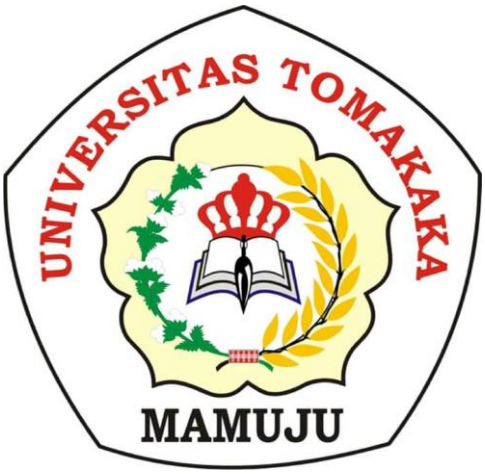
KIKI WANDA

200250502027



TEKNIK INFORMATIKA

UNIVERSITAS TOMAKAKA MAMUJU

SQL Create Database

Digunakan untuk membuat database baru.

Syntax dasar:

CREATE DATABASE database_nama

Contoh:

CREATE DATABASE databaru

SQL Create Table

Digunakan untuk membuat tabel data baru dalam sebuah database.

Syntax dasar:

CREATE TABLE

(

Column_name1 table_nama data_type

Column_name2 table_nama data_type

Column_name3 table_nama data_type

)

Contoh:

CREATE TABLE tabeldata

(Id int, Nama varchar (255), Email varchar(50),

Kota varchar(255))

SQL Select

Digunakan untuk memilih data dari table database.

Syntax dasar:

SELECT column_name(s)

FROM table_name

Atau

SELECT * FROM table_name

Contoh 1:

SELECT nama,email FROM tabeldata

Contoh 2:

SELECT * FROM tabeldata

SQL Select Distinct

Digunakan untuk memilih data-data yang berbeda (menghilangkan duplikasi) dari sebuah table database.

Syntax dasar:

SELECT DISTINCT column_name(s)

FROM table_name

Contoh:

SELECT DISTINCT kota FROM tabeldata

SQL Where

Digunakan untuk memfilter data pada perintah Select

Syntax dasar:

SELECT column name(s)

FROM table_name

WHERE column_name operator value

Contoh:

SELECT * FROM tabeldata WHERE kota='Kediri'

SQLOrder By

Digunakan untuk mengurutkan data berdasarkan kolom (field) tertentu. Secara default, urutan tersusun secara ascending (urut kecil ke besar). Anda dapat mengubahnya menjadi descending (urut besar ke kecil) dengan menambahkan perintah DESC.

Syntax dasar:

SELECT column_name(s)

FROM table_name

ORDER BY column_name(s) ASC|DESC

Contoh 1:

SELECT * FROM tabeldata ORDER BY nama

Contoh 2:

SELECT * FROM tabeldata ORDER BY id DESC

SQL Like

Digunakan bersama dengan perintah Where, untuk proses pencarian data dengan spesifikasi tertentu.

Syntax dasar:

SELECT column_name(s)

FROM table_name

WHERE column_name LIKE pattern

Contoh 1:

SELECT * FROM tabeldata WHERE nama LIKE ‘z%’

Keterangan :

Contoh di atas digunakan untuk pencarian berdasarkan kolom nama yang berhuruf depan “z”.

Contoh 2:

SELECT * FROM tabeldata WHERE nama LIKE ‘z%’

Keterangan :

Contoh di atas digunakan untuk pencarian berdberdasarkan kolom nama yang berhuruf belakang “z”.

SQL In

Digunakan untuk pencarian data menggunakan lebih dari satu filter pada perintah Where.

Syntax dasar :

SELECT column_name(s)

FROM table_name

WHERE column_name IN (value1,value2, . . .)

Contoh:

SELECT * FROM tabeldata WHERE kota IN (‘kediri’,’malang’)

SQL Between

Digunakan untuk menentukan jangkauan pencarian.

Syntax dasar:

SELECT column_name(s)

FROM table_name

WHERE column_name

BETWEEN value1 AND value2

Contoh :

SELECT * FROM tabeldata WHERE id BETWEEN 15 and 45

Keterangan :

Contoh di atas digunakan untuk mencari data yang memiliki nomor id antara 15 dan 45.

SQL Insert Into

Digunakan untuk menambahkan data baru di tabel database.

Syntax dasar :

INSERT INTO table_name

VALUES (value1,value2,value3, . . .)

Atau

INSERT INTO table_name (column1,column2,column3, . . .)

VALUES (value1,value2,value3, . . .)

