1. MODUL I  
   PERANCANGAN DIAGRAM DAN NORMALISASI
   1. **TUJUAN**
   2. Mahasiswa memahami permasalahan dan dapat menentukan entitas-entitas dari permasalahan tersebut.
   3. Mahasiswa dapat menentukan atribut-atribut dari suatu entitas.
   4. Mahasiswa dapat menentukan relasi antara masing-masing entitas
   5. Mahasiswa memahami dan dapat merancang suatu permasalahan yang diberikan berdasarkan teori yang telah dipelajari.
   6. Mahasiswa dapat melakukan proses normalisasi.
   7. **LANDASAN TEORI**
      * 1. **Elemen-elemen Diagram ER**

*Entity* *Relationship* *Diagram* (ERD) merupakan model data berupa notasi grafis dalam pemodelan data konseptual yang menggambarkan hubungan antara penyimpan. Adapun elemen-elemen diagram ER antara lain:

**Entitas**

Entitas adalah sesuatu atau objek di dunia nyata yang dapat dibedakan dari sesuatu atau objek lain [3]. Entitas menunjukkan objek-objek dasar yang terkait di dalam sistem yang dapat berupa objek fisik seperti rumah dan manusia ataupun objek konsep seperti pekerjaan dan perusahaan. Entitas biasanya diberi nama dengan kata benda. Terdapat dua jeNIS entitas, yaitu:

1. **Entitas Kuat**

Adalah entitas yang berdiri sendiri, keberadaannya tidak tergantung dengan entitas lain:

|  |
| --- |
|  |

**Gambar 1.1** Notasi entitas kuat.

1. **Entitas Lemah**

Entitas yang keberadaannya bergantung pada entitas lain, artinya entitas tersebut tidak berguna jika tidak ada entitas yang lain. Berikut adalah notasinya:

|  |
| --- |
|  |

**Gambar 1.2** Notasi entitas lemah.

**Atribut**

Atribut merupakan elemen dari sebuah entitas dan berfungsi mendeskripsikan karakter entitas. Atribut juga dikatakan sebagai properti deskriptif yang dimiliki oleh setiap anggota dari himpunan entitas [2]. Atribut berfungsi sebagai entitas penjelas. Simbol yang menyatakan atribut adalah simbol *elips*, dimana nama dari atribut dituliskan di dalam simbol. Nama dari atribut biasanya berupa kata tunggal.

Terdapat beberapa atribut yang digunakan untuk menggambarkan model ER diagram, yaitu:

* 1. **Atribut kunci**

Atribut yang digunakan untuk menentukan suatu entitas secara unik [2]. Contohnya id\_barang.

|  |
| --- |
|  |

**Gambar 1.3** Notasi atribut kunci.

* 1. **Atribut *simple***

Atribut yang bernilai tunggal. Contohnya yaitu nama, harga dan status [2].

|  |
| --- |
|  |

**Gambar 1.4** Notasi atribut *simple.*

* 1. **Atribut *multivalue***

Suatu atribut yang memiliki sekelompok nilai yang banyak [2]. Contohnya gelar dan hobi.

|  |
| --- |
|  |

**Gambar 1.5** Notasi atribut *multivalue.*

* 1. **Atribut *composite***

Atribut yang masih dapat diuraikan lagi [2]. Contohnya alamat dan waktu.

|  |
| --- |
|  |

**Gambar 1.6** Notasi atribut *composite.*

* 1. **Atribut derivatif**

Atribut yang dihasilkan dari turunan atribut yang lain [2]. Contohnya umur dari tanggal lahir.

|  |
| --- |
|  |

**Gambar 1.7** Simbol atribut derivatif.

**Relasi**

Relasi adalah hubungan antara satu himpunan entitas dengan himpunan entitas yang lain [2].Pada diagram ER, relasi digambarkan dengan sebuah bentuk belah ketupat. Pada umumnya, relasi diberi nama dengan kata kerja sehingga memudahkan untuk melakukan pembacaan relasi. Berikut adalah notasi dari relasi:

|  |
| --- |
|  |

**Gambar 1.8** Simbol relasi.

Kerelasian antar entitas dapat dikelompokkan dalam 3 jeNIS yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas lain yang biasa disebut dengan kardinalitas [3]. Tiga macam kardinalitas tersebut ialah:

1. **Relasi satu ke satu (*One to One*)**

Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B, dan begitu juga sebaliknya setiap entitas B pada himpunan entitas B berhubungan paling banyak satu entitas pada himpunan entitas A [2]. Simbol relasi satu ke satu yaitu:

|  |
| --- |
|  |

**Gambar 1.9** Notasi relasi satu ke satu.

1. **Relasi Satu ke Banyak (*One to Many*)**

Setiap entitas pada himpunan entitas A berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak satu entitas pada himpunan entitas A [2]. Simbol relasi satu ke banyak yaitu:

|  |
| --- |
|  |

**Gambar 1.10** Notasi relasi satu ke banyak.

1. **Relasi Banyak ke Satu (*Many to One*)**

Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan banyak entitas pada himpunan entitas A [2]. Simbol relasi banyak ke satu yaitu:

|  |
| --- |
|  |

**Gambar 1.11** Notasi relasi banyak ke satu.

1. **Relasi banyak ke banyak (*Many to Many*)**

Setiap entitas pada himpunan entitas A berhubungan dengan banyak entitas pada himpunan entitas B, dan begitu juga sebaliknya [2]. Simbol relasi banyak ke banyak yaitu:

|  |
| --- |
|  |

**Gambar 1.12** Notasi relasi banyak ke banyak.

* + - 1. **Proses Perancangan Diagram ER**

Langkah-langkah yang dapat dilakukan untuk menghasilkan diagram ER adalah sebagai berikut [2]:

1. Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlibat.
2. Menentukan atribut-atribut *key* dari masing-masing himpunan entitas.
3. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas-himpunan entitas yang ada beserta *foreign-key*-nya.
4. Menentukan derajat/kardinalitas relasi untuk setiap himpunan relasi.
5. Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif (*non key*).
   * + 1. **Normalisasi**

Normalisasi merupakan sebuah teknik dalam desain logika sebuah basis data yang mengelompokkan atribut dari suatu tabel/relasi sehingga membentuk struktur relasi yang baik (tanpa redundansi) [1].

Tujuan dilakukannya normalisasi adalah sebagai berikut:

1. Minimalisasi redundansi (pengulangan data).
2. Memudahkan identifikasi entitas.
3. Mempermudah pemodifikasian data.
4. Mencegah terjadinya anomali.

Tahapan normalisasi dimulai dari tahap paling ringan (1NF) hingga paling ketat (5NF). Urutan tahapnya yaitu 1NF, 2NF, 3NF, BCNF, 4NF, 5NF [1]. Biasanya hanya sampai pada tingkat 3NF, suatu tabel sudah dapat menghasilkan tabel yang berkualitas baik.

Berikut tahapan normalisasi dalam perancangan basis data:

* + - 1. **Bentuk Normal Pertama (1NF)**

Suatu tabel disebut memenuhi bentuk normal pertama jika:

1. Tidak adanya atribut *multivalue*, atribut *composite* atau kombinasinya.
2. Setiap atribut dalam tabel tersebut harus bernilai *atomic* (tidak dapat dibagi-bagi lagi).
3. Tidak ada perulangan grup atribut dalam satu baris.
   * + 1. **Bentuk Normal Kedua (2NF)**

Suatu tabel disebut memenuhi bentuk normal kedua jika:

1. Sudah memenuhi dalam bentuk normal pertama (1NF).
2. Semua atribut bukan kunci utama bergantung secara fungsional terhadap semua atribut kunci dan bukan hanya sebagian atribut kunci.
3. Jika ada ketergantungan parsial maka atribut tersebut harus dipisah pada tabel yang lain, dimana perlu ada tabel penghubung ataupun kehadiran *foreign-key* bagi atribut-atribut yang telah terpisah.
   * + 1. **Bentuk Normal Ketiga (3NF)**

Suatu tabel disebut memenuhi bentuk normal ketiga jika:

1. Sudah memenuhi bentuk normal kedua (2NF).
2. Setiap atribut yang bukan kunci tidak tergantung secara fungsional terhadap atribut bukan kunci yang lain dalam relasi tersebut (tidak ada ketergantungan transitif pada atribut bukan kunci).
   * + 1. **Keuntungan normalisasi dari permasalahan yang ada di dunia nyata**

Dari permasalahan yang ada di dunia nyata terdapat keuntungan normalisasi di antaranya sebagai berikut:

1. Meminimalkan ukuran penyimpanan yang diperlukan untuk penyimpanan data.
2. Meminimalkan resiko kandungan inkonsistensi data pada basis data.
3. Meminimalkan kemungkinan anomali pembaruan.
4. Memaksimalkan stabilitas struktur data.
5. Kinerja penelusuran yang lebih cepat.
6. Meningkatkan integritas data.
   1. **KASUS KELOMPOK**

Sekolah Dasar Internasional Wakanda (SBIW) akan merancang sebuah sistem informasi. Basis data yang dibentuk berdasarkan data-data berbagai entitas. Guru akan didata nama, kontak dan alamat sehingga terbentuk NUPTK. Lalu guru dibedaan menjadi guru honorer dan PNS yang memiliki NIP. Semua guru dapat mengajar mata pelajaran, dan guru pun menguasai tidak hanya satu saja mata pelajaran, namun terkadang mereka harus mengajar mata pelajaran yang tidak mereka pelajari, sehingga mata pelajaran dapat dikuasai dan diajar oleh banyak guru. Mata pelajaran hanya didata namanya saja, sehingga dibentuk kode mata pelajaran. Setiap mata pelajaran yang diajar guru akan didata pegisian kelas mana yang diajar. Sehingga guru dapat mengajar mata pelajaran yang sama di kelas yang berbeda, begitupun dikelas yang sama dapat mengajar mata pelajaran lain. Kelaspun harus di data nama kelasnya sehinggaa terbentuk kode kelas. Sebuah kelas dapat diwalikan oleh seorang guru non-honorer. Sebuah kelaspun memiliki banyak siswa, dimana dari mereka terdapat satu yang menjadi ketua kelas. Setiap siswa akan didata nama, alamay, nama wali, kontak wali, dan rangking mereka hingga dibentuk NIS. Setiap siswa dapat mempelajari banyak mata pelajaran, begitu pula sebaliknya setiap mata pelajaran dapat dipelajari banyak siswa. Setiap kegiatan belajar mengajar akan dilakukan pengambilan nilai.

* 1. **KASUS MODUL**
     + - 1. Melakukan perancangan ERD serta menentukan kamus data yang dimiliki diagram tersebut.
         2. Melakukan normalisasi 1NF, 2NF, dan 3NF terhadap ERD yang telah dirancang
  2. **ANALISIS KASUS**
     1. **Perancangan ED Diagram serta Kamus Data**

1. **Rancangan ERD Sebelum Normalisasi**

|  |
| --- |
|  |

**Gambar 1.13** ERDSD Int­ernasional Wakanda.

1. **Kamus data**

Guru : {NUPTK, nama\_guru, kontak\_gutu, alamat\_guru}

PNS : {NUPTK, NIP}

Mapel : {kode\_mapel, nama\_mapel}

Kelas : {kode\_kelas, nama\_kelas}

Siswa : {NIS, nama\_siswa, rangking, alamat\_siswa, nama\_wali, kontak\_wali}

* + 1. **Tabel *universal***

**Tabel 1.1** Tabel *universal*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NUPTK | nama\_guru | NIP | kontak\_guru | alamat\_guru | kode\_jadwal | ... |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ... | kode\_mapel | nama\_mapel | nilai | kode\_kelas | nama\_kelas | ... |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ... | NIS | nama\_siswa | rank | alamat | nama\_wali | kontak\_wali |
|  |  |  |  |  |  |  |

**tabel 1.1** merupakan tabelyang merangkum semua kelompok data yang saling berhubungan. Tabel ini diperoleh dari rancangan ERD yang belum mengalami proses normalisasi yang dimana pada tabel ini dapat diperoleh semua atribut yang berasal dari semua entitas di ERD termasuk relasinya.

* + 1. **Normalisasi 1NF, 2NF, dan 3NF terhadap ERD yang telah dirancang.**

Normalisasi (normalize) merupakan salah satu cara pendekatan atau teknik yang digunakan dalam membangun desain logic database relation dengan menerapakan sejumlah aturan dan kriteria standard. Tujuan dari normalisasi adalah untuk menghasilakan struktur tabel yang normal atau baik.

* + - * 1. **Tabel 1NF**

Bentuk normalisasi yang pertama adalah 1NF. Di dalam tabel bentuk normal pertama ini tidak diperbolehkan adanya atribut yang tidak bernilai tunggal, yaitu atribut komposit dan *multivalue.* Tabel 1NF yang dibuat harus bernilai tunggal (atomik).Di bawah ini adalah tabel 1NF Sistem Informasi “Sistem Informasi SDI Wakanda”:

**Tabel 1.2** Tabel *universal*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUPTK | nama\_guru | NIP | kontak\_guru | alamat\_guru | ... |
|  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ... | kode\_mapel | nama\_mapel | nilai | kode\_kelas | nama\_kelas | ... |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ... | NIS | nama\_siswa | rangking | alamat | nama\_wali | kontak\_wali |
|  |  |  |  |  |  |  |

Pada **Tabel 1.2**, sudah dapat dikatakan berbentuk 1NF dikarenakan pada tabel tersebut tidak ada kolom yang memiliki atribut bernilai banyak (*multivalued attribute*) atau lebih dari satu atribut dengan domain nilai yang sama.

* + - * 1. **Tabel 2NF**

Tahapan kedua normalisasi adalah 2NF. Syarat untuk tabel 2NF adalah memenuhi 1NF dan tidak ada ketergantungan parsial, maksudnya adalah ketergantungan atribut bukan kunci pada atribut kunci secara sebagian, bukan *fully functionally dependent.*

Karena ada dua jenis guru didalam ERD, yakni guru honorer dan PNS maka tabel guru dipisah menjadi dua bagian, yakni:

**Tabel 1.3** Tabel guru 2NF

|  |  |  |  |
| --- | --- | --- | --- |
| NUPTK | nama\_guru | kontak\_guru | alamat\_guru |
|  |  |  |  |

**Tabel 1.3** Tabel guru PNS 2NF

|  |  |
| --- | --- |
| NUPTK | NIP |
|  |  |

Tabel guru didekomposisi dari tabel keseluruhan karena atribut ‘NIP’ tidak bergantung pada ‘NUPTK’. Jadi, tabel guru dibuat menjadi suatu tabel tersendiri yang berisi atribut kunci ‘NUPTK’ tadi, dan atribut bukan kunci yang dipengaruhinya, yaitu ‘nama\_guru’, ‘kontak\_guru’, ‘alamat\_guru’ dan ‘NIP’. Atribut NIP merupakan pembeda antara guru biasa dengan guru PNS. Dengan dipisahkan sepeti tabel diatas, tidak akan da lagi nilai kosong di kolom ‘NIP’.

**Tabel 1.4** Tabel mapel 2NF

|  |  |
| --- | --- |
| kode\_mapel | nama\_mapel |
|  |  |

Tabel mapel didekomposisi dari tabel keseluruhan karena ada atribut bergantung pada ‘kode\_mapel’. Jadi, tabel tersebut menjadi suatu tabel tersendiri yang berisi atribut kunci ‘kode\_mapel’ tadi, dan atribut bukan kunci yang dipengaruhinya yakni ‘nama\_mapel’.

**Tabel 1.5** Tabel kelas 2NF

|  |  |  |  |
| --- | --- | --- | --- |
| kode\_kelas | nama\_kelas | NIM\_ketua | NUPTK\_walikelas |
|  |  |  |  |

Tabel kelas didekomposisi dari tabel keseluruhan karena ada atribut bergantung pada ‘kode\_kelas’. Jadi, tabel tersebut menjadi suatu tabel tersendiri yang berisi atribut kunci ‘kode\_kelas’ tadi, dan atribut bukan kunci yang dipengaruhinya yakni ‘nama\_kelas’.

**Tabel 1.6** Tabel siswa 2NF

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| NIS | nama\_siswa | rank | Alamat\_siswa | nama\_wali | kontak\_wali | kode\_kelas |
|  |  |  |  |  |  |  |

Tabel Siswa didekomposisi dari tabel keseluruhan karena beberapa atribut bergantung pada ‘NIS’. Jadi, tabel dosen dibuat menjadi suatu tabel tersendiri yang berisi atribut kunci ‘NIS’ tadi, dan atribut bukan kunci yang dipengaruhinya, yaitu ‘nama\_siswa’, ‘rank’, ‘alamat\_siswa’, ‘nama\_wali’ dan ‘kontak\_wali’.

* + - * 1. **Tabel 3NF**

Pada tahapan ketiga yaitu 3NF, tabel sudah dapat dikatakan tabel yang baik. Oleh karena itu, ini adalah tahapan normalisasi yang terakhir. Syarat bagi suatu tabel untuk memenuhi 3NF adalah memenuhi 2NF dan tidak ada ketergantungan transitif, yang berarti ketergantungan atribut bukan kunci pada atribut bukan kunci lainnya. Dari tabel-tabel 2NF sistem informasi “Sistem Informasi SDI Wakanada”, yang perlu perbaiki adalah tabel relasi.

**Tabel 1.8** tabel mengajar

|  |  |
| --- | --- |
| NUPTK | Kode\_mapel |
|  |  |

Tabel di atas merupakan implementasi relasi yang menghubungkan antara entitas guru dan entitas mata pelajaran yang diajar oleh guru yang bersangkutan.

**Tabel 1.8** Tabel menguasai

|  |  |
| --- | --- |
| NUPTK | kode\_mapel |
|  |  |

Tabel di atas merupakan implementasi relasi yang menghubungkan antara entitas guru dan entitas mata pelajaran yang dikuasainya.

**Tabel 1.9** tabel menjadwalkan

|  |  |  |
| --- | --- | --- |
| kode\_jadwal | kode\_kelas | kode\_guru |
|  |  |  |

Tabel di atas merupakan implementasi relasi yang menghubungkan antara agregasi entitas guru dan entitas mapel.

**Tabel 1.10** tabel menghadiri

|  |  |
| --- | --- |
| kode\_jadwal | kode\_kelas |
|  |  |

Tabel di atas merupakan implementasi relasi yang menghubungkan agregasi jadwal dengan kelas yang akan diajarkan oleh guru yang bersangkutan.

