|  |
| --- |
|  |
| Administrasi Dan Pengelolaan Database | |
| **Bahan Ajar (Materi Pelengkap Modul)**  Diklat Fungsional Pranata Komputer Tingkat Ahli  Badan Pusat Statistik  Angkatan 2 Tahun 2019 | |
|  | |
| **Jimmy Ludin, SST., M.Si.** | |
|  | |

|  |
| --- |
| Pusat Pendidikan dan Pelatihan Badan Pusat Statistik Tahun 2019 |

DAFTAR ISI

[DAFTAR ISI i](#_Toc383690682)

[BAB 1 1](#_Toc383690683)

[PENERAPAN DATABASE 1](#_Toc383690684)

[A. Database Management Sistem (DBMS) 1](#_Toc383690685)

[B. Gambaran Database dan Penerapannya. 1](#_Toc383690686)

[BAB 2 2](#_Toc383690687)

[MEMAHAMI KONSEP DATABASE 2](#_Toc383690688)

[A. Model Database Relasi 2](#_Toc383690689)

[B. Jenis Hubungan Antar Tabel 2](#_Toc383690690)

[C. Model Data Entity Relationship. 2](#_Toc383690691)

[D. Diagram Entity Relationship (Diagram E-R) 3](#_Toc383690692)

[E. Relasi Satu ke Satu (One to One) 4](#_Toc383690693)

[F. Relasi Satu ke Banyak (One to Many) 5](#_Toc383690694)

[G. Relasi Banyak ke Banyak (Many to Many) 6](#_Toc383690695)

[H. Tahapan Membuat Diagram E-R. 6](#_Toc383690696)

[BAHASA SQL (STRUCTURE QUERY LANGUAGE) 9](#_Toc383690697)

[A. Komponen SQL 9](#_Toc383690698)

[B. Data Definition Language 9](#_Toc383690699)

[DAFTAR PUSTAKA 13](#_Toc383690700)

BAB 1

PENERAPAN DATABASE

1. Database Management Sistem (DBMS)

**Database Management Sistem** atau disingkat DBMS adalah perangkat lunak (Software) yang berfungsi untuk mengelola database, mulai dari membuat database itu sendiri, sampai dengan proses-proses yang berlaku dalam database tersebut, baik berupa entry, edit, hapus, query terhadap data, membuat laporan dan lain sebagainya secara efektif dan efisien. Salah satu jenis DBMS yang sangat terkenal saat ini adalah Relational DBMS (RDBMS), yang merepresentasikan data dalam bentuk tabel-tabel yang saling berhubungan. Sebuah tabel disusun dalam bentuk baris (record) dan kolom (field)

Banyak sekali berkembang perangkat lunak RDBMS ini, misalnya MySQL, Oracle, Sybase, dBase, MS. SQL, Microsoft Access (MS. Access) dan lain-lain.

1. Gambaran Database dan Penerapannya.

Misalnya dalam dunia pendidikan atau lingkungan diklat pada umumnya, sering anda menjumpai pertanyaan-pertanyaan seperti berikut :

* + Berapa jumlah peserta yang mengikuti diklat Database Management?
  + Berapa jumlah peserta yang berjenis kelamin laki-laki dan yang berjenis kelamin perempuan ?
  + Dan banyak pertanyaan-pertanyaan yang lain.

Jawaban dari pertanyaan-pertanyaan di atas dapat dilakukan dengan cepat dan mudah, bila dalam pengelolaan sistem informasi diklat telah menggunakan sistem database. Tapi akan sangat membosankan dan memakan waktu yang lama jika masih dikelola secara manual.

Ini merupakan contoh kecil yang dihadapi dalam dunia diklat, tentu hal ini juga akan dihadapi dalam bidang kerja yang lain dengan format dan model yang lain pula.