Contoh 1:

INSERT INTO tabeldata VALUES (1,’redaksi’,’redaksi@kuliahkomputer.com’,’Kediri’)

Contoh 2 :

INSERT INTO tabeldata (id,nama,email,kota) VALUES
(1,’redaksi’,’redaksi@kuliahkomputer.com’,’Kediri’)

SQL Update

Digunakan untuk mengubah/memperbarui data di tabel database.

Syntax dasar:

UPDATE table_name

SET column1=value,column2=value, . . .

WHERE some_column=some_value

Contoh :

UPDATE tabeldata SET email='redaksi@kuliahkomputer.com', kota='Kediri'

WHERE id =1

SQL Delete

Digunakan untuk menghapus data di table database. Tambahkan perintah Where untuk memfilter data-data tertentu yang akan dihapus. Jika tanpa perintah Where, maka seluruh data dalam tabel akan terhapus.

Syntax dasar :

DELETE FROM table_name

WHERE some_column=some_value

Contoh:

DELETE FROM tabeldata WHERE id=1

SQL Inner Join

Digunakan untuk menghasilkan baris data dengan cara menggabungkan 2 buah tabel atau lebih menggunakan pasangan data yang match pada masing-masing tabel. Perintah ini sama dengan perintah join yang sering digunakan.

Syntax dasar :

SELECT column_name(s)

FROM table_name1

INNER JOIN table_name2

ON table_name1.column_name=table_name2

column-name

contoh :

SELECT tabeldata.nama,tabeldata.email,order.no_order

FROM tabeldata

INNER JOIN order

ON tabeldata.id=order.id

ORDER BY tabeldata.nama

SQL Left Join

Digunakan untuk menghasilkan baris data dari tabel kiri (nama tabel pertama) yang tidak ada pasangan datanya pada tabel kanan (nama tabel kedua).

Syntax dasar :

SELECT column_name(s)

FROM table_name1

LEFT JOIN table_name2

ON table_name1.column_name=table_name2.

column_name

contoh :

SELECT tabeldata.nama,tabeldata.email,order.no_order

FROM tabeldata

LEFT JOIN order

ON tabeldata.id=order.id

ORDER BY tabeldata.nama

SQL Right Join

Digunakan untuk menghasilkan baris data dari tabel kanan (nama tabel kedua) yang tidak ada pasangan datanya pada tabel kiri (nama tabel pertama).

Syntax dasar :

SELECT column_name(s)

FROM table_name1

RIGHT JOIN table_name2

ON table_name1.column_name=table_name2

column_name

contoh :

SELECT tabeldata.nama,tabeldata.emailmorder.no_order

FROM tabeldata

RIGHT JOIN order

ON tabeldata.id=order.i

ORDER BY tabeldata.nama

SQL Full Join

Digunakan untuk menghasilkan baris data jika ada data yang sama pada salah satu tabel.

Syntax dasar :

SELECT column_name(s)

FROM table_name1

FULL JOIN table_name2

ON table_name1.column_name=table_name2

column_name

Contoh :

SELECT tabeldata.nama,tabeldata.email,order.no_order

FROM tabeldata

FULL JOIN order

ON tabeldata.id=order.id

ORDER BY tabeldata.nama

SQL Union

Digunakan untuk menggabungkan hasil dari 2 atau lebih perintah Select.

Syntax dasar :

SELECT column_name(s)FROM table_name1

UNION column_name(s) FROM table_name2

Atau

SELECT column_name(s) FROM table_name1

UNION ALL

SELECT column_name(s) FROM table_name2

Contoh :

SELECT nama FROM mhs_kampus1

UNION

SELECT nama FROM mhs_kampus2

SQL Alter Table

Digunakan untuk menambah, menghapus, atau mengubah kolom (field) pada tabel yang sudah ada.

Syntax untuk menambah kolom :

ALTAR TABLE table_name

ADD column_name datatyoe

Contoh :

ALTER TABLE Persons

ADD DateOfBirth date

Syntax untuk menghapus kolom :

ALTER TABLE table_name

DROP COLUMN column_name

Contoh :

ALTER TABLE Persons

DROP COLUMN DateOfBirth

Syntax untuk mengubah kolom :

ALTER TABLE table_name

ALTER TABLE clumn_name datatype

Contoh :

ALTER TABLE Persons

ALTER COLUMN DateOfBirth year

19. Drop Table

Digunakan untuk menghapus tabel beserta seluruh datanya.

Syntax dasar :

DROP TABLE table_name

Contoh :

DROP TABLE mhs

SQL Drop Database

Digunakan untuk menghapus database.

Syntax dasar :

DROP DATABASE database_name

SQL Count

Digunakan untuk menghitung jumlah (cacah) suatu data.

