pada gambar 14.



Gambar 14 Visualisasi *system administrator*

Pada gambar 15, merupakan penggambaran sistem *database* dengan berbasis cloud. *Database* files dapat diakses dengan melalui *database* service Httpd atau html nodes. Selain itu sistem ini berbasis *Web based database application* sehingga membutuhkan *Graphic User Interface* (*KEY*) Ketika melakukan permintaan atau mengirimkan file dari lokal *database*.



Gambar 15 Alur kerja sistem *database* berbasis cloud

BAB 2

PEMBUATAN *DATABASE* BERBASIS DATA

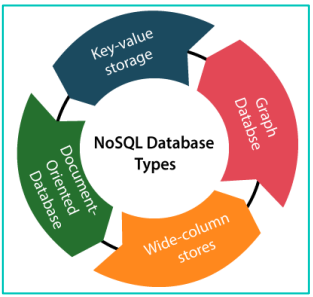
2.1. Tujuan Pembelajaran

Mahasiswa mampu membuat desain *database* berupa *Enhanced Entity Relationship* sesuai RDBMS dalam pengembangan Sistem Informasi sesuai studi kasus.

2.2. Tipe-Tipe Basis Data Relasional

Database dalam sebuah pengembangan sistem memiliki beberapa tipe tertentu berdasarkan kebutuhan pengguna baik dari penyimpanannya, pengembangannya serta penggunanya. Tipe – tipe *database* yang ada seperti *Operational Database, Analytical Database, Data Warehouse, Distributed Database, End-User Database, Real-Time Database, Navigational Database, Hypermedia Database, In Memory Database* dan *Document Oriented Database.*

Operational Database merupakan sistem manajemen basis data operasional, biasa digunakan untuk memperbarui data secara real-time, mendukung pertumbuhan penggunaan data tidak terstruktur (NoSQL) dan mesin DBMS NoSQL. Untuk noSQL *database* memiliki 4 tipe didalamnya yakni Graph *database*, *Key-Value storage, Document-Oriented database* dan *wide-column stores.* Dapat dilihat pada gambar 16.



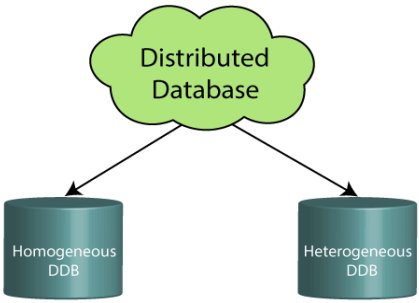
Gambar 16 Rangkaian tipe untuk NoSQL

Analytical Database merupakan *database* yang dapat menyimpan dan mengelola *big data*, termasuk bisnis, pasar, dan data pelanggan untuk analisis *business intelligence* (BI).

Data Warehouse adalah sistem yang menyimpan data dari waktu *key* waktu (data historis), menyimpan dari *database* operasional, menjadi sumber utama dalam mencari informasi yang telah diperiksa, diubah, dan terintegrasi.

Centralized Database adalah penyimpanan data di sistem *database* terpusat, tersimpan data dari lokasi yang berbeda melalui beberapa aplikasi, membiarkan pengguna mengakses data secara aman, pusat *database* dapat menjadi pusat data di perguruan tinggi / universitas / perusahaan .

Distributed Database sebagai *database* yang berfungsi untuk mendistribusikan *database* melalui *workgroup locale* di kantor regional, kantor cabang, serta lokasi kerja lainnya yang berkaitan. Contoh Apache, Cassandra, HBase, Ignite dan lain lain. Terdapat 2 cabang pada *database* terdistribusi dapat dilihat pada gambar 17.



Gambar 17. Visualisasi kedua cabang *distributed database*

Homogeneous DDB merupakan Sistem *database* yang menjalankan sistem operasi yang sama dan menggunakan proses aplikasi yang sama dan membawa perangkat keras yang sama. *Heterogeneous DDB* merupakan sistem *database* yang menjalankan pada berbeda sistem operasi di bawah berbeda aplikasi prosedur, dan membawa perangkat keras yang berbeda.

Selanjutnya, *End-User Database* merupakan basis data yang dikembangkan oleh *end-user* itu sendiri melalui *workstation* mereka seperti word *processing*, *spreadsheet* hingga download file.

Real-Time Database sebuah basis data seperti harga mata uang dollar yang mampu berubah setiap menit. Di balik itu semua, terdapat basis data *real-time* yang bekerja dengan cepat, biasanya digunakan oleh lembaga akuntansi, hukum, perbankan, multimedia, analisis data ilmiah, serta catatan medis.

Navigational Database sebagai *database* dengan pengguna lainnya bisa menemukan informasi yang dituju dengan memberikan sebuah kata kunci yang sesuai.

Sedangkan *Hypermedia Database* merupakan *database* yang memungkinkan berbagai laman multimedia yang saling berhubungan dan bisa diakses dengan fitur hyperlink.

In Memory Database basis data memori yang terdapat pada perangkat keras (microSD, harddisk, flashdisk) basis data ini digunakan untuk jaringan telekomunikasi yang memerlukan pengoperasian cepat secara darurat.

Document Oriented Database menjadi *database* yang menyimpan setiap catatan sebagai dokumen yang memiliki keunikan khusus. Sejumlah *field* apapun dapat ditambahkan melalui dokumen. *Field* tersebut juga dapat diisi dengan beberapa bagian data.

Relational Database merupakan *database* yang menyimpan data dalam bentuk baris (tuple) dan kolom (atribut), dan bersama-sama membentuk tabel (relasi), *database* relonal menggunakan SQL untuk menyimpan, memanipulasi, serta mempertahankan data.

Cloud Database adalah *database* yang disimpan dalam lingkungan virtual dan pelaksanaan atas platform komputasi awan, itu menyediakan pengguna berbagai komputasi awan (SaaS, PaaS, IaaS, DLL.), contoh: layanan Web Amazon (AWS, Microsoft Azure, Kamatera, PhonixNAP, ScienceSoft, Google cloud SQL, DLL).

Object-oriented Databases merupakan Basis data yang berorientasi pada objek Yang menggunakan pendekatan model data berbasis objek untuk menyimpan data dalam sistem *database*, data tersebut diwakili dan disimpan sebagai objek yang mirip dengan objek

yang digunakan dalam bahasa pemrograman yang berorientasi pada objek.

Hierarchical Databases merupakan *Database* yang menyimpan data dalam bentuk hubungan orang tua dan anak.

Network Databases adalah *database* yang biasanya mengikuti model data jaringan, memungkinkan setiap catatan memiliki beberapa node anak dan orangtua untuk membentuk struktur grafik generalisasi.

2.3. Model Basis Data Relasional

Relasi menghubungkan antar tabel, relasi memberikan dampak adanya kunci tamu pada salah satu tabel dalam relasi. Basis data relasional memiliki setidaknya 1 tabel dalam *database*nya. Setiap tabel memiliki lajur vertikal yang biasa disebut dengan kolom atribut (column/*field*). Setiap tabel memiliki lajur horizontal yang biasa disebut dengan baris data (row/record).

Pada setiap pertemuan kolom atribut dan baris data ditempatkan item-item data (satuan data terkecil). Memilah data *key* dalam berbagai tabel 2 dimensi. Dalam setiap entitas, dipilih setidaknya 1 atribut untuk dijadikan kunci utama(*primary key*). Basis data yang memperhatikan aturan relasi atau hubungan sehingga setiap tabel yang terhubung, atribut kunci pada 1 tabel menjadi kunci tamu (*foreign key*) pada tabel lainnya.(Julaeha et al., 2020)

Untuk menerapkan basis data relasional kita tetap menggunakan perangkat lunak sistem pengelola basis data (DBMS).

2.4. Membangun Basis Data Relasional

Dalam pembuatan *database* memiliki 2 hal dasar yang perlu diketahui. Pembuatan *database* dengan berbasis data dan berbasis objek yang dapat dilihat perbedaannya pada tabel 1.

Tabel 1. Perbedaan *Database* berbasis data dan objek

Berbasis Data	Berbasis Objek
o Melihat data yang dibutuhkan sistem o Teknik Normalisasi o Minimal 3NF o Membangun sistem dengan data yang SUDAH didefinisikan kebutuhannya o Memiliki contoh LAPORAN	o Melihat objek sistem yang dibangun o Menggunakan Entity Relationship Diagram o Data dan Laporan/output dari sistem belum ada/didefinisikan

Pembuatan *database* dengan berbasis data perlu adanya normalisasi. Secara singkat Teknik normalisasi adalah teknik membuat *relational database* yang tidak berkaitan dengan objek namun dengan data yang menerapkan sejumlah aturan standar sehingga menghasilkan struktur tabel yang normal. Pada pengolahannya, struktur tabel yang memiliki ketidak-konsistenan data akan berdampak pada data lainnya atau disebut anomali. Dalam membangun sistem yang berstandar tinggi maka kita juga perlu

memahami anomali – anomali apa saja yang perlu dihindari. Anomali terdapat 3 jenis yang berbeda, yaitu sebagai berikut:

1. Anomali insertion (penyisipan)

   Anomal insertion merupakan masalah yang timbul saat melakukan penambahan data. Contoh pada tabel 2, apabila pengguna ingin menambahkan sebuah value pada id_pemasok saja, maka value data barang akan menjadi kosong. Sehingga penambahan data pada pemasok **tidak dapat dilakukan** karena kunci utamanya ada pada id_brg. Apabila tetap ditambahkan ada menimbulkan sebuah anomali.

Tabel 2. Contoh Anomali Insertion

ID_BRG	NAMA_BRG	ID_PEMASOK	NAMA_PEMASOK
B001	Buku	P001	ATJ
B002	Pensil	P001	ATJ
B003	Printer	P002	EK
B004	Sepatu	P003	SAD
B005	Sandal	P003	SAD
		P004	PPN

2. Anomali deletion (penghapusan)

   Anomal deletion merupakan masalah yang timbul saat melakukan penghapusan data. Contoh pada tabel 3, apabila pengguna ingin menghapus id_brg dengan value B003 maka seluruh data yang berkaitan dengan id_brg tersebut akan ikut terhapus.

Tabel 3. Contoh Anomali Deletion

ID_BRG	NAMA_BRG	ID_PEMASOK	NAMA_PEMASOK
B001	Buku	P001	ATJ
B002	Pensil	P001	ATJ

B003	Printer	P002	EK
B004	Sepatu	P003	SAD
B005	Sandal	P003	SAD

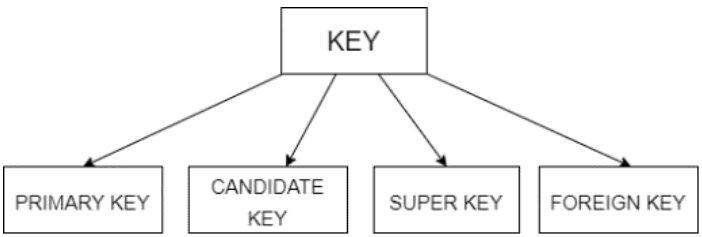
3. Anomali update (peremajaan)

Anomal update merupakan masalah yang timbul saat melakukan perubahan data. Contoh pada tabel 4, apabila nama_brg pensil memiliki pemasok baru dan berubah menjadi VOD. Maka akan ada sebuah anomaly bahwa dengan id_pemasok P001 memiliki 2 nama pemasuk baru dan lama.

Tabel 4. Contoh Anomali Update

ID_BRG	NAMA_BRG	ID_PEMASOK	NAMA_PEMASOK
B001	Buku	P001	ATJ
B002	Pensil	P001	VOD
B003	Printer	P002	EK
B004	Sepatu	P003	SAD
B005	Sandal	P003	SAD

2.5.Kunci Basis Data Relasional

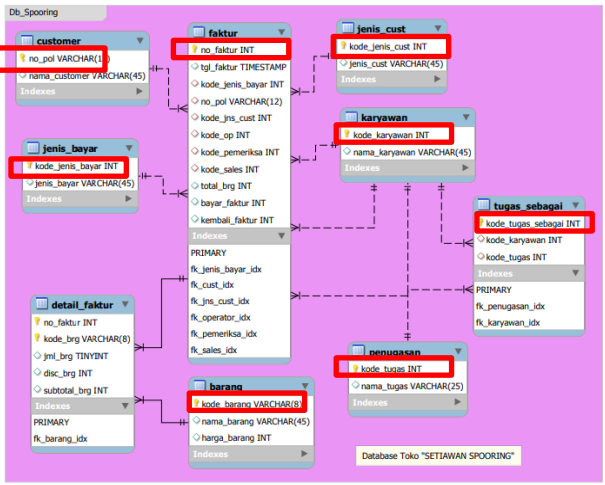


Gambar 18. Macam macam *key* relasional

Keys atau kunci pada *database* memberikan peran penting pada *database* dan bersifat unik. Bersifat unik bermakna tidak boleh ada data yang sama antar data yang lain. Sebuah atribut yang unik dapat dijadikan sebuah kunci pada entitas tersebut. Terdapat beberapa macam kunci yang diterapkan pada suatu tabel seperti gambar 18.

Primary Key

Disebut juga kunci utama, umumnya terdiri dari 1 atribut, namun bisa lebih dari 1 atribut. Terlihat pada gambar 19 bahwa contoh *primary key* adalah kode_barang ditandai dengan kotak merah seperti, No_pol menjadi PK dari tabel customer, kode_jenis_bayar menjadi PK dari tabel jenis bayar dan seterusnya.



Gambar 19 *Primary Key* pada Relasi ERD Spooring

Candidate *Key*

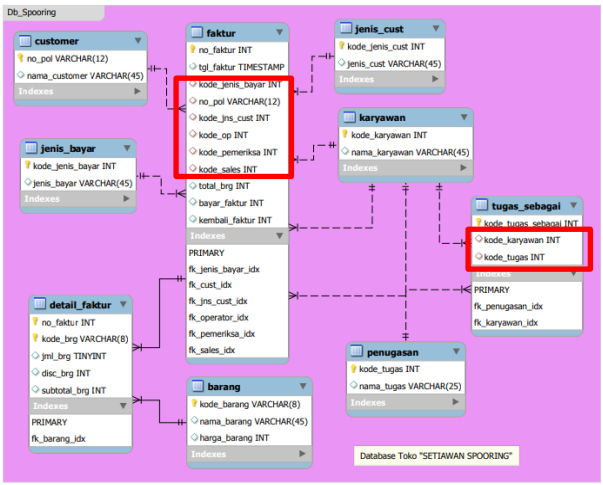
Disebut juga kunci kandidat. Umumnya terdiri dari 1 atribut, namun bisa lebih dari 1 atribut. memiliki nilai unik (sama seperti PK). Atribut selain PK dapat menjadi candidate *key*.

Super *Key*

Kumpulan atribut untuk memastikan baris itu unik. Super *key* adalah superset dari candidate *key*. Bisa diartikan bahwa super *key* kumpulan dari beberapa kunci yang memiliki baris yang unik.

Foreign Key

Foreign key (FK) bisa disebut kunci tamu adalah sebuah *primary key* yang diletakan pada tabel lain sehingga membentuk suatu relasi. Dapat dilihat pada gambar 20 bahwa no_pol, kode_jns_bayar yang semula menjadi PK pada masing-masing tabel apabila dimasukan pada tabel faktur menjadi sebuah FK dan membentuk relasi yang ditandai dengan garis putus-putus.



Gambar 20 Forgen *Key* pada Relasi ERD Spooring

2.6.Syarat – syarat Basis Data Relasional

Sehingga untuk menjadi sebuah struktur tabel yang normal perlu memeruhi syarat-syarat tertentu. Struktur tabel yang normal perlu adanya sebuah penguraian / pemecahan / dekomposisi tabel dari data yang dikelola terjamin aman. Selain itu struktur tabel perlu memenuhi *functional dependency* yang berlaku serta minimal 3NF.

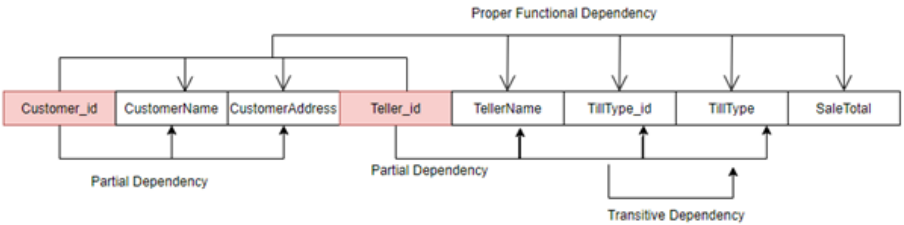
Functional dependency merupakan sebuah gambaran dari suatu relasi serta memberikan batasan hubungan antara suatu tabel dengan

tabel lainnya. Tujuannya agar semua hubungan sesuai dengan fungsi yang dibutuhkan, dapat menentukan kunci relasi serta mempermudah dalam melakukan normalisasi suatu tabel. Terdapat 3 macam f*unctional dependency* yaitu, *Full Functional Dependency*, *Partial Functional Dependency* dan *Transitive Functional Dependency.* Secara lebih jelas bisa dilihat pada gambar 21.

Full Functional Dependency, apabila suatu atribut memiliki ketergantungan secara fungsional terhadap atribut lain dan tidak kepada sebagian atau turunan dari atribut tersebut. Pada gambar 21, full dependency antara teller_id dan customer_id apabila customeradders dihapuskan karena teller_id tidak bergantung pada subset customer_id.

Partial Functional Dependency, keadaan dimana dalam sebuah tabel terdapat atribut atau sebagian atribut yang tidak bergantung sepenuhnya kepada *primary key* di tabel tersebut. Pada gambar 21, customeraddres tidak hanya begantung pada customers_id tapi juga bergantung pada custommername.

Transitive Functional Dependency, apabila pada sebuah tabel terdapat atribut yang tidak hanya tergantung kepada *primary* keynya, tetapi kepada atribut lain yang bukan kunci. Pada gambar 21, atribut tilltype tidak hanya bergantung pada teller_id namun juga pada tiitype_id.



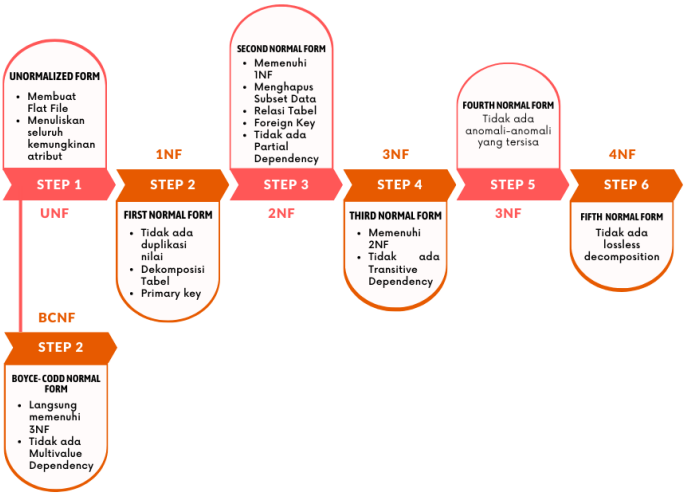
Gambar 21. *Functional Dependency*

2.7. Teknik Normalisasi

Teknik normalisasi adalah suatu teknik dengan pendekatan *button-up* yang digunakan untuk membantu mengidentifikasikan hubungan. Dimulai dari menguji hubungan, yaitu *functional dependencies* antara atribut. Tujuan utama normalisasi adalah mengidentifikasikan kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan, adapun karakteristik hubungan tersebut mencakup (Suryadi, 2019) :

1. Minimal jumlah atribut yang diperlukan untuk mendukung kebutuhan perusahaan.
2. Atribut dengan hubungan logika yang menjelaskan mengenai functional dependencies.
3. Minimal duplikasi untuk tiap atribut.

2.8. Tahapan Normalisasi



Gambar 22 Alur Normalisasi

Peranan normalisasi dalam pengembangan sistem adalah dalam penggunaan pendekatan *button-up* dan teknik validasi. Teknik validasi digunakan untuk memeriksa apakah struktur relasi yang dihasilkan oleh ER modeling itu baik atau tidak. Terdapat enam bentuk normal yang biasa digunakan dimulai dari tahap ringan (1NF) hingga paling ketat (5NF) dapat dilihat pada gambar 22. Namun pada studi kasus di perusahaan, Normalisasi cukup sampai pada tingkat 3NF atau BCNF karena sudah menghasilkan tabel-tabel yang berkualitas baik / *full dependency*. Berikut ini beberapa poin penting terkait normalisasi :

1. Bentuk Tidak Normal / *Unormalized From (*UNF)

   Membuat *Field* File dari contoh data serta menuliskan seluruh kemungkinan atribut dari data.

2. Bentuk Normal Pertama / *First Normal Form* (1NF)

Menghilangkan duplikasi dari data yang sudah ditulis sebelumnya. Memecahkan data menjadi beberapa tabel yang atributnya saling bergantung sehingga memunculkan *primary key* pada tabel tersebut.