BAB 2

MEMAHAMI KONSEP DATABASE

1. Model Database Relasi

Model database relasi merupakan model database yang paling banyak digunakan saat ini, karena paling sederhana dan mudah digunakan serta yang paling penting adalah kemampuannya dalam mengakomodasi berbagai kebutuhan pengelolaan database. Sebuah database dalam model ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (**record**) dan kolom (**field**), pertemuan antara baris dengan kolom disebut **item data (data value)**, tabel-tabel yang ada dihubungkan (**relationship**) sedemikian rupa menggunakan **field-field kunci (Key field)** sehingga dapat meminimalkan duplikasi data.

1. Jenis Hubungan Antar Tabel

Jenis hubungan antar tabel dalam model database relasi, juga didefinisikan dengan hubungan :

* Satu ke satu (One to One)
* Satu ke Banyak (One to Many)
* Banyak ke satu (Many to One)
* Banyak ke Banyak (Many to Many)

Untuk lebih jelasnya penggunaan hubungan ini, sering digunakan **Diagram Entity Relationship (Diagram E-R)** yang merupakan bagian dari **Model Data Entity Relationship.**

1. Model Data Entity Relationship.

Seperti telah dijelaskan di atas, bahwa hubungan antar data dan batasan-batasannya dalam suatu sistem database, dapat diolah secara hirarki, jaringan dan relasional. Ketiga tipe model data ini mengacu kepada hubungan antar record (**Record Based Data Models**) dalam masing-masing entity/tabel. Tapi disisi lain, hubungan dan batasan data ini, dapat juga berupa **Object Based Data Model** (Model Data Berbasis Object).

Konsep utama dalam model data berbasis object ini adalah penggunaan entity, atribut dan hubungan antar entitynya (**Entity Relationship**). Yang tergolong kedalam **Object Based Data Model** ini adalah **Model Data Entity Relationship**. Model data entity relationship sering dijadikan acuan dalam merancang suatu sistem database.

Pada **Model Data Entity Relationship** ini, data yang ada ditransformasikan dengan memanfaatkan sejumlah perangkat konseptual menjadi diagram data, yang sering disebut **Diagram Entity Relationship (Diagram E-R)**. Ada dua komponen utama pembentuk model data ini, yaitu :

* + - **Entity** beserta attributnya.
    - **Relasi** dan jenis hubungannya

1. Diagram Entity Relationship (Diagram E-R)

**Diagram E-R** digunakan untuk menggambarkan secara sistematis hubungan antar entity-entity yang ada dalam suatu sistem database menggunakan simbol-simbol sehingga lebih mudah dipahami. Simbol-simbol yang boleh digunakan adalah :

1. **Persegi Panjang**, berfungsi untuk menyatakan suatu entity.
2. **Elips**, berfungsi untuk menyatakan attribute, jika diberi garis bawah menandakan bahwa attribute tersebut merupakan attribute/field kunci.
3. **Belah Ketupat**, menyatakan jenis relasi.
4. **Garis**, penghubungan antara relasi dengan entity dan antara entity dengan attribute.

Misal : Hubungan antara entity Dosen dengan entity mata kuliah.



Dalam hubungan antar entity, juga harus ditentukan derajat relasi antar entity. Derajat relasi menunjukkan jumlah maksimum record suatu entity ber-relasi dengan record pada entity yang lainnya. Misalnya pada contoh sebelumnya, entity mahasiswa dapat berelasi dengan **lebih dari satu** record yang ada pada entity mata kuliah sebaliknya **satu** record pada entity mata kuliah hanya boleh ber-relasi dengan **satu** mahasiswa yang sama pada entity mahasiswa, begitu juga satu record pada entity mata kuliah berhubungan paling banyak satu record juga pada entity dosen, dan seterusnya.

1. Relasi Satu ke Satu (One to One)

Artinya **satu** record pada entity A ber-relasi paling banyak **satu** record juga pada entity B, begitu juga sebaliknya, satu record pada entity B, ber-relasi paling banyak satu record juga dengan entity A. Dalam diagram E-R, relasi ini disimbolkan dengan angka 1.