* + - * 1. **ERD setelah dinormalisasi**

**Kamus data :**

Guru : {NUPTK, nama\_guru, kontak\_gutu, alamat\_guru}

PNS : {NUPTK, NIP}

Mapel : {kode\_mapel, nama\_mapel, NUPTK\_walikelas, NIS\_ketua}

Kelas : {kode\_kelas, nama\_kelas}

Siswa : {NIS, nama\_siswa, rangking, alamat\_siswa, nama\_wali, kontak\_wali, kode\_kelas}

|  |
| --- |
|  |

**Gambar 1.14** ERDsetelah dinormalisasi

* 1. **KESIMPULAN**

Berdasarkan praktikum yang telah dilaksanakan, dapat diambil beberapa kesimpulan, antara lain:

1. Menentukan entitas dari suatu permasalahan, perlu dilakukan adalah menganalisa objek-objek penting yang ada di dalam suatu permasalahan.
2. Atribut-atribut dalam suatu entitas, dalam penentuannya diperlukan prediksi dan perkiraan data yang akan ada dalam entitas tersebut. Atribut kunci adalah atribut yang pertama ditentukan di antara semua atribut, sebab merupakan ciri khas yang dapat membedakan satu entitas dengan entitas lain.
3. Melakukan identifikasi himpunan relasi berarti harus menguasai dan memahami bagaimana himpunan entitas saling berhubungan dan saling mempengaruhi satu sama lain. Kardinalitas dan jeNIS derajat relasi juga sangat penting, sebab untuk mengetahui peletakan *foreign key.*
4. Perancangan ERD dan tabel implementasinya sangat memerlukan penguaaan dan pemahaman akan setiap elemen perancangan diagram ER.
5. Normalisasi bentuk pertama, semua atribut harus bernilai tunggal (tidak dapat dibagi-bagi lagi). Pada normalisasi bentuk kedua, perlu diketahui *functional dependency* dari atribut bukan kunci pada atribut kunci. Dan di normalisasi bentuk ketiga, harus dipastikan bahwa sudah tidak ada lagi ketergantungan transitif dari atribut bukan kunci pada atribut bukan kunci lainnya.
6. MODUL II  
   MANIPULASI STRUKTUR TABEL
7. **TUJUAN**
8. Mahasiswa dapat membaca diagram ER.
9. Mahasiswa dapat mengimplementasikan rancangan dalam struktur tabel.
10. Mahasiswa dapat membuat tabel dengan SQL.
11. **LANDASAN TEORI**
12. **Implementasi ERD ke Dalam Database SQL**

Teknik yang dilakukan untuk melakukan transformasi ERD ke dalam *database* SQL adalah [1]:

Setiap himpunan entitas akan diimplementasikan sebagai sebuah tabel (*file* data).

|  |
| --- |
|  |

**Gambar 2.1** Simbol entitas

**Tabel 2. 1** Tabel matakuliah

|  |  |  |
| --- | --- | --- |
| kd\_mk | nama\_mk | sks |
|  |  |  |

Relasi dengan derajat relasi satu ke satu yang menghubungkan dua buah himpunan entitas akan direpresentasikan dalam bentuk penambahan atau penyetaraan atribut-atribut relasi ke tabel yang mewakili salah satu dari kedua entitas.

|  |
| --- |
|  |

**Gambar 2.2** Relasi 1-1

**Tabel 2.2** Tabel mahasiswa

|  |  |  |
| --- | --- | --- |
| nim | Nama | No\_hp |
|  |  |  |

**Tabel 2.3** Tabel orangtua

|  |  |  |
| --- | --- | --- |
| nama | No\_hp | nim |
|  |  |  |

Relasi dengan derajat relasi 1-N (satu ke banyak) yang menghubungkan 2 buah entitas, juga akan direpresentasikan dalam bentuk pencantuman atribut *key* dari himpunan entitas pertama (yang berderajat 1) ke tabel entitas kedua (yang berderajat N).

|  |
| --- |
|  |

**Gambar 2.3** Relasi 1-M

**Tabel 2.4** Tabel mahasiswa

|  |  |
| --- | --- |
| Id mhs | Id\_jurusan |
|  |  |

**Tabel 2.5** Tabel jurusan

|  |  |
| --- | --- |
| Id jurusan | Id\_mhs |
|  |  |

Relasi dengan derajat relasi banyak ke banyak yang menghubungkan dua buah himpunan entitas, akan diwujudkan dalam bentuk tabel khusus yang mewakili *field* (tepatnya *foreign-key*) yang berasal dari *key*-*key* dari himpunan entitas yang dihubungkannya.

* + 1. **Membuat dan Menghapus Basis Data**

DDL (*Data Definition Language*) merupakan perintah-perintah yang digunakan untuk mendefinisikan suatu basis data. Umumnya DDL digunakan sekali saja pada perancangan basis data. Terdapat beberapa perintah pada DDL, antara lain CREATE untuk membuat objek baru, ALTER untuk mengubah objek yang sudah ada, DROP untuk menghapus objek, RENAME untuk mengubah nama objek yang sudah ada dan TRUNCATE untuk mengosongkan suatu objek [2].

**CREATE *Database***

CREATE merupakan salah satu perintah DDL yang digunakan untuk menciptakan basis data baru maupun objek-objek lain seperti tabel dalam basis data. SQL yang umum digunakan:

|  |
| --- |
| CREATE DATABASE nama\_basis\_data; |

Aturan penamaan pada sebuah *database* sama seperti aturan penamaan variabel yaitu nama terdiri dari huruf, angka, dan *underscode* (\_). Contoh untuk membuat suatu *database* dengan nama praktikum\_sbd menggunakan *query* sebagai berikut:

|  |
| --- |
| CREATE DATABASE praktikum\_sbd; |

**DROP *Database***

DROP merupakan salah satu perintah DDL. DDL digunakan untuk menghapus basis data yang sudah ada maupun objek-objek lain seperti tabel dalam basis data. SQL yang umum digunakanyaitu :

|  |
| --- |
| DROP DATABASE nama\_basis\_data; |

Contoh untuk menghapus suatu *database* dengan nama praktikum\_sbd menggunakan *query* sebagai berikut:

|  |
| --- |
| DROP DATABASE praktikum\_sbd; |

* + 1. **Membuat dan Menghapus Tabel**

1. **CREATE Tabel**

Untuk membuat suatu tabel pada SQL, *query* yang umum digunakan yaitu :

|  |
| --- |
| CREATE TABLE nama\_tabel (  nama\_kolom1 tipe\_data (size) nama\_konstrain,  nama\_kolom2 tipe\_data (size) nama\_konstrain,  ...); |

1. nama\_tabel adalah nama tabel yang akan dibuat.
2. nama\_kolom1, nama\_kolom2 adalah nama kolom yang akan dibuat.
3. tipe\_data tergantung implementasi sistem manajemen basis data, misalnya pada MySQL tipe data *Varchar*, *Text*, *Integer*, *Datetime*, dan sebagainya.
4. nama\_konstrain adalah batasan yang diberikan untuk tiap kolom. Misalnya NOT NULL, UNIQUE, PRIMARY KEY, dan FOREIGN KEY.

Contoh untuk membuat suatu tabel dosen pada SQL, *query* yang digunakan yaitu:

|  |
| --- |
| CREATE TABLE dosen (  nip INT PRIMARY KEY,  nama VARCHAR (30),  alamat VARCHAR (50)); |

1. **DROP Tabel**

Untuk menghapus suatu tabel pada SQL, *query* yang umum digunakan yaitu:

|  |
| --- |
| DROP TABLE nama\_tabel; |

Contoh untuk menghapus suatu tabel dosen pada SQL, *query* yang digunakan yaitu:

|  |
| --- |
| DROP TABLE dosen; |

* + 1. **Merubah Struktur Tabel**

Untuk merubah suatu struktur tabel maka dapat menggunakan perintah sebagai berikut [1] :

1. **ALTER TABEL**

ALTER TABEL adalah perintah SQL yang digunakan untuk merubah struktur kolom pada tabel.

1. **RENAME TO**

Digunakan untuk merubah nama tabel.

|  |
| --- |
| ALTER TABLE nama\_tabel RENAME TO nama\_tabel\_baru; |

1. **ADD**

Digunakan untuk menambah kolom pada sebuah tabel.

|  |
| --- |
| ALTER TABLE nama\_tabel ADD nama\_kolom tipe\_data (size); |

1. **ALTER COLOUMN**

Digunakan untuk merubah struktur kolom pada sebuah tabel.

|  |
| --- |
| ALTER TABLE nama\_tabel ALTER COLOUMN nama\_kolom tipe\_data (size); |

1. **DROP**

Digunakan untuk menghapus kolom atau *field*.

|  |
| --- |
| ALTER TABLE nama\_tabel DROP COLOUMN nama\_kolom; |

* + 1. **Melihat informasi tabel**

Digunakan untuk melihat tabel apa saja pada *database* tertentu.

|  |
| --- |
| SHOW TABLES FROM nama\_db; |

*Query* di atas untuk melihat deskripsi tabel tertentu.

|  |
| --- |
| DESC nama\_tabel; |

* + 1. **Mendapatkan Informasi Tabel**

Digunakan untuk melihat isi tabel seluruhnya.

|  |
| --- |
| SELECT \*FROM nama\_tabel; |

Untuk menampilkan isi dari kolom-kolom tertentu.

|  |
| --- |
| SELECT kolom1, kolom2,… FROM nama\_tabel; |

Sedangkan untuk menampilkan isi dari kolom dengan kondisi tertentu.

|  |
| --- |
| SELECT kolom1 FROM nama\_tabel WHERE kolom2=isi kolom; |

* + 1. **Modifikasi Informasi dalam Tabel**

Digunakan untuk menambah baris baru berupa informasi dalam tabel.

|  |
| --- |
| INSERT INTO nama\_tabel (nama\_kolom) VALUES (isi\_kolom); |

1. **KASUS KELOMPOK**

Sekolah Dasar Internasional Wakanda (SBIW) akan merancang sebuah sistem informasi. Basis data yang dibentuk berdasarkan data-data berbagai entitas. Guru akan didata nama, kontak dan alamat sehingga terbentuk NUPTK. Lalu guru dibedaan menjadi guru honorer dan PNS yang memiliki NIP. Semua guru dapat mengajar mata pelajaran, dan guru pun menguasai tidak hanya satu saja mata pelajaran, namun terkadang mereka harus mengajar mata pelajaran yang tidak mereka pelajari, sehingga mata pelajaran dapat dikuasai dan diajar oleh banyak guru. Mata pelajaran hanya didata namanya saja, sehingga dibentuk kode mata pelajaran. Setiap mata pelajaran yang diajar guru akan didata pegisian kelas mana yang diajar. Sehingga guru dapat mengajar mata pelajaran yang sama di kelas yang berbeda, begitupun dikelas yang sama dapat mengajar mata pelajaran lain. Kelaspun harus di data nama kelasnya sehinggaa terbentuk kode kelas. Sebuah kelas dapat diwalikan oleh seorang guru non-honorer. Sebuah kelaspun memiliki banyak siswa, dimana dari mereka terdapat satu yang menjadi ketua kelas. Setiap siswa akan didata nama, alamay, nama wali, kontak wali, dan rangking mereka hingga dibentuk NIS. Setiap siswa dapat mempelajari banyak mata pelajaran, begitu pula sebaliknya setiap mata pelajaran dapat dipelajari banyak siswa. Setiap kegiatan belajar mengajar akan dilakukan pengambilan nilai.

|  |
| --- |
|  |

**Gambar 2.4** Rancangan ERD

1. **KASUS MODUL**
   * 1. Setiap kelompok membagi jumlah table ke masing-masing anggota kelompok
     2. Untuk setiap tabel dilakukan perintah berikut:
2. Membuat *database*
3. Membuat tabel
4. Menampilkan tabel
5. Menampilkan struktur tabel
6. Menambah *field*
7. Menambah *primary key*
8. Mengubah nama *field*
9. Mengubah struktur *field*
10. Mengubah nama tabel dengan perintah RENAME
11. Mengubah nama tabel dengan perintah ALTER
12. Menghapus *field*
13. Menghapus tabel
14. **ANALISIS KASUS**
15. **Membuka XAMPP**

|  |
| --- |
| xampp.PNG |

**Gambar 2.5** Tampilan XAMPP

Pada **Gambar 2.5** merupakan tampilan utama XAMPP, dan setelah itu klik *start* pada *actions Apache,* dan *MySQL*. *start* pada *actions Apache,* berfungsi untuk membuka *PHP my.admin*, sedangkan *start* pada *actions MySQL* berfungsi untuk membuka *MySQL.*

1. **Membuka CMD**

|  |
| --- |
|  |

**Gambar 2.6** Tampilan CMD

Setelah apikasi XAMPP telah dibuka dan sudah memulai proses *running* maka langkah yang harus dilakukan selanjutnya adalah membuka CMD. CMD disini berfungsi sebagai pembuatan tabel-tablel yang diperlukan sesuai dengan entitas yang terdapat dalam kasus kelompok. **Gambar 2.6** merupakan tampilan CMD ketika pertama kali dibuka untuk melakukan proses pembuatan *database*.

1. **Membuat dan Menggunakan *Database***

|  |
| --- |
|  |

**Gambar 2.7** Membuat dan menggunakan *database*

Perintah “show databases” digunakan untuk menampilkan *database* yang tersedia atau yang sebelumnya telah dibuat. Untuk menggunkan *database* yang tersedia maka digunakan perintah “use data base ririn”.

1. **Tabel Guru**
   1. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.8** Membuat tabel Guru

Perintah *Query* pada **Gambar 2.8** merupakan cara untuk membuat tabel Guru, dengan menggunakan perintah “create table” yang diikuti nama tabenya, dilanjutkan dengan tanda kurung yang didalamnya adalah kolom kolom yang tersedia pada tabel tersebut yakni “alamat guru” dengan tipe data “varchar(50)”. “kontak\_guru” dengan tipe data “int”, “nama\_guru” dengan tipe data “varchar(30)”, dan “NUPTK” dengan tipe data “varchar(10)”.

* 1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.9** Menampilkan tabel Guru

Pada **Gambar 2.9** merupakan hasil dari perintah “show tables” yang digunakan untuk menampilakn tabel-tabel yang telah dibuat.

* 1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.10** Menampilkan struktur tabel Guru

Perintah *Query* “desc Guru” pada **Gambar 2.10** merupakan perintah *query* dalam mySQL untuk menampilkan struktur tabel “Guru” dari *database*. Hasilnya adalah kolom-kolom beserta tipe datanya

* 1. **Menambah *Primary Key***

|  |
| --- |
|  |

**Gambar 2.11** Menambahkan *primary key* tabel Guru

Pada **Gambar 2.11** memperlihatkan *query* untuk menjadikan suatu *field* dalam tabel menjadi atribut kunci. Pada *query* di atas, dalam tabel “Guru”, *field* “NUPTK” dijadikan sebagai atribut kunci.

* 1. **Menambah *Field***

|  |
| --- |
|  |

**Gambar 2.12** Menambahkan *field* baru pada tabel Guru

Perintah *query* pada **Gambar 2.12** yakni “alter table Guru add status\_guru varchar(10);” digunakan untuk merubah struktur tabel degan menambah kolom “status\_guru” dan dengan tipe data “varchar(10)” pada tabel “Guru”.

* 1. **Mengubah Nama *Field***

|  |
| --- |
|  |

**Gambar 2.13** Mengubah *field* pada tabel Guru

Perintah *query* **Gambar 2.13** yakni “alter table Guru change status\_guru status varchar(10)” digunakan untuk merubah struktur tabel dengan mengubah nama kolom “status\_guru” menjadi “status” dengan tipe data “varchar(10)” pada tabel “Guru”.

* 1. **Mengubah Struktur *Field***

|  |
| --- |
|  |

**Gambar 2.14** Mengubah struktur *field* pada tabel Guru

Perintah *query* **Gambar 2.14** yakni “alter table Guru change status status int;” digunakan untuk merubah struktur tabel “status” yang awalnya bertipe data “varchar” diubah menjadi tipe data “int”.

* 1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.15** Mengubah nama tabel anggota dengan perintah *rename*

Perintah *query* **Gambar 2.15** yakni “rename table Giri to Guru;” merupakan salah satu perintah DDL yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Giri” namanya diganti dengan nama tabel “Guru”.

* 1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.16** Mengubah nama tabel Guru dengan perintah *alter*

Perintah *Query* **Gambar 2.16** “alter table Guru rename Giri;” merupakan salah satu perintah DDL yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Guru” namanya diganti dengan tabel “Giri”.

* 1. **Menghapus *Field***

|  |
| --- |
|  |

**Gambar 2.17** Menghapus *field* dalam tabel Guru

Perintah *query* **Gambar 2.17** yakni “alter table Guru drop status;” merupakan salah satu perintah yang digunakan untuk menghapus sebuah kolom pada tabel. Potongan script “DROP” diatas merupakan perintah yang digunakan untuk menghapus *field* tersebut.

* 1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.18** Menghapus tabel Guru

Perintah **Gambar 2.18** “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Guru” diartikan sebagai perintah untuk menghapus tabel “Guru”.

1. **Tabel Mata\_Pelajaran**
   1. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.19** Membuat tabel Mata\_Pelajaraan

Perintah *query* **Gambar 2.19** merupakan cara untuk membuat tabel Mata\_Pelajaran, dengan menggunakan perintah “create”, selanjutnya untuk menambahkan atribut-atribut atau kolom untuk tabel “Mata\_Pelajaran” dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “kode\_mapel” dengan tipe data “int” dan “nama\_mapel” dengan tipe data “varchar(20)”.

* 1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.20** Menampilkan tabel Mata\_Pelajaraan

Pada **Gambar 2.20** merupakan perintah *query* “show tables” yang digunakan untuk menampilkan tabel-tabel yang sebelumnya telah dibuat. Dapat terlihat bahwa basis data ririn berisi tabel Mata\_Pelajaran.

* 1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.21** Menampilkan struktur tabel Mata\_Pelajaraan

Perintah *Query* **Gambar 2.21**“desc Mata\_Pelajaran” merupakan perintah *query* dalam mySQL untuk menampilkan struktur tabel “Mata\_Pelajaran” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

* 1. **Menambah *Primary Key***

|  |
| --- |
|  |

**Gambar 2.22** Menambahkan *primary key* tabel Mata\_Pelajaraan

Pada **Gambar 2.22** memperlihatkan *query* untuk menjadikan suatu *field* dalam tabel menjadi atribut kunci. Pada *query* di atas, dalam tabel “Mata\_Pelajaran”, *field* “kode­\_mapel” dijadikan sebagai atribut kunci.

* 1. **Menambah *Field***

|  |
| --- |
|  |

**Gambar 2.23**Menambahkan *field* baru pada tabel Mata\_Pelajaraan

Perintah *query* **Gambar 2.23** yakni “alter table Mata\_Pelajaran add guru\_mapel varchar(20);” merupakan salah satu perintah untuk menambah kolom baru dengan nama “guru\_mapel” dan dengan tipe data “varchar(20)” pada tabel “Mata\_Pelajaran”. Potongan *script* “add” diatas merupakan perintah yang digunakan untuk menambah kolom.