3. Bentuk Normal Kedua / *Second Normal Form (2NF)*

   Setelah memenuhi dari 1NF maka langkah selanjutnya adalah menghilangkan Ketergantungan Parsial atau *partial dependency.* Menghapus subset data serta membuat suatu relasi tabel. Relasi tabel akan ditandai dengan munculnya sebuah *key* baru yaitu *Forgein key.* Mulai dari Tahap ini terdapat 2 jenis tabel yaitu tabel Mater dan Tabel Implementasi Relasi. Tabel master adalah tabel yang berdiri sendiri dan tidak memiliki *foreign key*. Sedangkan tabel implementasi relasi merupakan tabel yang terdapat *foreign key* sehingga dapat membentuk suatu relasi dari tabel master

4. Bentuk Normal Ketiga / *Third Normal Form (3NF)*

   Pada bentuk ini, data harus telah memenuhi 2NF serta Menghilangkan Ketergantungan Transitif atau *transitive dependency.*

5. Bentuk Normal *Boyce-Code Form (BCNF)*

   Menghilangkan anomali-anomali hasil dari ketergantungan fungsional Bentuk BCNF terpenuhi dalam sebuah tabel, jika untuk setiap *Functional Dependency* terhadap setiap atribut atau gabungan atribut dalam bentuk : X --> Y maka X adalah Super *Key*. Tabel tersebut harus di dekomposisi berdasarkan *Functional Dependency* yang ada, sehingga X menjadi super *key* dari tabel-tabel hasil

dekomposisi. Setiap tabel dalam BCNF merupakan 3NF. Akan tetapi setiap 3NF belum tentu termasuk BCNF. Perbedaannya, untuk *Functional Dependency* X--> A, BCNF tidak membolehkan A sebagai bagian dari *primary key*.

6. Bentuk Normal Keempat / *Fourth Normal Form (4NF)* :

   Menghilangkan ketergantungan multivalue Bentuk normal 4NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk BCNF, dan tabel tersebut tidak boleh memiliki lebih dari sebuah *multivalued attribute.* Untuk setiap multivalued attribute (MVD) juga harus merupakan *Functional Dependency.*

7. Bentuk Normal Kelima / *Fifth Normal Form (5NF)* :

   Menghilangkan anomali-anomali yang tersisa. Bentuk normal 5NF terpenuhi jika memiliki sebuah *loseloss decomposition* menjadi tabel-tabel yang lebih kecil. Jika 4 bentuk normal sebelumnya dibentuk berdasarkan *Functional Dependency,* 5NF dibentuk berdasarkan konsep *Join Dependence.* Yakni apabila sebuah tabel telah didekomposisi menjadi tabel-tabel lebih kecil, harus bisa digabungkan lagi untuk membentuk tabel semula.

8. Overnormalisasi

   Analisa Overnormalisasi diperlukan jika *database* ini digunakan untuk sistem multi user. Tabel-tabel yang sudah normal ini digabungkan dengan fungsi lain yang ada di lapangan, misalnya; untuk fungsi retur, untuk fungsi inventori, untuk fungsi sales order maupun order pembelian, untuk fungsi keamanan *database*, dan lain-lain.

2.9.Relasi

Relasi adalah hubungan antar tabel. Setiap relasi yang menghubungkan 2 tabel memiliki derajat yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain, hal ini disebut sebagai **Kardinalitas**. (Octafian, 2011) Jenis Kardinalitas ada 3 macam yaitu ***One to One*** **(1:1)**, ***One to Many*** **(1:N)** dan ***Many to Many*** **(N:M)**.

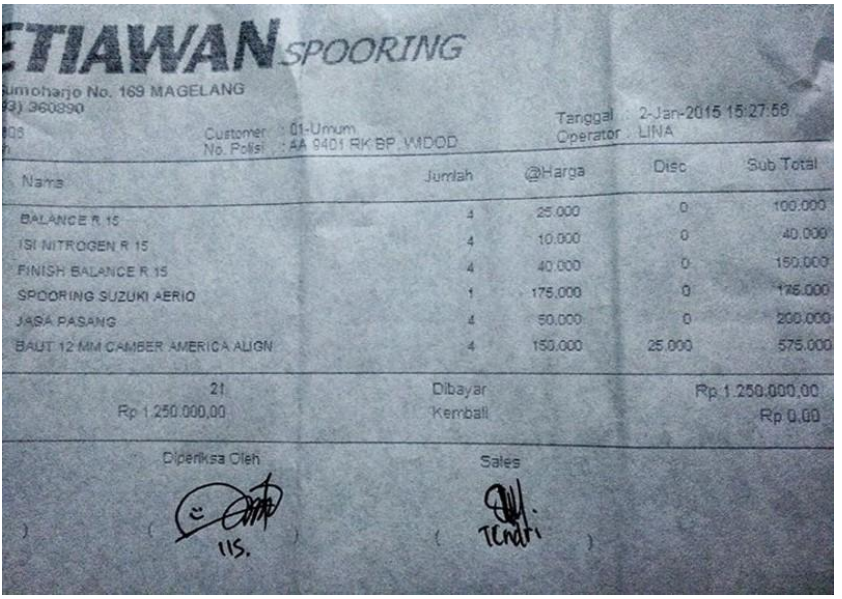
One to One **(1:1),** merupakan suatu hubungan yang hanya terbuat dari satu entitas pada tabel pertama dengan entitas lainnya pada tabel kedua. Contoh : siswa memakai dasi, berarti satu siswa hanya memakai satu dasi. Sopir mengemudi motor, berarti seorang sopir hanya mengemudi satu buah motor.

One to Many **(1:N),** merupakan hubungan yang terbuat dari satu entitas pada tabel pertama namun dapat berhubungan dengan satu atau lebih dari entitas pada tabel kedua. Contoh : siswa memiliki sepatu, berarti seorang siswa bisa jadi memiliki lebih dari satu sepatu.

Many to Many **(N:M)**, merupakan hubungan yang terbuat dari satu atau lebih entitas pada tabel pertama dengan entitas satu atau lebih pada tabel kedua. Contoh : siswa mengikuti komunitas, berarti suatu komunitas bisa diikuti oleh lebih dari satu siswa dan banyak siswa bisa jadi mengikuti banyak komunitas.

2.10. Contoh Normalisasi

Sebelum kita melakukan normalisasi kita perlu memahami ruang lingkupnya perusahaan. Sus bab ini menggunakan contoh nyata dari Toko Setiawan Spooring yang akan dibuatkan *database*-nya dengan Teknik normalisasi seperti gambar 22. Toko Setiawaan Spooring memiliki data-data seperti penjualan pada pelayanan kasir, pengeluaran stok Gudang, pengelolaan data pelanggan dan pencatatan kinerja sales dan inovator. Gambar 23 merupakan salah satu contoh invoice Nota Setiawan Spooring untuk memudahkan Teknik normalisasi.


ETIAWAN SPOORING
Sumoharjo No. 169 MAGELANG
93) 360890
Customer : 01-Umum
No. Polisi : AA 9401 RK BP WIDOD
Tanggal : 2-Jan-2015 15:27:56
Operator : LINA

Nama	Jumlah	@Harga	Disc	Sub Total
BALANCE R 15	4	25.000	0	100.000
ISI NITROGEN R 15	4	10.000	0	40.000
FINISH BALANCE R 15	4	40.000	0	160.000
SPOORING SUZUKI AERIO	1	175.000	0	175.000
JASA PASANG	4	50.000	0	200.000
BAUT 12 MM CAMBER AMERICA ALIGN	4	150.000	25.000	575.000

21 Rp 1.250.000,00 | Dibayar Rp 1.250.000,00 | Kembali Rp 0,00

Diperiksa Oleh (115.) Sales (Tenri)

Gambar 23 Alur Normalisasi

Unnormalized Form

Tahap pertama membuat *unnormalized form* dengan menuliskan *field file* atau menuliskan semua kemungkinan yang tertulis pada

invoice. Selanjutnya melakukan persiapan untuk memasuki tahap berikutnya dengan memberi warna. Terdapat 5 warna biru, orange, merah muda, kuning, abu-abu dan hijau. Untuk warna biru adalah kemungkinan entitas yang berkaitan dengan barang. Warna hijau adalah kemungkinan entitas yang berkaitan dengan faktur. Warna orange adalah kemungkinan entitas yang berkaitan dengan jenis bayar. Warna merah muda adalah kemungkinan entitas yang berkaitan dengan barang jenis customer. Warna kuning adalah kemungkinan entitas yang berkaitan dengan identitas customer. Dan warna abu abu adalah kemungkinan entitas yang berkaitan dengan pegawai. Dapat dilihat pada gambar 24.

No_Faktur	Jenis_bayar	Jenis_Cust	No_Pol	Nama_Cust	Tgl_Faktur	Op_Faktur	Kode_Brg
E418	Cash	01-Umum	AA 9401 RK	BP WIDODO	1/2/2015	LINA	TYT036
							TYT037
							TYT038
							TYT039
							TYT040
							TYT041

Nama_Brg	Jml_Brg	Harga_Brg	Disc_Brg	Subtotal_Brg	Total_Brg	Bayar_Faktur	Kembali Faktur
BALANCE R15	4	Rp25,000	0	Rp100,000	Rp1,250,000	Rp1,250,000	0
ISI NITROGEN 15	4	Rp10,000	0	Rp40,000			
FINISH BALANCE R15	4	Rp40,000	0	Rp160,000			
SPOORING SUZUKI AERIO	1	Rp175,000	0	Rp175,000			
JASA PASANG	4	Rp50,000	0	Rp200,000			
BAUT 12 mm CAMBER AMERICA ALIGN	4	Rp150,000	Rp25,000	Rp575,000			

Pemeriksa_Faktur	Sales_faktur
IIS	TENDRI

Gambar 24 Tahap Normalisasi *Unormalized Form*

1NF*(First Normal Form)*

Tahap berikutnya merupakan *first normal form* dengan syarat tidak ada duplikasi data, dekomposisi tabel atau memisahkan antar

entitas (kasus ini berarti memisahkan antar warna atau entitas yang sudah dibuat pada tahap sebelumnya), Serta memunculkan sebuah *key* baru yaitu *primary key*.

Entitas pertama dapat dilihat pada gambar 25 yaitu faktur. Diketahui bahwa no_faktur menentukan tgl_faktur, jml_brg, dics_brg, subtotal_brg, total brg, bayar_faktur dan Kembali_faktur. Entitas yang ditentukan oleh no_faktur berarti bergantung kepada no_faktur sehingga kita menjadikan no_faktur sebagai *primary key* atau PK dan beri tanda garis bawah dan bintang satu untuk membedakannya.

No_Faktur*	Tgl_Faktur	Jml_Brg	Disc_Brg	Subtotal_Brg
E418	1/2/2015	4	0	Rp100,000
		4	0	Rp40,000
		4	0	Rp160,000
		1	0	Rp175,000
		4	0	Rp200,000
		4	Rp25,000	Rp575,000

Total_Brg	Bayar_Faktur	Kembali Faktur
Rp1,250,000	Rp1,250,000	0

Gambar 25 Entitas Faktur *first normal form*

Entitas berikutnya dapat dilihat pada gambar 26 yaitu customer. Diketahui bahwa no_pol menentukan nama_cust. Sehingga nama_cust ditentukan bergantung kepada no_pol sehingga kita menjadikan no_pol sebagai *primary key* atau PK dan beri tanda garis bawah dan bintang satu untuk membedakannya.

No_Pol*	Nama_Cust
AA 9401 RK	BP WIDODO

Gambar 26 Entitas customer *first normal form*

Entitas berikutnya dapat dilihat pada gambar 27 yaitu jenis bayar. Diketahui bahwa jenis bayar hanya memiliki atribut satu, maka

kita perlu memberikan atribut tambahan sebagai bentuk imajinasi kita bahwa pada entitas jenis bayar membutuhkan atribut sebagai *primary key* yaitu kode_jenis_byr dengan tanda garis bawah dan bintang satu untuk membedakannya.

Kode_jns_byr*	Jenis_bayar
	Cash
	Credit Card
	Invoice

Gambar 27 Entitas Jenis bayar *first normal form*

Entitas berikutnya dapat dilihat pada gambar 28 yaitu jenis customer. Sama seperti jenis bayar, customer hanya memiliki atribut satu, maka kita perlu memberikan atribut tambahan sebagai bentuk imajinasi kita bahwa pada entitas customer membutuhkan atribut sebagai *primary key* yaitu kode_jenis_cust dengan tanda garis bawah dan bintang satu untuk membedakannya.

Kode_jns_Cust*	Jenis_Cust
	01-Umum
	02-Member
	03-Kantor
	04-Mitra

Gambar 28 Entitas jenis customer *first normal form*

Entitas berikutnya dapat dilihat pada gambar 29 yaitu barang. Diketahui bahwa nama_brg dan harga_brg bergantung kepada no_brg maka kita menjadikan kode_brg sebagai *primary key* dengan tanda garis bawah dan bintang satu untuk membedakannya.

Kode_Brg*	Nama_Brg	Harga_Brg
TYT036	BALANCE R15	Rp25,000
TYT037	ISI NITROGEN 15	Rp10,000
TYT038	FINISH BALANCE R15	Rp40,000
TYT039	SPOORING SUZUKI AERIO	Rp175,000
TYT040	JASA PASANG	Rp50,000
TYT041	BAUT 12 mm CAMBER AMERICA ALIGN	Rp150,000

Gambar 29 Entitas Barang *first normal form*

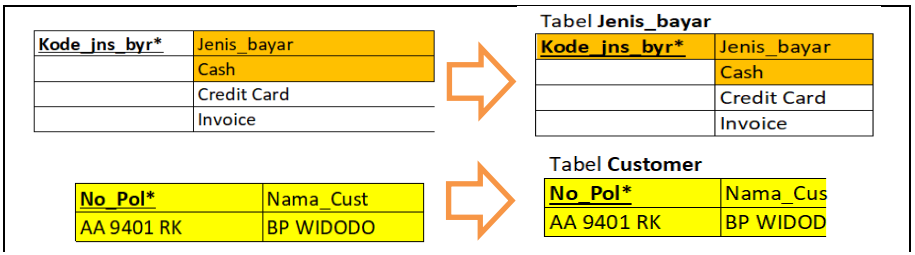
Entitas berikutnya adalah pegawai atau karyawan. Apabila kita memperhatikan dari sudut pandang nama pegawai seperti iis, sisca, puput maka *primary key* yang digunakan adalah kode_karyawan. Apabila kita lebih memperhatikan pada bagian tugas seperti op, pemerika, sales maka yang digunakan adalah kode_tugas. Sehingga kedua hal tersebut memiliki keunikan yang berbeda. Seperti pada gambar 30.

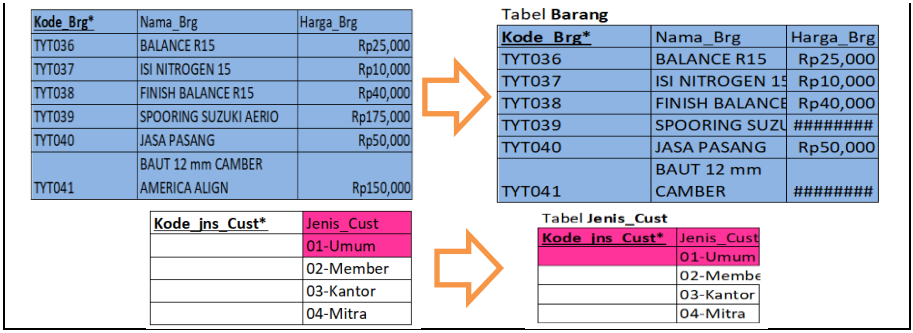
Kode_karyawan*	**Kode_tugas***	Op_Faktur	Pemeriksa_Faktur	Sales_faktur
		iis	Sisca	Puput

Gambar 30 Entitas karyawan *first normal form*

2NF*(Second Normal Form)*

Tahap berikutnya adalah *second normal form* dengan syarat sudah memenuhi 1NF, menghapus subset data serta memberi *foreign key* untuk membuat sebuah relasi serta tidak adanya partial dependency. Pertama memilih entitas yang khusus sudah 2NF dari hasil 1NF berarti entitas tersebut memang tidak memiliki subset data dan *partial dependency*. Kemudian memberi nama tabel entittas tersebut. Seperti pada gambar 31.





Gambar 31 contoh entitas yang udah memenuhi *second normal form*

Pada entitas karyawan terdapat ketergantungan antara kode_tugas dan kode_karyawan seperti pada gambar 30 itu yang disebut *partial dependency*. Untuk menghilangkan *partial dependency* maka perlu dekomposisi tabel (pemisahan/pemecahan tabel menjadi seperti gambar 32 serta penambahan tabel baru pada gambar 33 sebagai implementasi relasi yang memiliki atribut *foreign key* yaitu kode_karyawan dan kode_tugas yang ditandai dengan bintang 2.

Tabel **Karyawan**

Kode_karyawan*	Nama Karyawan
K_001	iis
K_002	Sisca
K_003	Puput
K_004	Andre
K_005	Bobi
K_006	Charlie

Tabel **Penugasan**

Kode_tugas*	Nama_tugas
TG_01	Op_Faktur
TG_02	Pemeriksa_Faktur
TG_03	Sales_faktur
TG_04	Manajer
TG_05	Supervisor
TG_06	Kepala_Cabang

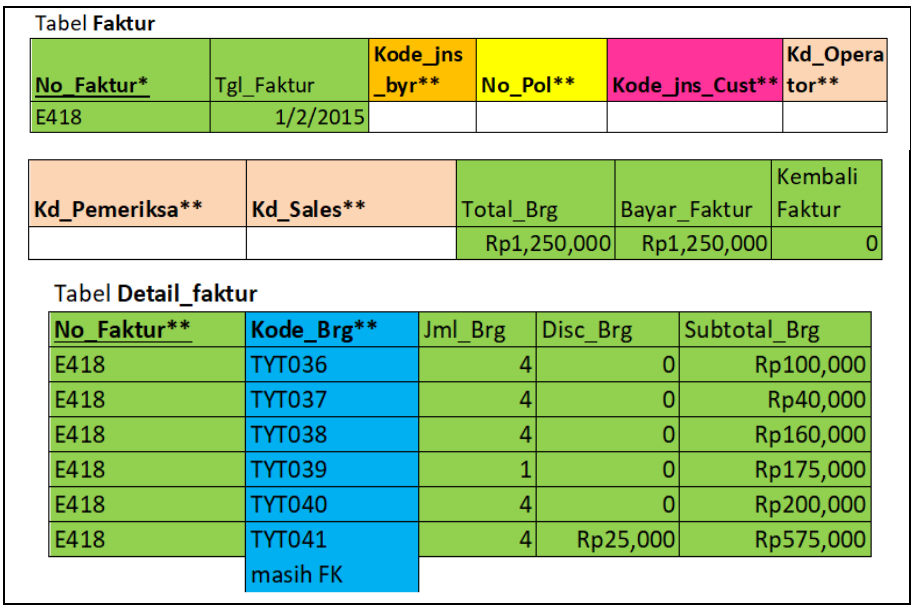
Gambar 32 Implementasi dekomposisi tabel *second normal form*

Tabel **Tugas_sebagai**

Kode_Tgs_Sebagai*	Kode_karyawan**	Kode_tugas**
S_001	K_001	TG_01
S_002	K_002	TG_02
S_003	K_003	TG_03
S_004	K_004	TG_04
S_005	K_005	TG_05

Gambar 33 Implementasi relasi tabel tugas sebagai *second normalize form*

Implementasi relasi lainnya terlihat pada gambar 34 bahwa, tabel faktur yang memiliki *foreign key* seperti kode_jnd_byr, no_pol, kode_jns_cust dll. Selain itu munculnya tabel detail faktur karena adanya atribut yang multi value seperti jml_brg, disc_brg dan subtotal_brg. Missal suatu customer memungkinkan membeli barang yang banyak maka agar tidak terlalu banyak value yang diulang maka cukup dipecah saja bagian detail faktur. Sehingga bagian faktur cukup ringkas dengan implementasi relasi dan cukup pada bagian detail fakturnya yang menjelaskan barang apa saja.



Tabel **Faktur**

No_Faktur*	Tgl_Faktur	Kode_jns_byr**	No_Pol**	Kode_jns_Cust**	Kd_Operator**
E418	1/2/2015				

Kd_Pemeriksa**	Kd_Sales**	Total_Brg	Bayar_Faktur	Kembali Faktur
		Rp1,250,000	Rp1,250,000	0

Tabel **Detail_faktur**

No_Faktur**	Kode_Brg**	Jml_Brg	Disc_Brg	Subtotal_Brg
E418	TYT036	4	0	Rp100,000
E418	TYT037	4	0	Rp40,000
E418	TYT038	4	0	Rp160,000
E418	TYT039	1	0	Rp175,000
E418	TYT040	4	0	Rp200,000
E418	TYT041	4	Rp25,000	Rp575,000
	masih FK			

Gambar 34 Implementasi relasi faktur dan detail faktur *second normal form*

Sehingga tahap 2NF menghasilkan total 9 entitas dengan 2 kategori yaitu 6 tabel master (jenis bayar, customer, jenis_cust, barang, karyawan, penugasan) dan 3 tabel implementasi relasi (tugas_sebagai, faktur, detail_faktur).

3NF(Third Normal Form)

Tahap kalii ini Kembali memilah tabel yang sudah 3NF dari tahap sebelumnya seperti pada gambar 35 dengan syarat tidak memiliki *transitif dependency*.