Contoh : Dalam proses belajar mengajar secara privat misalnya, seorang (**satu**) tutor hanya mengajar **satu** siswa, begitu juga sebaliknya, satu siswa hanya diajar oleh satu tutor. Hubungan antar entity tutor dengan siswa ini dapat digambar dengan diagram E-R sebagai berikut :



**Keterangan :**

* Entity tutor mempunya dua attribute, yaitu kode tutor (**kd\_tutor**) yang berfungsi sebagai field kunci, dan nama tutor (**nm\_tutor**).
* Entity siswa juga mempunyai dua attribute, yaitu kode siswa (**kode\_sis**) sebagai field kunci dan nama siswa (**nm\_sis**).
* Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode tutor (**kd\_tutor**) dan kode siswa (**kd\_sis**) yang berfungsi sebagai kunci tamu (foreign key) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan tempat mengajar (**tempat**).
* Derajat relasi dinyatakan dengan **1 : 1**, yang menandakan bahwa hubungan antar entity adalah satu ke satu, seperti terlihat pada gambar.
* Dalam diagram E-R, attribute yang berfungsi sebagai field kunci harus di beri garis bawah, sebagai pembeda dengan attribute yang lain.

1. Relasi Satu ke Banyak (One to Many)

Artinya **satu** record pada entity A ber-relasi dengan beberapa record pada entity B, tapi tidak sebaliknya, setiap record pada entity B ber-relasi paling banyak satu record dengan entity A. Dalam diagram E-R, relasi ini disimbolkan dengan angka **1** untuk menyatakan satu dan huruf **M** atau **N** untuk menyatakan banyak.

Contoh : Dalam proses belajar mengajar di sekolah dasar misalnya, **satu** orang guru mengajar beberapa (**banyak**) murid, tetapi satu kelas (beberapa murid) hanya di ajaroleh satu guru. Hubungan antara entity guru dan murid seperti ini dapat dibuat diagram E-R, sebagai berikut :



**Keterangan :**

* Entity guru mempunya dua attribute, yaitu kode guru (**kd\_guru**) yang berfungsi sebagai field kunci, dan nama guru (**nm\_guru**).
* Entity murid juga mempunyai dua attribute, yaitu kode murid (kd\_murid) sebagai field kunci dan nama murid (nm\_murid).
* Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode guru (**kd\_guru**) dan kode murid (**kd\_murid**) yang berfungsi sebagai kunci tamu (foreign key) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan lokal mengajar (**lokal**).
* Derajat relasi dinyatakan dengan **1 : M**, yang menandakan bahwa hubungan antar entity adalah satu ke banyak, seperti terlihat pada gambar diatas.

1. Relasi Banyak ke Banyak (Many to Many)

Artinya beberapa record pada entity A dapat ber-relasi dengan beberapa record juga pada entity B, begitu juga sebaliknya, beberapa record pada entity B dapat ber-relasi dengan beberapa record juga pada entity A.. Dalam diagram E-R, relasi ini disimbolkan dengan huruf **M** atau **N** untuk menyatakan banyak..

Contoh : Dalam hubungan antara mahasiswa dengan dosen pada perguruan tinggi, yaitu seorang seorang dosen mengajar banyak mahasiswa, sebaliknya seorang mahasiswa dapat diajar oleh beberapa dosen, sehingga terjadihubungan **banyak ke banyak**. Perhatikan diagram E-R berikut :



**Keterangan :**

* Entity dosen mempunya dua attribute, yaitu kode dosen (**kd\_dos**) yang berfungsi sebagai field kunci, dan nama dosen (**nm\_dos**).
* Entity mahasiswa juga mempunyai dua attribute, yaitu No. Buku Pokok (**NoBP**) sebagai field kunci dan nama mahasiswa (**nm\_mhs**).
* Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode dosen (**kd\_dos**) dan No.Buku Pokok mahasiswa (**NoBP**) yang berfungsi sebagai kunci tamu (foreign key) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan tempat mengajar (**lokal**).
* Derajat relasi dinyatakan dengan **M : N**, yang menandakan bahwa hubungan antar entity adalah banyak ke banyak, seperti terlihat pada gambar diatas.