* 1. **Mengubah Nama *Field***

|  |
| --- |
|  |

**Gambar 2.24**Mengubah *field* pada tabel Mata\_Pelajaraan

Perintah *query* **Gambar 2.24** “alter table Mata\_Pelajaran change guru\_mapel gurumapel varchar(20)” merupakan salah satu perintah yang digunakan untuk menguubah kolom “guru\_mapel” dan yang diubah namanya menjadi “gurumapel” dengan tipe data “varchar(20)” pada tabel “Mata\_Pelajaran”. Sedangkan potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah nama kolom.

* 1. **Mengubah Struktur *Field***

|  |
| --- |
|  |

**Gambar 2.25**Mengubah struktur *field* pada tabel Mata\_Pelajaraan

Perintah *query* **Gambar 2.25** “alter table Mata\_Pelajaran change gurumapel gurumapel int;” merupakan salah satu perintah yang digunakan untuk mengubah struktur tabel yakni untuk tipe data dari suatu kolom. tipe data kolom “gurumapel” yang tadi awalnya bertipe data “varchar” diubah menjadi tipe data “int”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah struktur kolom.

* 1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.26** Mengubah nama tabel admin dengan perintah Mata\_Pelajaraan

Perintah *query* **Gambar 2.26** “rename table Mata\_Pelajaran to MAPEL;” merupakan salah satu perintah DDL yang digunakan untuk mengubah nama sebuah tabel, *command* “RENAME” pada tabel dengan nama “Mata\_Pelajaran” namanya diganti menjadi tabel “MAPEL”.

* 1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.27** Mengubah nama tabel admin dengan perintah Mata\_Pelajaraan

Perintah *query* **Gambar 2.27** “alter table MAPEL rename Mata\_Pelajra;” merupakan salah satu perintah DDL yang digunakan untuk mengubah nama sebuah tabel, *command* “ALTER” pada tabel dengan nama “MAPEL” namanya diganti menjadi tabel “Mata\_Pelajaran”.

* 1. **Menghapus *Field***

|  |
| --- |
|  |

**Gambar 2.28** Menghapus *field* dalam tabel Mata\_Pelajaraan

Perintah *query* **Gambar 2.28** “alter table Mata\_Pelajaran drop gurumapel;” merupakan salah satu perintah DDL yaitu “ALTER” yang digunakan untuk menghapus sebuah kolom pada tabel. Potongan script “DROP” diatas merupakan perintah yang digunakan untuk menghapus *field* tersebut.

* 1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.29** Menghapus tabel Mata\_Pelajaraan

Pada **Gambar 2.29** Perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Mata\_Pelajaran” diartikan sebagai perintah untuk menghapus tabel “Mata\_Pelajaran”.

1. **Tabel PNS**
   1. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.30** Membuat tabel PNS

Perintah *query* **Gambar 2.30** merupakan cara untuk membuat tabel PNS, dengan menggunakan perintah *create*, dimana selanjutnya, penambahan atribut-atribut atau kolom untuk tabel “PNS” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “NIP” dengan tipe data “varchar(10)” dan “NUPTK” dengan tipe data “varchar(20)”.

* 1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.31** Menampilkan tabel PNS

Pada **Gambar 2.31** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel PNS.

* 1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.32** Menampilkan struktur tabel PNS

Perintah *query* **Gambar 2.32** “desc PNS” diatas merupakan perintah *query* dalam mySQL untuk menampilkan struktur tabel “PNS” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

* 1. **Menambah *Field***

|  |
| --- |
|  |

**Gambar 2.33** Menambahkan *field* baru pada tabel PNS

Perintah *query* **Gambar 2.33** “alter table PNS add guru varchar(10);” merupakan salah satu perintah DDL yang digunakan untuk menambah kolom baru dengan nama “guru” dan dengan tipe data “varchar(10)” pada tabel “PNS”. Potongan *script* “add” diatas merupakan perintah yang digunakan untuk menambah kolom.

* 1. **Mengubah Nama *Field***

|  |
| --- |
|  |

**Gambar 2.34** Mengubah *field* pada tabel PNS

Perintah *query* **Gambar 2.34** “alter table PNS change guru namaguru varchar(10)” merupakan salah satu perintah yang digunakan untuk mengubah nama kolom “guru” menjadi “namaguru” dengan tipe data “varchar(10)” pada tabel “PNS”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah nama kolom.

* 1. **Mengubah Struktur *Field***

|  |
| --- |
|  |

**Gambar 2.35** Mengubah struktur *field* pada tabel PNS

Perintah *query* **Gambar 2.35** “alter table PNS change namaguru namaguru int;” merupakan salah satu perintah yang digunakan untuk mengubah tipe data kolom “namaguru” yang tadi awalnya bertipe data “varchar” diubah menjadi tipe data “int”. Potongan *script* “change” diatas merupakan perintah yang digunakan suntuk merubah struktur kolom.

* 1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.36** Mengubah nama tabel PNS dengan perintah *rename*

Perintah *query* **Gambar 2.36** “rename table PNS to pegawainegerisispil;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *command* “RENAME” pada tabel dengan nama “PNS” diganti dengan nama tabel “pegawainegerisipil”.

* 1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.37** Mengubah nama tabel PNS dengan perintah *alter*

Perintah *query* **Gambar 2.37** “alter table pegawainegerisipil rename PNS;” merupakan salah satu perintah DDL yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “pegawainegerisipil” diganti menjadi “PNS”.

* 1. **Menghapus *Field***

|  |
| --- |
|  |

**Gambar 2.38** Menghapus *field* dalam tabel PNS

Perintah *query* **Gambar 2.38** “alter table PNS drop namaguru;” merupakan salah satu perintah yang digunakan untuk menghapus sebuah kolom pada tabel. Potongan script “DROP” diatas merupakan perintah yang digunakan untuk menghapus *field* tersebut.

* 1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.39** Menghapus tabel PNS

Pada **Gambar 2.39** perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table PNS” diartikan sebagai perintah untuk menghapus tabel “PNS”.

1. **Tabel Honorer**
2. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.40** Membuat tabel Honorer

Pada **Gambar 2.40** perintah *query* diatas merupakan cara untuk membuat tabel Honorer, dengan menggunakan perintah *create*, dimana selanjutnya, penambahan atribut-atribut atau kolom untuk tabel “Honorer” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “NUPTK” dengan tipe data “varchar(20)”.

1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.41** Menampilkan tabel Honorer

Pada **Gambar 2.41** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Honorer.

1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.42** Menampilkan struktur tabel Honorer

Pada **Gambar 2.42** perintah *query* “desc Honorer” diatas merupakan perintah *query* dalam mySQL untuk menampilkan struktur tabel “Hanorer” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.43** Mengubah nama tabel Honorer dengan perintah *rename*

Pada **Gambar 2.43**perintah *query* “rename table Honorer to Honor;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Honorer” diganti menjadi “Honor”.

1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.44** Mengubah nama tabel Honorer dengan perintah *alter*

Pada **Gambar 2.44** perintah *query* “alter table Honor rename Honorer;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Honor” namanya diganti dengan tabel “Honorer”.

1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.45** Menghapus tabel Honorer

Pada **Gambar 2.45**perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Honorer” diartikan sebagai perintah untuk menghapus tabel “Honorer”.

1. **Tabel Siswa**
   1. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.46** Membuat tabel Siswa

Pada **Gambar 2.46** perintah *query* diatas merupakan cara untuk membuat tabel Siswa, dengan menggunakan perintah *create*, dimana selanjutnya, penambahan atribut-atribut atau kolom untuk tabel “Siswa” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “NIS” dengan tipe data “varchar(20)”, “nama\_siswa” dengan tipe data “varchar(30)”, “alamat\_siswa” dengan tipe data “varchar(50)”, “nama\_wali” dengan tipe data “varchar(20)”, “kontak\_wali” dengan tipe data “varchar(20)”, “kode\_mapel” dengan tipe data “int”, dan “kode\_kelas” dengan tipe data “varchar(5)”.

* 1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.47** Menampilkan tabel Siswa

Pada **Gambar 2.47** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Siswa.

* 1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.48** Menampilkan struktur tabel Siswa

Pada **Gambar 2.48** perintah *query* “desc Siswa” merupakan perintah *query* yang digunakan untuk menampilkan struktur tabel “Siswa” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

* 1. **Menambah *Primary Key***

|  |
| --- |
|  |

**Gambar 2.49** Menambahkan *primary key* tabel Siswa

Pada **Gambar 2.49** memperlihatkan *query* untuk menjadikan suatu *field* dalam tabel menjadi atribut kunci. Pada *query* di atas, dalam tabel “Siswa”, *field* “NIS” dijadikan sebagai atribut kunci.

* 1. **Menambah *Field***

|  |
| --- |
|  |

**Gambar 2.50** Menambahkan *field* baru pada tabel Siswa

Pada **Gambar 2.50** perintah *query* “alter table Siswa add guru varchar(20);” merupakan salah satu perintah yang digunakan untuk menambah kolom baru dengan nama “guru” dan dengan tipe data “varchar(20)” pada tabel “Siswa”. Potongan *script* “add” diatas merupakan perintah yang digunakan untuk menambah kolom.

* 1. **Mengubah Nama *Field***

|  |
| --- |
|  |

**Gambar 2.51** Mengubah *field* pada tabel Siswa

Pada **Gambar 2.51** perintah *query* “alter table Siswa change guru namaguru varchar(20)” merupakan salah satu perintah yang digunakan untuk mengubah nama kolom “guru” dan menjadi “namaguru” dengan tipe data “varchar(20)” pada tabel “Siswa”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah nama kolom.

* 1. **Mengubah Struktur *Field***

|  |
| --- |
|  |

**Gambar 2.52** Mengubah struktur *field* pada tabel Siswa

Pada **Gambar 2.52** perintah *query* “alter table Siswa change namaguru namaguru int;” merupakan salah satu perintah yang digunakan untuk mengubah tipe data kolom “namaguru” yang tadi awalnya bertipe data “varchar” diubah menjadi tipe data “int”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah struktur kolom.

* 1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.53** Mengubah nama tabel Siswa dengan perintah *rename*

Pada **Gambar 2.53** perintah *query* “rename table Siswa to Siswi;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Siswa” menjadi tabel “Siswi”.

* 1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.54** Mengubah nama tabel Siswa dengan perintah *alter*

Pada **Gambar 2.54** perintah *query* “alter table Siswi rename Siswa;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Siswi” diganti menjadi “Siswa”.

* 1. **Menghapus *Field***

|  |
| --- |
|  |

**Gambar 2.55** Menghapus *field* dalam tabel Siswa

Pada **Gambar 2.55** perintah *query* “alter table Siswa drop namaguru;” merupakan salah satu perintah yang digunakan menghapus sebuah kolom pada tabel. Potongan script “DROP” diatas merupakan perintah yang digunakan untuk menghapus *field* tersebut.

* 1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.56** Menghapus tabel Siswa

Pada **Gambar 2.56** perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Siswa” diartikan sebagai perintah untuk menghapus tabel “Siswa”.

1. **Tabel Kelas**
   1. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.57** Membuat tabel Kelas

Pada **Gambar 2.57** perintah *query* diatas merupakan cara untuk membuat tabel Kelas, dengan menggunakan perintah *create*, dimana selanjutnya, penambahan atribut-atribut atau kolom untuk tabel “Kelas” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “kode\_kelas” dengan tipe data “varchar(20)”, “nama\_kelas” dengan tipe data “varchar(20)”, “NIS\_ketua” dengan tipe data “varchar(10)”, dan “NUPTK\_walikelas” dengan tipe data “varchar(20)”.

* 1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.58** Menampilkan tabel Kelas

Pada **Gambar 2.58** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Kelas.

* 1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.59** Menampilkan struktur tabel Kelas

Pada **Gambar 2.59** perintah *query* “desc Kelas” diatas merupakan perintah *query* untuk menampilkan struktur tabel “Kelas” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

* 1. **Menambah *Primary Key***

|  |
| --- |
|  |

**Gambar 2.60** Menambahkan *primary key* tabel Kelas

Pada **Gambar 2.60** memperlihatkan *query* untuk menjadikan suatu *field* dalam tabel menjadi atribut kunci. Pada *query* di atas, dalam tabel “Kelas”, *field* “kode\_kelas” dijadikan sebagai atribut kunci.

* 1. **Menambah *Field***

|  |
| --- |
|  |

**Gambar 2.61** Menambahkan *field* baru pada tabel Kelas

Pada **Gambar 2.61** perintah *query* “alter table Kelas add guru varchar(20);” merupakan salah satu perintah DDL yaitu “ALTER” yang digunakan untuk merubah struktur tabel, dalam hal ini menambah kolom baru dengan nama “guru” dan dengan tipe data “varchar(20)” pada tabel “Kelas”. Potongan *script* “add” diatas merupakan perintah yang digunakan untuk menambah kolom.

* 1. **Mengubah Nama *Field***

|  |
| --- |
|  |

**Gambar 2.62** Mengubah *field* pada tabel Kelas

Pada **Gambar 2.62** perintah *query* “alter table Kelas change guru namaguru varchar(20)” merupakan salah satu perintah yang digunakan untuk mengubah nama kolom “guru” menjadi “namaguru” dengan tipe data “varchar(20)” pada tabel “Kelas”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah nama kolom.

* 1. **Mengubah Struktur *Field***

|  |
| --- |
|  |

**Gambar 2.63** Mengubah struktur *field* pada tabel Kelas

Pada **Gambar 2.63** perintah *query* “alter table Kelas change namaguru namaguru int;” merupakan salah satu perintah yang digunakan mengubah tipe data kolom “namaguru” yang tadi awalnya bertipe data “varchar” diubah menjadi tipe data “int”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah struktur kolom.

* 1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.64** Mengubah nama tabel Kelas dengan perintah *rename*

Pada **Gambar 2.64** perintah *query* “rename table Kelas to Ruangan;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Kelas” diganti menjadi “Ruangan”.

* 1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.65** Mengubah nama tabel Kelas dengan perintah *alter*

Pada **Gambar 2.65** perintah *query* “alter table Ruangan rename Kelas;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Ruangan” diganti menjadi “Kelas”.

* 1. **Menghapus *Field***

|  |
| --- |
|  |

**Gambar 2.66** Menghapus *field* dalam tabel Kelas

Pada **Gambar 2.66** perintah *query* “alter table Kelas drop namaguru;” merupakan salah satu perintah yang digunakan untuk menghapus sebuah kolom pada tabel. Potongan script “DROP” diatas merupakan perintah yang digunakan untuk menghapus *field* tersebut.

* 1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.67** Menghapus tabel Kelas

Pada **Gambar 2.67** paerintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Kelas” diartikan sebagai perintah untuk menghapus tabel “Kelas”.

1. **Tabel Menguasai**
2. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.68** Membuat tabel Menguasai

Pda **Gambar 2.68** perintah *query* diatas merupakan cara untuk membuat tabel Menguasai, dengan menggunakan perintah *create*, dimana selanjutnya, penambahan atribut-atribut atau kolom untuk tabel “Menguasai” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “NUPTK” dengan tipe data “varchar(10)” dan “Kode\_mapel” dengan tipe data “varchar(10)”.

1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.69** Menampilkan tabel Menguasai

Pada **Gambar 2.69** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Menguasai.

1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.70** Menampilkan struktur tabel Menguasai

Pada **Gambar 2.70** perintah *query* “desc Menguasai” diatas merupakan perintah *query* untuk menampilkan struktur tabel “Menguasai” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.71** Mengubah nama tabel Menguasai dengan perintah *rename*

Pada **Gambar 2.71**perintah *query* “rename table Menguasai to Kuasai;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Menguasai” namanya diganti dengan nama tabel “Kuasai”.

1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.72** Mengubah nama tabel Menguasai dengan perintah *alter*

Pada **Gambar 2.72** perintah *query* “alter table Kuasai rename Menguasai;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Kuasai” namanya diganti dengan tabel “Menguasai”.

1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.73** Menghapus tabel Menguasai

Pada **Gambar 2.73**perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Menguasai” diartikan sebagai perintah untuk menghapus tabel “Menguasai”.

1. **Tabel Mengajar**
   * + 1. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.74** Membuat tabel Mengajar

Pada **Gambar 2.74** perintah *query* diatas merupakan cara untuk membuat tabel Mengajar, dengan menggunakan perintah *create.* Selanjutnya, penambahan atribut-atribut atau kolom untuk tabel “Mengajar” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “NUPTK” dengan tipe data “varchar(10)” dan “Kode\_mapel” dengan tipe data “varchar(10)”.

* + - 1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.75** Menampilkan tabel Mengajar

Pada **Gambar 2.75** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Mengajar.

* + - 1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.76** Menampilkan struktur tabel Mengajar

Pada **Gambar 2.76** perintah *query* “desc Mengajar” diatas merupakan perintah *query* untuk menampilkan struktur tabel “Mengajar” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

* + - 1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.77** Mengubah nama tabel Mengajar dengan perintah *rename*

Pada **Gambar 2.77**perintah *query* “rename table Mengajar to Ajar;” merupakan salah satu perintah digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Mengajar” menjadi “Ajar”.

* + - 1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.78** Mengubah nama tabel Mengajar dengan perintah *alter*

Pada **Gambar 2.78** perintah *query* “alter table Ajar rename Mengajar;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Ajar” diubah menjadi “Mengajar”.

* + - 1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.79** Menghapus tabel Mengajar

Pada **Gambar 2.79**perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Mengajar” diartikan sebagai perintah untuk menghapus tabel “Mengajar”.

1. **Tabel Menghadiri**
2. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.80** Membuat tabel Menghadiri

Pada **Gambar 2.80** paerintah *query* diatas merupakan cara untuk membuat tabel Menghadiri, dengan menggunakan perintah *create*. Selanjutnya penambahan atribut-atribut atau kolom untuk tabel “Menghadiri” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “NUPTK” dengan tipe data “varchar(10)”, “Kode\_mapel” dengan tipe data “varchar(10)”, dan “Kode\_kelas” dengan tipe data “varchar(10)”.

1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.81** Menampilkan tabel Menghadiri

Pada **Gambar 2.81** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Menghadiri.

1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.82** Menampilkan struktur tabel Menghadiri

Pada **Gambar 2.82** perintah *query* “desc Menghadiri” diatas merupakan perintah *query* dalam mySQL untuk menampilkan struktur tabel “Menghadiri” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.83** Mengubah nama tabel Menghadiri dengan perintah *rename*

Pada **Gambar 2.83**paerintah *query* “rename table Menghadiri to Hadir;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “Menghadiri” diganti menjadi “Hadir”.

1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.84** Mengubah nama tabel Menghadiri dengan perintah *alter*

Pada **Gambar 2.84** perintah *query* “alter table Hadir rename Menghadiri;” merupakan salah satu perintah DDL yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “Hadir” diganti menjaadi “Menghadiri”.