Tabel **Jenis_bayar**

Kode_jns_byr*	Jenis_bayar
	Cash
	Credit Card
	Invoice

Tabel **Customer**

No_Pol*	Nama_Cus
AA 9401 RK	BP WIDOD

Tabel **Barang**

Kode_Brg*	Nama_Brg	Harga_Brg
TYT036	BALANCE R15	Rp25,000
TYT037	ISI NITROGEN 15	Rp10,000
TYT038	FINISH BALANCE	Rp40,000
TYT039	SPOORING SUZU	Rp175,000
TYT040	JASA PASANG	Rp50,000
TYT041	BAUT 12 mm CAMBER	Rp150,000

Tabel **Jenis_Cust**

Kode_jns_Cust*	Jenis_Cust
	01-Umum
	02-Membe
	03-Kantor
	04-Mitra

Tabel **Karyawan**

Kode_karyawan*	Nama Karyawan
K_001	iis
K_002	Sisca
K_003	Puput
K_004	Andre
K_005	Bobi
K_006	Charlie

Tabel **Penugasan**

Kode_tugas*	Nama_tugas
TG_01	Op_Faktur
TG_02	Pemeriksa_Faktur
TG_03	Sales_faktur
TG_04	Manajer
TG_05	Supervisor
TG_06	Kepala_Cabang

Tabel **Tugas_sebagai**

Kode_Tgs_Sebagai*	Kode_karyawan**	Kode_tugas**
S_001	K_001	TG_01
S_002	K_002	TG_02
S_003	K_003	TG_03
S_004	K_004	TG_04
S_005	K_005	TG_05

Tabel **Faktur**

No_Faktur*	Tgl_Faktur	Kode_jns_byr**	No_Pol**	Kode_jns_Cust**	Kd_Operator**
E418	1/2/2015				

Kd_Pemeriksa**	Kd_Sales**	Total_Brg	Bayar_Faktur	Kembali Faktur
		Rp1,250,000	Rp1,250,000	0

Gambar 35. Entitas yang telah memenuhi *third normal form*

Entitas yang memiliki transitive dependency dapat didekomposisitabel atau dirubah status menjadi atributnya. Pada gambar 34 implementasi relasi untuk tabel detail faktur, transitive dependency ditandai dengan jml_brg, diskon_brg dan sub total yang bergantung kepada no faktur dan juga kepada kode barang. Seharusnya 3 aribut tadi hanya bergantung pada PK. Sehingga perlu adanya Tindakan untuk menghindari transitive dependency dengan merubah status atribut. Pada gambar 36, dibuahnya statsu FK kode_brg menjadi PK kode_brg sehingga tabel detil faktur memiliki 2 PK yaitu no_faktur dan kode_brg.

Tabel **Detail_faktur**

No_Faktur*	Kode_Brg*	Jml_Brg	Disc_Brg	Subtotal_Brg
E418	TYT036	4	0	Rp100,000
E418	TYT037	4	0	Rp40,000
E418	TYT038	4	0	Rp160,000
E418	TYT039	1	0	Rp175,000
E418	TYT040	4	0	Rp200,000
E418	TYT041	4	Rp25,000	Rp575,000

Gambar 36. Hasil penghilangan *transitive dependency* pada *third normal form*

2.11. Implementasi Relasi pada Setiawan Spooring

Contoh dalam *database* SETIAWAN SPOORING yang terdiri dari tabel Karyawan, tabel Tugas_sebagai dan penugasan. Pada gambar 37 Tabel karyawan tersusun dari 2 kolom, yaitu : kode_karyawan dan Nama_karyawan serta Tabel penugasan terdiri dari 2 kolom yaitu : kode_tugas dan nama_tugas. Pada gambar 38 Tabel tugas_sebagai tersusun dari 3 kolom yaitu : kode_tugas_sebagai, kode_karyawan, kode_tugas.

PK

Tabel **Karyawan**

Kode_karyawan*	Nama Karyawan
K_001	Iis
K_002	Sisca
K_003	Puput
K_004	Andre
K_005	Bobi
K_006	Charlie

PK

Tabel **Penugasan**

Kode_tugas*	Nama_tugas
TG_01	Op_Faktur
TG_02	Pemeriksa_Faktur
TG_03	Sales_faktur
TG_04	Manajer
TG_05	Supervisor
TG_06	Kepala_Cabang

Gambar 37contoh tabel karyawan dan penugasan

PK

Tabel **Tugas_sebagai**

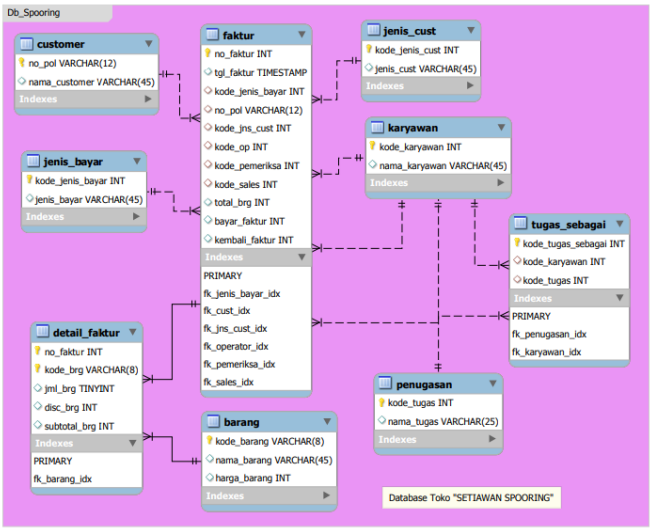
FK

Kode_Tgs_Sebagai*	Kode_karyawan**	Kode_tugas**
S_001	K_001	TG_01
S_002	K_002	TG_02
S_003	K_003	TG_03
S_004	K_004	TG_04
S_005	K_005	TG_05

Gambar 38 Implementasi relasi tabel tugas_sebagai

Database SETIAWAN SPOORING tersebut direlasikan berdasarkan kunci *primary* (PK) dan kunci tamu (FK). Tabel karyawan dan tabel_tugas sebagai direlasikan berdasarkan kode_karyawan. Kode_karyawan di tabel karyawan sebagai kunci *primary* dan kode_karyawan pada tabel tugas_sebagai, sebagai kunci tamu (FK). Tabel tugas_sebagai dan tabel penugasan direlasikan berdasar kode_tugas. Kode_tugas pada tabel penugasan sebagai kunci *primary* (PK) dan kode_tugas pada tabel tugas_sebagai, sebagai kunci tamu (FK).

Gambar 39 adalah implementasi relasi dari 3NF yang telah dilakukan pada *database* SETIAWAN SPOORING. Ditandai dengan adanya garis putus-putus yang menghubungkan antar entitas dengan entitas lainnya.



Gambar 39 Implementasi relasi *database* Setiawan spooring

BAB 3

PEMBUATAN *DATABASE* BERBASIS OBJEK

3.1. Tools Pengembangan *Database*

Data yang dikelola dalam sebuah *database* memerlukan *tools* yang dapat mengakses data baik untuk mendefinisikan struktur *database* dan tabel / entitas ataupun memanipulasi data dalam *database*. *Tools* yang dimaksud berupa software yang dikenal dengan nama *Database Management System* (DBMS). *Database* MySQL merupakan software yang berbentuk *database* relasional atau disebut *Relational Database Management System* (RDBMS). RDBMS pada umumnya menggunakan suatu bahasa permintaan yang bernama SQL *(Structured Query Language)* atau lebih dikenal dengan Query. (Letwoski, 2015)

Pada mata kuliah Basis Data 1 ini, kita akan melakukan Pemodelan Data (Desain) yang memungkinkan membuat model skema *database* berbasis objek yang dikenal dengan *Enhanced Entity Relationship* (EER). EER yang akan kita pelajari tidak luput dari DBMS yang akan kita gunakan. Pada sub bab selanjutnya akan dijelaskan DBMS yang akan kita gunakan selama proses perkuliahan hingga tingkat akhir.

1. MySQL VS MariaDB

DBMS yang akan kita bahas pada sub bab ini adalah My SQL dan MariaDB. Sebagaimana yang disebutkan pada situs resmi MySQL,

bahwasanya pengguna MySQL adalah YouTube, Paypal, Google, Facebook, Twitter, Netflix, Github, LinkedIn, dan lain-lain (My SQL, 2020). Sedangkan MariaDB menyatakan bahwa Red Hat, Samsung, Nokia, Microsoft, Booking.com, Alibaba Cloud, Tencent Cloud, Development Bank of Singapore (DBS), Gaming Innovation Group (GIG) dan IBM merupakan pengguna MariaDB (MariaDB, 2020a, 2020b).

Pengembangan pertama pada tahun 1994 dibelakangi oleh perusahaan asal Swedia, MySQL dikembangkan 3 orang founder yaitu David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Pada tahun 2008, MySQL AB diakuisisi oleh Sun Microsystems. Pengembangan terus berlanjut hingga pada tahun 2010 Oracle mengakuisisi Sun Microsystem (Wikipedia, 2015).

Dampak dari akuisisi tersebut salah satu founder dari MySQL Michael "Monty" Widenius memulai projek baru dengan nama MariaDB. Dia juga menarik beberapa developer dari MySQL itu sendiri. Hal itu dia lakukan karena ada perbedaan sudut pandang antara Michael "Monty" Widenius dan Oracle dalam arah pengembangan MySQL (Huda, 2020). MySQL dan MariaDB memiliki banyak kesamaan. Khususnya dari segi struktur data dan query.

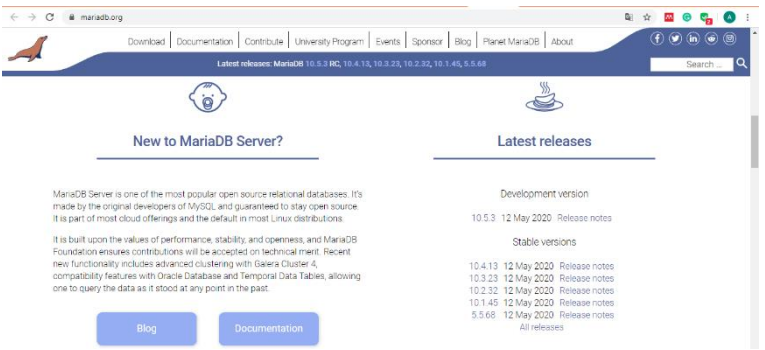
MySQL seperti halnya RDBMS yang lain, ia menyimpan datanya dalam bentuk tabel. Setiap tabel memiliki kolom. Dan setiap baris pada satu tabel memiliki jumlah kolom yang sama. MySQL menggunakan *primary key* untuk membedakan antara satu baris data dengan baris

data yang lain, ia juga menggunakan *foreign key* untuk membangun relasi antar satu baris data pada satu tabel tertentu, dengan baris data di tabel yang lain. MySQL juga memiliki fitur *constraint, views, stored procedures* dan komponen inti lainnya yang bisa kita gunakan untuk menunjang kebutuhan bisnis kita. Gambar 40 menunjukan situs MySQL official.

Gambar 40 MySQL Official Site

MariaDB karena ia adalah fork dari MySQL, otomatis ia juga mendukung semua fitur di atas. Kecuali fitur-fitur baru dari MySQL yang dikembangkan setelah bagian dari MariaDB seperti support JSON dan sebagainya. Dengan keidentikan yang dimiliki MariaDB dan MySQL ini, kita bisa dengan mudah saat berpindah dari MySQL *key* MariaDB atau sebaliknya. Bahkan perintah command line-nya pun sama yaitu dengan nama program mysql. Dari segi query, tidak ada perbedaan antara MySQL dan MariaDB. Ini membuat kita bisa mengimpor/mengekspor data dengan mudah dari MySQL *key* MariaDB dan begitu pula sebaliknya (Huda, 2020).

Perbedaan antara MySQL dan MariaDB adalah dari sisi pengembangan. MySQL yang awalnya adalah Open Source, saat diakuisisi oleh Oracle akhirnya pengembangannya dikendalikan oleh Oracle secara tertutup. Semua keputusan arah pengembangan tidak dikeluarkan *key* publik. Sedangkan MariaDB memang betul-betul murni dikendalikan oleh komunitas alias bersifat *community-driven project*. Segala macam bentuk keputusan yang berkaitan dengan arah pengembangan, atau bahkan review, debat, diskusi mengenai projek MariaDB, bisa kita akses melalui *mailing-list* yang bersifat publik. Kode sumber MySQL bisa kita lihat dan bisa kita akses di repositori resmi MySQL di github (https://github.com/mysql/mysql-server). Kode sumber MariaDB bisa kita lihat dan bisa kita akses di repositori resmi MariaDB di github (https://github.com/MariaDB/server) pada gambar 41.



Gambar 41 MariaDB Official Site

2. XAMPP

XAMPP adalah sebuah paket kumpulan software yang terdiri dari apache, MariaDB, PHP, dan Perl. Pengalaman dari pengembang web

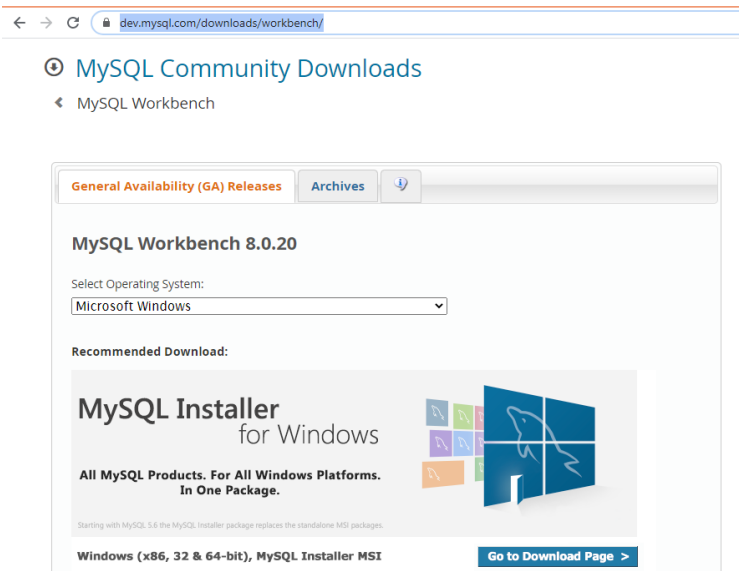
bahwa tidak mudah untuk menginstal server web Apache dan semakin sulit jika Anda ingin menambahkan MariaDB, PHP dan Perl. XAMPP memberi kemudahan instalasi bagi para pengembang untuk masuk *key* dunia Apache. Agar nyaman bagi pengembang, XAMPP dikonfigurasi dengan semua fitur yang dihidupkan. Saat ini ada distribusi untuk Windows, Linux, dan OS X. Untuk master software XAMPP bisa didownload gratis di situs resminya www.apachefriends.org/en/xampp.html seperti pada gambar 42.

Gambar 42 XAMPP Official Site

Pada proses perkuliahan kedepannya, mahasiswa akan banyak menggunakan XAMPP sebagai salah satu tools dalam membangun website menggunakan bahasa PHP. Sehingga mahasiswa dituntut untuk familiar mengembangkan *database* menggunakan MariaDB. Dikarenakan struktur data MariaDB identik dengan MySQL, maka dalam pengembangan Pemodelan Data (Desain) juga dapat menggunakan tools MySQL.

3. MySQL Worbench

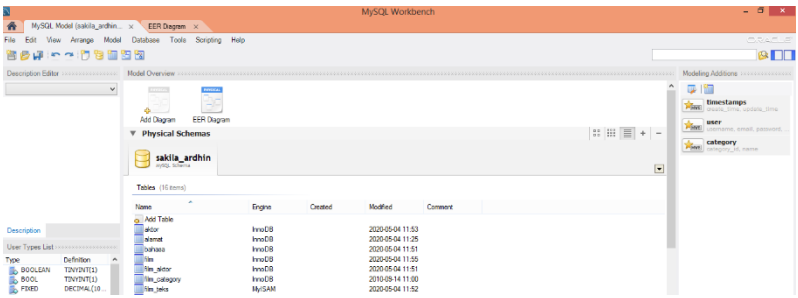
MySQL *Workbench* adalah alat grafis untuk bekerja dengan server dan *database* MySQL. MySQL *Workbench* terdiri dari 2 edisi yaitu Edisi Komersial dan Edisi Komunitas. Dalam perkuliahan ini kita akan menggunakan Edisi Komunitas yang merupakan open source. MySQL *Workbench* dapat diunduh di https://dev.mysql.com/downloads/*workbench*/ seperti pada gambar 43.



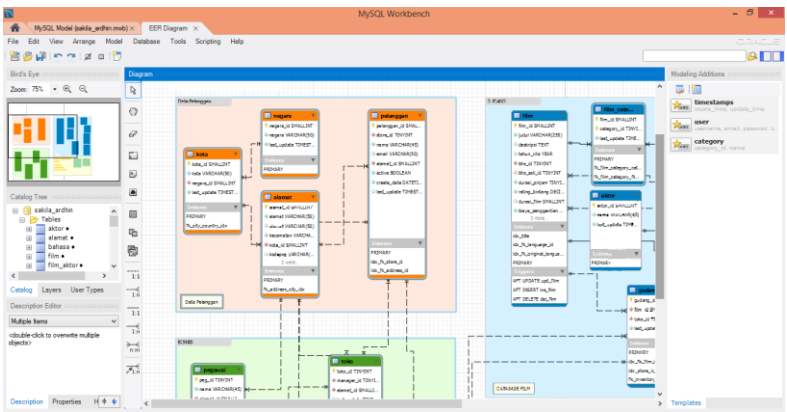
Gambar 43 Link download MySQL *Workbench*

MySQL *Workbench* sangat membantu dalam pemodelan data (desain). ERD ataupun hasil normalisasi dapat secara otomatis dibuat dalam bentuk grafis yang mudah untuk dipahami dan sudah menyesuaikan aturan-aturan dasar dalam pembuatan *database*. Sehingga sangat membantu mahasiswa dalam proses memahami pembuatan ERD yang benar. Teknik ini dikenal sebagai *Enhanced*

Entity Relationship (EER). Gambar 44 dan 45 merupakan tampilan yang digunakan selama desain *database* secara grafis.



Gambar 44 Tampilan Skema Fisik pada MySQL *Workbench*



Gambar 45 Tampilan Diagram EER pada MySQL *Workbench*

3.2. Mengenal MySQL *Workbench*

MySQL *Workbench* memiliki beberapa seri sebagai bentuk perbaikan. Dirilis pada https://dev.mysql.com/doc/relnotes/*workbench*/en/ secara rutin, terlihat bahwa saat ini MySQL *Workbench* yang baru saja dirilis merupakan seri 8.0. Penjelasan setiap perbaikan di setiap seri juga dapat diakses melalui link di atas.

Secara umum, MySQL *Workbench* adalah alat grafis untuk bekerja dengan server dan *database* MySQL. MySQL *Workbench* mendukung penuh server MySQL versi 5.6 dan lebih tinggi. Ini juga kompatibel dengan versi 5.x server MySQL yang lebih lama, kecuali dalam situasi tertentu (seperti menampilkan daftar proses) karena tabel sistem berubah. Itu tidak mendukung versi server MySQL 4.x. MySQL *Workbench* tersedia dalam dua edisi yaitu Edisi Komunitas dan Edisi Komersial. Edisi Komunitas tersedia gratis. Edisi Komersial menyediakan fitur Perusahaan tambahan, seperti MySQL Enterprise Backup, MySQL Firewall, dan Audit MySQL.

Dalam penggunaan secara menyeluruh fungsi MySQL *Workbench* mencakup lima topik utama:

a. Pengembangan SQL yang membantu dalam pembuatan dan pengelolaan koneksi *key* server *database*. Selain itu juga untuk mengkonfigurasi parameter koneksi, MySQL *Workbench* menyediakan kemampuan untuk mengeksekusi query SQL pada koneksi *database* menggunakan Editor SQL bawaan.
b. Pemodelan Data (Desain) yang membantu dalam pembuatan model skema basis data Anda secara grafis, merekayasa balik dan meneruskan antara skema dan basis data secara langsung, dan mengubah semua aspek *database* menggunakan Table Editor yang komprehensif. Table Editor menyediakan fasilitas yang mudah digunakan untuk mengubah *Tables*, *Columns*,

Indexes, Triggers, Partitioning, Options, Inserts dan *Privileges, Routines* serta *Views.*

c. Administrasi Server yang membantu dalam pengelolaan instance server MySQL dengan mengelola user, melakukan *backup and recovery*, memeriksa data audit, melihat kesehatan *database*, dan memantau kinerja server MySQL.
d. Migrasi Data yang memudahkan dalam bermigrasi dari Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL, dan tabel, objek dan data RDBMS lainnya *key* MySQL. Migrasi juga mendukung migrasi dari versi MySQL sebelumnya *key* rilis terbaru.
e. MySQL *Enterprise Support* yaitu dukungan untuk produk-produk Enterprise seperti MySQL *Enterprise Backup*, MySQL Firewall, dan Audit MySQL.

Namun demikian, pada mata kuliah Basis Data ini hanya akan dibahas dalam Pemodelan Data (Desain) yang membantu secara grafis dalam mendesain sebuah skema *database* dan diagram EER.

1. Instalasi

Pada mata kuliah Basis Data ini menggunakan MySQL *Workbench* 6.0 dengan pertimbangan bahwa seri ini adalah seri terakhir yang minimalis pada OS Windows 8 dan 10. Komputer dengan OS dibawah Windows 8 dan 10 dapat menggunakan MySQL *Workbench* 5.0. Pengembangan setelah versi 6.0 lebih dikhususkan pada fungsi MySQL sebagai administrasi server dan migrasi data. Selain itu setelah

versi 6.0 membutuhkan minimum *requirement hardware* yang lebih tinggi pada RAM sehingga dapat membebani komputer.