Syntax dasar :

SELECT COUNT (column_name) FROM table_name

Contoh :

SELECT COUNT(id) AS Jumlah_tamu FROM tabeldata Max()

2021

Digunakan untuk mendapatkan nilai terbesar dari data-data yang ada.

Syntax dasar :

SELECT MAX (column_name) FROM table_name

Contoh :

SELECT MAX(harga) AS Harga_termahal FROM order

SQL Min

Digunakan untuk mendapatkan nilai terkecil dari data-data yang ada.

Syntax dasar :

SELECT MIN (column_name) FROM table_name

Contoh:

SELECT MIN(harga) AS Harga_termurah FROM order

SQL Sum

Digunakan untuk mendapatkan nilai total penjumlahan dari data-data yang ada.

Syntax dasar :

SELECT SUM (column_name) FROM table_name

Contoh :

SELECT SUM(harga) AS Harga_total FROM order

SQL AVG

Digunakan untuk menghitung nilai-rata-rata dari suatu data.

Syntax dasar :

SELECT AVG (column_name) FROM table_name

Contoh :

SELECT AVG(harga) AS Harga_rata2FROM order

SQL Group By

Digunakan untuk mengelompokkan data dengan kriteria tertentu.

Syntax dasar :

SELECT column_name,aggregate_function(column_name)

FROM table_name

WHERE column_name operator value

GROUP BY column_name

Contoh :

SELECT nama_customer,SUM(harga) FROM order GROUP BY nama_customer

SQL. Having

Digunakan untuk memfilter data dengan fungsi tertentu.

Syntax dasar :

SELECT column_name,aggregate_function(column_name)

FROM table_name

WHERE column_name operator value

GROUP BY column_name

HAVING aggregate_function(column_name) operator value

Contoh :

SELECT nama_customer,SUM(harga) FROM order

WHERE nama_customer='Rico' OR nama_customer='Bastian'

GROUP BY nama_customer

HAVING SUM (harga)>125000

SQL Insert Into

Digunakan untuk menambahkan data baru di tabel database.

Syntax dasar :

INSERT INTO table_name

VALUES (value1,value2,value3, . . .)

Atau

INSERT INTO table_name (column1,column2,column3, . . .)

VALUES (value1,value2,value3, . . .)

Contoh 1:

INSERT INTO tabeldata VALUES (1,'redaksi','redaksi@kuliahkomputer.com','Kediri')

Contoh 2:

INSERT INTO tabeldata (id,nama,email,kota) VALUES
(1,'redaksi','redaksi@kuliahkomputer.com','Kediri')

SQL And, Or, Not

Operator AND, OR dan NOT merupakan perintah dasar SQL yang biasanya di kombinasikan dengan perintah WHERE. Ketiganya di gunakan untuk mem-filter record berdasarkan suatu kondisi, operator AND akan menampilkan record apabila semua kondisi bernilai TRUE, operator OR akan menampilkan record apabila salah satu kondisi bernilai TRUE, sedangkan operator NOT akan menampilkan record apabila semua kondisi bernilai FALSE.

Sintaks AND

SELECT kolom1, kolom2, ... FROM nama_tabel WHERE kondisi1 AND kondisi2 AND kondisi3 ...;

Contoh AND

SELECT nis, nama FROM siswa WHERE alamat='Jakarta' AND tahun_lahir='2000'

Perintah di atas akan menampilkan record NIS dan NAMA dari tabel SISWA dengan ALAMAT di JAKARTA dan TAHUN_LAHIR “2000”, artinya record siswa yang lahir di tahun “2000” namun tidak beralamat di “Jakarta: atau siswa yang beralamat di “Jakarta” namaun lahir bukan pada tahun “2000” tidak akan di tampilkan.

Sintaks OR

SELECT kolom1, kolom2, ... FROM nama_tabel WHERE kondisi1 OR kondisi2 OR kondisi3 ...;

Contoh OR

SELECT nis, nama FROM siswa WHERE alamat='Jakarta' OR alamat='Bandung'

Perintah di atas akan menampilkan record NIS dan NAMA dari tabel SISWA dengan ALAMAT di “JAKARTA” atau di “Bandung”, artinya record siswa yang beralamat di “Jakarta” dan di “Bandung” saja yang akan di tampilkan.