1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.85** Menghapus tabel Menghadiri

Pada **Gambar 2.85**perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Menghadiri” diartikan sebagai perintah untuk menghapus tabel “Menghadiri”.

1. **Tabel Mempelajari**
2. **Pembuatan Tabel**

|  |
| --- |
|  |

**Gambar 2.86** Membuat tabel Mempelajari

Pada **Gambar 2.86** perintah *query* diatas merupakan cara untuk membuat tabel Mempelajari, dengan menggunakan perintah *create*. Selanjutnya penambahan atribut-atribut atau kolom untuk tabel “Mempelajari” dapat dilakukan. Dalam pembuatan kolom dilakukan dengan cara menulis nama kolom terlebih dahulu dan tipe data dari kolom tersebut. Kolom-kolom yang akan dibuat pada tabel anggota diatas yaitu, “nilai” dengan tipe data “int”, “Kode\_mapel” dengan tipe data “varchar(10)”, dan “Kode\_kelas” dengan tipe data “varchar(10)”.

1. **Menampilkan Tabel**

|  |
| --- |
|  |

**Gambar 2.87** Menampilkan tabel Mempelajari

Pada **Gambar 2.87** merupakan penampilan tabel-tabel yang ada dalam basis data dengan menggunakan *query* “show tables”. Dapat terlihat bahwa basis data ririn berisi tabel Mempelajari.

1. **Menampilkan Struktur Tabel**

|  |
| --- |
|  |

**Gambar 2.88** Menampilkan struktur tabel Mempelajari

Pada **Gambar 2.88** perintah *query* “desc Mempelajari” diatas merupakan perintah *query* untuk menampilkan struktur tabel “Mempelajari” dari suatu *database*. Dan dari gambar diatas, sudah dapat dilihat bahwa tabel dan semua kolom yang ditambahkan sebelumnya sudah terdaftar di dalam *database*.

1. **Menambah *Field***

|  |
| --- |
|  |

**Gambar 2.89** Menambahkan *field* baru pada tabel Mempelajari

Pada **Gambar 2.89** perintah *query* “alter table Mempelajari add pelajar varchar(10);” merupakan salah satu perintah yang digunakan untuk menambah kolom baru dengan nama “pelajar” dan dengan tipe data “varchar(10)” pada tabel “Mempelajari”. Potongan *script* “add” diatas merupakan perintah yang digunakan untuk menambah kolom.

1. **Mengubah Nama *Field***

|  |
| --- |
|  |

**Gambar 2.90** Mengubah *field* pada tabel Mempelajari

Pada **Gambar 2.90** perintah *query* “alter table Mempelajari change pelajar ajar varchar(10)” merupakan salah satu perintah yang digunakan untuk mengubah nama kolom “pelajar” menjadi “ajar” dengan tipe data “varchar(10)” pada tabel “Mempelajari”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah nama kolom.

1. **Mengubah Struktur *Field***

|  |
| --- |
|  |

**Gambar 2.91** Mengubah struktur *field* pada tabel Mempelajari

Pada **Gambar 2.91** perintah *query* “alter table Mempelajari change ajar ajar int;” merupakan salah satu perintah yang digunakan untuk mengubah strukturt ipe data kolom “ajar” yang tadi awalnya bertipe data “varchar” menjadi tipe data “int”. Potongan *script* “change” diatas merupakan perintah yang digunakan untuk merubah struktur kolom.

1. **Mengubah Nama Tabel dengan Perintah *Rename***

|  |
| --- |
|  |

**Gambar 2.92** Mengubah nama tabel Mempelajari dengan perintah *rename*

Pada **Gambar 2.92** perintah *query* “rename table mempelajari to ajar;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, *query* yang digunakan menggunakan *command* “RENAME” pada tabel dengan nama “mempelajari” namanya diganti dengan nama tabel “ajar”.

1. **Mengubah Nama Tabel dengan Perintah *Alter***

|  |
| --- |
|  |

**Gambar 2.93** Mengubah nama tabel Mempelajari dengan perintah *alter*

Pada **Gambar 2.93** perintah *query* “alter table ajar rename mempelajari;” merupakan salah satu perintah yang digunakan untuk mengubah nama sebuah tabel, pada kasus ini, *query* yang digunakan menggunakan *command* “ALTER” pada tabel dengan nama “ajar” diganti menjadi “mempelajari”.

1. **Menghapus *Field***

|  |
| --- |
|  |

**Gambar 2.94** Menghapus *field* dalam tabel Mempelajari

Pada **Gambar 2.94** perintah *query* “alter table Mempelajari drop statussiswa;” merupakan salah satu perintah yang digunakan untuk menghapus sebuah kolom pada tabel. Potongan script “DROP” diatas merupakan perintah yang digunakan untuk menghapus *field* tersebut.

1. **Menghapus Tabel**

|  |
| --- |
|  |

**Gambar 2.95** Menghapus tabel Mempelajari

Pada **Gambar 2.95** perintah “drop” dapat juga digunakan untuk menghapus tabel. Caranya dengan menuliskan perintah “drop table” disertai nama tabel yang ingin dihapus. “drop table Mempelajari” diartikan sebagai perintah untuk menghapus tabel “Mempelajari”.

1. **KESIMPULAN**

Berdasarkan praktikum yang dilakukan, dapat diambil beberapa kesimpulan yaitu sebagai berikut:

* + - 1. Pembacaan ERD dapat dilakukan dengan membaca setiap entitas, dimana setiap entitas memiliki atributnya masing masing baik itu atribut sendiri ataupun atribut yang diambil dari entitas lain sebagai *foreign key.*
      2. Pengimplmentasian rancangan ERD ke dalam struktur tabel adalah dengan menempatkan setiap atribut kedalam tabel entitasnya masing masing.
    1. Pembuatan tabel dalam SQL harus diawali dengan membuat *database*  yang berfungsi untuk menyimpan semua elemen tabel yang ada termasuk relasi-relasinya.

**MODUL III  
MANIPULASI TABEL**

* 1. **TUJUAN**

1. Mahasiswa dapat memasukkan data.
2. Mahasiswa dapat menghapus data.
3. Mahasiswa dapat meng-*update* data.
4. Mahasiswa dapat melakukan indeksing terhadap suatu tabel.
   1. **LANDASAN TEORI**
5. ***Data Manipulation Language***

Terdapat beberapa ragam dalam perintah SQL, salah satunya yaitu DML. DML atau *Data Manipulation Language* merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data atau *record* dalam tabel. Perintah SQL yang termasuk dalam DML antara lain yaitu SELECT, INSERT, UPDATE, DELETE [2].

1. **Memasukkan Data**

Perintah yang digunakan untuk memasukkan atau menyisipkan data ke dalam tabel adalah perintah INSERT. Terdapat dua cara memasukkan data ke dalam tabel, yaitu:

1. Dengan mengetahui urutan nama kolom (urutan data *default*)

Memasukkan data secara berturut-turut dengan memastikan urutan nilai-nilai yang dimasukkan dalam urutan yang sama seperti urutan kolom atau *field* pada tabel. Bentuk umum perintah dari cara dengan mengetahui urutan nama kolom yaitu sebagai berikut.

|  |
| --- |
| INSERT INTO nama\_tabel VALUES (value1, value2, … ); |

Contoh:

|  |
| --- |
| INSERT INTO mahasiswa VALUES (101,’Bayu Sasmita’,’Mataram’); |

1. Tanpa mengetahui urutan nama kolom

Bentuk umum perintah untuk memasukkan data tanpa mengetahui urutan nama kolom yaitu sebagai berikut:

|  |
| --- |
| INSERT INTO namaTabel (kolom1,kolom2,…) VALUES (value1, value2,…); |

Dimana:

1. “nama\_tabel” adalah nama tabel yang akan mengalami penambahan data atau *record*.
2. “kolom1, kolom2, …” adalah nama-nama kolom yang akan diisi nilai.
3. “value1, value2, …” adalah nilai-nilai yang akan mengisi kolom tersebut [2].

Contoh:

|  |
| --- |
| INSERT INTO mahasiswa (NIM,nama,alamat)  VALUES (101,’Bayu Sasmita’,’Mataram’); |

1. **Menghapus Data**

Perintah yang digunakan untuk menghapus *record* atau baris dari suatu tabel yaitu “DELETE”. Bentuk umum perintah SQL untuk menghapus data atau *record* adalah sebagai berikut:

|  |
| --- |
| DELETE FROM nama\_tabel WHERE kondisi; |

Dimana:

1. “DELETE FROM” adalah perintah dasar untuk menghapus suatu data dari tabel.
2. “nama\_tabel” adalah nama tabel yang akan dihapus datanya.
3. “WHERE kondisi” adalah klausa untuk menentukan data atau *record* mana yang akan dihapus. Klausa “WHERE” boleh ada, boleh juga tidak. Namun, jika “WHERE” tidak ditambahkan, maka semua *record* dalam tabel akan terhapus.

Contoh:

|  |
| --- |
| DELETE FROM mahasiswa WHERE nim=”101”; |

MySQL tidak akan menanyakan konfirmasi ulang untuk menghapus atau membatalkan penghapusan data. Maka dari itu, pastikan bahwa data benar-benar akan dihapus dan data yang telah dihapus tidak dapat dikembalikan lagi [1].

1. **Meng-*update* Data**

Perintah yang digunakan untuk meng-*update* data atau *record* dari suatu tabel yaitu dengan menggunakan perintah UPDATE. Bentuk umum dari perintah UPDATE adalah sebagai berikut:

|  |
| --- |
| UPDATE nama\_tabel SET nama\_kolom=nilai\_baru WHERE kondisi; |

Dimana :

1. “nama\_kolom=nilai\_baru” adalah ekspresi pemberian nilai baru untuk suatu kolom yang akan diubah atau di-*update*.
2. “nama\_tabel” merupakan nama tabel tempat dimana kolom yang akan diubah atau di-*update* berada.
3. “kondisi” merupakan kriteria untuk pemilihan *record* atau data yang akan diubah. Jika klausa WHERE tidak digunakan maka perubahan akan dilakukan pada semua *record* atau data dalam tabel.

Contoh:

1. Mengubah sebuah kolom untuk data baris tertentu.

|  |
| --- |
| UPDATE mahasiswa SET nama=’Achmad Fauzan’ WHERE nim=’101’; |

1. Merubah beberapa kolom sekaligus.

|  |
| --- |
| UPDATE mahasiswa SET nama=’Achmad Fauzan’, ‘alamat=’Narmada’ WHERE nim=’101’; |

1. Mengubah beberapa baris sekaligus.

|  |
| --- |
| UPDATE mahasiswa SET alamat=’Mataram’ WHERE nim=’101’ OR nim=’102’; |

1. Mengubah semua data yang berada pada kolom tertentu.

|  |
| --- |
| UPDATE mahasiswa SET prodi=’Informatika’; |

1. ***Indexing* Data**

*Indexing* data dibuat pada suatu tabel yang sudah ada yang memiliki kegunaan untuk mempercepat pencarian terhadap suatu set data dengan kondisi tertentu. Indeks juga memiliki kegunaan untuk pengujian data unik. Indeks otomatis dibuatkan saat suatu kolom dianggap sebagai *primary key* atau *unique* [2]. Berikut merupakan bentuk pernyataan indeks:

1. Indeks Unik

Indeks unik berarti bahwa dua baris tidak dapat mempunyai nilai indeks yang sama. Berikut adalah bentuk pernyataan indeks unik.

|  |
| --- |
| CREATE UNIQUE INDEX nama\_indeks ON nama\_tabel (nama\_kolom); |

Dimana, “nama\_kolom” adalah kolom yang akan diberikan indeks.

Contoh:

|  |
| --- |
| CREATE UNIQUE INDEX pelanggan ON pelanggan (NO\_PELANGGAN); |

1. Indeks Sederhana

Jika kata *unique* dihilangkan maka nilai yang sama diperbolehkan.

|  |
| --- |
| CREATE INDEX nama\_indeks ON nama\_tabel (nama\_kolom); |

Contoh:

|  |
| --- |
| CREATE INDEX alamatmhs ON mahasiswa (alamat); |

1. Melihat Indeks

Indeks dapat dilihat menggunakan *query* sebagai berikut:

|  |
| --- |
| SHOW INDEX FROM nama\_tabel; |

1. Menghapus Indeks

Bentuk perintah untuk menghapus indeks yaitu sebagai berikut:

|  |
| --- |
| DROP INDEX nama\_indeks ON nama\_tabel; |

* 1. **KASUS KELOMPOK**

Sekolah Dasar Internasional Wakanda (SBIW) akan merancang sebuah sistem informasi. Basis data yang dibentuk berdasarkan data-data berbagai entitas. Guru akan didata nama, kontak dan alamat sehingga terbentuk NUPTK. Lalu guru dibedakan menjadi guru honorer dan PNS yang memiliki NIP. Semua guru dapat mengajar mata pelajaran, dan guru pun menguasai tidak hanya satu saja mata pelajaran, namun terkadang mereka harus mengajar mata pelajaran yang tidak mereka pelajari, sehingga mata pelajaran dapat dikuasai dan diajar oleh banyak guru. Mata pelajaran hanya didata namanya saja, sehingga dibentuk kode mata pelajaran. Setiap mata pelajaran yang diajar guru akan didata pegisian kelas mana yang diajar. Sehingga guru dapat mengajar mata pelajaran yang sama di kelas yang berbeda, begitupun dikelas yang sama dapat mengajar mata pelajaran lain. Kelaspun harus di data nama kelasnya sehinggaa terbentuk kode kelas. Sebuah kelas dapat diwalikan oleh seorang guru non-honorer. Sebuah kelaspun memiliki banyak siswa, dimana dari mereka terdapat satu yang menjadi ketua kelas. Setiap siswa akan didata nama, alamat, nama wali, kontak wali, dan rangking mereka hingga dibentuk NIS. Setiap siswa dapat mempelajari banyak mata pelajaran, begitu pula sebaliknya setiap mata pelajaran dapat dipelajari banyak siswa. Setiap kegiatan belajar mengajar akan dilakukan pengambilan nilai.

|  |
| --- |
|  |

**Gambar 3.1** Rancangan ERD

* 1. **KASUS MODUL** 
     1. Meng-*insert* melalui dua cara yaitu tunggal dan jamak untuk masing-masing tabel minimal 10 data
     2. Meng-*update* data masing-masing tabel dengan kondisi *primary key* jika ada atau dengan satu kondisi jika tidak ada *primary key* dan dengan dua atau lebih kondisi
     3. Menghapus suatu data di masing-masing tabel dengan kondisi *primary key* jika ada, jika tidak ada *primary key* gunakan satu kondisi dan juga hapus suatu data dengan dua atau lebih kondisi.
  2. **ANALISIS KASUS**

Berdasarkan *database* “Sistem Informasi Perpustakaan” dilakukan manipulasi tabel yang meliputi “INSERT”, “UPDATE” dan “DELETE” sebagai berikut:

1. **Tabel Guru**
   * + 1. ***Insert***

|  |
| --- |
|  |

**Gambar 3.2** *insert* data tunggal

*Query* pada **Gambar 3.2** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas, kolom “alamat\_guru”, “kontak\_guru”, “nama\_guru” dan “NUPTK” yang diisi dengan *record* secara berurut “Cakra”, “0814xx”, “Susi Nuryanti” dan “182001”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.3** *insert* data jamak

*Query* pada **Gambar 3.3** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 9 nilai sehingga diperoleh 10 data setiap kolom.

* + - 1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.4** *update* dengan *primary key*

Pada **Gambar 3.4** perintah *update* menggunakan satu kondisi dimana data yang akan diubah adalah “alamat\_guru” menjadi “Ampenan” ketika “NUPTK” sama dengan “182001”.

|  |
| --- |
|  |

**Gambar 3.5** *update* dengan 2 kondisi

Pada **Gambar 3.5** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “alamat\_guru” menjadi “Gunung Sari” ketika kondisi “NUPTK” sama dengan “182005” dan “kontak\_guru” sama dengan “893”. Melalui *query update* pada tabel tersebut, data yang memiliki hubungan yang sesuai dengan kondisi yang telah ada pada *query* tersebut akan otomatis berubah mengikuti perintah *set*.