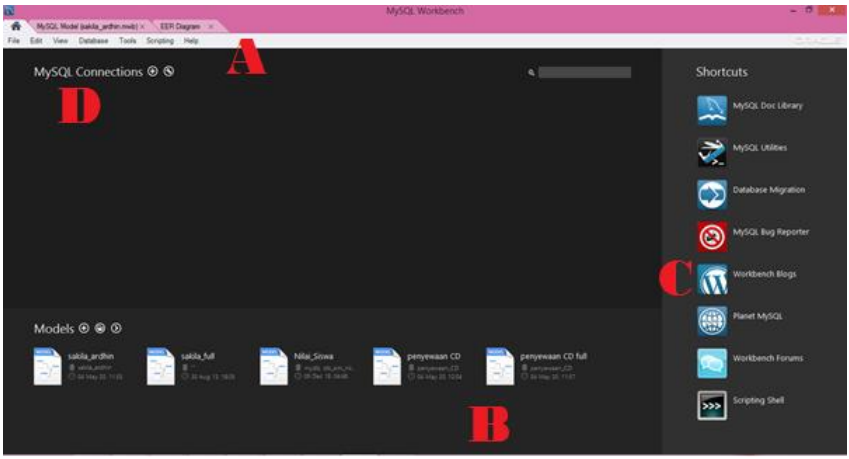
Untuk master program dapat diunduh pada https://dev.mysql.com/downloads/*workbench*/ atau dapat mengakses pada https://moca.ummgl.ac.id/. Dalam proses instalasi MySQL *Workbench* membutuhkan Microsoft .NET Framework dan Microsoft Visual *KEY*++ Redistributable.

Dalam proses instalasi, maka pilih Application dan klik tombol tersebut kemudian ikuti langkahnya hingga akhir. Yang harus diperhatikan saat instalasi, adalah letak aplikasi tersebut. Usahakan path yang anda pilih adalah drive yang sama dengan anda menginstall XAMPP. Sehingga MySQL *Workbench* dan XAMPP sukses terinstall pada komputer anda.

2. Tampilan Pemodelan Data (Desain)

Pada saat pembuatan *database*, pada pertemuan sebelumnya telah disampaikan bahwa ada 2 teknik pembuatan *database*. Pembuatan *database* berorientasi pada objek dari sistem yang dibangun menggunakan teknik *Entity Relationship Diagram* (ERD). Pembuatan *database* berorientasi pada data dari sistem yang dibangun menggunakan teknik Normalisasi. Setelah desain selesai dilakukan baik menggunakan ERD/Normalisasi maka dapat kita gambarkan secara grafis agar memudahkan dalam simulasi dan mengecek keabsahan *database* yang kita kembangkan. Tools yang kita gunakan

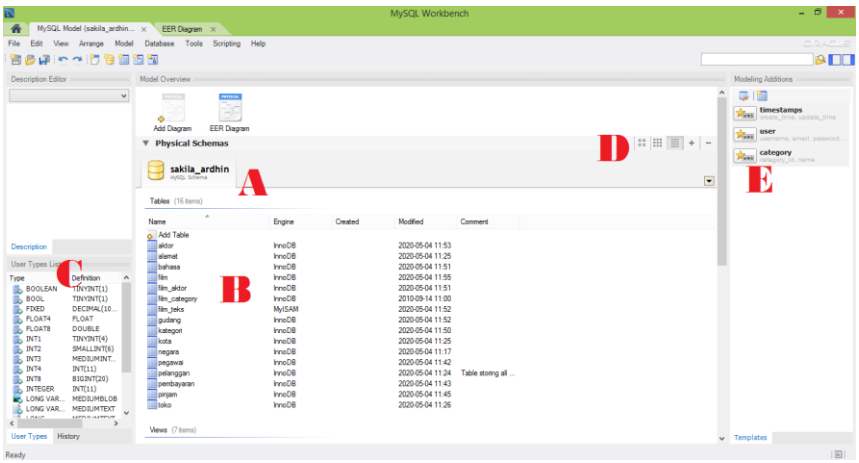
dalam pembuatan skema *database* secara grafis adalah MySQL *Workbench* dan XAMPP sebagai engine dalam server *database*.



Gambar 46 Tampilan awal MySQL *Workbench* 6.0

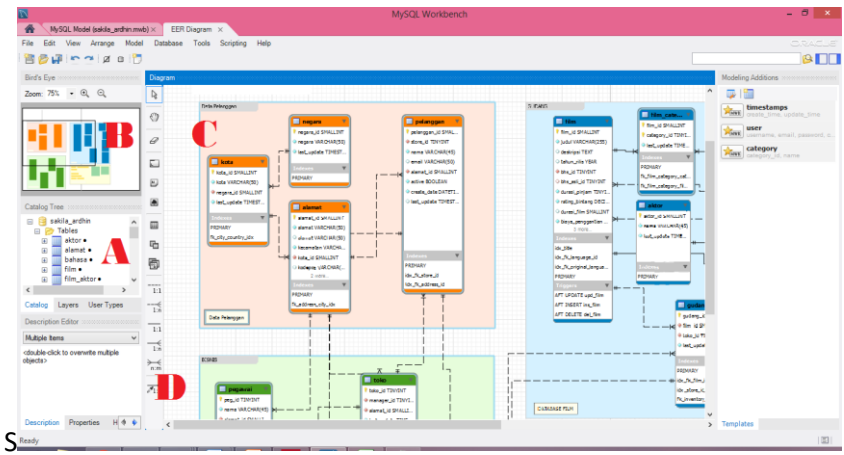
Pada gambar 46 diatas merupakan tampilan awal saat MySQL *Workbench* diakses. Pada label A merupakan menu yang aktif yaitu MySQL Model dan EER Diagram. MySQL Model untuk membuat skema *database* secara fisik dan EER Diagram secara grafis. Pada label B merupakan desain *database* pada project *workbench* yang pernah kita buat sebelumnya. Pada label *KEY* terdapat tombol pintasan MySQL Doc Library, MySQL Utilities, *Database* Migration, MySQL Bug Reporter, *Workbench* Blogs, Planet MySQL, *Workbench* Forums dan Scripsting Shell. Tombol pintasan yang sangat membantu adalah MySQL Doc Library karena merupakan panduan dalam menggunakan MySQL *Workbench*. Label D merupakan tombol untuk menghubungkan MySQL *Workbench* dengan server *database* dari engine di XAMPP.

Pada gambar 47 di bawah ini merupakan tampilan setelah kita memilih desain *workbench* yang pernah kita kembangkan pada tampilan MySQL Model. Pada tampilan ini kita dapat membuat *database* hasil ERD/Normalisasi dengan memilih entity, views dan routine yang akan kita kembangkan. Pada label A merupakan identitas dari *database* yang kita kembangkan. Nantinya saat diekspor pada server *database* yang dijalankan di XAMPP nama *database* yang dikenali sesuai nama pada label A. Label B merupakan daftar entity, views dan routine yang dibuat pada skema *database* terpilih pada label A. Label *KEY* merupakan tipe data yang dapat dipilih pengembang *database* saat membuat *entity, views* dan *routine.* Pada label D merupakan tampilan yang dapat dipilih saat menampilkan entity, views dan routine. Tampilan dapat berupa icon, list ataupun daftar secara detail. Label E merupakan template yang disediakan MySQL *Workbench* untuk membuat timestamps, user dan category.



Gambar 47 Tampilan MySQL Model

Pada gambar 48 di bawah ini label A merupakan *Catalog Tree* yang menghubungkan antara skema *database* yang kita buat secara fisik dengan *database* yang akan kita buat secara grafis. Setiap penambahan entity, views dan routine pada skema fisik secara otomatis ditambahkan pada *Catalog Tree* (label A). Sedangkan pada label B merupakan tampilan *bird's eye* yang secara keseluruhan menunjukkan tabel yang sudah kita buat EER diagram nya. Tampilan pada label B merupakan tampilan versi utuh dari tampilan pada label *KEY*. Saat kita butuh tampilan di zoom in atau zoom out maka posisi layar di tampilan label *KEY* akan terlihat pada label B. Tampilan pada label D merupakan fitur yang tersedia untuk menggambar EER diagram secara manual.



Gambar 48 Tampilan EER Diagram

3.3. Desain *database*

Setelah desain selesai dilakukan baik menggunakan ERD/Normalisasi maka dapat kita gambarkan secara grafis agar

memudahkan dalam simulasi dan mengecek keabsahan *database* yang kita kembangkan. (Kroenke & Auer, 2015)

Tahap Persiapan

Sebelum mendesain secara grafis, hal-hal yang harus dipersiapkan adalah :

Desain *Database*

Nama *database* yang akan secara permanen dikenali dalam server *database*. Nama *database* tidak dapat dipisahkan dengan spasi. Pastikan jika mengandung 2 kata dituliskan dengan tanda garis bawah (_ *underscore*).

Desain Entitas (entitiy) atau tabel

Nama entitas (*entitiy*) atau tabel yang sudah sesuai dengan hasil teknik ERD ataupun normalisasi. Nama entitas tidak dapat dipisahkan dengan spasi. Pastikan jika mengandung 2 kata dituliskan dengan tanda garis bawah (_ *underscore*). Saat kita menggunakan spasi, tidak akan muncul eror saat didesain menggunakan MySQL *Workbench*, namun saat diimplementasikan pada server *database* akan muncul pesan eror.

Implementasi *Primary key* pada MySQL

Atribut (*field*) sebagai *primary key* akan lebih mudah dikenali jika menggunakan id_namatabel atau kode_namatabel. Walaupun pada beberapa kasus *primary key* menggunakan istilah yang lebih dikenali

masyarakat luas seperti NIK (Nomor Induk Kepegawaian). Atribut *primary key* tidak boleh berisi nilai NULL (**Not NULL**).

Atribut yang dijadikan *primary key* dapat terdiri dari lebih dari 1 atribut. Walaupun secara performa akan lebih cepat entitas yang hanya memiliki *primary key* 1 atribut saja.

MySQL bekerja lebih cepat jika dengan bilangan bulat, sehingga usahakan tipe data atribut *primary key* harus berupa **integer** misalnya **INT, BIGINT**. Tipe integer yang lebih kecil yaitu **TINYINT, SMALLINT**, dll.

Keunggulan menggunakan tipe bilangan bulat sebagai atribut *primary key* dengan adanya **AUTO_INCREMENT** yang menghasilkan urutan unik untuk kunci secara otomatis. Kunci utama baris berikutnya lebih besar dari yang sebelumnya. Pastikan bahwa kisaran nilai dari tipe integer untuk *primary key* cukup untuk menyimpan semua kemungkinan baris yang mungkin dimiliki oleh tabel. MySQL membuat **indeks bernama *PRIMARY*** dengan tipe *PRIMARY* untuk *primary key* dalam sebuah tabel.

Namun demikian, penggunaan tipe data berupa **varchar** juga menjadi pilihan beberapa pengembang *database* agar memudahkan memberi tanda pada atribut *primary key*. Contohnya seperti TG_002, AB_101, dll.

Desain Atribut (*field*)

Atribut secara umum diberi tipe data *integer* untuk bilangan dan *varchar* dengan *length* tertentu untuk *text*. Saat merubah *length* pada *varchar* dapat dilakukan saat membuat atribut melalui MySQL Model.

Desain Relasi

Pada pembuatan *database*, baik menggunakan teknik ERD ataupun Normalisasi pasti akan menghasilkan relasi. Relasi adalah hubungan antar tabel. Setiap relasi yang menghubungkan 2 tabel memiliki derajat yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain, hal ini disebut sebagai **Kardinalitas**. Jenis Kardinalitas ada 3 macam yaitu ***One to One*** **(1:1)**, ***One to Many*** **(1:N)** dan ***Many to Many*** **(N:M)**.

Implementasi *Foreign key* pada MySQL

Atribut sebagai *foreign key* pada tabel lain harus memiliki tipe data yang **PERSIS SAMA** dengan atribut pada tabel asalnya. Nama dapat dideklarasikan berbeda, namun tipe data haruslah sama.

Ada 2 perbedaan pengertian *foreign key* secara grafis dalam MySQL *Workbench* yaitu :

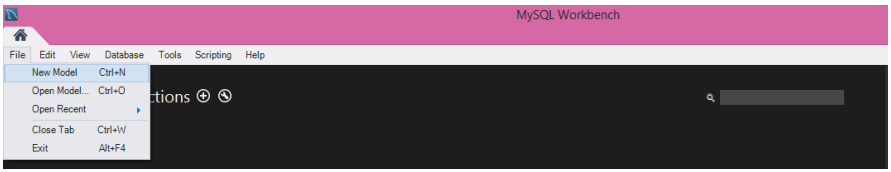
1) Hubungan yang **mengidentifikasi** diidentifikasi oleh **garis yang lurus** antara tabel

   Hubungan identifikasi adalah hubungan di mana tabel anak tidak dapat diidentifikasi secara unik tanpa tabel orang tuanya. Biasanya ini terjadi di mana tabel perantara dibuat untuk menyelesaikan hubungan banyak *key* banyak. Dalam kasus tersebut, kunci utama biasanya merupakan kunci komposit yang terdiri dari kunci utama dari dua tabel asli.
2) Hubungan yang **tidak mengidentifikasi** diidentifikasi oleh **garis putus-putus** di antara tabel

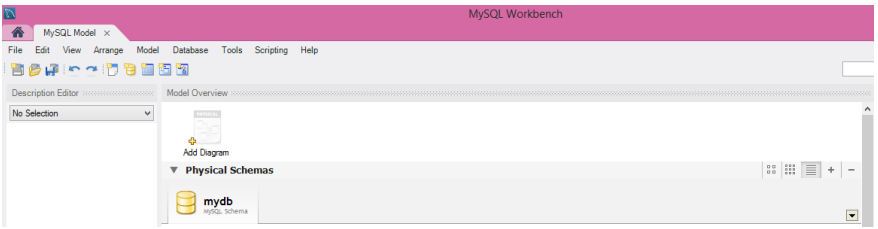
Dalam desain *database* kita akan mengenali ***primary key, unique*** **dan** index. Secara fungsi atau kegunaan *primary key* dan *unique* keduanya sama-sama digunakan untuk menjamin bahwa tidak ada duplikasi data pada atribut kunci. Atribut yang diset sebagai *primary key* atau *unique* secara otomatis akan di set sebagai *index*. *Index* sangat bermanfaat dalam proses pencarian data (*data seeking*) baik secara langsung dari DBMS ataupun saat *database* dipanggil oleh fungsi/prosedur sebuah aplikasi pemrograman.

Membuat *Database* pada MySQL *Workbench*

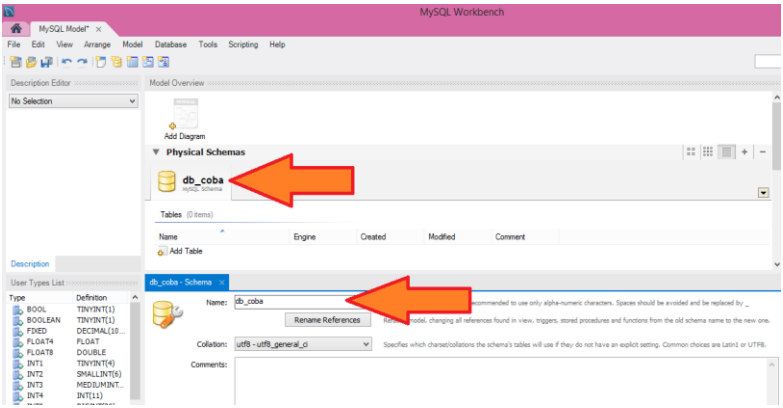
Dalam membuat *database* kita membuat sebuah model baru dengan memilih new mode sama pada gambar 49. Hingga muncul tampilan pada gambar 50. Setelah itu mengganti nama *database* sesuai gambar 51. Dan menyimpan *database* pada penyimpanan local dengan format file*.mwb seperti gambar 52.



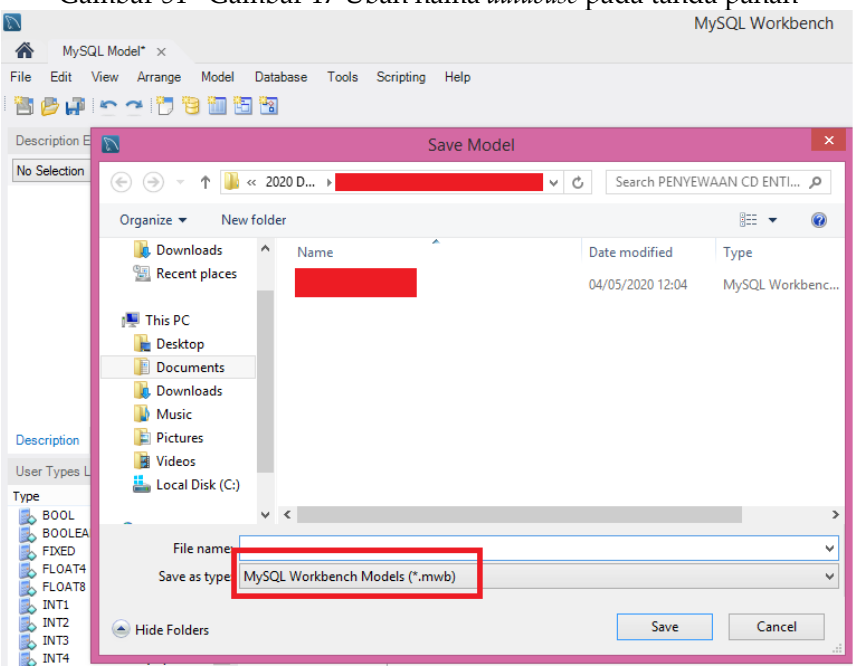
Gambar 49 Tampilan awal saat membuat model baru, pilih New Model



Gambar 50 Gambar 16 Sesaat setelah New Model di klik



Gambar 51 Gambar 17 Ubah nama *database* pada tanda panah

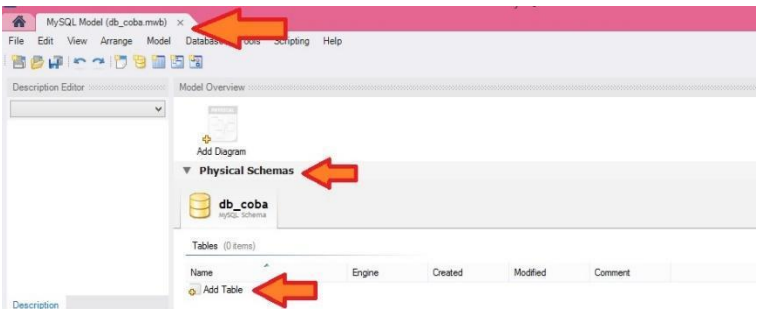


Gambar 52 Gambar 18 Simpan pada drive yang anda pilih dalam bentuk file *.mwb

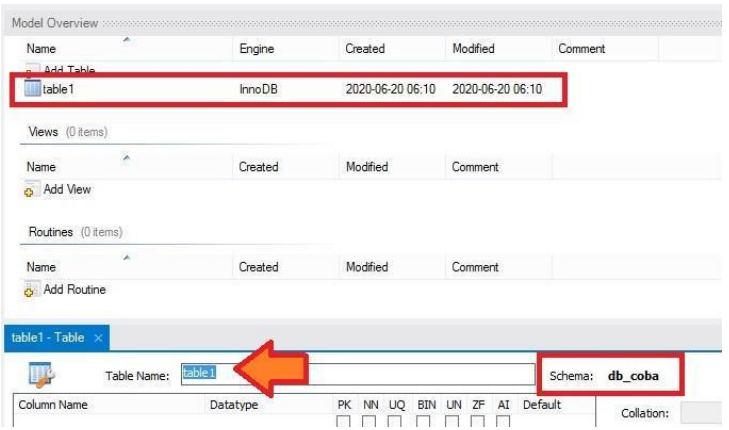
Membuat Tabel / Entitas / Entity pada MySQL *Workbench*

Membuat sebuah tabel pada MySQL *Workbench* terdapat 2 cara dengan langsung *key physical schemas* atau melalui EER. Menambahkan Tabel *key* Skema Fisik (*Physical Schemas*) caranya dengan memilih *physical*

schemas kemudian *add table* langsung memberi nama sesuai desain seperti pada gambar 53 dan 54.

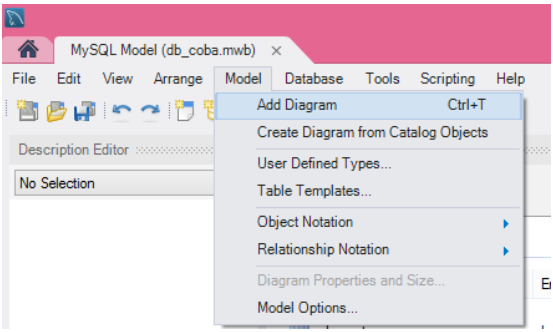


Gambar 53 Menambahkan tabel *key* skema fisik

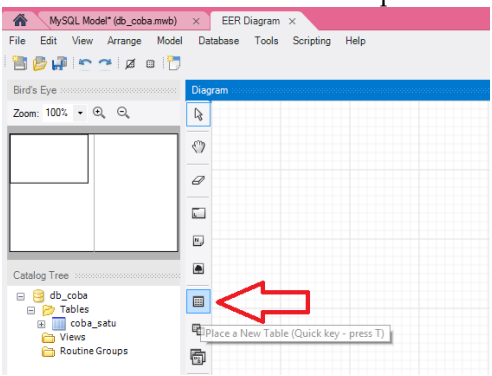


Gambar 54 Menambahkan tabel 1

Untuk menambahkan tabel melalui diagram bisa dengan memilih menu model kemudian klik add diagram seperti pada gambar 55. Kemudian tampilan akan berubah seperti pada gambar 56. Setelah itu klik place new tabel.