1. Tahapan Membuat Diagram E-R.

Berikut panduan bagaimana tahapan dalam membuat Diagram E-R;

1. **Mengidentifikasi dan menetapkan seluruh entity yang terlibat dalam sistem database tersebut.**

Sebagaimana telah dijelaskan diawal, bahwa entity merupakan sesuatu yang dapat diidentifikasi dengan mudah dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian atau konsep yang informasinya akan disimpan. Dari sekian banyak kemungkinan entity yang ada, anda harus memilah-milah entity mana saja yang sesuai dan mampu mengakomodasi kebutuhan sistem yang akan dirancang. Misalnya dalam ruang lingkup perkuliahan, ada banyak entity yang mungkin, misalnya entity mahasiswa, mata kuliah, dosen, asistensi, jadwal, jurusan dan lain sebagainya. Namun dalam ruang lingkup perkuliahan sederhana, anda dapat menggunakan tiga entity saja, yaitu entity mahasiswa, dosen dan mata kuliah. Proses menentukan entity ini memang agak sulit tapi akan menjadi mudah apabila diserta latihan dengan berbagai macam kasus database.

1. **Menentukan attribute-attribute atau field dari masing-masing entity beserta kunci (key)-nya.**

Menentukan attribute dari suatu entity sangat menentukan baik atau tidaknya sistem database yang dirancang, karena attribute ini sangat menentukan nantinya dalam proses relasi. Attribute merupakan ciri khas yang melekat pada suatu entity, misalnya attribute pada mahasiswa dapat berupa nobp, nama, tempat lahir, tanggal lahir, alamat, nama orang tua, pekerjaan orang tua dan lain-lain. Dari sekian banyak kemungkinan attribute yang ada pada entity mahasiswa, anda dapat menggunakan hanya yang perlu saja. Setelah menentukan attributenya selanjutnya adalah menentukan field kunci. Field kunci adalah penanda entity tersebut sehingga bisa digunakan untuk relasi nantinya dan field kunci ini harus bersifat unik. Misalnya pada entity mahasiswa, attribute nobp bisa dijadikan field kunci, karena bersifat unik dan tidak ada mahasiswa yang mempunyai nobp sama.

1. **Mengidentifikasi dan menetapkan seluruh himpunan relasi diantara himpunan-himpunan entity yang ada beserta kunci tamu (foreign key)-nya.**

Setelah menentukan entity dan attribute beserta field kuncinya, maka selanjutnya adalah menentukan entity yang terbentuk akibat adanya relasi antar entity. Misalnya antara entity mahasiswa dengan entity dosen, terjadi suatu hubungan proses mengajar, maka proses mengajar ini merupakan entity baru. Entity mengajar ini harus anda tentukan juga attribute yang melekat padanya beserta kunci tamu (foreign key). Kunci tamu adalah field kunci utama pada tabel lain, dan field tersebut digunakan juga pada tabel yang satu lagi. Misalnya nobp adalah field kunci dari entity mahasiswa, pada entity mengajar terdapat juga attribute NoBP, maka keberadaan attribute nobp pada entity mengajar disebut sebagai kunci tamu. Proses menentukan hubungan antar entity juga sangat menentukan kualitas sistem database yang dirancang.

1. **Menentukan derajat relasi untuk setiap himpunan relasi.**

Setelah semua entity dan attribute yang dibutuhkan terbentuk, maka selanjutnya adalah menentukan derajat relasi antar entity tersebut, apakah satu kesatu, satu ke banyak atau sebaliknya, atau banyak ke banyak. Berhati-hatilah dalam menentukan derajat relasi ini, karena nantinya akan berhubungan dengan proses query terhadap data.

BAB 3

BAHASA SQL (STRUCTURE QUERY LANGUAGE)

1. Komponen SQL
   1. Data Definition Language (DDL) :

Digunakan untuk mendefinisikan data dengan menggunakan perintah : create, drop, alter.