* + - 1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah/dihapus.

|  |
| --- |
|  |

**Gambar 3.6** *delete* dengan *primary key*

Pada **Gambar 3.6** perintah *delete* dilakukan pada tabel “Guru” dengan kondisi “NUPTK=182001”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.7** *delete* dengan dua kondisi

Pada **Gambar 3.7** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Guru” dengan kondisi “NUPTK=182001” atau “kontak\_guru=886”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel Mata\_Pelajaran**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.8** *insert* data tunggal

*Query* pada **Gambar 3.8** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “kode\_mapel” dan “nama\_mapel” yang diisi dengan *record* secara berurut “a01” dan “Pendidikan Agama”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.9** *insert* data jamak

*Query* pada **Gambar 3.9** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 9 nilai sehingga diperoleh 10 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.10** *update* dengan *primary key*

Pada **Gambar 3.10** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “nama\_mapel” menjadi “PpKN” ketika kondisi “kode\_mapel” sama dengan “p01”.

|  |
| --- |
|  |

**Gambar 3.11** *update* dengan dua kondisi

Pada **Gambar 3.11** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “nama\_mapel” menjadi “SBD” ketika kondisi “kode\_mapel” sama dengan “s01” dan “nama\_mapel” sama dengan “Seni Budaya dan kete”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.12** *delete* dengan *primary key*

Pada **Gambar 3.12** perintah *delete* dilakukan pada tabel “Mata\_Pelajaran” dengan kondisi “kode\_mapel=a01”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.13** *delete* dengan dua kondisi

Pada **Gambar 3.13** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Mata\_Pelajaran” dengan kondisi “kode\_mapel=b01” dan “nama\_mapel= ‘Bahasa Indonesia’”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel PNS**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.14** *insert* data tunggal

*Query* pada **Gambar 3.14** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “NIP” dan “NUPTK” yang diisi dengan *record* secara berurut “1 50058 65” dan “182003”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.15** *insert* data jamak

*Query* pada **Gambar 3.15** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 7 nilai sehingga diperoleh 8 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.16** *update* dengan *primary key*

Pada **Gambar 3.16** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NUPTK” menjadi “182001” ketika kondisi “NUPTK” sama dengan “182002”.

|  |
| --- |
|  |

**Gambar 3.17** *update* dengan dua kondisi

Pada **Gambar 3.17** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “NUPTK” menjadi “182002” ketika kondisi “NUPTK” sama dengan “182003” dan “NIP” sama dengan “1 50085 65”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.18** *delete* dengan *primary key*

Pada **Gambar 3.18** perintah *delete* dilakukan pada tabel “PNS” dengan kondisi “NIP=’1 54114 90’”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.19** *delete* dengan dua kondisi

Pada **Gambar 3.19** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “PNS” dengan kondisi “NIP=’1 50085 65’” dan “NUPTK=182002”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel Siswa**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.20** *insert* data tunggal

*Query* pada **Gambar 3.20** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “NIS”, “nama\_siswa”, “alamat\_siswa” “nama\_wali”, “kontak\_wali”, “rank” dan “kode\_kelas” yang diisi dengan *record* secara berurut “2705001”, “Budi Setiawan”, “Ampenan” “Renaldi”, “876”, “1” dan “3a”. Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.21** *insert* data jamak

*Query* pada **Gambar 3.21** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 11 nilai sehingga diperoleh 12 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.22** *update* dengan *primary key*

Pada **Gambar 3.22** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NIS” menjadi “2705009” ketika kondisi “NIS” sama dengan “2705001”.

|  |
| --- |
|  |

**Gambar 3.23** *update* dengan dua kondisi

Pada **Gambar 3.23** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “NIS” menjadi “2705004” ketika kondisi “NIS” sama dengan “2705035” dan “rank” sama dengan “9”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.24** *delete* dengan *primary key*

Pada **Gambar 3.24** perintah *delete* dilakukan pada tabel “Siswa” dengan kondisi “NIS=2705064”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.25** *delete* dengan dua kondisi

Pada **Gambar 3.25** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Siswa” dengan kondisi “NIS=2705076” dan “rank=2”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel Menguasai**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.26** *insert* data tunggal

*Query* pada **Gambar 3.26** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “NUPTK” dan “kode\_mapel” yang diisi dengan *record* secara berurut “182001” dan “a01”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.27** *insert* data jamak

*Query* pada **Gambar 3.27** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 9 nilai sehingga diperoleh 10 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.28** *update* dengan *primary key*

Pada **Gambar 3.28** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NUPTK” menjadi “182005” ketika kondisi “NUPTK” sama dengan “182001”.

|  |
| --- |
|  |

**Gambar 3.29** *update* dengan dua kondisi

Pada **Gambar 3.29** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “NUPTK” menjadi “182001” ketika kondisi “NUPTK” sama dengan “182005” dan “kode\_mapel” sama dengan “a01”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.30** *delete* dengan *primary key*

Pada **Gambar 3.30** perintah *delete* dilakukan pada tabel “Menguasai” dengan kondisi “NUPTK=182007”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.31** *delete* dengan dua kondisi

Pada **Gambar 3.31** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Menguasai” dengan kondisi “NUPTK=182007” dan “kode\_mapel=p02”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel Mengajar**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.32** *insert* data tunggal

*Query* pada **Gambar 3.32** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “NUPTK” dan “kode\_mapel” yang diisi dengan *record* secara berurut “182001” dan “a01”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.33** *insert* data jamak

*Query* pada **Gambar 3.33** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 9 nilai sehingga diperoleh 10 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.34** *update* dengan *primary key*

Pada **Gambar 3.34** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NUPTK” menjadi “182004” ketika kondisi “NUPTK” sama dengan “182008”.

|  |
| --- |
|  |

**Gambar 3.35** *update* dengan dua kondisi

Pada **Gambar 3.35** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “NUPTK” menjadi “182008” ketika kondisi “NUPTK” sama dengan “182004” dan “kode\_mapel” sama dengan “p01”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.36** *delete* dengan *primary key*

Pada **Gambar 3.36** perintah *delete* dilakukan pada tabel “Mengajar” dengan kondisi “NUPTK=182005”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.37** *delete* dengan dua kondisi.

Pada **Gambar 3.37** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Mengajar” dengan kondisi “NUPTK=182001” dan “kode\_mapel=‘a01’”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel Honorer**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.38** *insert* data tunggal

*Query* pada **Gambar 3.38** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “NUPTK”, yang diisi dengan *record* secara berurut “182006”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.39** *insert* data jamak

*Query* pada **Gambar 3.39** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 2 nilai sehingga diperoleh 3 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.40** *update* dengan *primary key*

Pada **Gambar 3.40** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NUPTK” menjadi “182006” ketika kondisi “NUPTK” sama dengan “182005”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.41** *delete* dengan *primary key*

Pada **Gambar 3.41** perintah *delete* dilakukan pada tabel “Honorer” dengan kondisi “NUPTK=182006”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

1. **Tabel Kelas**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.42** *insert* data tunggal

*Query* pada **Gambar 3.42** menggunakan perintah penambahan data dengan memilih kolom-kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “kode\_kelas”, “nama\_kelas”, “NIS\_ketua” dan “NUPTK\_walikelas” yang diisi dengan *record* secara berurut “1”, “satu”, “2705101” dan “182001”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.43** *insert* data jamak

*Query* pada **Gambar 3.43** tidak menggunakan penentuan kolom dimana semua kolom harus diisi dengan *record* sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 9 nilai sehingga diperoleh 10 data setiap kolom

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.44** *update* dengan *primary key*

Pada **Gambar 3.44** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “kode\_kelas” menjadi “18” ketika kondisi “kode\_kelas” sama dengan “1A”.

|  |
| --- |
|  |

**Gambar 3.45** *update* dengan dua kondisi

Pada **Gambar 3.45** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “kode\_kelas” menjadi “2B” ketika kondisi “kode\_kelas” sama dengan “2A” dan “nama\_kelas” sama dengan “dua A”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.46** *delete* dengan *primary key*

Pada **Gambar 3.46** perintah *delete* dilakukan pada tabel “Kelas” dengan kondisi “kode\_kelas=’1B’”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.47** *delete* dengan dua kondisi

Pada **Gambar 3.47** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Kelas” dengan kondisi “kode\_Kelas=’2B’” dan “nama\_kelas=’2A’”, dimana *query* tersebut menggunakan “and” berarti data terhapus ketika kedua kondisi terpenuhi.

1. **Tabel Menghadiri**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.48** *insert* data tunggal

*Query* pada **Gambar 3.48** menggunakan perintah penambahan data dengan memilih kolom kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “NUPTK”, “kode\_mapel” dan “kode\_kelas” yang diisi dengan *record* secara berurut “182001”, “a01” dan “1”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.49** *insert* data jamak

*Query* pada **Gambar 3.49** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 11 nilai sehingga diperoleh 12 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.50** *update* dengan *primary key*

Pada **Gambar 3.50** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NUPTK” menjadi “182006” ketika kondisi “NUPTK” sama dengan “182010”.

|  |
| --- |
|  |

**Gambar 3.51** *update* dengan dua kondisi

Pada **Gambar 3.51** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “NUPTK” menjadi “182010” ketika kondisi “NUPTK” sama dengan “182006” dan “kode\_kelas” sama dengan “6B”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.52** *delete* dengan *primary key*

Pada **Gambar 3.52** perintah *delete* dilakukan pada tabel “Menghadiri” dengan kondisi “NUPTK=182010”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.53** *delete* dengan dua kondisi

Pada **Gambar 3.53** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Menghadiri” dengan kondisi “NUPTK=182005” dan “kode\_mapel=s01”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

1. **Tabel Mempelajari**
2. ***Insert***

|  |
| --- |
|  |

**Gambar 3.54** *insert* data tunggal

*Query* pada **Gambar 3.54** menggunakan perintah penambahan data dengan memilih kolom kolom yang diinginkan. Dari gambar diatas ditentukan bahwa kolom “nilai”, “kode\_mapel” dan “NIS” yang diisi dengan *record* secara berurut “90”, “a01” dan “2705002”.

Dimasukkan nama kolom, beserta nilai pada setiap baris dimana *query*-nya seperti berikut:

|  |
| --- |
|  |

**Gambar 3.55** *insert* data jamak

*Query* pada **Gambar 3.55** menggunakan perintah penambahan data tanpa menggunakan pemilihan kolom dimana semua kolom harus diisi dengan *record*. sehingga apabila terjadi bahwa nilai kolom kurang maka akan terjadi *error*. Setiap kolom diisi dengan 11 nilai sehingga diperoleh 12 data setiap kolom.

1. ***Update***

*Update* digunakan untuk mengubah nilai suatu data, agar data yang dituju sesuai dengan yang diinginkan maka harus diberikan sebuah kondisi.

|  |
| --- |
|  |

**Gambar 3.56** *update* dengan *primary key*

Pada **Gambar 3.56** perintah *update* menggunakan satu kondisi dimana data yang diubah adalah “NIS” menjadi “2705010” ketika kondisi “NIS” sama dengan “2705057”.

|  |
| --- |
|  |

**Gambar 3.57** *update* dengan dua kondisi

Pada **Gambar 3.57** perintah *update* menggunakan dua kondisi dimana data yang diubah adalah “NIS” menjadi “2705057” ketika kondisi “NIS” sama dengan “275010” dan “kode\_mapel” sama dengan “p03”.

1. ***Delete***

*Delete* digunakan untuk menghapus data. Agar sesuai dengan data yang dituju, maka harus disertakan setidaknya satu kodisi. Dimana apabila kondisi tidak terpenuhi maka tidak ada *record* yang berubah.

|  |
| --- |
|  |

**Gambar 3.58** *delete* dengan *primary key*

Pada **Gambar 3.58** perintah *delete* dilakukan pada tabel “Mempelajari” dengan kondisi “NIS=2705096”. Data pada tabel akan otomatis terhapus, ketika ada data yang memenuhi kondisi yang ada pada *query delete*, dikarenakan penghapusan menggunakan *primary key*, kondisi yang digunakan cukup satu saja dikarenakan nilai dari setiap *primary key* memiliki keunikan atau dalam artian lain tidak ada *primary key* yang memiliki nilai yang sama pada sebuah tabel.

|  |
| --- |
|  |

**Gambar 3.59** *delete* dengan dua kondisi

Pada **Gambar 3.59** merupakan perintah yang digunakanuntuk menghapus data dengan menggunakan 2 kondisi untuk melakukan penghapusan. Perintah yang digunakan untuk melakukan *delete*/menghapus data yaitu “delete from nama\_tabel where kondisi1 and kondisi2;”. Perintah *delete* dilakukan pada tabel “Mempelajari” dengan kondisi “NIS=2705101” dan “kode\_mapel=’a01’;”, dimana *query* tersebut menggunakan “and” yang artinya *delete* terjadi apabila kedua kondisi terpenuhi.

* 1. **KESIMPULAN**

Berdasarkan praktikum yang dilakukan, dapat diambil beberapa kesimpulan yaitu sebagai berikut:

1. Memasukkan data pada tabel dilakukan dengan menentukan nama tabel yang akan dimasukkan datanya dan nilai dari data yang akan dimasukkan dengan perintah “INSERT”.
2. Menghapus data pada tabel dilakukan dengan menentukan nama tabel tempat data yang akan dihapus dan memilih data yang akan dihapus dengan memberikan kondisi data/record yang ingin dihapus.
3. Meng-*update* data dilakukan dengan menentukan nama tabel yang akan di-*update*, kemudian menentukan nilai dari data yang akan di-*update*. Sesuai dengan kondisi yang dideklarasikan (mendeklarasikan salah satu nilai data tersebut pada salah satu kolom).
4. Membuat *index* dari suatu tabel dilakukan dengan membuat nama *index* pada suatu tabel yang dipilih, kemudian mendeklarasikan kolom-kolom pada tabel tersebut yang akan dibuatkan *index*.

**MODUL IV  
QUERY TABEL TUNGGAL DAN BEBERAPA TABEL**

* 1. **TUJUAN**
     1. Mahasiswa dapat menampilkan data tabel tunggal baik dengan kondisi maupun tidak.
     2. Mahasiswa dapat menampilkan tabel dalam urutan tertentu.
     3. Mahasiswa dapat mengelompokkan data pada tabel.
     4. Mahasiswa dapat menampilkan satu atau lebih data dari beberapa tabel sekaligus.
     5. Mahasiswa dapat menampilkan data pada tabel berdasarkan data beberapa tabel lain.
  2. **LANDASAN TEORI**

1. ***Query* Untuk Menampilkan Data pada Tabel Tunggal**

*Query* adalah perintah–perintah untuk mengakses data pada sistem basis data. *Query* merupakan bahasa untuk manipulasi basis data yang telah distandarkan dan lebih dikenal dengan nama SQL (*Standard Query Language*). Manfaat *query* dalam basis data adalah:

1. Menampilkan data–data tertentu pada suatu tabel.
2. Menampilkan data–data dari dua tabel atau lebih dengan kondisi tertentu.
3. Dapat melakukan operasi perhitungan.

Berikut adalah beberapa penggunaan *query*:

* + - 1. ***Query* untuk menampilkan data pada tabel tunggal**

Perintah – perintah yang biasanya digunakan untuk menampilkan data pada tabel tunggal anatara lain yaitu:

* + - * 1. Perintah “SELECT”

“SELECT” adalah pernyataan yang digunakan untuk mengambil data tertentu dalam tabel [1]. Berikut ini adalah bentuk sederhana dari pernyataan “SELECT”:

|  |
| --- |
| SELECT \* FROM nama\_tabel; |

*Query* yang digunakan untuk memilih semua kolom untuk suatu tabel dimana klausa “SELECT” menentukan kolom yang ditampilkan pada hasil. Klausa “FROM” akan menentukan *cross product* serta “nama\_tabel” adalah nama dari tabel yang semua isi kolomnya akan ditampilkan.

* + - * 1. Perintah “DISTINCT”

“DISTINCT” merupakan perintah yang digunakan untuk memilih hanya nilai yang berbeda pada suatu tabel. Bentuk *query* pernyataan “DISTINCT” yaitu:

|  |
| --- |
| SELECT DISTINCT nama\_kolom FROM nama\_tabel; |

Pada kunci “DISTINCT” bersifat pilihan yang dihitung sebagai jawaban dari *query* ini tidak memiliki duplikat, yaitu memiliki 2 salinan pada baris yang sama [2].

* + - * 1. Fungsi *Agregat*

Fungsi *agregat* dilakukan untuk perhitungan terhadap hasil *query* yang berupa numerik dari pernyataan “SELECT”. Fungsi ini pada dasarnya menyimpulkan hasil dari *query* pada kolom tertentu. Fungsi – fungsi agregat yaitu:

“MIN”, mengembalikan nilai terkecil pada suatu kolom.

|  |
| --- |
| SELECT MIN (nama\_kolom) FROM nama\_tabel; |

“MAX”, mengembalikan nilai terkecil pada suatu kolom.

|  |
| --- |
| SELECT MAX (nama\_kolom) FROM nama\_tabel; |

“SUM”, mengembalikan jumlah nilai numerik pada suatu kolom.

|  |
| --- |
| SELECT SUM (nama\_kolom) FROM nama\_tabel; |

“AVG”, mengembalikan nilai rata – rata pada suatu kolom.

|  |
| --- |
| SELECT AVG (nama\_kolom) FROM nama\_tabel; |

“COUNT (\*)”, mengembalikan jumlah baris pada suatu tabel.

|  |
| --- |
| SELECT COUNT(\*) FROM nama\_tabel; |

“COUNT”, mengembalikan jumlah total nilai pada suatu kolom.

|  |
| --- |
| SELECT COUNT (nama\_kolom) FROM nama\_tabel; |

* + - * 1. “ALIAS” atau “AS”

Pernyataan “ALIAS” pada SQL digunakan untuk mengganti nama kolom atau nama tabel. *Query* yang digunakan untuk mengganti nama kolom yaitu:

|  |
| --- |
| SELECT nama\_kolom AS nama\_alias\_kolom FROM nama\_tabel; |

*Query* yang digunakan untuk mengganti nama tabel yaitu:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel AS nama\_alias\_tabel; |

* + - 1. ***Query* Untuk Menampilkan Data pada Tabel Tunggal dengan Kondisi**

Untuk pencarian sebuah atau sekelompok data dari *database* diperlukan satu atau lebih kondisi. Artinya, bila diberikan suatu kondisi pada perintah SQL, maka hanya data-data tertentu yang akan ditampilkan. Kata kunci untuk pemberian kondisi adalah “WHERE” yang dituliskan setelah penulisan nama tabel pada perintah SQL. *Query* yang digunakan yaitu[1]:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE kondisi; |

Macam-macam kondisi atau operator dalam perintah SQL, yaitu:

1. **Operator logika**
2. = : sama dengan
3. != : tidak sama dengan.
4. .< : lebih kecil dari.
5. >= : lebih besar sama dengan.
6. <= : lebih kecil sama dengan.
7. ***Like***

Digunakan untuk menentukan pencarian berdasarkan pola tertentu pada suatu kolom. Tanda “%” dapat digunakan untuk menentukan sembarang karakter, baik sebelum maupun sesudah pola.

1. Untuk menghasilkan data yang dimulai dengan satu huruf, “LIKE “A%””.
2. Untuk menghasilkan data yang diakhiri dengan satu huruf, “LIKE “%A””.
3. Untuk menghasilkan data yang mengandung satu huruf, “LIKE “&A%””.

*Query* “LIKE” adalah:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE nama\_kolom LIKE ‘pola’; |

1. **AND dan OR**

Digunakan untuk menggabungkan dua atau lebih kondisi pada klausa WHERE. Operator “AND” menampilkan data jika semua kondisi yang diminta bernilai benar. Operator “OR” menampilkan data jika kondisi ada yang bernilai benar *Query*-nya yaitu:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE kondisi1 AND kondisi2 |

1. **Operasi logika**
2. “IN” : berfungsi untuk menampilkan data yang berada dalam kriteria yang dipilih.
3. “NOT IN” : berfungsi untuk menampilkan data selain dari kriteria yang ditentukan.
4. “BETWEEN” : berfungsi untuk menampilkan data di antara dua nilai.
5. “NOT BETWEEN” : berfungsi untuk menampilkan data yang tidak berada di antara dua nilai.

*Query* untuk operator “IN”, yaitu:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE nama\_kolom IN(nilai1, nilai2); |

*Query* untuk operator BETWEEN, yaitu:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE nama\_kolom BETWEEN nilai1 AND nilai2; |

* + - 1. ***Query* Untuk Mengurutkan Data pada Tabel Tunggal**

Klausa “ORDER BY” digunakan untuk mengurutkan hasil seleksi menampilkan data pada suatu tabel. *Query* yang digunakan untuk mengurutkan hasil seleksi :

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE kondisi ORDER BY nama\_kolom [desc/asc]; |

Opsi “DESC” digunakan untuk menampilkan data berdasarkan urutan data dari yang terbesar ke data yang terkecil. Opsi “ASC” digunakan untuk mengurutkan data dari yang terkecil ke yang terbesar.