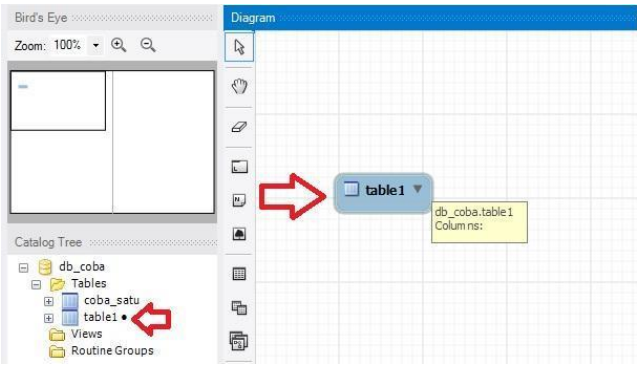


Gambar 55 Tombol menambahkan tabel pada EER Diagram

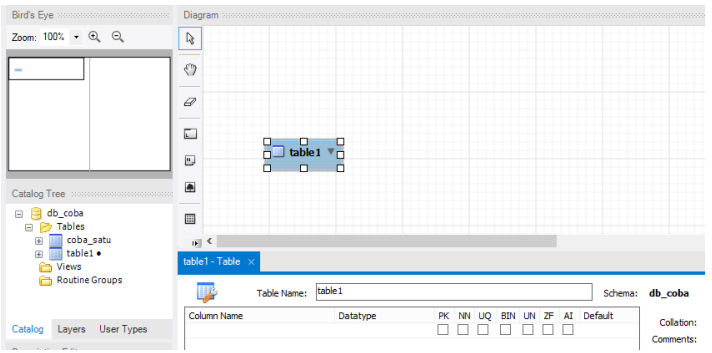


Gambar 56 Tombol untuk menambahkan EER Diagram baru

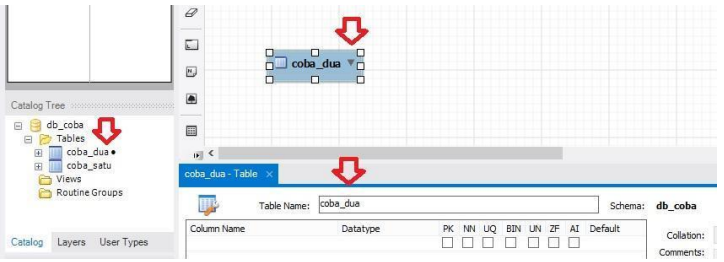
Kemudian akan muncil tabel baru seperti pada gambar 57, dan melakukan drag drop pada halaman tabel. Selanjutnya kita bisa klik dua kali dan muncul menu pada gambar 58. Sehingga kita bisa melakukan penambahan atribut serta pergantian nama tabel



Gambar 57 Hasil pada layar saat ditambahkan tabel



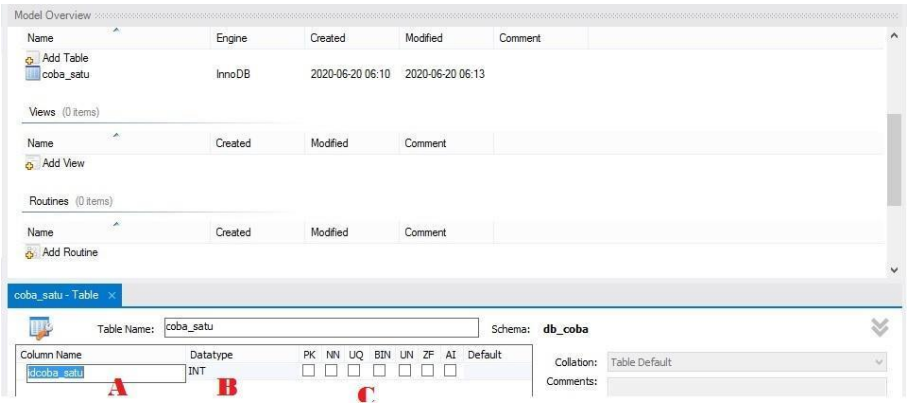
Gambar 58 Menambahkan atribut pada tabel 1



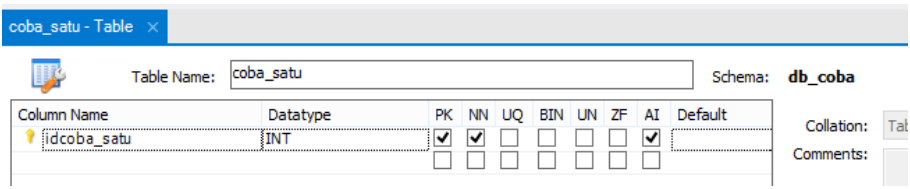
Gambar 59 Mengubah nama tabel 1 menjadi coba_dua

Membuat *Primary Key* pada MySQL *Workbench*

Untuk membuat *primary key* kitab isa membuat entitasnya terlebih dahulu. Kemudian mencentang kotak kecil yang ada pada kolom PK atau *primary key* seperti pada gambar 60 dan 61.



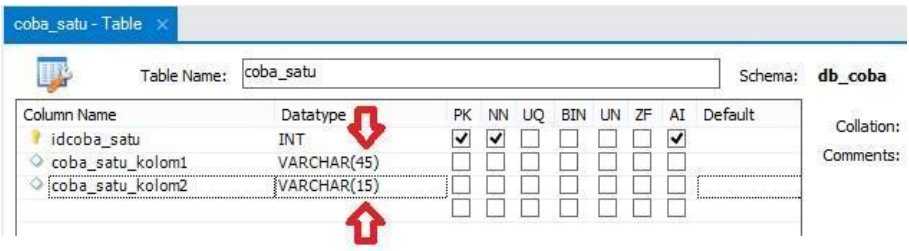
Gambar 60 Menambahkan *Primary Key*



Gambar 61 Contoh pada PK Tabel Coba_satu

Menambahkan Atribut / Kolom / *Field* pada MySQL *Workbench*

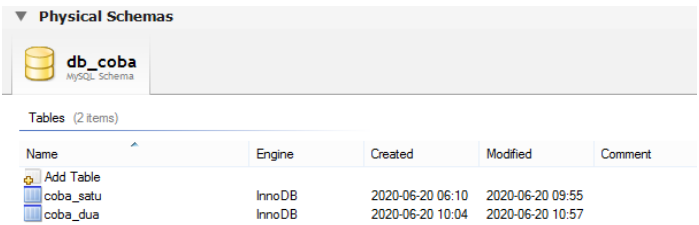
Untuk menambah atribut – atribut lainnya kita bisa melakukan klik baris dibawahnya dan mengisikan value selanjutnya seperti pada gambar62.



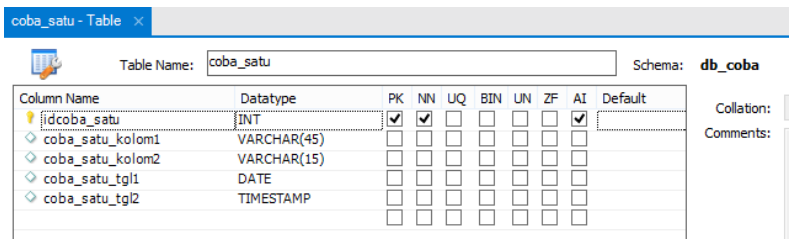
Gambar 62 Menambahkan atribut pada tabel/zentitas dengan tipe data varchar

Membuat *Foreign Key* pada MySQL *Workbench*

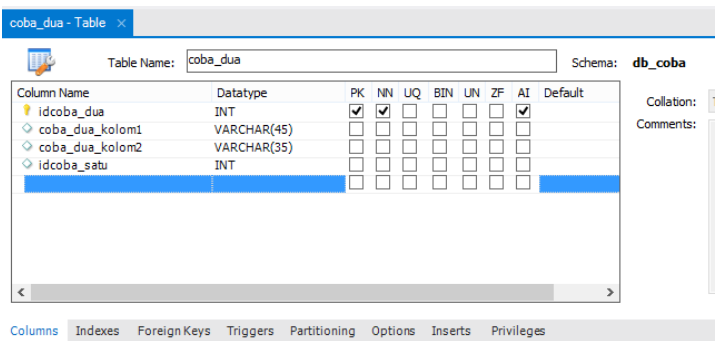
Untuk membuat *foreign key* kita menyiapkan 2 tabel yang akan dibuat sebuah relasi. Pada salah satu tabel kita membuat baris baru yang isinya diambil dari *primary* tabel lainnya. Kemudian kita bisa memilih *foreign key* serta menambahkan atribut yang akan dijadikan FK. Untuk Langkah-langkah secara detail bisa dilihat pada gambar 63 sampai 71.



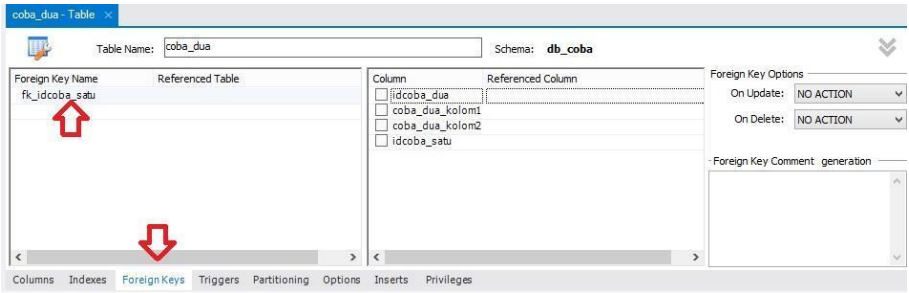
Gambar 63 *database* db_coba terdapat 2 entitas



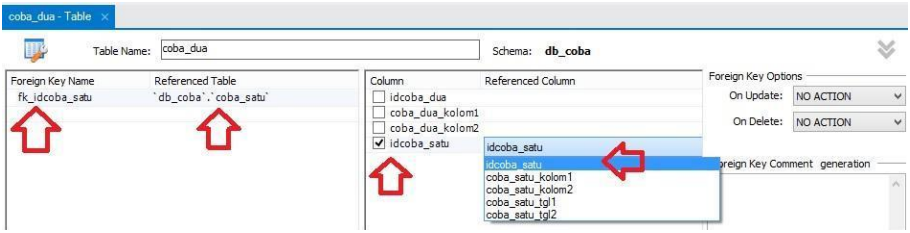
Gambar 64 Struktur tabel coba_satu



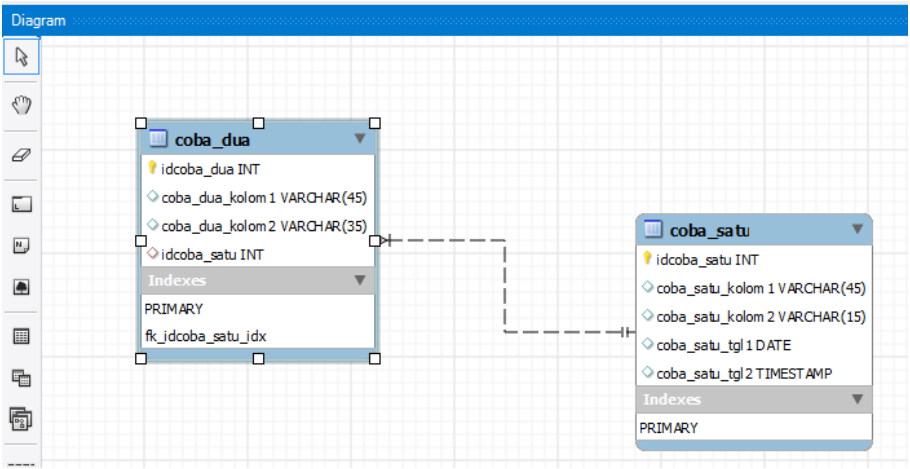
Gambar 65 Menambahkan atribut idcoba_satu untuk disetting sebagai FK



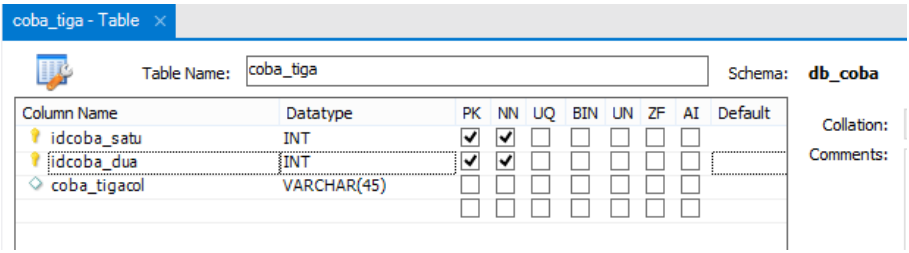
Gambar 66 Membuat *Foreign Key*



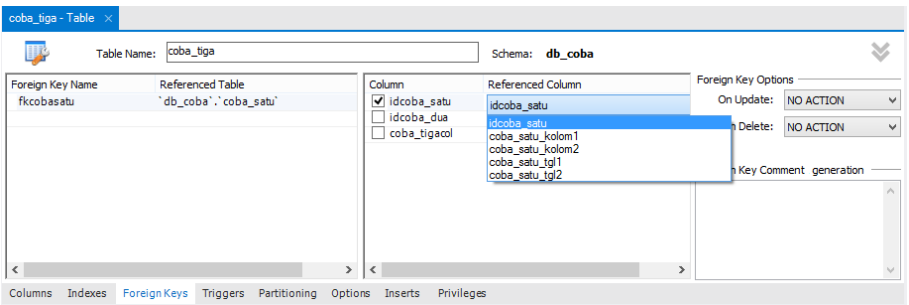
Gambar 67 Memilih reference table



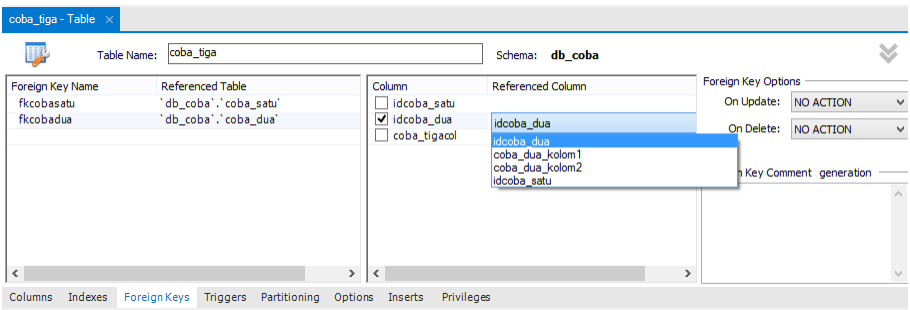
Gambar 68 Hasil pembuatan FK yang tampak pada EER Diagram



Gambar 69 Membuat PK sekaligus FK yang sering disebut Candidate *Key*



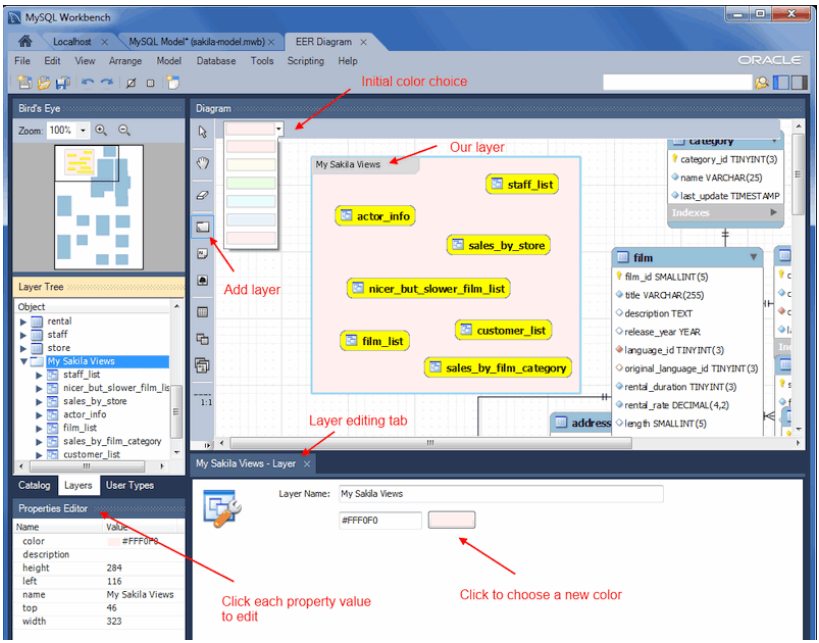
Gambar 70 Membuat reference table pada coba_satu



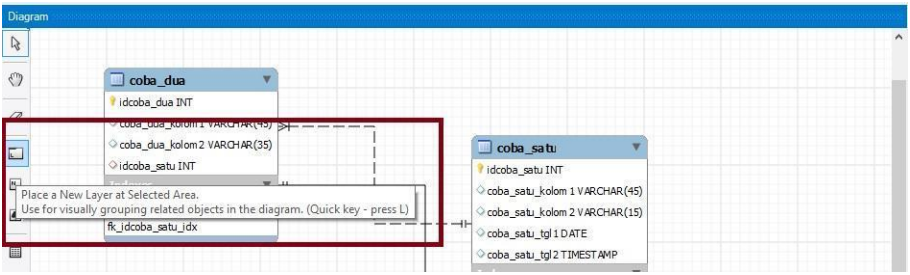
Gambar 71 Membuat reference table pada coba_dua

Layer

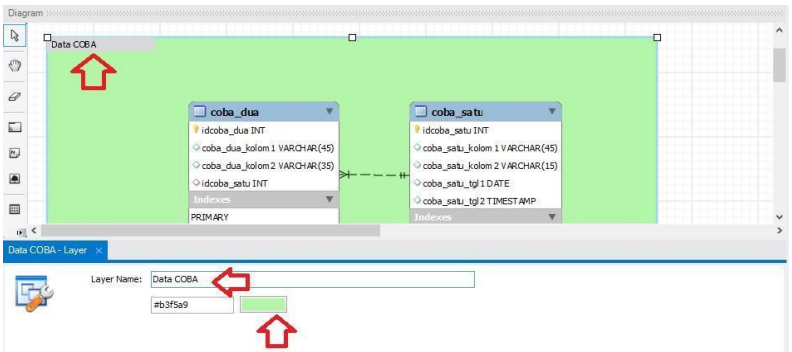
Untuk memperindah tampilan kita bisa menambahkan layer dibelakang tabel tabel. Tujuannya selain untuk memperindah juga dapat mengelompokan tabel sesuai dengan bagian bagiannya agar memudahkan dalam mencari databse pada kelompok tertentu. Langkah langkahnya bisa dilihat pada gambar 72 - 74.



Gambar 72 Tampilan dalam pembuatan layer



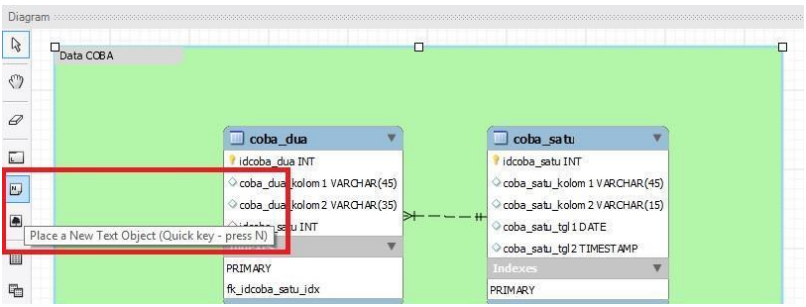
Gambar 73 Tombol Add Layer



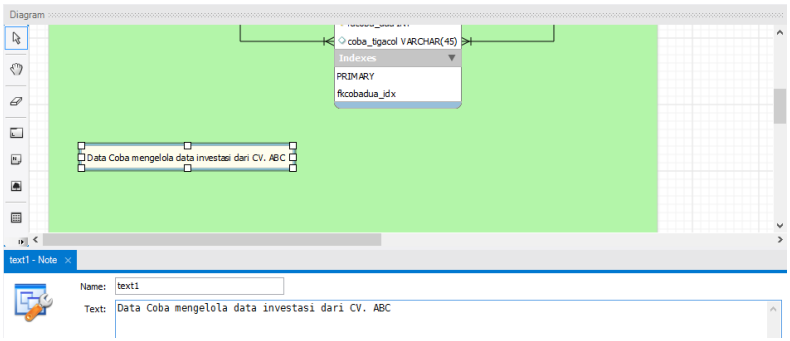
Gambar 74 Contoh menambahkan layer

Text Objects

Text objects untuk memberikan caption atau tulisan yang berisi catatan mengenai *database* tersebut. Bisa juga berisi mengenai bagian deskripsi *database* tersebut. Langkah langkahnya bisa dilihat pada gambar 75 dan 76.