Sintaks NOT

SELECT kolom1, kolom2, ... FROM nama_tabel WHERE NOT kondisi;

Contoh NOT

SELECT nis, nama FROM siswa WHERE NOT alamat='Jakarta'

case

Ucase() : Berfungsi untuk mengubah huruf pada data tertentu menjadi huruf besar, Contoh : $sql=mysql_query(“SELECT UCASE (nama_lengkap) as nama FROM mahasiswa”);

Lcase() : Berfungsi untuk mengubah huruf pada data tertentu menjadi huruf kecil, Contoh : $sql=mysql_query(“SELECT LCASE(nama_lengkap) as nama FROM mahasiswa”);

SQL SELF JOIN

Self join adalah suatu bentuk kondisi join tau penggabungan yang terjadi pada dua tabel yang sama kondisinya dari kedua tabel tersebut dan menambahkan suatu kata dengan kondisi yang sama antara kedua tabel

Contoh sintak

Select tabel1.namakolom ||' works for' || tabel2.namakolom

from namatabel1, namatabel2

where namatabel1.kolompersmaantabel1 = namakolom2.kolompersamaantabel2

SQL view

view merupakan bentuk alternatif penyajian data dari satu tabel atau lebih, beberapa tujuan membuat view adalah meningkatkan keamanan data serta penyederhanaan bagi para pengguna.

Contoh sintak

Create view namaview (kolom1, kolom2, ...)

as select statement from namatabel

[with check option]

Keterangan :

Namaview : nama view yang dibuat

Column : nama atribut untuk view

Statement : atribut yang dipilih dari tabel database

Namatabel : nama tabel yang ada pada basis data

SQL index

yaitu berfungsi untuk membuat index

Contoh sintak

create [unique] index namaindex

on namatabel (namakolom)

NULL

digunakan untuk menguji nilai kosong (nilai NULL).

Syntax in Null

SELECT column_names

FROM table_name

WHERE column_name IS NULL;

Syntal is not null

SELECT column_names

FROM table_name

WHERE column_name IS NOT NULL;

contoh :

SELECT NamaPelanggan, NamaKontak, Alamat

FROM Pelanggan

WHERE Alamat IS NULL;

SQL SELECT TOP

digunakan untuk membatasi jumlah baris data yang dihasilkan oleh suatu perintah atau query SELECT. Pembatasan jumlah baris data hasil query ini diperlukan apabila suatu query menghasilkan jumlah baris data yang sangat banyak sehingga mempengaruhi kecepatan eksekusi query tersebut. Perintah SELECT TOP mempunyai format yang berbeda-beda untuk setiap tipe database.

contoh :

SELECT TOP number|percent column_name(s)

FROM table_name

WHERE condition;

Berikut ini menunjukkan sintaks dari klausa TOP dengan pernyataan SELECT:

Berikut ini menunjukkan sintaks dari klausa TOP dengan pernyataan SELECT:

SELECT TOP (expression)

[PERCENT] [WITH TIES]

FROM table_name

ORDER BY column_name;

SQL EILDCARDS

Karakter wildcard digunakan untuk menggantikan satu atau lebih karakter dalam string.

Pernyataan SQL berikut memilih semua customer dengan Kota yang diawali dengan “ber”:

Contoh :

```
SELECT * FROM Customer

WHERE Kota LIKE 'ber%';
```

contoh :

```
SELECT * FROM Customer

WHERE Kota LIKE 'L_n_on';
```

SQL Aliases

alias SQL digunakan untuk memberikan tabel database, atau kolom dalam sebuah tabel, nama sementara. Pada dasarnya alias diciptakan untuk membuat nama kolom lebih mudah dibaca.

SQL Alias sintaks untuk Kolom :

```
SELECT column_name AS alias_name

FROM table_name;
```

SQL Alias sintaks untuk Tabel :

```
SELECT column_name(s)

FROM table_name AS alias_name;
```

Contoh:

Aliases untuk tabel kolom

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]

FROM Customers;
```

Aliases untuk Tabel

```
SELECT o.OrderID, o.OrderDate, c.CustomerName

FROM Customers AS c, Orders AS o

WHERE c.CustomerName="Around the Horn" AND c.CustomerID=o.CustomerID;
```

SQL EXISTS

digunakan untuk menguji keberadaan rekaman apa pun di subkueri. Operator EXISTS dapat mengembalikan nilai true jika subkueri mengembalikan satu atau lebih rekaman.

Syntax :

SELECT column_name(s)

FROM table_name

WHERE EXISTS

(SELECT column_name FROM table_name WHERE condition);

Contoh:

SELECT NamaSupplier

FROM Suppliers

WHERE EXISTS (SELECT NamaProduk FROM Produk WHERE Produk.IdSupplier = Suppliers.IdSupplier AND Harga < 30);

SQL ANY dan ALL

dapat digunakan dengan klausa WHERE atau HAVING. Operator ANY akan mengembalikan true jika salah satu nilai subquery memenuhi kondisi.

Operator ALL akan mengembalikan true jika semua nilai subkueri memenuhi kondisi.

Syintax ANI

SELECT column_name(s)

FROM table_name

WHERE column_name operator ANY

(SELECT column_name FROM table_name WHERE condition);

Syintax ALL

SELECT column_name(s)

FROM table_name

WHERE column_name operator ALL

(SELECT column_name FROM table_name WHERE condition);

Catatan: : Operator harus menjadi operator perbandingan standar (=, <>,! =,>,> =, <, Atau <=)

Contoh :