* + - 1. ***Query* Untuk Mengelompokkan Data pada Tabel Tunggal**

Klausa “GROUP BY” digunakan untuk mengelompokkan data berdasarkan kolom tertentu. *Query* yang digunakan untuk mengelompokkan data adalah [2]:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel GROUP BY [kolom\_group]; |

1. ***Sub Query* pada Beberapa Tabel**

*Sub query* ada yang memiliki *query* lain yang tersembunyi di dalamnya. *Sub* *query* biasanya muncul pada klausa “WHERE” dari *query*. *Sub* *query* pada beberapa tabel digunakan ketika nilai pada basis data untuk ditampilkan belum diketahui pasti. Berikut adalah bentuk umum *sub* *query* pada beberapa tabel:

|  |
| --- |
| SELECT nama\_kolom FROM nama\_tabel WHERE operator (SELECT nama\_kolom FROM nama\_tabel); |

Adapun jenis-jenis *sub* *query* antara lain yaitu:

1. ***Single row* *sub* *query***

*Single row* *sub* *query* memberikan hasil hanya satu baris pada bagian *sub* *query*. Operator yang dapat digunakan yaitu operator pembanding yaitu =,>,<.>=,<= atau <>. Serta operator logika “AND” dan “OR”. Selain itu, dapat menggunakan *group function* seperti “AVG”, “MAX” dan “MIN”. bentuk umum penulisan *single row* *sub* *query* yaitu:

|  |
| --- |
| SELECT [nama\_kolom1, …] FROM nama\_tabel WHERE nama\_kolom\_kondisi operator\_pembanding (SELECT nama\_kolom FROM nama\_tabel WHERE nama\_kolom\_kondisi operator\_nilai); |

1. ***Multiple row* *sub* *query***

Adalah *sub* *query* yang menghasilkan lebih dari satu baris. Operator yang digunakan yaitu “IN”, “ANY” dan “ALL”.

1. Operator “IN”

“IN” adalah klausa yang menyatakan bahwa *query* utama akan menghasilkan nilai yang sama pada *sub* *query*[2]. *Query* operator “IN” yaitu:

|  |
| --- |
| SELECT [nama\_kolom1. …] FROM nama\_tabel WHERE nama\_kolom\_kondisi IN (SELECT nama\_tabel, nama\_kolom FROM namatabel1, nama\_kolom operator nama\_tabel2, nama\_kolom2); |

1. Operator “ANY”

“ANY” adalah klausa yang menggunakan operator pembanding aritmatika selain “=”. Hal tersebut disebabkan apabila menggunakan operator relasi “=”, maka fungsi operator “ANY” sama dengan operator “IN”, sehingga kondisi tersebut tidak dianjurkan[1]. *Query* operator “ANY” yaitu:

|  |
| --- |
| SELECT [nama\_kolom1. …] FROM nama\_tabel WHERE nama\_kolom\_kondisi ANY (SELECT nama\_tabel, nama\_kolom FROM namatabel1, nama\_kolom operator nama\_tabel2, nama\_kolom2); |

1. Oparator “ALL”

“ALL” adalah klausa yang digunakan untuk menghasilkan baris yang membuat perbandingan menghasilkan *true*. *Query* operator “ALL” yaitu:

|  |
| --- |
| SELECT [nama\_kolom1. …] FROM nama\_tabel WHERE nama\_kolom\_kondisi ALL (SELECT nama\_kolom FROM nama\_tabel WHERE kondisi); |

1. **Multi Column Sub *Query***

*Multi column* *sub* *query* adalah *sub* *query* yang menghasilkan lebih dari satu kolom. Jumlah dan posisi kolom dari *outer* *query* yang akan dibandingkan harus sama dengan jumlah dan posisi kolom pada *sub* *query*. Penghubung yang biasa digunakan adalah “IN”. *Query* multi *column* *sub* *query* yaitu:

|  |
| --- |
| SELECT [nama\_kolom1. …] FROM nama\_tabel WHERE nama\_kolom\_kondisi ALL (SELECT nama\_kolom FROM nama\_tabel GROUP BY nama\_kolom); |

1. ***Sub Query* dengan Kondisi**

*Sub* *query* pada beberapa tabel dengan kondisi terdiri dari beberapa jenis seperti di bawah ini:

1. **Klausa** “**HAVING**”

Klausa “HAVING” digunakan untuk menyaring data berdasarkan fungsi kelompok. Bentuk umum klausa “HAVING” yaitu:

|  |
| --- |
| SELECT [DISTINCT] [nama\_kolom1, nama\_kolom, ..] FROM nama\_tabel WHERE kondisi GROUP BY kondisi2 HAVING fungsi-agregasi (nama\_kolom); |

1. **Klausa** “**WHERE**”

Klausa “WHERE” disebut juga klausa bersyarat yang digunakan untuk menyaring hasil serta menentukan kriteria seleksi untuk memilih tatanan ang diperlukan dari tabel[1]. *Query* klausa “WHERE” yaitu:

|  |
| --- |
| SELECT [nama\_kolom1, nama\_kolom2, …] FROM [nama\_tabel1, nama\_tabel2, …] WHERE kondisi [AND/OR] kondisi2; |

* 1. **KASUS KELOMPOK**

Berikut adalah rancangan diagram ER yang akan diimplementasikan ke dalam struktur tabel *database* dengan menggunakan SQL:

|  |
| --- |
|  |

**Gambar 4.1** Rangkaian ERD setelah normalisasi

* 1. **KASUS MODUL**

1. Lakukan *query* tabel tunggal dengan ketentuan-ketentuan berikut pada basis data yang sudah dibuat :
2. Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.
3. LIKE dengan karakter *wildcard*
4. NOT LIKE dengan karakter *wildcard*
5. NULL
6. NOT NULL
7. GROUP BY
8. ORDER BY
9. Operator Logika
10. IN
11. BETWEEN
12. NOT BETWEEN
13. SUBSTRING
14. COUNT
15. Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.

Minimal 2 tabel untuk satu *query*

1. Memuat *Sub* *query* pada 3 tabel (minimal menampilkan 2 *field*)
2. Memuat *Sub* *query* pada 4 tabel (minimal menampilkan 2 *field*)
3. Membuat *sub* *query* sebelum kondisi atau setelah perintah SELECT (minimal 3 *query*)
   1. **ANALISIS KASUS**

Berdasarkan *database* “Sistem Informasi Sekolah Wakanda” dilakukan manipulasi tabel dengan *query* pada tabel tunggal dan beberapa tabel seperti berikut:

* + 1. **Lakukan *query* tabel tunggal dengan ketentuan yang disediakan pada basis data yang sudah dibuat.**
       - 1. **Tabel Guru**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.2** Tampilan data terakhir dari tabel Guru

Pada *query* **Gambar 4.2** berfungsi untuk menampilkan semua data terakhir pada tabel “Guru” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.3** Perintah *Like* pada tabel Guru

Pada *query* **Gambar 4.3** merupakan perintah “like” yang berfungsi untuk menampilkan seluruh data pada pada tabel “Guru” dengan kondisi yaitu “Nama\_guru” yang memiliki awalan huruf S.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.4** Perintah *Not Like* pada tabel Guru

Pada *query* **Gambar 4.4** merupakan perintah “not like” yang berfungsi untuk menampilkan seluruh data pada tabel “Guru” dengan kondisi data yaitu “nama\_guru” yang tidak memiliki awalan huruf S. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “not like ‘S%’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.5** Perintah *Null* pada tabel Guru

Pada *query* **Gambar 4.5** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan kondisi kolom “nama\_guru” bernilai “null” menggunakan perintah “is null”. Akan tetapi data yang ditampilkan tidak ada pada kolom “nama\_guru” dikarenakan tidak ada data yang bernilai “null”.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.6** Perintah *Not Null* pada tabel Guru

Pada *query* **Gambar 4.6** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan kondisi kolom “nama\_guru” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “nama\_guru” semua data akan ditampilkan selama bernilai “not null”.

1. ***Group by***

|  |
| --- |
|  |

**Gambar 4.7** Perintah *Group by* pada tabel Guru

Pada *query* **Gambar 4.7** berfungsi untuk melakukan pengelompokan pada data berdasarkan suatu kondisi tertentu. Sehingga pada tabel “Guru” dengan perintah “select” akan menampilkan semua kolom yang dikelompokkan berdasarkan data pada kolom “nama\_guru”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.8** Perintah *Order by* pada tabel Guru

Pada *query* **Gambar 4.8** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “nama\_guru” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.9** Perintah operator logika pada tabel Guru

Pada *query* **Gambar 4.9** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan perintah “select” dimana kondisi data pada kolom “NUPTK=’1923-4252-0982-1092’”, maka akan menampilkan semua data dengan nilai “NUPTK” sama dengan “1923-4252-0982-1092”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.10** Perintah *in* pada tabel Guru

Pada *query* **Gambar 4.10** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (Mataram)” artinya akan menampilkan data yang data “alamat\_guru” yang isi datanya “(Mataram)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.11** Perintah *between* pada tabel Guru

Pada *query* **Gambar 4.11** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan perintah “select” yang mana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “alamat\_guru” yang berada diantara nilai “’b’ and ‘m’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.12** Perintah *not between* pada tabel Guru

Pada *query* **Gambar 4.12** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “alamat\_guru” yang tidak berada diantara nilai “’c’ and ‘j’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.13** Perintah *substring* pada tabel Guru

Pada *query* **Gambar 4.13** berfungsi untuk menampilkan data pada tabel “Guru” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(alamat\_guru,1,4))” artinya data akan menampilkan *string* atau karakter dalam suatu kolom “alamat\_guru” dengan jumlah “4” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.14** Perintah *count* pada tabel Guru

Pada *query* **Gambar 4.14** berfungsi untuk menampilkan jumlah data pada tabel “Guru” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database***

|  |
| --- |
|  |

**Gambar 4.15** Tampilan pada tabel Guru

Pada *query* **Gambar 4.15** berfungsi untuk menampilkan semua data pada tabel “Guru” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Kelas**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.16** Tampilan data terakhir dari tabel Kelas

Pada *query* **Gambar 4.16** berfungsi untuk menampilkan semua data terakhir pada tabel “Kelas” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.17** Perintah *Like* pada tabel Kelas

Pada *query* **Gambar 4.17** berfungsi untuk menampilkan data pada tabel “Kelas” dengan kondisi data “kode\_kelas” yang memiliki awalan huruf A. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘A%’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.18** Perintah *Not Like* pada tabel Kelas

Pada *query* **Gambar 4.18** berfungsi untuk menampilkan data pada tabel “Kelas” dengan kondisi data “kode\_kelas” yang tidak memiliki akhiran huruf A. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘%A’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.19** Perintah *Null* pada tabel Kelas

Pada *query* **Gambar 4.19** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan kondisi kolom “kode\_kelas” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “kode\_kelas” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.20** Perintah *Not Null* pada tabel Kelas

Pada *query* **Gambar 4.20** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan kondisi kolom “nama\_kelas” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “nama\_kelas” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.21** Perintah *Group by* pada tabel Kelas

Pada *query* **Gambar 4.21** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “kode\_kelas”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.22** Perintah *Order by* pada tabel Kelas

Pada *query* **Gambar 4.22** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “kode\_kelas” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.23** Perintah operator logika pada tabel Kelas

Pada *query* **Gambar 4.23** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dimana kondisi data pada kolom “kode\_kelas=’4B’”, maka akan menampilkan semua data dengan nilai “kode\_kelas” sama dengan “4B”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.24** Perintah *in* pada tabel Kelas

Pada *query* **Gambar 4.24** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘3B’)” artinya akan menampilkan data yang data “kode\_kelas” yang isi datanya “(‘3B’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.25** Perintah *between* pada tabel Kelas

Pada *query* **Gambar 4.25** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_kelas” yang berada diantara nilai “’1’ and ‘2’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.26** Perintah *not between* pada tabel Kelas

Pada *query* **Gambar 4.26** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_kelas” yang tidak berada diantara nilai “’1’ and ‘2’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.27** Perintah *substring* pada tabel Kelas

Pada *query* **Gambar 4.27** berfungsi untuk menampilkan data pada tabel “Kelas” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(kode\_kelas,1,2))” artinya data akan menampilkan isi kolom “kode\_kelas” dengan jumlah “2” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.28** Perintah *count* pada tabel Kelas

Pada *query* **Gambar 4.28** berfungsi untuk menampilkan jumlah data pada tabel “Kelas” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.29** Tampilan pada tabel Kelas

Pada *query* **Gambar 4.29** berfungsi untuk menampilkan semua data pada tabel “Kelas” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Siswa**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.30** Tampilan data terakhir dari tabel Siswa

Pada *query* **Gambar 4.30** berfungsi untuk menampilkan semua data terakhir pada tabel “Siswa” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.31** Perintah *Like* pada tabel Siswa

Pada *query* **Gambar 4.31** berfungsi untuk menampilkan data pada tabel “Siswa” dengan kondisi data “nama\_siswa” yang memiliki awalan huruf S. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘S%’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.32** Perintah *Not Like* pada tabel Siswa

Pada *query* **Gambar 4.32** berfungsi untuk menampilkan data pada tabel “Siswa” dengan kondisi data “nama\_siswa” yang tidak memiliki awalan huruf S. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘S%’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.33** Perintah *Null* pada tabel Siswa

Pada *query* **Gambar 4.33** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan kondisi kolom “nama\_siswa” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “nama\_siswa” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.34** Perintah *Not Null* pada tabel Siswa

Pada *query* **Gambar 4.34** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan kondisi kolom “nama\_siswa” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “nama\_siswa” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.35** Perintah *Group by* pada tabel Siswa

Pada *query* **Gambar 4.35** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “nama\_siswa”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.36** Perintah *Order by* pada tabel Siswa

Pada *query* **Gambar 4.36** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “nama\_siswa” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.37** Perintah operator logika pada tabel Siswa

Pada *query* **Gambar 4.37** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dimana kondisi data pada kolom “rank=2”, maka akan menampilkan semua data dengan nilai “rank” sama dengan “2”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.38** Perintah *in* pada tabel Siswa

Pada *query* **Gambar 4.38** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘mataram’)” artinya akan menampilkan data yang data “alamat\_siswa” yang isi datanya “(‘mataram’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.39** Perintah *between* pada tabel Siswa

Pada *query* **Gambar 4.39** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “alamat\_siswa” yang berada diantara nilai “’a’ and ‘k’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.40** Perintah *not between* pada tabel Siswa

Pada *query* **Gambar 4.40** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “alamat\_siswa” yang tidak berada diantara nilai “’c’ and ‘j’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.41** Perintah *substring* pada tabel Siswa

Pada *query* **Gambar 4.41** berfungsi untuk menampilkan data pada tabel “Siswa” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(alamat\_siswa,1,5))” artinya data akan menampilkan isi kolom “alamat\_siswa” dengan jumlah “5” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.42** Perintah *count* pada tabel Siswa

Pada *query* **Gambar 4.42** berfungsi untuk menampilkan jumlah data pada tabel “Siswa” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.43** Tampilan pada tabel Siswa

Pada *query* **Gambar 4.43** berfungsi untuk menampilkan semua data pada tabel “Siswa” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Mata\_Pelajaran**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.44** Tampilan data terakhir dari tabel Mata\_Pelajaran

Pada *query* **Gambar 4.44** berfungsi untuk menampilkan semua data terakhir pada tabel “Mata\_Pelajaran” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.45** Perintah *Like* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.45** berfungsi untuk menampilkan data pada tabel “Mata\_Pelajaran” dengan kondisi data “nama\_mapel” yang memiliki awalan huruf S. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘S%’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.46** Perintah *Not Like* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.46** berfungsi untuk menampilkan data pada tabel “Mata\_Pelajaran” dengan kondisi data “nama\_mapel” yang tidak memiliki awalan huruf S. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘S%’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.47** Perintah *Null* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.47** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan kondisi kolom “nama\_mapel” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “nama\_mapel” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.48** Perintah *Not Null* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.48** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan kondisi kolom “nama\_mapel” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “nama\_mapel” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.49** Perintah *Group by* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.49** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “nama\_mapel”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.50** Perintah *Order by* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.50** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “nama\_mapel” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.51** Perintah operator logika pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.51** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dimana kondisi data pada kolom “kode\_mapel=7”, maka akan menampilkan semua data dengan nilai “kode-mapel” sama dengan “7”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.52** Perintah *in* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.52** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘agama’)” artinya akan menampilkan data yang data “nama\_mapel” yang isi datanya “(‘agama’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.53** Perintah *between* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.53** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “nama\_mapel” yang berada diantara nilai “’b’ and ‘f’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.54** Perintah *not between* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.54** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “nama\_mapel” yang tidak berada diantara nilai “’b’ and ‘f’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.55** Perintah *substring* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.55** berfungsi untuk menampilkan data pada tabel “Mata\_Pelajaran” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(nama\_mapel,1,3))” artinya data akan menampilkan isi kolom “nama\_mapel” dengan jumlah “3” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.56** Perintah *count* pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.56** berfungsi untuk menampilkan jumlah data pada tabel “Mata\_Pelajaran” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.57** Tampilan pada tabel Mata\_Pelajaran

Pada *query* **Gambar 4.57** berfungsi untuk menampilkan semua data pada tabel “Mata\_Pelajaran” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Honorer**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.58** Tampilan data terakhir dari tabel Honorer

Pada *query* **Gambar 4.58** berfungsi untuk menampilkan semua data terakhir pada tabel “Honorer” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.59** Perintah *Like* pada tabel Honorer

Pada *query* **Gambar 4.59** berfungsi untuk menampilkan data pada tabel “Honorer” dengan kondisi data “NUPTK” yang memiliki awalan angka 1123. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘1123%’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.60** Perintah *Not Like* pada tabel Honorer

Pada *query* **Gambar 4.60** berfungsi untuk menampilkan data pada tabel “Honorer” dengan kondisi data “NUPTK” yang tidak memiliki awalan angka 11235. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘11235%’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.61** Perintah *Null* pada tabel Honorer

Pada *query* **Gambar 4.61** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan kondisi kolom “NUPTK” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “NUPTK” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.62** Perintah *Not Null* pada tabel Honorer

Pada *query* **Gambar 4.62** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan kondisi kolom “NUPTK” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “NUPTK” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.63** Perintah *Group by* pada tabel Honorer

Pada *query* **Gambar 4.63** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “NUPTK”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.64** Perintah *Order by* pada tabel Honorer

Pada *query* **Gambar 4.64** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “NUPTK” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.65** Perintah operator logika pada tabel Honorer

Pada *query* **Gambar 4.65** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dimana kondisi data pada kolom “NUPTK=’1235-2345-8902-1234’”, maka akan menampilkan semua data dengan nilai “NUPTK” sama dengan “1235-2345-8902-1234”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.66** Perintah *in* pada tabel Honorer

Pada *query* **Gambar 4.66** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘1747-7336-5020-0002’)” artinya akan menampilkan data yang data “NUPTK” yang isi datanya “(‘1747-7336-5020-0002’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.67** Perintah *between* pada tabel Honorer

Pada *query* **Gambar 4.67** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “NUPTK” yang berada diantara nilai “’1235’ and ‘1747’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.68** Perintah *not between* pada tabel Honorer

Pada *query* **Gambar 4.68** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “NUPTK” yang tidak berada diantara nilai “’1123’ and ‘1747’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.69** Perintah *substring* pada tabel Honorer

Pada *query* **Gambar 4.69** berfungsi untuk menampilkan data pada tabel “Honorer” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(NUPTK,1,4))” artinya data akan menampilkan isi kolom “NUPTK” dengan jumlah “4” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.70** Perintah *count* pada tabel Honorer

Pada *query* **Gambar 4.70** berfungsi untuk menampilkan jumlah data pada tabel “Honorer” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.71** Tampilan pada tabel Honorer

Pada *query* **Gambar 4.71** berfungsi untuk menampilkan semua data pada tabel “Honorer” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel. Tidak ada data yang tertampilkan karena hanya ada 4 baris dalam tebel tersebut.