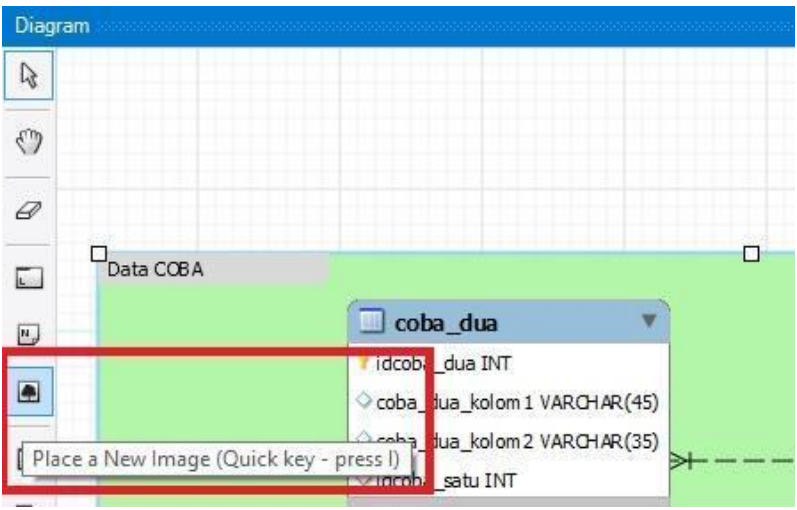
Gambar 75 Tampilan Add Text Objects



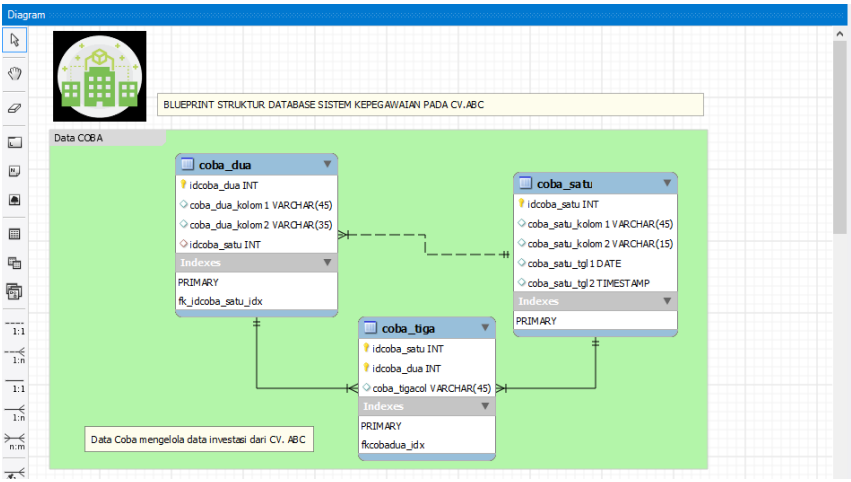
Gambar 76 Contoh menambahkan Text Object

Image

Selain itu kita juga bisa menambahkan sebuah gambar pada kelompok *database*. Langkah langkahnya bisa dilihat pada gambar 77 dan 78.



Gambar 77 Tampilan Add Image



Gambar 78 Contoh penambahan image

3.4. Studi Kasus *Database* Setiawan Spooring

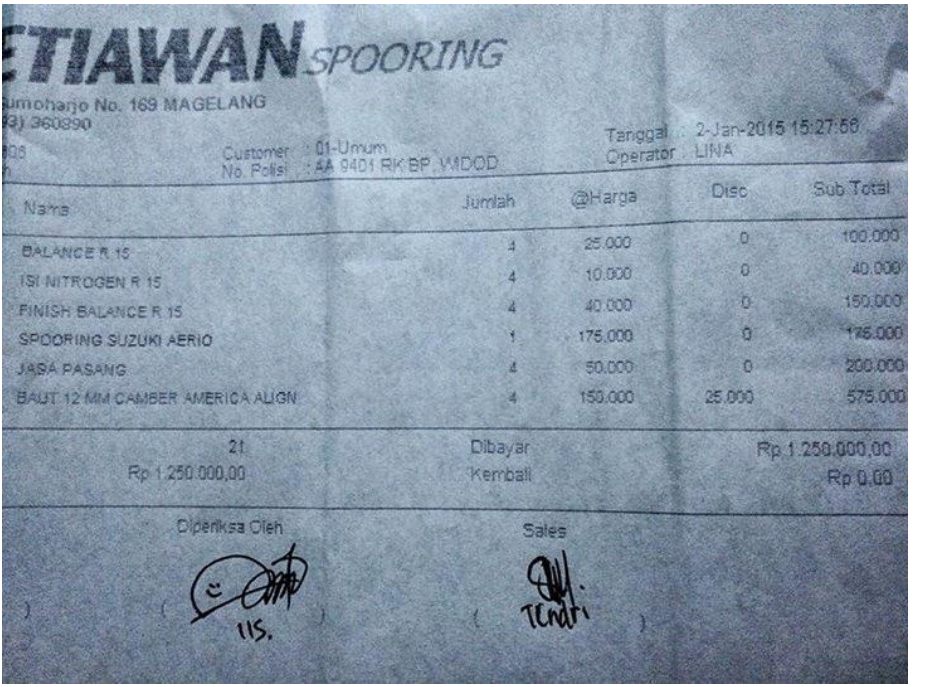
1. Tahap Persiapan

Sebelum mendesain secara grafis, hal-hal yang harus dipersiapkan adalah :

Menyiapkan Data

Dalam pembuatan *database* untuk *input* sebuah sistem, pastikan konsep yang dibuat memilih antara teknik berorientasi objek (ERD) atau berorientasi data (Normalisasi).

Pada contoh kali ini menggunakan teknik Normalisasi dengan data dari Toko "SETIAWAN SPOORING" yang didapatkan dari kwitansi/invoice hasil transaksi (pada gambar di bawah ini).



ETIAWAN SPOORING

umoharjo No. 169 MAGELANG
3) 360890

Customer : 01-Umum
No. Polisi : AA 9401 RK BP WIDOD

Tanggal : 2-Jan-2015 15:27:58
Operator : LINA

Nama	Jumlah	@Harga	Disc	Sub Total
BALANCE R 15	4	25.000	0	100.000
ISI NITROGEN R 15	4	10.000	0	40.000
FINISH BALANCE R 15	4	40.000	0	160.000
SPOORING SUZUKI AERIO	1	175.000	0	175.000
JASA PASANG	4	50.000	0	200.000
BAUT 12 MM CAMBER AMERICA ALIGN	4	150.000	25.000	575.000

21
Rp 1.250.000,00

Dibayar Rp 1.250.000,00
Kembali Rp 0,00

Diperiksa Oleh

Sales

Gambar 79 Nota/kwitansi/invoice "SETIAWAN SPOORING"

Menuliskan Entitas Hasil Normalisasi

Pada materi sebelumnya telah ditunjukkan simulasi dalam sebuah normalisasi data Toko "SETIAWAN SPOORING". Hasil dari Normalisasi 3NF terdapat 9 entitas keseluruhan dengan **Tabel Master** sejumlah 6 Entitas dan **Tabel Implementasi Relasi** sejumlah 3 Entitas. Deskripsi mengenai atribut masing-masing entitas beserta *data dummy* sebagai contoh dalam pemilihan tipe data setiap atribut. Dalam pengembangan desain *database* baik menggunakan MySQL *Workbench* ataupun program sejenis, **tabel master harus dibuat terlebih dahulu** dilanjutkan dengan tabel implementasi relasi. Hal ini **berhubungan dengan pembuatan *foreign key*** yang membutuhkan tabel referensi *key* tabel master dan tabel lainnya.

a) Tabel Master sejumlah 6 Entitas :

-Tabel Jenis Bayar

-Tabel Customer

-Tabel Jenis_Cust

-Tabel Barang

-Tabel Karyawan

-Tabel Penugasan

b) Tabel Implementasi Relasi sejumlah 3 Entitas :

-Tabel Tugas_sebagai

-Tabel Faktur

-Tabel Detail_Faktur

1. Tabel **Jenis_bayar**

Kode jns byr	Jenis_bayar
	Cash
	Credit Card
	Invoice

2. Tabel **Customer**

No Pol	Nama_Cust
AA 9401 RK	BP WIDODO

3. Tabel **Barang**

Kode Brg	Nama_Brg	Harga_Brg
TYT036	BALANCE R15	Rp25.000
TYT037	ISI NITROGEN 15	Rp10.000
TYT038	FINISH BALANCE R15	Rp40.000
TYT039	SPOORING SUZUKI AE	Rp175.000
TYT040	JASA PASANG	Rp50.000
TYT041	BAUT 12 mm CAMBER AMERICA	Rp150.000

4. Tabel **Jenis_Cust**

Kode jns Cust	Jenis_Cust
	01-Umum
	02-Member
	03-Kantor
	04-Mitra

5. Tabel **Karyawan**

Kode karyawan	Nama Karyawan
K_001	iis
K_002	Sisca
K_003	Puput
K_004	Andre
K_005	Bobi
K_006	Charlie

6. Tabel **Penugasan**

Kode tugas	Nama_tugas
TG_01	Op_Faktur
TG_02	Pemeriksa_Faktur
TG_03	Sales_faktur
TG_04	Manajer
TG_05	Supervisor
TG_06	Kepala_Cabang

Gambar 80 Sejumlah 6 Tabel Master hasil

7. Tabel **Tugas_sebagai**

Kode Tgs Sebag	Kode_karyawan**	Kode_tugas**
S_001	K_001	TG_01
S_002	K_002	TG_02
S_003	K_003	TG_03
S_004	K_004	TG_04
S_005	K_005	TG_05

8. Tabel **Faktur**

No_Faktur	Tgl_Faktur	Kode_jns_byr**	No_Pol**	Kode_jns_Cust**	
E418	02/01/2015				
Kd_Operator**	Kd_Pemeriksa**	Kd_Sales**	Total_Brg	Bayar_Faktur	Kembali_Faktur
			Rp1.250.000	Rp1.250.000	0

9. Tabel **Detail_faktur**

No_Faktur	Kode_Brg	Jml_Brg	Disc_Brg	Subtotal_Brg
E418	TYT036	4	0	Rp100.000
E418	TYT037	4	0	Rp40.000
E418	TYT038	4	0	Rp160.000
E418	TYT039	1	0	Rp175.000
E418	TYT040	4	0	Rp200.000
E418	TYT041	4	Rp25.000	Rp575.000

Gambar 81 Sejumlah 3 Tabel Implementasi Relasi Hasil Normalisasi

Database yang akan dibuat dengan ketentuan sebagai berikut :

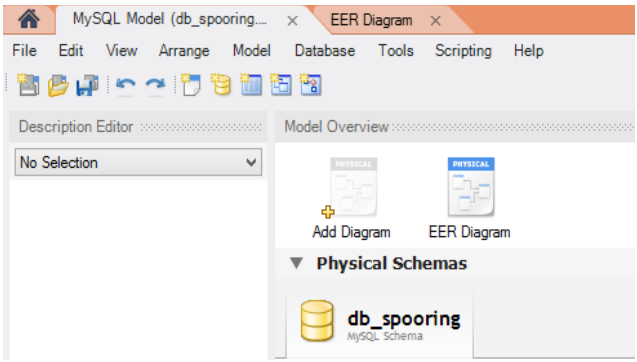
- Nama *database* **db_spooring**
- *Primary Key* (PK) menggunakan **tipe data INT** (*integer*) agar dapat diberikan kondisi AI (*Auto Increment*) **kecuali PK dari tabel customer dan tabel barang**. Pertimbangannya adalah karena tabel customer menggunakan PK dari nomor polisi kendaraan bermotor yang maksimal memiliki 12 karakter. Sedangkan tabel barang mempertimbangkan kategori produk yang memiliki beberapa *foreign* dalam sparepart kendarjaan.
- *Foreign key* (FK) yang dibuat berdasarkan tabel yang memiliki warna yang sama.

2. Membuat *Database* DBSpooring pada MySQL *Workbench*

Pada bagian ini kita akan mencoba membuat databse spooring menggunakan MySQL *Workbench* dari normalisasi yang telah

dilakukan pada tahap sebelumnya dengan berpedoman pada Langkah-langkah yang sudah dijelaskan pada SUBBAB 3.4

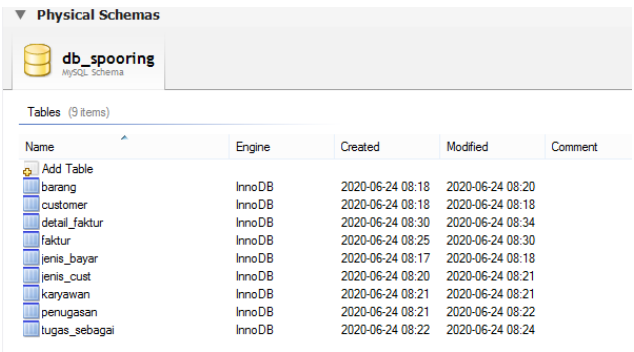
Pertama dengan membuat *database* dengan nama db_spooring sesuai gambar 82.



Gambar 82 Membuat db_spooring

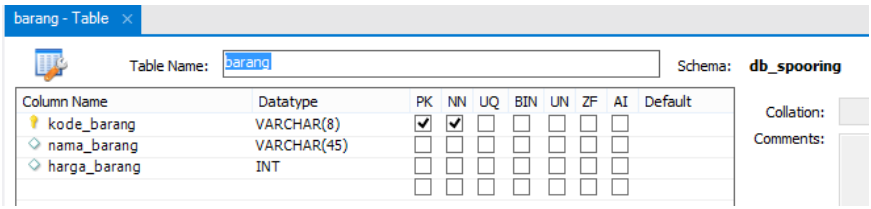
3. Membuat Entitas dan *Primary Key* DBSpooring pada MySQL *Workbench*

Selanjutnya kita bisa membuat tabel dari db spooring yang terdiri atas 9 entitas yaitu jenis bayar, customer, jenis_cust, barang, karyawan, penugasan, tugas_sebagai, faktur, detail_faktur sesuai gambar 83.

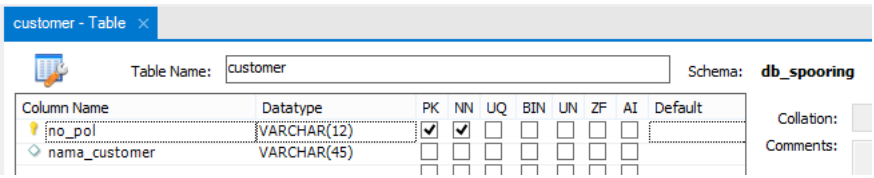


Gambar 83 Db_spooring memiliki 9 entitas

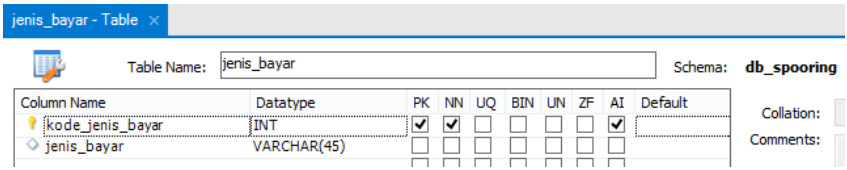
Untuk melihat rincian tabel *key*-9 entitas bisa pada gambar 84 sampai gambar 92.



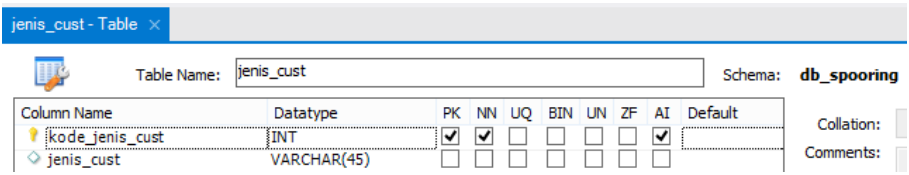
Gambar 84 Tabel barang



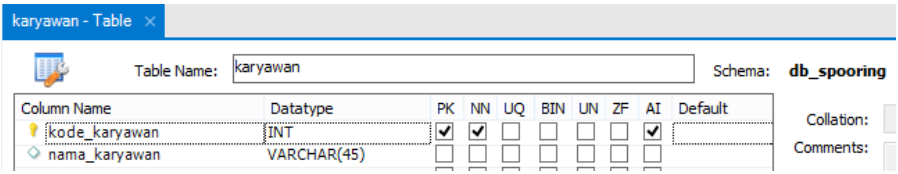
Gambar 85 Tabel customer



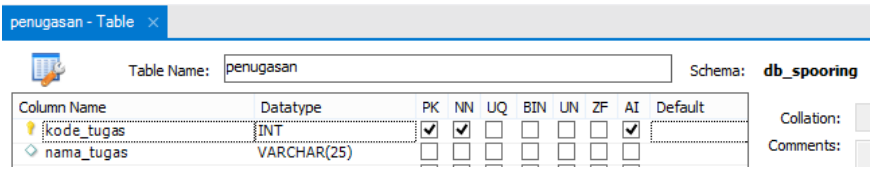
Gambar 86 Tabel Jenis_bayar



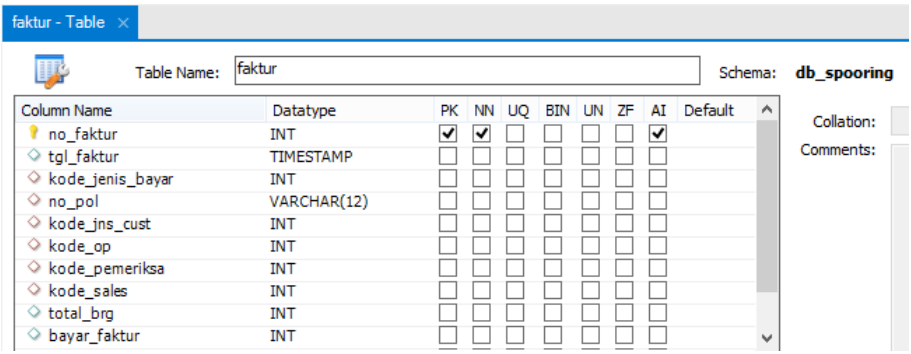
Gambar 87 Tabel Jenis_cust



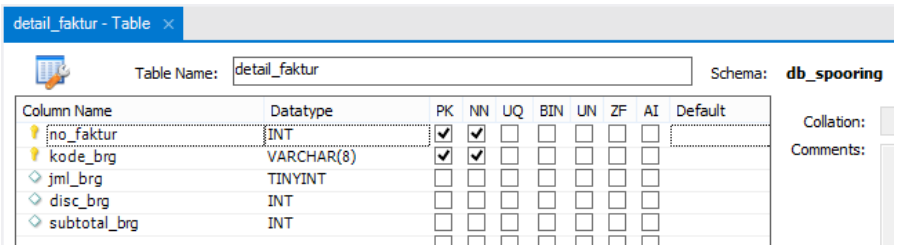
Gambar 88 Tabel karyawan



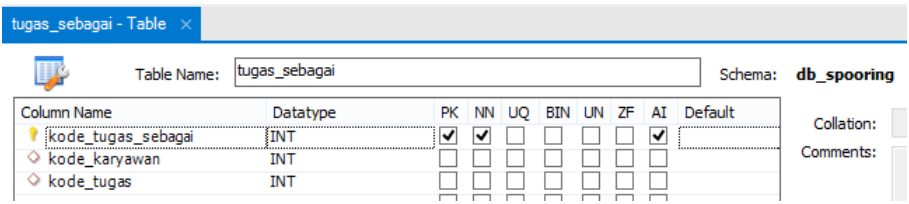
Gambar 89 Tabel penugasan



Gambar 90 Tabel faktur



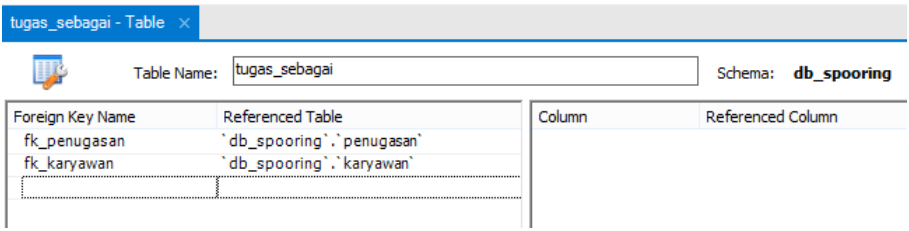
Gambar 91 Tabel detail_faktur



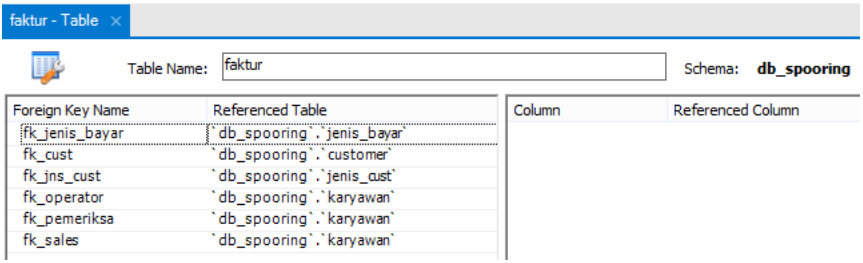
Gambar 92 Tabel tugas_sebagai

4. Membuat *Foreign Key* DBSpooring pada MySQL *Workbench*

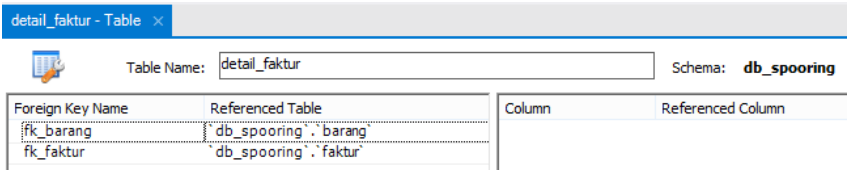
Selanjutnya dari *key* 9 entitas kita akan membuat sebuah *foreign key* dari tabel implementasi relasi yaitu hanya tabel tugas sebagai dengan risncial pada gamabr 93, tabel faktur dengan risncian pada gambar 94 dan tabel detail_faktur pada gambar 95.