* 1. Data Manipulation Language (DML) :

Digunakan untuk memanipulasi data dengan menggunakan perintah : select, insert, update, delete. Data Manipulation Language merupakan bagian terpadu bahasa SQL. Perintah-perintahnya dapat dibuat secara interaktif atau ditempelkan pada sebuah program aplikasi. Pemakai hanya perlu menentukan 'APA' yang ia inginkan, DBMS menentukan 'BAGAIMANA' cara mendapatkannya.

* 1. Data Control Language (DCL) :

Digunakan untuk mengontrol hak para pemakai data dengan perintah : grant, revoke

1. Data Definition Language
   * 1. CREATE TABLE

Fungsi : membuat tabel

Sintaks : CREATE TABLE tbname

(col 1 data type data spec,

col 2 data type data spec,

.

.

PRIMARY KEY (col1,……))

Contoh :

CREATE TABLE PERSONEL

(REGNO CHAR(10) NOT NULL,

NAME CHAR(45) NOT NULL, ADDRESS CHAR(45),

BIRTH DATE NOT NULL WITH DEFAULT,

PRIMARY KEY (REGNO))

**NULL**

Spesifikasi NULL, NOT NULL, NOT NULL WITH DEFAULT

NULL :

dapat diinterpretasikan sebagai nilai yang tidak diketahui atau tidak tersedianya suatu nilai. Null bukan berarti kosong (blank) atau 0 (Nol)

NOT NULL :

pemakai atau program harus memberikan nilai-nilai pada saat memasukkan record

NOT NULL WITH DEFAULT :

nilai default disimpan pada saat record dimasukkan tanpa nilai yang ditentukan untuk kolom ini.

Nilai default-nya :

Nol untuk tipe field NUMERIC

Blank untuk tipe field CHARACTER

CURRENT DATE untuk tipe field DATE

CURRENT TIME untuk tipe field TIME

Pada saat membuat tabel, salah satu atribut tersebut di atas dispesifikasikan pada sebuah kolom.

* + 1. CREATE VIEW

Fungsi : membuat tabel view.

View merupakan bentuk alternatif penyajian data dari satu atau lebih tabel. View dapat berisi semua atau sebagian kolom yang terdapat pada tabel dimana kolom tersebut didefinisikan.

Tujuan membuat view :

* + Meningkatkan keamanan data
  + Meningkatkan kemandirian data
  + Penyederhanaan bagi end user (data yang sedikit, nama-nama kolom yang baru dan dapat dibaca dengan lebih baik)

Properti :

* + Tidak terdapatnya data tambahan
  + View mencakup subset kolom dan / atau baris
  + View dapat berisikan data dari beberapa tabel dan / atau tabel-tabel view lainnya
  + View dapat berisikan perolehan data, misal : nilai rata-rata

Sintaks : CREATE VIEW viewname (column1, column2, ……..)

AS SELECT statement FROM tbname

[WITH CHECK OPTION]

Keterangan : View-name : nama view yang akan dibuat. Column : nama atribut untuk view Statement : atribut yang dipilih dari tabel basis data. Tabel-name : nama tabel basis data.

Contoh :

CREATE VIEW VPERSON (REGNO, NAME) AS

SELECT REGNO, NAME FROM PAUL.PERSONEL

* + 1. CREATE INDEX

Fungsi : membuat index

Sintaks : CREATE [UNIQUE] INDEX indexname ON nama\_table (nama\_kolom)

Contoh :

CREATE UNIQUE INDEX PRSONIDX

ON PERSONEL(REGNO)

Dengan indeks memungkinkan suatu tabel diakses dengan urutan tertentu tanpa harus merubah urutan fisik dari datanya dan dapat pula diakses secara cepat melalui indeks yang dibuat berdasar nilai field tertentu. Spesifikasi UNIQUE akan menolak key yang sama dalam file.