* + - * 1. **Tabel PNS**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.72** Tampilan data terakhir dari tabel PNS

Pada *query* **Gambar 4.72** berfungsi untuk menampilkan semua data terakhir pada tabel “PNS” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.73** Perintah *Like* pada tabel PNS

Pada *query* **Gambar 4.73** berfungsi untuk menampilkan data pada tabel “PNS” dengan kondisi data “NUPTK” yang memiliki awalan angka 19. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘19%’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.74** Perintah *Not Like* pada tabel PNS

Pada *query* **Gambar 4.74** berfungsi untuk menampilkan data pada tabel “PNS” dengan kondisi data “NUPTK” yang tidak memiliki awalan angka 19. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘19%’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.75** Perintah *Null* pada tabel PNS

Pada *query* **Gambar 4.75** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan kondisi kolom “NIP” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “NIP” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.76** Perintah *Not Null* pada tabel PNS

Pada *query* **Gambar 4.76** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan kondisi kolom “NIP” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “NIP” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.77** Perintah *Group by* pada tabel PNS

Pada *query* **Gambar 4.77** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “NIP”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.78** Perintah *Order by* pada tabel PNS

Pada *query* **Gambar 4.78** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “NIP” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.79** Perintah operator logika pada tabel PNS

Pada *query* **Gambar 4.79** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dimana kondisi data pada kolom “NUPTK=’2132-4521-2157-5543’”, maka akan menampilkan semua data dengan nilai “NUPTK” sama dengan “2132-4521-2157-5543”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.80** Perintah *in* pada tabel PNS

Pada *query* **Gambar 4.80** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘197503303003121002’)” artinya akan menampilkan data yang data “NIP” yang isi datanya “(‘197503303003121002’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.81** Perintah *between* pada tabel PNS

Pada *query* **Gambar 4.81** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “NIP” yang berada diantara nilai “’1985033’ and ‘1975043’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.82** Perintah *not between* pada tabel PNS

Pada *query* **Gambar 4.82** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “NIP” yang tidak berada diantara nilai “’1985033’ and ‘1975043’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.83** Perintah *substring* pada tabel PNS

Pada *query* **Gambar 4.83** berfungsi untuk menampilkan data pada tabel “PNS” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(NIP,1,4))” artinya data akan menampilkan isi kolom “NIP” dengan jumlah “4” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.84** Perintah *count* pada tabel PNS

Pada *query* **Gambar 4.84** berfungsi untuk menampilkan jumlah data pada tabel “PNS” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.85** Tampilan pada tabel PNS

Pada *query* **Gambar 4.85** berfungsi untuk menampilkan semua data pada tabel “PNS” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan data ke-5 terakhir pada tabel.

* + - * 1. **Tabel Mengajar**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.86** Tampilan data terakhir dari tabel Mengajar

Pada *query* **Gambar 4.86** berfungsi untuk menampilkan semua data terakhir pada tabel “Mengajar” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.87** Perintah *Like* pada tabel Mengajar

Pada *query* **Gambar 4.87** berfungsi untuk menampilkan data pada tabel “Mengajar” dengan kondisi data “kode\_mapel” yang memiliki akhiran huruf A. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘%A’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.88** Perintah *Not Like* pada tabel Mengajar

Pada *query* **Gambar 4.88** berfungsi untuk menampilkan data pada tabel “Mengajar” dengan kondisi data “kode\_mapel” yang tidak memiliki akhiran huruf A. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘%A’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.89** Perintah *Null* pada tabel Mengajar

Pada *query* **Gambar 4.89** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan kondisi kolom “kode\_mapel” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “kode\_mapel” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.90** Perintah *Not Null* pada tabel Mengajar

Pada *query* **Gambar 4.90** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan kondisi kolom “kode\_mapel” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “kode\_mapel” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.91** Perintah *Group by* pada tabel Mengajar

Pada *query* **Gambar 4.91** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “kode\_mapel”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.92** Perintah *Order by* pada tabel Mengajar

Pada *query* **Gambar 4.92** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “nama\_mapel” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.93** Perintah operator logika pada tabel Mengajar

Pada *query* **Gambar 4.93** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dimana kondisi data pada kolom “kode\_mapel=7”, maka akan menampilkan semua data dengan nilai “kode\_mapel” sama dengan “7”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.94** Perintah *in* pada tabel Mengajar

Pada *query* **Gambar 4.94** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘8’)” artinya akan menampilkan data yang data “kode\_mapel” yang isi datanya “(‘8’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.95** Perintah *between* pada tabel Mengajar

Pada *query* **Gambar 4.95** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_mapel” yang berada diantara nilai “’2’ and ’6’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.96** Perintah *not between* pada tabel Mengajar

Pada *query* **Gambar 4.96** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_mapel” yang tidak berada diantara nilai “’2’ and ’6’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.97** Perintah *substring* pada tabel Mengajar

Pada *query* **Gambar 4.97** berfungsi untuk menampilkan data pada tabel “Mengajar” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(NUPTK,1,4))” artinya data akan menampilkan isi kolom “NUPTK” dengan jumlah “4” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.98** Perintah *count* pada tabel Mengajar

Pada *query* **Gambar 4.98** berfungsi untuk menampilkan jumlah data pada tabel “Mengajar” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.99** Tampilan pada tabel Mengajar

Pada *query* **Gambar 4.99** berfungsi untuk menampilkan semua data pada tabel “Mengajar” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Menguasai**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.100** Tampilan data terakhir dari tabel Meguasai

Pada *query* **Gambar 4.100** berfungsi untuk menampilkan semua data terakhir pada tabel “Menguasai” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.101** Perintah *Like* pada tabel Meguasai

Pada *query* **Gambar 4.101** berfungsi untuk menampilkan data pada tabel “Menguasai” dengan kondisi data “kode\_mapel” yang memiliki awalan huruf S. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘S%’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.102** Perintah *Not Like* pada tabel Meguasai

Pada *query* **Gambar 4.102** berfungsi untuk menampilkan data pada tabel “Menguasai” dengan kondisi data “kode\_mapel” yang tidak memiliki akhiran huruf S. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘%S’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.103** Perintah *Null* pada tabel Meguasai

Pada *query* **Gambar 4.103** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan kondisi kolom “kode\_mapel” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “kode\_mapel” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.104** Perintah *Not Null* pada tabel Meguasai

Pada *query* **Gambar 4.104** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan kondisi kolom “kode\_mapel” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “kode\_mapel” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.105** Perintah *Group by* pada tabel Meguasai

Pada *query* **Gambar 4.105** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “kode\_mapel”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.106** Perintah *Order by* pada tabel Meguasai

Pada *query* **Gambar 4.106** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “kode\_mapel” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.107** Perintah operator logika pada tabel Meguasai

Pada *query* **Gambar 4.107** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dimana kondisi data pada kolom “kode\_mapel=8”, maka akan menampilkan semua data dengan nilai “kode\_mapel” sama dengan “8”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.108** Perintah *in* pada tabel Meguasai

Pada *query* **Gambar 4.108** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘7’)” artinya akan menampilkan data yang data “kode\_mapel” yang isi datanya “(‘7’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.109** Perintah *between* pada tabel Meguasai

Pada *query* **Gambar 4.109** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_mapel” yang berada diantara nilai “’1123’ and ’6239’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4110** Perintah *not between* pada tabel Meguasai

Pada *query* **Gambar 4.110** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_mapel” yang tidak berada diantara nilai “’1123’ and ’6239’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.111** Perintah *substring* pada tabel Meguasai

Pada *query* **Gambar 4.111** berfungsi untuk menampilkan data pada tabel “Menguasai” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(kode\_mapel,1,4))” artinya data akan menampilkan isi kolom “kode\_mapel” dengan jumlah “4” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.112** Perintah *count* pada tabel Meguasai

Pada *query* **Gambar 4.112** berfungsi untuk menampilkan jumlah data pada tabel “Menguasai” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.113** Tampilan pada tabel Meguasai

Pada *query* **Gambar 4.113** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Menghadiri**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.114** Tampilan data terakhir dari tabel Menghadiri

Pada *query* **Gambar 4.114** berfungsi untuk menampilkan semua data terakhir pada tabel “Menghadiri” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.115** Perintah *Like* pada tabel Menghadiri

Pada *query* **Gambar 4.115** berfungsi untuk menampilkan data pada tabel “Menghadiri” dengan kondisi data “kode\_kelas” yang memiliki akhiran huruf A. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘%A’“.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.116** Perintah *Not Like* pada tabel Menghadiri

Pada *query* **Gambar 4.116** berfungsi untuk menampilkan data pada tabel “Menghadiri” dengan kondisi data “kode\_kelas” yang tidak memiliki akhiran huruf A. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘%A’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.117** Perintah *Null* pada tabel Menghadiri

Pada *query* **Gambar 4.117** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan kondisi kolom “kode\_kelas” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “kode\_kelas” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.118** Perintah *Not Null* pada tabel Menghadiri

Pada *query* **Gambar 4.118** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan kondisi kolom “kode\_kelas” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “kode\_kelas” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.119** Perintah *Group by* pada tabel Menghadiri

Pada *query* **Gambar 4.119** berfungsi untuk menampilkan semua data pada tabel “Menguasai” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “kode\_kelas”.

1. ***Order by***

|  |
| --- |
|  |

**Gambar 4.120** Perintah *Order by* pada tabel Menghadiri

Pada *query* **Gambar 4.120** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “kode\_kelas” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.121** Perintah operator logika pada tabel Menghadiri

Pada *query* **Gambar 4.121** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan perintah “select” dimana kondisi data pada kolom “kode\_kelas=’3C’”, maka akan menampilkan semua data dengan nilai “kode\_kelas” sama dengan “3C”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.122** erintah *in* pada tabel Menghadiri

Pada *query* **Gambar 4.122** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘3B’)” artinya akan menampilkan data yang data “kode\_kelas” yang isi datanya “(‘3B’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.123** Perintah *between* pada tabel Menghadiri

Pada *query* **Gambar 4.123** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_kelas” yang berada diantara nilai “’1’ and ‘2’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.124** Perintah *not between* pada tabel Menghadiri

Pada *query* **Gambar 4.124** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_kelas” yang tidak berada diantara nilai “’1’ and ‘2’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.125** Perintah *substring* pada tabel Menghadiri

Pada *query* **Gambar 4.125** berfungsi untuk menampilkan data pada tabel “Menghadiri” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(kode\_kelas,1,2))” artinya data akan menampilkan isi kolom “kode\_kelas” dengan jumlah “2” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.126** Perintah *count* pada tabel Menghadiri

Pada *query* **Gambar 4.126** berfungsi untuk menampilkan jumlah data pada tabel “Menghadiri” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.127** Tampilan pada tabel Menghadiri

Pada *query* **Gambar 4.127** berfungsi untuk menampilkan semua data pada tabel “Menghadiri” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + - * 1. **Tabel Mempelajari**

1. **Tampilkan data terakhir yang ada pada suatu tabel dari *database* yang sudah anda buat.**

|  |
| --- |
|  |

**Gambar 4.128** Tampilan data terakhir dari tabel Mempelajari

Pada *query* **Gambar 4.128** berfungsi untuk menampilkan semua data terakhir pada tabel “Memeplajari” dengan perintah “select \*” kemudian menggunakan perintah “order by” untuk mengurutkan data dari data terakhir dan menampilkan 1 data terakhir dengan menggunkan perintah “limit 1”.

1. ***Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.129** Perintah *Like* pada tabel Mempelajari

Pada *query* **Gambar 4.129** berfungsi untuk menampilkan data pada tabel “Mempelajari” dengan kondisi data “kode\_mapel” yang memiliki akhiran angka 1. hal ini dapat dilakukan dengan menggunakan perintah “select” dan “like ‘%1’”.

1. ***Not* *Like* dengan karakter *wildcard***

|  |
| --- |
|  |

**Gambar 4.130** Perintah *Not Like* pada tabel Mempelajari

Pada *query* **Gambar 4.130** berfungsi untuk menampilkan data pada tabel “Mempelajari” dengan kondisi data “kode\_mapel” yang tidak memiliki akhiran angka 1. hal ini dapat dilakukan dengan mennakan perintah “select” dan “not like ‘%1’“. Kondisi ini merupakan kebalikan dari perintah “like”.

1. ***Null***

|  |
| --- |
|  |

**Gambar 4.131** Perintah *Null* pada tabel Mempelajari

Pada *query* **Gambar 4.131** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan kondisi kolom “kode\_mapel” bernilai “null” menggunakan perintah “is null”. Akan tetapi pada kolom “kode\_mapel” tidak ada data bernilai “null” maka tidak ada data yang ditampilkan.

1. ***Not* *Null***

|  |
| --- |
|  |

**Gambar 4.132** Perintah *Not Null* pada tabel Mempelajari

Pada *query* **Gambar 4.132** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan kondisi kolom “kode\_mapel” bernilai tidak “null” menggunakan perintah “is not null”. Berdasarkan pada kolom “kode\_mapel” semua data bernilai “not null” maka semua data yang ditampilkan.

1. ***Group* *by***

|  |
| --- |
|  |

**Gambar 4.133** Perintah *Group by* pada tabel Mempelajari

Pada *query* **Gambar 4.133** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dan menampilkan salah satu data jika memiliki data sama. dengan perintah “group by” berdasarkan data pada kolom “kode\_mapel”.

1. ***Order*  *by***

|  |
| --- |
|  |

**Gambar 4.134** Perintah *Order by* pada tabel Mempelajari

Pada *query* **Gambar 4.134** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dan mengurutkannya dengan perintah “order by”. Pengurutan data berdasarkan data pada kolom “kode\_mapel” secara *descending* atau mengurutkan dari yang terbesar ke yang terkecil.

1. **Operator logika**

|  |
| --- |
|  |

**Gambar 4.135** Perintah operator logika pada tabel Mempelajari

Pada *query* **Gambar 4.135** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dimana kondisi data pada kolom “kode\_mapel=3”, maka akan menampilkan semua data dengan nilai “kode\_mapel” sama dengan “3”.

1. **IN**

|  |
| --- |
|  |

**Gambar 4.136** Perintah *in* pada tabel Mempelajari

Pada *query* **Gambar 4.136** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “in (‘4’)” artinya akan menampilkan data yang data “kode\_mapel” yang isi datanya “(‘4’)”.

1. ***Between***

|  |
| --- |
|  |

**Gambar 4.137** Perintah *between* pada tabel Mempelajari

Pada *query* **Gambar 4.137** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_mapel” yang berada diantara nilai “’1’ and ‘3’”.

1. ***Not* *Between***

|  |
| --- |
|  |

**Gambar 4.138** Perintah *not between* pada tabel Mempelajari

Pada *query* **Gambar 4.138** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dimana pada *query* diatas memiliki kondisi “not between” artinya akan menampilkan data tabel pendaftar berdasarakan data “kode\_mapel” yang tidak berada diantara nilai “’1’ and ‘3’”.

1. ***Substring***

|  |
| --- |
|  |

**Gambar 4.139** Perintah *substring* pada tabel Mempelajari

Pada *query* **Gambar 4.139** berfungsi untuk menampilkan data pada tabel “Mempelajari” dengan perintah “select” kemudian dilanjutkan dengan perintah “(substring(kode\_mapel,1,2))” artinya data akan menampilkan isi kolom “kode\_mapel” dengan jumlah “2” *string* dari urutan pertama data yang sebenarnya.

1. ***Count***

|  |
| --- |
|  |

**Gambar 4.140** Perintah *count* pada tabel Mempelajari

Pada *query* **Gambar 4.140** berfungsi untuk menampilkan jumlah data pada tabel “Mempelajari” dengan menggunakan perintah “count (\*)” yang merupakan salah satu fungsi agregat yang berfungsi untuk menghitung jumlah data pada suatu tabel atau kolom tertentu.

1. **Tampilkan 5 sampai dengan 10 data terakhir dari suatu tabel pada *database*.**

|  |
| --- |
|  |

**Gambar 4.141** Tampilan pada tabel Mempelajari

Pada *query* **Gambar 4.141** berfungsi untuk menampilkan semua data pada tabel “Mempelajari” dengan perintah “select” dan perintah selanjutnya yaitu “limit 5, 10” yang berfungsi untuk menampilkan 5 data terakhir pada tabel.

* + 1. **Memuat *Sub* *query* pada 3 tabel (minimal menampilkan 2 *field*).**

|  |
| --- |
|  |

**Gambar 4.142** Tampilan *sub query* pada 3 tabel

Pada *query* **Gambar 4.142** merupakan *sub querry* yang berfungsi untuk menampilkan data pada kolom kelas yang mana perintah “select” pertama berfungsi untuk menampilkan data pada kolom “kode\_kelas” pada tabel “kelas”. Perintah “select” kedua berfungsi untuk mengatur kondisi “kode\_kelas” yang melibatkan tabel “menghadiri”.

* + 1. **Membuat *Sub* *query* pada 4 tabel (minimal menampilkan 2 *field*).**

|  |
| --- |
|  |

**Gambar 4.143** Tampilan *sub query* pada 4 tabel

Pada *query* **Gambar 4.143** berfungsi untuk menampilkan data pada kolom “nama\_mapel” pada tabel “Mata\_Pelajaran” yang dikuasai oleh guru honorer dengan perintah “select” dengan kondisi *subquery* pada4 tabel, tabel yang digunakan antara lain “Menguasai” , “Guru”, dan “Honorer”. Pada kondisi *subquery* pada tabel terakhir akan terdapat kondisi data yang mana akan ditampilkan.

* + 1. **Membuat *sub* *query* sebelum kondisi atau setelah perintah SELECT (minimal 3 *query*).**

|  |
| --- |
|  |

**Gambar 4.144** Tampilan *sub query* dengan 3 *query*

Pada *query* **Gambar 4.144** berfungsi untuk menampilkan data “kode\_kelas” dengan kondisi “kodekelas = ‘1A’”, “nama\_guru” dengan kondisi “nama\_guru = ’Anton Kurniawan” dari tabel “guru”, dan “nama\_mapel” dengan kondisi “nama\_mapel = ‘Agama’” dari tabel “mata\_pelajaran”.