Gambar 93 *Foreign Key* pada Tabel tugas_sebagai



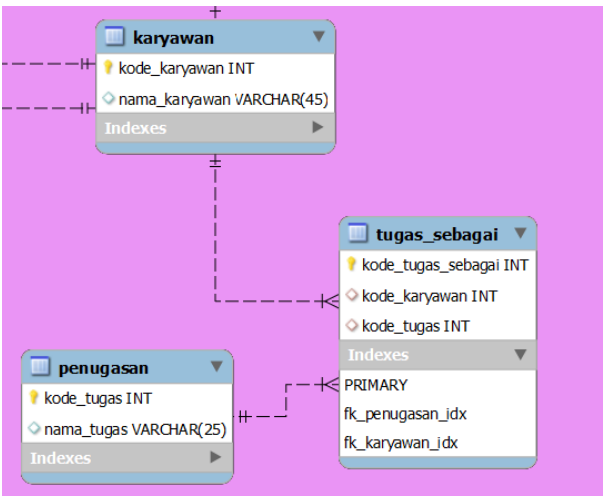
Gambar 94 *Foreign Key* pada Tabel faktur



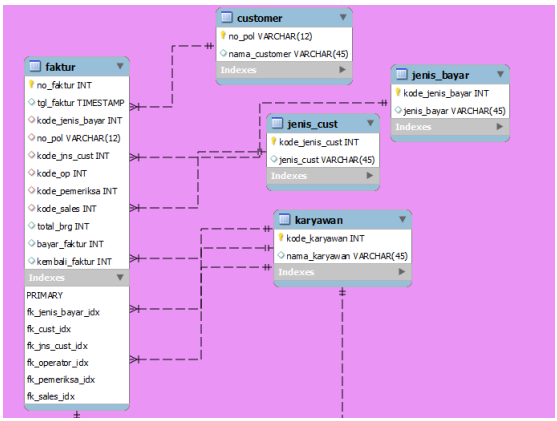
Gambar 95 *Foreign Key* pada Tabel detail_faktur

5. Membuat EER Diagram DBSpooring pada MySQL *Workbench*

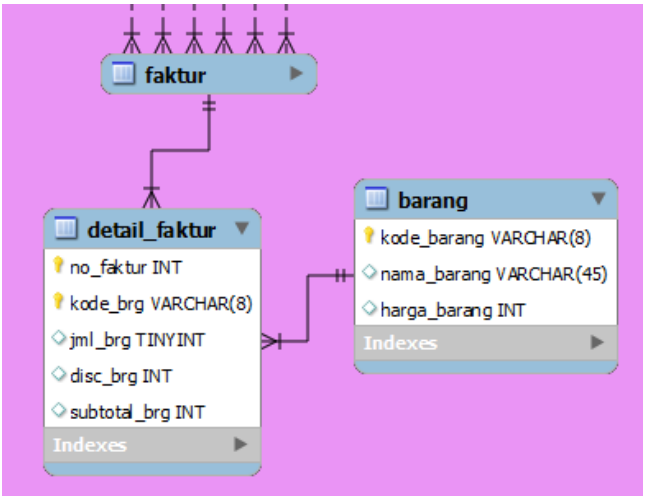
Setelah *foreign key* sudah dibuat maka pada MySQL *Workbench* akan otomatis terbentuk relasinya seperti pada gambar 96 yang memvisualisasikan relasi tabel tugas_sebagai, gambar 97 yang memvisualisasikan relasi tabel faktur dan gambar 98 memvisualisasikan relasi tabel detail faktur.



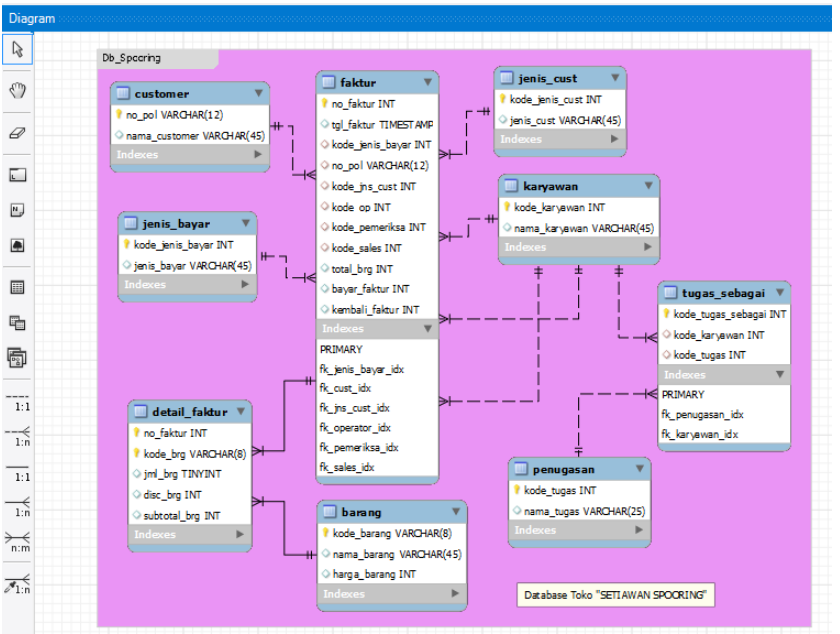
Gambar 96 Relasi pada tabel tugas_sebagai



Gambar 97 Relasi pada tabel Faktur



Gambar 98 Relasi pada tabel detail_faktur



Gambar 99 EER Diagram db_spooring

Gambar 99 merupakan gambaran relasi antar 9 entitas secara utuh pada *database* db_spooring.

BAB 4

STRUCTURED QUERY LANGUAGE (SQL)

4.1. Mengenal SQL

SQL adalah singkatan dari Structured Query Language. Sedangkan pengertian SQL adalah suatu bahasa (language) yang digunakan untuk mengakses data di dalam sebuah *database* relasional. SQL adalah sebuah bahasa permintaan *database* yang terstruktur. Bahasa SQL ini dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam *database* maupun merelasikan antar *database* .

SQL sering juga disebut dengan istilah **query**, dan bahasa SQL secara praktiknya digunakan sebagai bahasa standar untuk manajemen *database* relasional. Hingga saat ini hampir seluruh server *database* atau software *database* mengenal dan mengerti bahasa SQL (Setyawati et al., 2020).

Dalam penggunaan SQL terdapat beberapa perintah yang berguna untuk mengakses dan memanajemen data yang terdapat dalam *database*. Jenis peringah SQL secara umum dibagi kepada tiga sub perintah, yaitu DDL *(Data Definition Language),* DML *(Data Manipulation Language),* dan DCL *(Data Control Language).* Ketiga sub perintah tersebut sangat perlu untuk dipahami bagi anda yang ingin menguasai bahasa sql dan mahir dalam pembuatan *database*.

1. DDL *(Data Definition Language)*

DDL adalah sebuah metode Query SQL yang berguna untuk mendefinisikan data pada sebuah *Database*, Query yang dimiliki DDL adalah :

- CREATE : Digunakan untuk membuat *Database* dan Tabel
- DROP : Digunakan untuk menghapus Tabel dan *Database*
- ALTER : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, baik menambah *Field* (Add), mengganti nama *Field* (Change) ataupun menamakannya kembali (Rename), dan menghapus *Field* (Drop).

2. DML *(Data Manipulation Language)*

DML adalah sebuah metode Query yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari Query DML ini untuk melakukan pemanipulasian *database* yang telah dibuat. Query yang dimiliki DML adalah :

- INSERT : Digunakan untuk memasukkan data pada Tabel *Database*
- UPDATE : Digunakan untuk pengubahan terhadap data yang ada pada Tabel *Database*
- DELETE : Digunakan untuk Penhapusan data pada tabel *Database*

3. DCL *(Data Control Language)*

DCL adalah sub bahasa SQL yang berfungsi untuk melakukan pengontrolan data dan server *database*nya, seperti manipulasi user dan hak akses (privileges).

Yang termasuk perintah dalam DCL ada dua, yaitu GRANT dan REVOKE.

a) GRANT adalah perintah ini digunakan untuk memberikan hak akses oleh admin *key* salah satu user atau pengguna.
   Hak akses tersebut bisa berupa hak membuat (CREATE), mengambil data (SELECT), menghapus data (DELETE), mengubah data (UPDATE), dan hak khusus lainnya yang berhubungan dengan sistem *database*.

b) REVOKE adalah perintah yang digunakan untuk mencabut hak akses yang telah diberikan kepada user. Dalam ini merupakan kebalikan dari perintah GRANT.

4.2. MEMULAI DDL (Data Definition Language)

1. Query membuat *database* / tabel baru

Query : **CREATE *DATABASE*** nama_*database*;

2. Query membuat tabel

Query : **CREATE TABLE** nama_tabel(

Field1 TipeData1,

Field2 TipeData2,

……

FieldN TipeDataN);

3. Query menghapus sebuah *database*

Query : **DROP** ***DATABASE*** nama_*database*

4. Query menghapus struktur tabel

Query : **DROP TABLE** nama_tabel**;**

5. Query merubah nama *database*

Buat *database* baru dan ganti nama semua tabel di *database* lama menjadi *database* baru:

Query :

CREATE ***database*** **new_db_name;**

RENAME TABLE db_name.table1 TO new_db_name,

db_name.table2 TO new_db_name;

DROP ***database*** **db_name;**

6. Query merubah struktur tabel

Perubahan struktur ini dapat berupa penambahan *field* baru, perubahan *field* yang sudah ada, maupun menghapus *field* yang sudah ada.

Query :

- **ALTER TABLE** nama_tabel **ADD** new_*field* tipedata
- **ALTER TABLE** nama_tabel **CHANGE** *field*_lama *field*_baru tipedata
- **ALTER TABLE** nama_tabel **DROP** nama_*field*
- **ALTER TABLE** nama_tabel **RENAME** nama_*field*

4.3. QUERY *PRIMARY KEY* DAN *FOREIGN KEY*

1. Query membuat *Primary Key*

Query : **CREATE TABLE** nama_tabel(Nama_*field* tipe_*field* *PRIMARY KEY* NOT NULL);

2. Query membuat *Foreign Key*

Query : **CREATE TABLE** nama_tabel(Nama_*field* tipe_*field* *FOREIGN KEY* REFERENCES NOT NULL);

3. Query merubah *Primary Key* atau *Foreign Key*

Query :

- **ALTER TABLE** nama_tabel **ADD *PRIMARY KEY*(**nama_*field*)**;**
- **ALTER TABLE** nama_tabel **ADD *FOREIGN KEY*(**nama_*field*)**REFERENCES** nama_tabel1(nama_field1)**;**

4. Query menghapus *Primary Key* atau *Foreign Key*

Query :

- **ALTER TABLE** nama_tabel **DROP *PRIMARY KEY***(nama_*field*)**;**

ALTER TABLE nama_tabel **DROP *FOREIGN KEY*(**nama_*field*)**REFERENCES** nama_tabel1(nama_field1)**;**

4.4. INDEX

Indeks merupakan suatu objek yang berfungsi untuk mempercepat proses **pengambilan** data, pengurutan dan pencarian data dari suatu tabel. Data pada tabel yang sudah diindeks akan diurutkan berdasarkan kolom indeks. dengan demikian proses

pencarian data dapat lebih cepat. Saat data baru sedikit mungkin pengindekan belum terlalu berguna, namun jika data sudah mencapai ribuan atau bahkan ratusan ribu barulah terasa perbedaanya jika *field* tersebut belum diindeks.

4.5. CONSTRAINT/BATASAN

Constraint adalah batasan atau aturan yang ada pada tabel. MySQL menyediakan beberapa tipe constraint

1. NOT NULL

Suatu kolom yang didefinisikan dengan constraint NOT NULL tidak boleh berisi nilai NULL. Kolom yang befungsi sebagai kunci *primary* (*primary key*) otomatis tidak boleh NULL.

2. UNIQUE

Mendefinisikan suatu kolom menjadi bersifat unik, artinya antara satu data dengan data lainnya namanya tidak boleh sama, misal alamat email.

3. CONSTRAINT *PRIMARY KEY*

Constraint *PRIMARY KEY* membentuk *key* yang unik untuk suatu table.

4. CONSTRAINT *FOREIGN KEY*

FOREIGN KEY constraint didefinisikan pada suatu kolom yang ada pada suatu table, dimana kolom tersebut juga dimiliki oleh table yang lain sebagai suatu *PRIMARY KEY*.

- Untuk Menghapus constraint perintah yang dapat digunakan yaitu “DROP ”

Query : ALTER TABLE table DROP CONSTRAINT type (column);

- Untuk Mengaktifkan constraint perintah yang dapat digunakan yaitu “ENABLE”

  Query : ALTER TABLE table ENABLE CONSTRAINT type (column);

- Untuk Mematikan constraint perintah yang dapat digunakan yaitu “DISABLE ”

  Query : ALTER TABLE table DISABLE CONSTRAINT type (column);

4.6. REFERENTIAL INTERGRITY CONSTRAINT

Refential *Integrity Constraint* adalah aturan untuk relasi antar tabel untuk menjamin validasi hubungan antar record di dalam tabel – tabel yang terkait.

1. Aturan untuk Update

Cascade : pembaharuan sebuah baris data diikuti oleh pembaharuan baris data pada tabel anak yang terelasi

Restrict : mencegah pembaharuan data jika terdapat baris data di tabel anak yang terhubungkan

Ignore : mengabaikan referensi. Boleh memperbarui data pada tabel parent, tetapi tidak memperbarui data pada tabel child

No Action : Tidak ada aksi apapun

2. Aturan untuk Delete

Cascade : menghapus seluruh baris data pada tabel child yang terhubung

Restrict : mencegah penghapusan jika terdapat baris data yang terhubung tabel child

Ignore : boleh menghapus data, tapi tidak akan berpengaruh pada tabel anaknya

No Action : Tidak ada aksi apapun

3. Aturan untuk Insert

Restrict : tidak boleh menambah data pada tabel child jika nilai yang dimasukkan pada kolom yang berelasi tidak terdapat pada parent tabelnya

Ignore : boleh menambah data pada tabel child meskipun nilai yang dimasukkan pada kolom yang berelasi tidak terdapat pada tabel parentnya

No Action : Tidak ada aksi apapun

4.7. TIPE DATA

Dalam bahasa SQL pada umumnya informasi tersimpan dalam tabel-tabel yang secara logik merupakan struktur dua dimensi terdiri dari baris (row atau *record*) dan kolom (*column* atau *field*). Sedangkan dalam sebuah *database* dapat terdiri dari beberapa tabel. Macam-macam Tipe Data pada MySQL, secara umum tipe-tipe data MySQL ini ada empat (4), diantaranya yaitu:

1. Tipe Data Numeric

Tipe data numerik yaitu tipe data yang digunakan untuk menyimpan data numerik (angka).

Tabel 5. Tabel Data Numeric

No	Nama	Fungsi	Jangkauan	Ukuran
1	TINYINT	Menyimpan data bilangan bulat positif dan negatif.	-128 s/d 127	1 byte (8 bit).
2	SMALLINT	menyimpan data bilangan bulat positif dan negatif.	: -32.768 s/d 32.767	: 2 byte (16 bit).
3	MEDIUMINT	menyimpan data bilangan bulat positif dan negatif.	-8.388.608 s/d 8.388.607	Ukuran : 3 byte (24 bit).
4	INT	menyimpan data bilangan bulat positif dan negative	-2.147.483.648 s/d 2.147.483.647	4 byte (32 bit).

5	BIGINT	menyimpan data bilangan bulat positif dan negatif.	± 9,22 x 1018	8 byte (64 bit).
6	*FIELD*	menyimpan data bilangan pecahan positif dan negatif presisi tunggal	-3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38.	4 byte (32 bit)
7	DOUBLE	menyimpan data bilangan pecahan positif dan negatif presisi ganda.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	8 byte (64 bit)
8	REAL	menyimpan data bilangan pecahan positif dan negatif presisi ganda.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	8 byte (64 bit).
9	DECIMAL	menyimpan data bilangan pecahan positif dan negatif.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	8 byte (64 bit).

10	NUMERIC	menyimpan data bilangan pecahan positif dan negatif.	-1.79...E+308 s/d -2.22...E-308, 0, dan 2.22...E-308 s/d 1.79...E+308.	8 byte (64 bit).

2. Tipe Data String

Tipe data string yaitu tipe data yang digunakan untuk menyimpan data string (text).

Tabel 6. Tabel Data String

No	Nama	Fungsi	Jangkauan
1	CHAR	menyimpan data string ukuran tetap.	0 s/d 255 karakter
2	VARCHAR	menyimpan data string ukuran dinamis.	0 s/d 255 karakter (versi 4.1), 0 s/d 65.535
3	TINYTEXT	menyimpan data text.	0 s/d 255 karakter (versi 4.1), 0 s/d 65.535
4	TEXT	menyimpan data text.	0 s/d 65.535
5	MEDIUMTEXT	menyimpan data text	0 s/d 224 - 1 karakter
6	LONGTEXT	menyimpan data text.	0 s/d 232 - 1 karakter

3. Tipe Data Date

Tipe data date dan time yaitu tipe data yang digunakan untuk menyimpan data tanggal dan waktu.

Tabel 7. Tabel Data Date

No	Nama	Fungsi	Jangkauan	Ukuran
1	DATE	menyimpan data tanggal	1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)	3 byte.
2	TIME	menyimpan data waktu	-838:59:59 s/d +838:59:59 (HH:MM:SS)	3 byte
3	DATETIME	menyimpan data tanggal dan waktu.	'1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'	8 byte
4	YEAR	menyimpan data tahun dari tanggal	1900 s/d 2155	1 byte

4. Tipe Data BLOB

Tipe data blob digunakan untuk menyimpan data biner.

Tabel 8. Tabel Data Blob

No	Nama	Fungsi	Jangkauan
1	BIT	Menyimpan data biner.	64 digit biner
2	TINYBLOB	menyimpan data biner/ Gambar ukuran kecil	255 byte
3	BLOB	Menyimpan data biner seperti files, gambar, suara, dll	4

4	MEDIUMBLOB	Menyimpan data biner/ Gambar kuran sedang	224-1 byte
5	LONGBLOB	Menyimpan data biner/ Gambar ukuran besar	232- 1 byte

5. Tipe Data yang lain

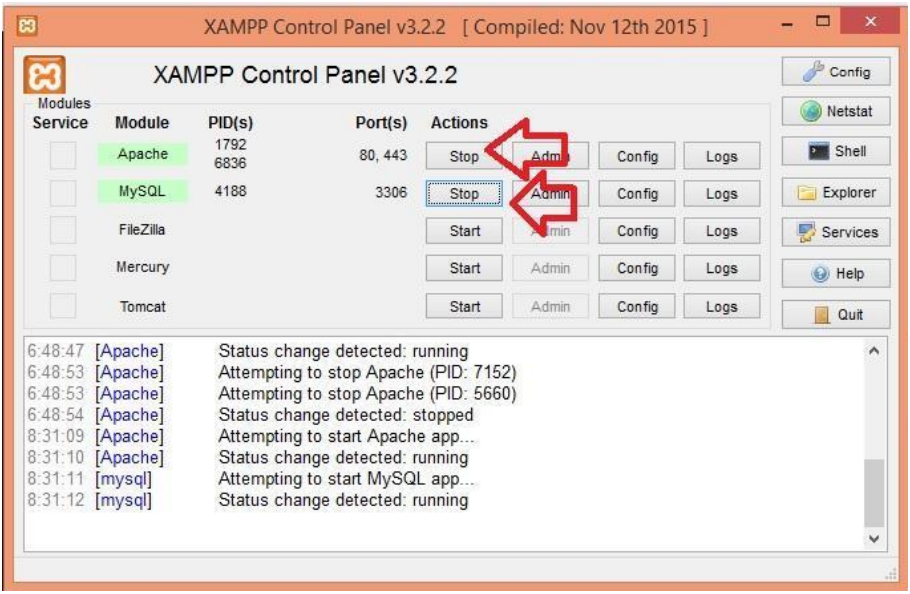
Selain tipe data di atas, MySQL juga menyediakan tipe data yang lain, diantaranya adalah :

Tabel 9. Tabel Data Lainnya

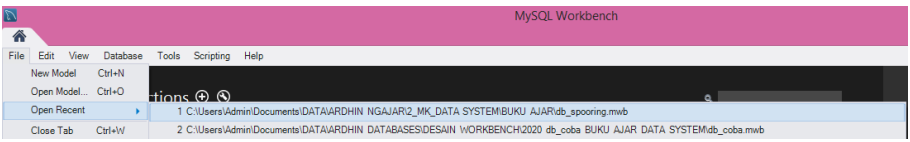
No	Nama	Fungsi	Jangkauan
1	ENUM	enumerasi (kumpulan data).	sampai dengan 65535 string.
2	SET	combination (himpunan data).	sampai dengan 255 string anggota

4.8. SCRIPT DDL *DATABASE* SETIAWAN SPOORING

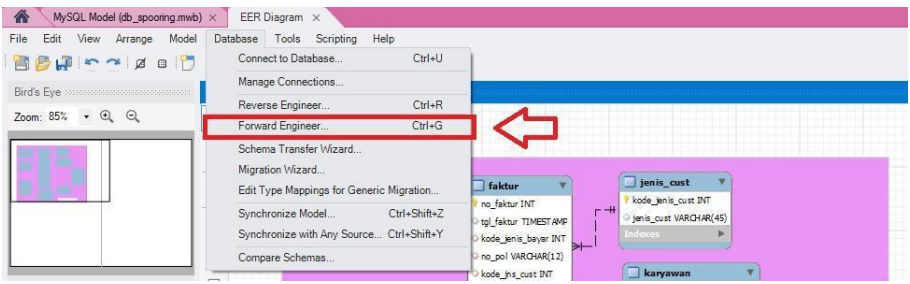
Setelah DBSpooring sudah terbentuk, mulai dari pembuatan tabel, *primary key*, forgenkey dan EERnya maka kita dapat mengubah menjadi Script DDL dengan menggunakan MySQL *Workbench*. Langkah Langkah script DDL membuat *database* DBSpooring dapat diperhatikan dari gambar 100 sampai 109.