* + 1. DROP TABLE

Fungsi : menghapus Tabel

Sintaks : DROP TABLE tbname

Contoh : DROP TABLE PERSONEL

Dengan perintah itu obyek lain yang berhubungan dengan tabel tersebut otomatis akan dihapus atau tidak akan berfungsi seperti :

- semua record dalam tabel akan terhapus

- index dan view pada tabel akan hilang

- deskripsi tabel akan hilang

* + 1. DROP VIEW

Fungsi : menghapus view

Sintaks : DROP VIEW viewname

Contoh : DROP VIEW VPERSON

* + 1. DROP INDEX

Fungsi : menghapus index Sintaks : DROP INDEX indexname

Contoh : DROP INDEX PRSONIDX

* + 1. ALTER

Fungsi : merubah atribut pada suatu tabel

Sintaks : ALTER TABLE tbname

MODIFY (nama\_kolom tipe\_kolom)

ADD (nama\_kolom tipe\_kolom [[before, nama\_kolom]])

DROP (nama\_kolom tipe\_kolom)

Contoh : merubah Tabel TABX dengan menambah Field D.

ALTER TABLE TABX

ADD D CHAR(3)

1. DATA MANIPULATION LANGUAGE
2. INSERT

Fungsi : menambah baris (record) baru

Sintaks : INSERT INTO tbname (col1, ...) VALUES (value1, ...)

Catatan : Sintaks tersebut dapat digunakan jika jumlah kolom = jumlah nilai, tetapi jika dalam tabel semua kolom akan diisi dapat digunakan sintaks berikut ini :

Sintaks : INSERT INTO tbname VALUES (value1, value2, ...)

Nilai-nilai diisikan sebanyak kolom yang terdapat di tabel tersebut.

1. UPDATE

Fungsi : merubah record

Sintaks : UPDATE tbname SET field = ekspresi WHERE kondisi

1. DELETE

Fungsi : menghapus record

Sintaks : DELETE FROM tbname WHERE kondisi

1. SELECT

Fungsi : menampilkan record

Sintaks : SELECT [DISTINCT] colname FROM tbname [WHERE kondisi]

[GROUP BY kondisi] [HAVING kondisi] [ORDER BY kondisi]

Contoh Kasus DML :

* Menambah record (INSERT)

INSERT INTO S VALUES ('S1','Smith',20,'London'); INSERT INTO S VALUES ('S2','Jones,10,'Paris'); INSERT INTO S VALUES ('S3','Blake',30,'Paris')

* Merubah record (UPDATE)

1. Merubah data (record) pada tabel P yang mempunyai nomor part P2, warnanya dirubah menjadi Kuning dan beratnya ditambah 5

UPDATE P SET Warna = 'Yellow', Weight = Weight + 5 WHERE Pn = 'P2'

1. Merubah record pada tabel S, statusnya menjadi dua kali status awal untuk supplier yang bertempat tinggal di kota London

UPDATE S SET Status = 2 \* Status WHERE City = 'London'

* Menghapus record (DELETE)

Menghapus record pada tabel S yang nomor supplier-nya S5

DELETE FROM S WHERE Sn ='S5'

* Menampilkan record (SELECT 1 tabel)

1. Menampilkan semua data supplier SELECT \* FROM S

atau

SELECT Sn, Sname, Status, City FROM S

1. Menampilkan semua nilai Pn pada tabel SP

SELECT Pn FROM SP

1. Menampilkan nomor supplier dan status untuk supplier yang tinggal di Paris

SELECT Sn, Status FROM S WHERE City ='Paris'

1. Menampilkan no.supplier yang tinggal di Paris dengan status > 20

SELECT Sn FROM S WHERE City ='Paris" AND Status > 20

1. Menampilkan jumlah pengiriman P1

SELECT COUNT(\*) FROM SP WHERE Pn = 'P1'