* 1. **KESIMPULAN**

Berdasarkan praktikum yang dilakukan dapat diambil kesimpulan sebagai berikut:

1. Dalam menampilkan sebuah data pada tabel dapat menggunakan perintah SELECT dimana perintah SELECT dapat memiliki suatu kondisi maupun tidak.
2. Dalam sebuah kondisi tertentu, Sebuah *query* bisa digunakan untuk memanipulasi data yang diinginkan pada suatu tabel, kondisi tersebut dapat digunakan dengan menggunakan perintah WHERE.
3. *Query* dapat langsung dilakukan pada beberapa tabel dengan menggunakan sebuah *subquery*, namun penggunaan *subquery* perlu mengetahui rincian ataupun jenis *subquery* yang digunakan agar dapat menentukan jenis operator yang benar sesuai dengan *subquery* yang akan digunakan.

untuk menampilkan suatu data pada tabel relasi sebelumnya data tersebutsudah dapat terhubung. Disini dapat menggunakan perintah SELECT lalu setelah itu masukkan *subquery* yang diinginkan, namun *subquery* yang digunakan harus berdasarkan pada relasi pada tabel.

MODUL V  
JOIN

* 1. **TUJUAN**
     1. Mahasiswa dapat melakukan *query* pada beberapa tabel menggunakan klausa JOIN.
     2. Mahasiswa dapat menggunakan klausa JOIN dengan benar.
  2. **LANDASAN TEORI**

1. **Jenis-Jenis JOIN**

Secara umum terdapat 5 jenis operasi untuk JOIN, yaitu:

1. **CROSS atau CARTESIAN JOIN**

CROSS JOIN digunakan untuk mendapatkan berbagai kemungkinan kombinasi. Hasil CROSS JOIN ialah kolom yang merupakan gabungan dari semua tabel yang di-JOIN dan baris merupakan semua kombinasi dari semua baris yang di-JOIN [1].

1. **INNER atau EQUI JOIN**

Dapat menampilkan data-data dari kedua tabel dengan men-JOIN-kan *foreign key* dari satu tabel dengan *primary key* dari tabel lainnya. EQUI adalah relasi yang dinyatakan dengan bentuk persamaan (=).

1. **INNER atau NON-EQUI JOIN**

NON-EQUI adalah relasi antara dua atau lebih tabel dimana nilai dari kolom-kolom yang dihubungkan tidak saling terhubung secara langsung (tidak ada hubungan *primarykey* dan *foreignkey*). Dalam NON-EQUI JOIN digunakan operator selain *equal* (=), seperti >, <, <>, >=, <= [3].

1. **OUTER JOIN**

OUTER JOIN digunakan untuk mendapatkan informasi dari suatu tabel yang datanya sesuai dan tidak sesuai. OUTER JOIN terbagi menjadi 3, yaitu: [2]

1. **LEFT OUTER JOIN**

Mencari informasi yang relasinya ada pada kedua sisi tabel dan yang relasi tabel pada sisi kiri tidak dijumpai pada tabel sisi kanan [2].

1. **RIGHT OUTER JOIN**

Mencari informasi yang relasi ada pada kedua sisi tabel dan relasi tabel yang ada pada sisi kanan tabel dan tidak dijumpai pada sisi kiri tabel.

1. **FULL OUTER JOIN**

Melakukan JOIN yang menghasilkan semua baris dari tabel kiri dan tabel kanan. Namun, saat ini FULL OUTER JOIN sudah tidak ada pada SQL.Maka untuk melakukan FULL OUTER JOIN dilakukan dengan menggabungkan LEFT OUTER JOIN dengan RIGHT OUTER JOIN.

1. **SELF JOIN**

SELF JOIN sama dengan JOIN lainnya, hanya melibatkan tabel yang sama.

1. ***Query* JOIN pada Beberapa Tabel**

Masing-masing operasi untuk JOIN, memiliki *query*-nya masing-masing, yaitu:

1. **CROSS atau Kartesian JOIN**

*Query* umum untuk CROSS JOIN adalah:

|  |
| --- |
| SELECT \*FROM nama\_tabel1, nama\_tabel2; |

1. **INNER atau EQUI JOIN**

*Query* umum untuk INNER JOIN:

|  |
| --- |
| SELECT \*FROM nm\_tabel1 INNER JOIN nm\_tabel2 ON tabel1.kolom1=tabel2.kolom2; |

Atau:

|  |
| --- |
| SELECT \*FROM nm\_tabel1, nm\_tabel2 WHERE tabel1.kolom1= tabel2.kolom2; |

1. **INNER atau NON-EQUI JOIN**

*Query* umum untuk INNER atau NON-EQUI JOIN:

|  |
| --- |
| SELECT \*FROM nm\_tabel1 INNER JOIN nm\_kolom1 ON tabel1.kolom1 [>||<||<>||>=||<=] tabel2.kolom2; |

Atau:

|  |
| --- |
| SELECT \*FROM nm\_tabel1, nm\_tabel2 WHERE tabel1.kolom1 [>||<||<>||>=||<=] tabel2.kolom2; |

1. **OUTER JOIN**
2. **LEFT OUTER JOIN**

|  |
| --- |
| SELECT \*FROM nm\_tabel1 LEFT JOIN nm\_tabel2 ON nm\_tabel1.nm\_kolom1=nm\_tabel2.nm\_kolom2; |

1. **RIGHT OUTER JOIN**

|  |
| --- |
| SELECT \*FROM nm\_tabel1 RIGHT JOIN nm\_tabel2 ON nm\_tabel1.nm\_kolom1=nm\_tabel2.nm\_kolom2; |

1. **FULL OUTER JOIN**

|  |
| --- |
| SELECT \*FROM nm\_tabel1 LEFT JOIN nm\_tabel2 ON nm\_tabel1.nm\_kolom1=nm\_tabel2.nm\_kolom2  UNION  SELECT \*FROM nm\_tabel1 RIGHT JOIN nm\_tabel2 ON nm\_tabel1.nm\_kolom1=nm\_tabel2.nm\_kolom2; |

1. **SELF JOIN**

*Query* umum untuk SELF JOIN:

|  |
| --- |
| SELECT alias1\_tabel1.kolom1, alias2\_tabel2.kolom2 FROM tabel as alias1, tabel AS alias2 WHERE alias1\_tabel.kolom1= alias2\_tabel1.kolom; |

* 1. **KASUS KELOMPOK**

Sekolah Dasar Internasional Wakanda (SBIW) akan merancang sebuah sistem informasi. Basis data yang dibentuk berdasarkan data-data berbagai entitas. Guru akan didata nama, kontak dan alamat sehingga terbentuk NUPTK. Lalu guru dibedaan menjadi guru honorer dan PNS yang memiliki NIP. Semua guru dapat mengajar mata pelajaran, dan guru pun menguasai tidak hanya satu saja mata pelajaran, namun terkadang mereka harus mengajar mata pelajaran yang tidak mereka pelajari, sehingga mata pelajaran dapat dikuasai dan diajar oleh banyak guru. Mata pelajaran hanya didata namanya saja, sehingga dibentuk kode mata pelajaran. Setiap mata pelajaran yang diajar guru akan didata pegisian kelas mana yang diajar. Sehingga guru dapat mengajar mata pelajaran yang sama di kelas yang berbeda, begitupun dikelas yang sama dapat mengajar mata pelajaran lain. Kelaspun harus di data nama kelasnya sehinggaa terbentuk kode kelas. Sebuah kelas dapat diwalikan oleh seorang guru non-honorer. Sebuah kelaspun memiliki banyak siswa, dimana dari mereka terdapat satu yang menjadi ketua kelas. Setiap siswa akan didata nama, alamay, nama wali, kontak wali, dan rangking mereka hingga dibentuk NIS. Setiap siswa dapat mempelajari banyak mata pelajaran, begitu pula sebaliknya setiap mata pelajaran dapat dipelajari banyak siswa. Setiap kegiatan belajar mengajar akan dilakukan pengambilan nilai.

|  |
| --- |
| X:\Kampus\Kuliah Angga\Semester 3\Sistem Basis Data\WhatsApp Image 2019-11-01 at 22.16.01.jpeg |

**Gambar 5.1** ERD setelah di normalisasi

* 1. **KASUS MODUL**

1. Membuat *query inner join*, *non equi join*, *cross join*, *natural join*, *left join*, dan *right join* pada 2 tabel.
2. Membuat *queryinnerjoin*, *non equi join*, *cross join*, *natural join*, *left join*, dan *right join* pada 3 tabel.
   1. **ANALISA KASUS**

Berikut merupakan analisis dari hasil kasus modul dan kasus kelompok

* + 1. ***Join* dua tabel dengan kondisi**

Berikut merupakan penjelasan tentang *join* yang dilakukan terhadap dua tabel.

1. ***Inner Join***

|  |
| --- |
|  |

**Gambar 5.2** *Inner join* dua tabel

Pada **Gambar 5.2** duah buah tabel yaitu “guru” dan “mengajar”, dari kedua buah tabel tersebut ditampilkan kolom “nama\_guru” dan “kode\_mapel” yang diajar oleh guru-guru tersebut, “nama\_guru” dan “kode\_mapel” yang diajar diperoleh dari *query* “select nama\_guru,kode\_mapel from guru inner join mengajar on guru.nuptk=mengajar.nuptk;”, jadi bisa disimpulkan bahwa tabel tersebut merupakan relasi antara “guru” dengan “kode\_mapel” apa yang diajar.

1. ***Non equi Join***

|  |
| --- |
|  |

**Gambar 5.3** *Non equi join* dua tabel

Pada **Gambar 5.3** dua buah tabel yaitu “siswa” dan “kelas”, pada tabel tersebut ditampilkan “nama\_siswa”, “nis”, “nama\_wali”, dan “NIS\_ketuakelas” dimana pada tabel tersebut ada kondisi dimana “nama\_siswa” tidak boleh ada yang bernama “I Putu Angga Purnama Widiarta” untuk ditampilkan pada tabel atau dalam artian lain, tabel tersebut menampilkan data diluar dari nama yang ada pada kondisi tersebut. Hal ini merupakan hasil dari *query* “select nama\_siswa,nis,nama\_wali,NIS\_ketuakelas from siswa inner join kelas where nama\_siswa!=”I Putu Angga Purnama Widiarta” limit 5;”, fungsi *limit* pada *query* tersebut digunakan untuk membatasi data yang ingin ditampilkan.

1. ***Cross Join***

|  |
| --- |
|  |

**Gambar 5.4** *Cross join* dua tabel

Pada **Gambar 5.4** dua buah tabel yaitu “mata\_pelajaran” dan “mengajar”, “kode\_mapel” pada kedua tabel dimana pada “mata\_pelajaran”bernilai sama dengan “mengajar” dengan *query* “select \* from mata\_pelajaran cross join mengajar where kode\_mapel=5 limit 3;” ditambahkan dengan kondisi dimana “kode\_mapel” bernilai 5.

1. ***Natural Join***

|  |
| --- |
|  |

### **Gambar 5.5** *Natural join* dua tabel

Pada **Gambar 5.5** dua buah tabel yaitu “guru” dan “mengajar”, “nama\_guru” pada kedua tabel dimana pada “guru”bernilai sama dengan “mengajar” dengan *query* “select \* from guru natural join mengajar where nama\_guru=”Imam Suryadi”;” ditambahkan dengan kondisi dimana “nama\_guru” bernama “Imam Suryadi”.

1. ***Left Join***

|  |
| --- |
|  |

### **Gambar 5.6** *Left join* dua tabel

Pada **Gambar 5.6** dua buah tabel yaitu “guru” dan “mengajar”, “nuptk” pada kedua tabel dimana pada “guru”bernilai sama dengan “mengajar” dengan *query* “select \* from guru left join mengajar on guru.nuptk=mengajar.nuptk limit 6;”. Pada *query* tersebut ada suatu kondisi dimana ada seorang “guru” yang belum mengajar “mata\_pelajaran” apapun, kondisi tersebut dapat diketahui dikarenakan adanya penggunaan dari *left join* itu sendiri, hal ini dapat disimpulkan melalui “kode\_mapel” yang bernilai *null*

1. ***Right* *Join***

|  |
| --- |
|  |

### **Gambar 5.7** *Right join* dua tabel

Pada **Gambar 5.7** dua buah tabel yaitu “guru” dan “mengajar”, “nuptk” pada kedua tabel dimana pada “guru”bernilai sama dengan “mengajar” dengan *query* “select \* from tiket right join membeli on tiket.harga=membeli.id\_penonton limit 2;”. Pada tabel tersebut dapat dilihat adanya “kode\_mapel” yang belum memiliki pengajar, kondisi tersebut terjadi karena fungsi dari *right join* yang menampilkan ketidaksesuaian pada suatu tabel.

* + 1. ***Join* tiga tabel**

Berikut merupakan penjelasan tentang *join* yang dilakukan terhadap tiga tabel berbeda

1. ***Inner join***

|  |
| --- |
|  |

### **Gambar 5.8** *Inner join* pada tiga tabel

Pada **Gambar 5.8** tiga buah tabel yaitu “guru”, “mengusai”, pada tabel tersebut ditampilkan “nuptk”, “nama\_guru”, “kode\_mapel”, dan “nilai”. Ada sebuah kondisi dimana “nilai” yang diminta untuk ditampilkan hanya “nilai” yang bernilai sama dengan “100”, *query* dari tabel tersebut adalah “select guru.nuptk, nama\_guru, menguasai.kode\_mapel,nilai from guru inner join menguasai inner join mempelajari where nilai=100 limit 3;”. *Limit* pada *query* diatas difungsikan untuk membatasi data yang dapat ditampilkan pada tabel yang dimana jumlah data yang dapat ditampilkan hanya maksimal 3 data saja.

1. ***Non equi Join***

|  |
| --- |
|  |

### **Gambar 5.9** *Non equi join* pada tiga tabel

Pada **Gambar 5.9** tiga buah tabel yaitu “guru”, “menguasai”, dan “mempelajari”, dimana data pada tabel tersebut hanya menampilkan data dengan “nilai” dibawah “100”. Data-data pada tabel tersebut ditampilkan melalui *query* “select guru.nuptk, nama\_guru, menguasai.kode\_mapel,nilai from guru inner join menguasai inner join mempelajari where nilai<100 limit 3;”. *Limit* pada *query* diatas difungsikan untuk membatasi data yang dapat ditampilkan pada tabel yang dimana jumlah data yang dapat ditampilkan hanya maksimal 3 data saja.

1. ***Cross Join***

|  |
| --- |
|  |

### **Gambar 5.10** *Cross join* pada tiga tabel

Pada **Gambar 5.10** tiga buah tabel yaitu “guru”, “membeli”, dan “penonton”, dimana pada tabel tersebut ditampilkan gabungan dari setiap kolom yang ada dari ketiga tabel diatas. Pada kolom “nilai”, “nilai” yang hanya bisa ditampilkan yaitu “nilai” yang sama dengan “100”, pada tabel ini tidak ada hubungan benar-benar saling berelasi, *cross join* hanya digunakan untuk menggabungkan kolom-kolom yang ada pada setiap tabel yang ada pada sebuah *database*.

1. ***Natural Join***

|  |
| --- |
|  |

### **Gambar 5.11** *Natural join* tiga tabel

Pada **Gambar 5.11** tiga buah tabel yaitu “guru”, “menguasai”, dan “mempelajari”, dimana pada tabel ini akan menampilkan “guru” yang dimana “mata\_pelajaran” yang dikuasai olehnya, siswa-siswinya mendapatkan nilai “100” hal ini terjadi dikarenakan fungsi dari *query* “select guru.nuptk,nama\_guru,kode\_mapel,nilai from guru natural join menguasai natural join mempelajari where nilai=100;”. Hal ini dapat terjadi dikarenakan ketiga tabel tersebut saling berhubungan antara satu dengan lainnya, fungsi dari *primary key* dan *foreign key* dapat kita buktikan melalui tampilan dari tabel diatas, dikarenakan kita dapat mengetahui “guru” yang “menguasai” suatu “mata\_pelajaran” dimana siswa-siswinya dapat memperoleh “nilai” yang sama dengan “100”.

1. ***Left Join***

|  |
| --- |
|  |

### **Gambar 5.12** *Left join* tiga tabel

Pada **Gambar 5.12** tiga buah tabel yaitu “guru”, “menguasai”, dan “mempelajari”, dimana “nuptk” pada “guru”bernilai sama dengan “menguasai” dan “kode\_mapel” pada “mempelajari”bernilai sama dengan “mempelajari” dengan *query* “select guru.nuptk,nama\_guru,menguasai.kode\_mapel,nilai from guru left join menguasai on guru.nuptk=menguasai.nuptk left join mempelajari on menguasai.kode\_mapel=mempelajari.kode\_mapel order by kode\_mapel limit3;”.

1. ***Right Join***

|  |
| --- |
|  |

### **Gambar 5.13** *Right join* tiga tabel

Pada **Gambar 5.13** tiga buah tabel yaitu “guru”, “menguasai”, dan “mempelajari”, dimana “nuptk” pada “guru”bernilai sama dengan “menguasai” dan “kode\_mapel” pada “mempelajari”bernilai sama dengan “mempelajari” dengan *query* “select guru.nuptk,nama\_guru,menguasai.kode\_mapel,nilai from guru right join menguasai on guru.nuptk=menguasai.nuptk right join mempelajari on menguasai.kode\_mapel=mempelajari.kode\_mapel order by kode\_mapel limit3;”.

* 1. **KESIMPULAN**

Berdasarkan praktikum yang telah dilakukan pada Modul 5 JOIN, dapat disimpulkan bahwa:

1. Terdapat 5 operasi dalam *join* yaitu *inner join*, *outer join*, *natural join*, *cross join*, dan *self join*.
2. *Inner join* dan *equi join* memiliki kesamaan dalam melakukan penampilan data. Perbedaannya hanya pada klausa penghubungnya, yaitu “inner join” pada *inner join* dan tanda koma “,” pada *equi join*. O*uter join* terdiri dari dua jenis *join*, yaitu *left join* dan *right join*. Pada *left join*, yang menjadi patokan adalah tabel sisi kiri, sedangkan pada *right join*, yang menjadi patokan adalah tabel sisi kanan. *Cross join* merupakan kombinasi perkalian dari semua baris yang ada dalam semua tabel yang digabungkan.*Natural join* hampir sama dengan *cross join*, hanya saja pada *natural join* diberikan kondisi sehingga baris hasilnya tidak banyak. *Self join*, tabel yang terlibat dalam penggabungan hanya satu. Penggabungan ini berguna jika ada dua kolom dalam tabel yang sama, namun ingin digunakan dua kali.