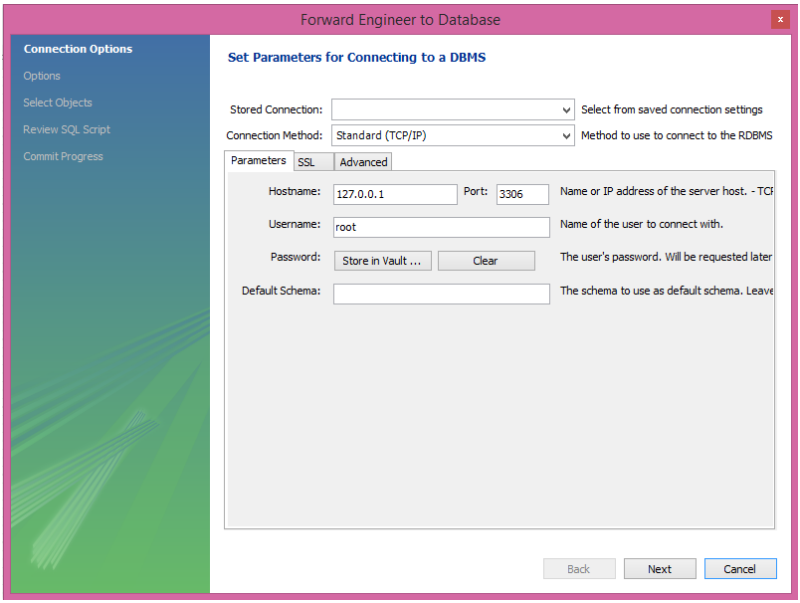
Gambar 100 Mengaktifkan engine Apache dan MySQL



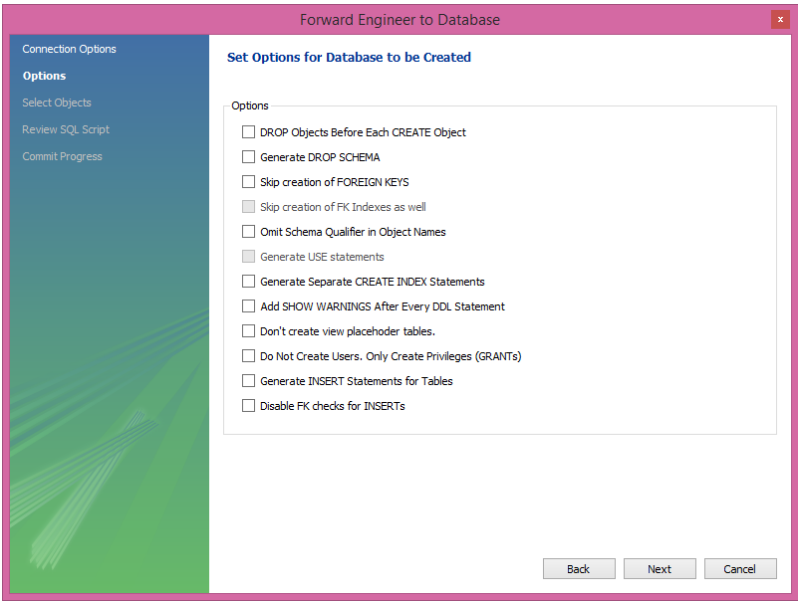
Gambar 101 Membuka desain *database* db_spooring



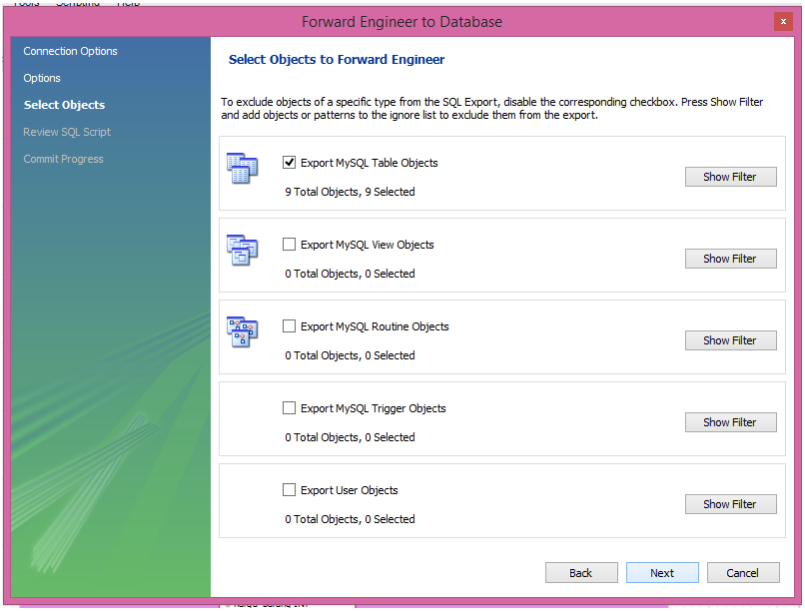
Gambar 102 Memilih Menu *Database*>Forward Engineer



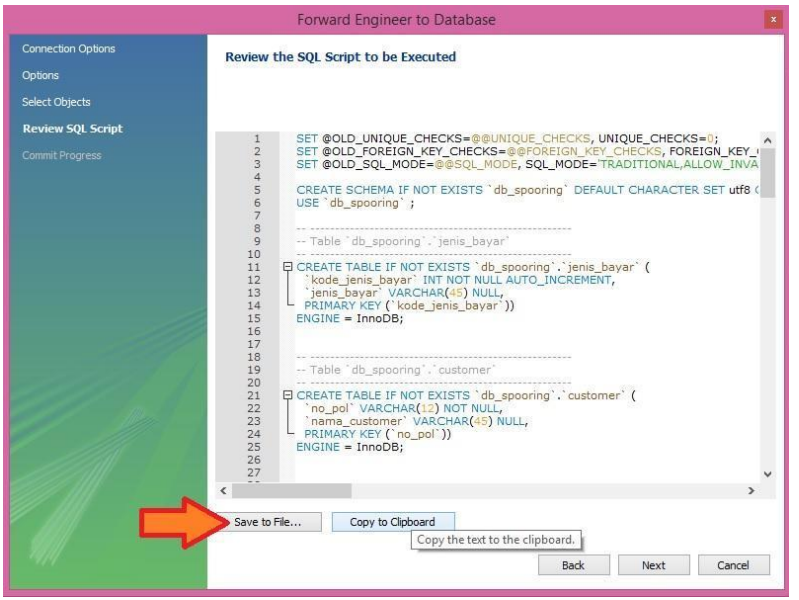
Gambar 103 Tampilan Menu Forward Engineer dan pilih Next



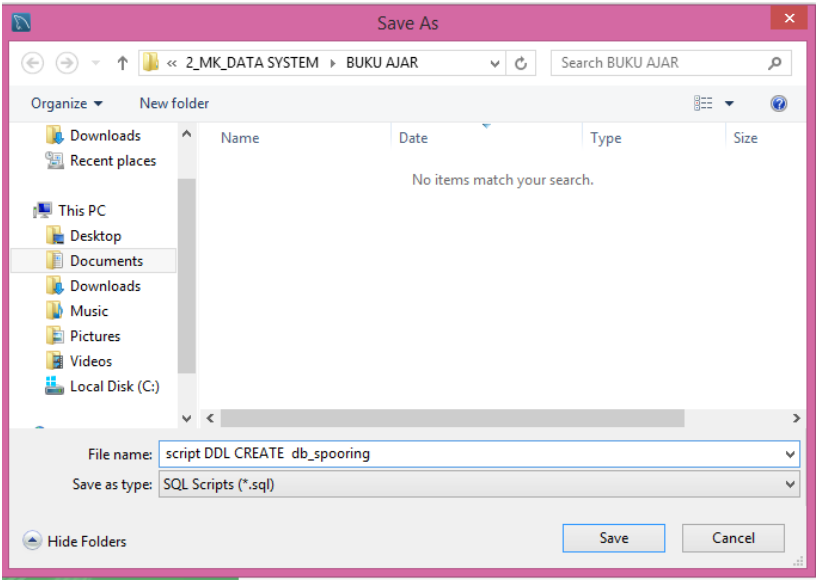
Gambar 104 Tampilan Menu Forward Engineer dan lanjutkan dengan pilih Next



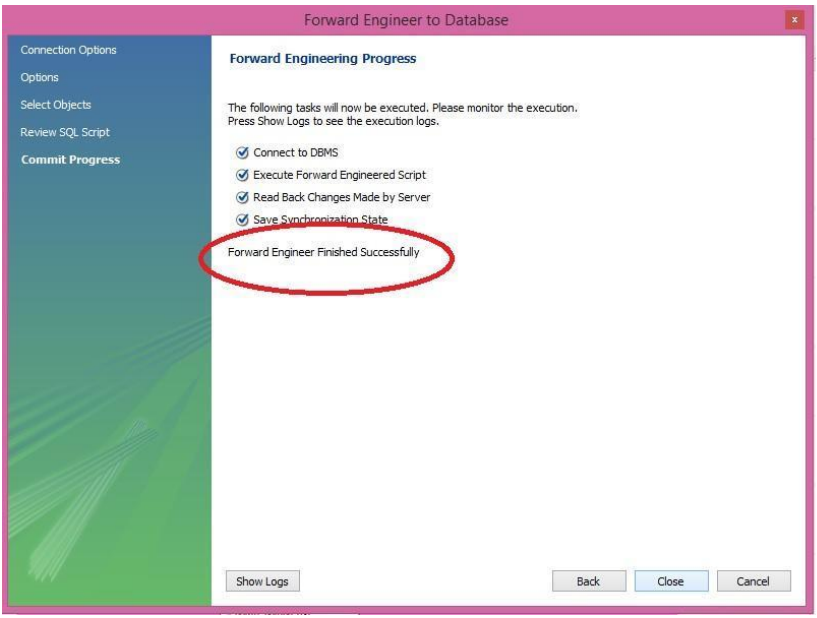
Gambar 105 Pastikan jumlah tabel sesuai dan lanjutkan dengan Next



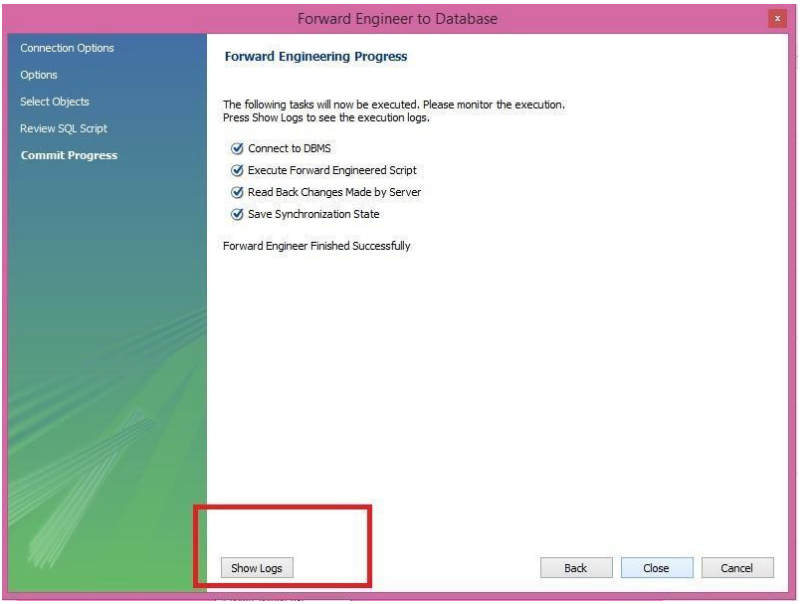
Gambar 106 Pilih save file



Gambar 107 Beri nama script DDL CREATE sesuai nama db masing-masing

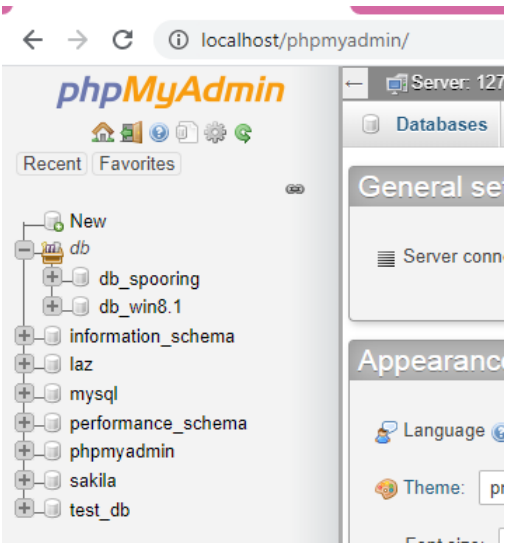


Gambar 108 Pastikan Forward Enginer sukses berjalan tanpa eror

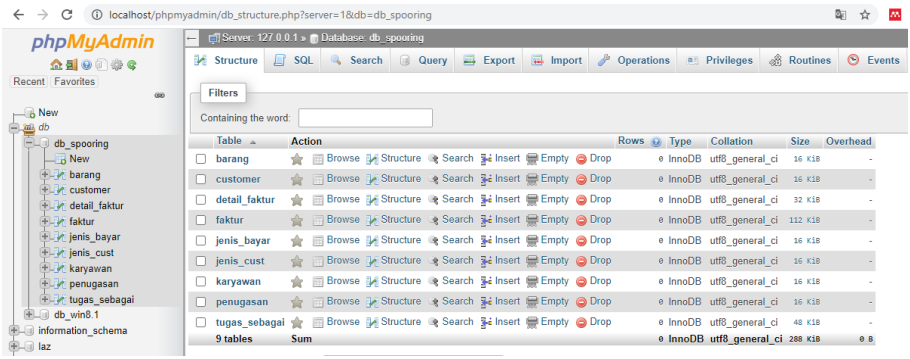


Gambar 109 Jika ada eror pastikan melihat pada button Show Log

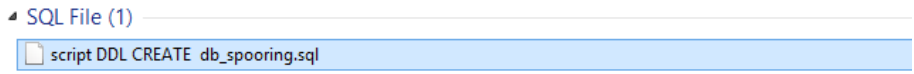
Gambar 110 Aktifkan browser dan masuk localhost



Gambar 111 Pilih *database* db_spooring



Gambar 112 Entitas berjumlah 9 pada db_spooring sudah terimplementasi



Gambar 113 Masuk explore dan pilih script DDL berupa SQL file dibuka menggunakan text editor

Di bawah ini merupakan syntax dalam membuat tabel, atribut beserta *primary key* dan *foreign key* hasil menggunakan forward engineer yang tersimpan dalam sebuah **script DDL CREATE db_spooring.sql**. Script ini dapat diubah jika ada penyesuaian struktur tabel yang dibangun sehingga memudahkan dalam melakukan proses restore saat *database* ada masalah.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
```

```
CREATE SCHEMA IF NOT EXISTS `db_spooring` DEFAULT CHARACTER
SET utf8 COLLATE utf8_general_ci ;

USE `db_spooring` ;


-- -----------------------------------------------------

-- Table `db_spooring`.`jenis_bayar`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `db_spooring`.`jenis_bayar` (

  `kode_jenis_bayar` INT NOT NULL AUTO_INCREMENT,

  `jenis_bayar` VARCHAR(45) NULL,

  PRIMARY KEY (`kode_jenis_bayar`))

ENGINE = InnoDB;


-- -----------------------------------------------------

-- Table `db_spooring`.`customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `db_spooring`.`customer` (

  `no_pol` VARCHAR(12) NOT NULL,

  `nama_customer` VARCHAR(45) NULL,

  PRIMARY KEY (`no_pol`))

ENGINE = InnoDB;


-- -----------------------------------------------------

-- Table `db_spooring`.`barang`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `db_spooring`.`barang` (

  `kode_barang` VARCHAR(8) NOT NULL,
```

```
  `nama_barang` VARCHAR(45) NULL,
  `harga_barang` INT NULL,
  PRIMARY KEY (`kode_barang`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `db_spooring`.`jenis_cust`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_spooring`.`jenis_cust` (
  `kode_jenis_cust` INT NOT NULL AUTO_INCREMENT,
  `jenis_cust` VARCHAR(45) NULL,
  PRIMARY KEY (`kode_jenis_cust`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `db_spooring`.`karyawan`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_spooring`.`karyawan` (
  `kode_karyawan` INT NOT NULL AUTO_INCREMENT,
  `nama_karyawan` VARCHAR(45) NULL,
  PRIMARY KEY (`kode_karyawan`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `db_spooring`.`penugasan`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `db_spooring`.`penugasan` (
  `kode_tugas` INT NOT NULL AUTO_INCREMENT,
  `nama_tugas` VARCHAR(25) NULL,
  PRIMARY KEY (`kode_tugas`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `db_spooring`.`tugas_sebagai`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_spooring`.`tugas_sebagai` (
  `kode_tugas_sebagai` INT NOT NULL AUTO_INCREMENT,
  `kode_karyawan` INT NULL,
  `kode_tugas` INT NULL,
  PRIMARY KEY (`kode_tugas_sebagai`),
  INDEX `fk_penugasan_idx` (`kode_tugas` ASC),
  INDEX `fk_karyawan_idx` (`kode_karyawan` ASC),
  CONSTRAINT `fk_penugasan`
    FOREIGN KEY (`kode_tugas`)
    REFERENCES `db_spooring`.`penugasan` (`kode_tugas`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_karyawan`
    FOREIGN KEY (`kode_karyawan`)
    REFERENCES `db_spooring`.`karyawan` (`kode_karyawan`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `db_spooring`.`faktur`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_spooring`.`faktur` (
  `no_faktur` INT NOT NULL AUTO_INCREMENT,
  `tgl_faktur` TIMESTAMP NULL,
  `kode_jenis_bayar` INT NULL,
  `no_pol` VARCHAR(12) NULL,
  `kode_jns_cust` INT NULL,
  `kode_op` INT NULL,
  `kode_pemeriksa` INT NULL,
  `kode_sales` INT NULL,
  `total_brg` INT NULL,
  `bayar_faktur` INT NULL,
  `kembali_faktur` INT NULL,
  PRIMARY KEY (`no_faktur`),
  INDEX `fk_jenis_bayar_idx` (`kode_jenis_bayar` ASC),
  INDEX `fk_cust_idx` (`no_pol` ASC),
  INDEX `fk_jns_cust_idx` (`kode_jns_cust` ASC),
  INDEX `fk_operator_idx` (`kode_op` ASC),
  INDEX `fk_pemeriksa_idx` (`kode_pemeriksa` ASC),
  INDEX `fk_sales_idx` (`kode_sales` ASC),
  CONSTRAINT `fk_jenis_bayar`
    FOREIGN KEY (`kode_jenis_bayar`)
    REFERENCES `db_spooring`.`jenis_bayar` (`kode_jenis_bayar`)
    ON DELETE NO ACTION
```

```
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_cust`
  FOREIGN KEY (`no_pol`)
  REFERENCES `db_spooring`.`customer` (`no_pol`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_jns_cust`
  FOREIGN KEY (`kode_jns_cust`)
  REFERENCES `db_spooring`.`jenis_cust` (`kode_jenis_cust`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_operator`
  FOREIGN KEY (`kode_op`)
  REFERENCES `db_spooring`.`karyawan` (`kode_karyawan`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_pemeriksa`
  FOREIGN KEY (`kode_pemeriksa`)
  REFERENCES `db_spooring`.`karyawan` (`kode_karyawan`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_sales`
  FOREIGN KEY (`kode_sales`)
  REFERENCES `db_spooring`.`karyawan` (`kode_karyawan`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `db_spooring`.`detail_faktur`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `db_spooring`.`detail_faktur` (
  `no_faktur` INT NOT NULL,
  `kode_brg` VARCHAR(8) NOT NULL,
  `jml_brg` TINYINT NULL,
  `disc_brg` INT NULL,
  `subtotal_brg` INT NULL,
  PRIMARY KEY (`no_faktur`, `kode_brg`),
  INDEX `fk_barang_idx` (`kode_brg` ASC),
  CONSTRAINT `fk_barang`
    FOREIGN KEY (`kode_brg`)
    REFERENCES `db_spooring`.`barang` (`kode_barang`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_faktur`
    FOREIGN KEY (`no_faktur`)
    REFERENCES `db_spooring`.`faktur` (`no_faktur`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Daftar Pustaka

Nugroho, S. and Primadewi, A. (2021) 'Penerapan Web Service untuk Integrasi Data Simperpus dan SIAK', *Jurnal Komtika (Komputasi dan Informatika)*, 4(2), pp. 71–81.

Andaru, A. (2018). Pengertian Database Secara Umum. *Fakultas Komputer* Section Class Content Istilah, 1–7. https://doi.org/10.1145/1147282.1147284

Julaeha, S., Kustian, N., & Parulian, D. (2020). Pemetaan Tabel Relationship dalam Visualisasi Diagram Relasi untuk Eksplorasi Data Pada Database. STRING (Satuan Tulisan Riset dan Inovasi Teknologi), 5(2), 126–133. https://doi.org/10.30998/string.v5i2.6653

Kroenke, D. M., & Auer, D. J. (2015). Database Concepts (7 ed.). https://doi.org/10.1007/978-3-662-65167-4_2

Letwoski, J. . (2015). Doing Database Design with MySQL. Journal of Technology Research, 6(February), 1–15.

Octafian, D. T. (2011). Desain Database Sistem Informasi Penjualan Barang (Studi Kasus : Minimarket "Grace" Palembang). Jurnal Teknologi Dan Informatika (Teknomatika), 1(2), 148–157.

Setyawati, E., Sarwani, Wijoyo, H., & Soeharmoko, N. (2020). Relational Database Management System (RDBMS). In Database Management System. https://doi.org/10.1201/9780429282843-3

Suryadi, S. (2019). Implementasi Normalisasi Dalam Perancangan Database Relational. Jurnal Teknik Informatika, 3(2), 1–5. https://doi.org/10.52332/u-net.v3i2.7

UNIMMA
PRESS
UNIVERSITAS MUHAMMADIYAH MAGELANG