1. Perintah untuk menghindari hasil data yang sama terulang kembali (distinct)

SELECT DISTINCT Pn FROM SP

1. Menampilkan no.supplier dan status bagi supplier yang tinggal di Paris dalam urutan status menurun

SELECT Sn,Status FROM S WHERE City = 'Paris'

ORDER BY Status desc

1. Menampilkan no.Part dari semua part yang dipasok oleh lebih dari seorang supplier

SELECT Pn FROM SP GROUP BY Pn HAVING COUNT(\*) > 1

1. Menampilkan semua part yang nomornya dimulai dengan huruf C

SELECT \* FROM P WHERE Pname LIKE 'C%'

* Menampilkan record (SELECT lebih dari satu tabel / JOIN)

1. Menampilkan semua supplier dan part yang keduanya bertempat tinggal pada kota yang sama

SELECT Sn, Sname,S tatus, S.City , Pn, Pname, Warna, Weight FROM S,P WHERE S.City = P.City

1. Menampilkan nama supplier yang memasok barang dengan nomor part P2

SELECT Sname FROM S, SP WHERE S.Sn = SP.Sn AND SP.Pn = 'P2'

Menampilkan nama supplier yang memasok part berwarna merah

SELECT Sname FROM S, SP, P WHERE S.Sn = SP.Sn AND SP.Pn = P.Pn AND P.COLOR = 'RED'

* Menampilkan record (SELECT lebih dari satu tabel / SELECT Bertingkat)

1. Menampilkan nama supplier yang memasok barang dengan nomor part P2

SELECT Sname FROM S WHERE Sn IN (SELECT Sn FROM SP WHERE Pn = 'P2')

atau

SELECT Sname FROM S WHERE Sn = ANY (SELECT Sn FROM SP WHERE Pn = 'P2')

1. Menampilkan nama supplier yang memasok part berwarna merah

SELECT Sname FROM S WHERE Sn IN (SELECT Sn FROM SP WHERE Pn IN

(SELECT Pn FROM P WHERE Warna = 'Red'))

1. Menampilkan no.supplier dengan nilai status lebih kecil daripada nilai maksimum status yang ada pada tabel S

SELECT Sn FROM S WHERE Status < (SELECT MAX(Status) FROM S)

1. Menampilkan nama supplier yang tidak memasok barang dengan nomor part P2

SELECT Sname FROM S WHERE Sn NOT IN (SELECT Sn FROM SP WHERE Pn = 'P2')

1. Menampilkan semua nomor supplier yang sama lokasinya dengan S1

SELECT Sn FROM S WHERE CITY = (SELECT CITY FROM S WHERE Sn = 'S1')

* Fungsi Perhitungan

COUNT : jumlah baris dan kolom SUM : jumlah nilai dam kolom AVG : rata - rata nilai dalam kolom MAX : nilai terbesar dalam kolom MIN : nilai terkecil dalam kolom

Untuk SUM dan AVG nilainya harus numerik (INT, SMALLINT, FLOAT). Fungsi- fungsi tsb jika dikenakan pada nilai yang NULL maka nilainya akan diabaikan kecuali untuk COUNT(\*)

1. Menghitung jumlah supplier

SELECT COUNT(\*) FROM S

atau

SELECT COUNT (Sn) FROM S

1. Menampilkan nomor part dan total kuantitas pengiriman dari setiap part SELECT Pn, SUM(QTY) FROM SP GROUP BY Pn
2. Menghitung jumlah kuantitas dari P2 yang telah disupply SELECT SUM (QTY) FROM SP WHERE Pn = 'P2'
3. Menampilkan jumlah pengiriman barang dengan nomor P4 dan dipasok oleh nomor suppplier S1

SELECT COUNT(\*) FROM SP WHERE Pn = 'P4' AND Sn = 'S1'

1. Menampilkan nomor part dan total kuantitas dari masing-masing part

SELECT Pn, SUM(QTY) FROM SP GROUP BY P3

DAFTAR PUSTAKA

Yuhefizar, “Database Management Menggunakah MS Access 2003”, Elex Media, , 2008

­­­­