```
SELECT NamaProduk
FROM Products
WHERE IdProduct = ANY (SELECT IdProduct FROM OrderDetails WHERE Quantity = 12);
```

SQL Select Into

statement SELECT INTO digunakan untuk dapat menyalin data dari satu tabel ke tabel baru.

Syntax select into

salin semua kolom kedalam tabel baru

```
SELECT *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

Syntax :

salin hanya beberapa kolom ke dalam tabel baru :

```
SELECT *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

contoh :

Pernyataan SQL berikut membuat salinan cadangan customer.

```
SELECT * INTO SalinDataCustomer2017
FROM Customers;
```

pernyataan SQL berikut menggunakan klausa IN untuk menyalin tabel ke tabel baru di database lain:

```
SELECT * INTO SalinDataCustomer2017 IN 'Backup.mdb'
FROM Customers;
```

SQL Stored Procedured

(Prosedur tersimpan) adalah kode SQL yang disiapkan untuk dapat di simpan, sehingga kode tersebut dapat digunakan kembali berulang kali. Jadi, jika memiliki kueri SQL yang ingin ditulis berulang kali, simpan sebagai store procedure (prosedur tersimpan), lalu panggil saja untuk menjalankannya. Kita juga bisa menggunakannya untuk meneruskan parameter ke store procedure (prosedur tersimpan), sehingga store procedure (prosedur tersimpan) dapat bertindak berdasarkan nilai parameter yang diteruskan.

Syntax Stored Procedure

CREATE PROCEDURE procedure_name

AS

sql_statement

Contoh :

CREATE PROCEDURE SelectAllCustomers

AS

SELECT * FROM Customers

GO;

SQL Comment

digunakan untuk menjelaskan bagian dari pernyataan SQL atau untuk mencegah eksekusi pernyataan SQL. Catatan: Contoh ini tidak akan berfungsi di Firefox dan Microsoft Edge! Komentar tidak didukung di database Microsoft Access.

Single Line Comments

Single Line Comments dimulai dengan —.

Teks apa pun di antara — dan akhir baris akan diabaikan (tidak akan dieksekusi) Contoh berikut akan menggunakan single line comments sebagai penjelasan:

Contoh

--Select all:

SELECT * FROM Customers;

Contoh berikut menggunakan single line comments untuk mengabaikan akhir baris:

Contoh

SELECT * FROM Customers -- WHERE Kota='Berlin';

Contoh berikut menggunakan single line comments untuk mengabaikan pernyataan:

contoh

--SELECT * FROM Customers;

SELECT * FROM Produk;

Multi-line Comments

Multi-line Comments dimulai dengan /* dan diakhiri dengan */.

Teks apa pun antara /* dan */ akan diabaikan.

Contoh berikut akan menggunakan multi-line comments sebagai penjelasan:

Contoh

/*Pilih semua kolom

dari semua catatan

di tabel Customers:*/SELECT * FROM Customers;

Contoh berikut menggunakan multi-line comments untuk mengabaikan banyak pernyataan:

Contoh

/*SELECT * FROM Customers;

SELECT * FROM Products;

SELECT * FROM Orders;

SELECT * FROM Categories;*/SELECT * FROM Suppliers;

Untuk mengabaikan hanya sebagian dari pernyataan, gunakan juga comments /* */ .

Contoh berikut menggunakan komentar untuk mengabaikan bagian dari baris:

Contoh

SELECT NamaCustomer, /*Kota,*/ Negara FROM Customers;

Contoh berikut menggunakan komentar untuk mengabaikan bagian dari pernyataan:

Contoh

SELECT * FROM Customers WHERE (NamaCustomer LIKE 'L%'

OR NamaCustomer LIKE 'R%' /*OR NamaCustomer LIKE 'S%'

OR NamaCustomer LIKE 'T%'*/ OR NamaCustomer LIKE 'W%')

AND Negara='USA'

ORDER BY NamaCustomer;

SQL Operators

adalah kata atau karakter khusus yang digunakan pada SQL terutama dalam klausa WHERE dimana statement SQL untuk melakukan operasi, seperti perbandingan dan operasi aritmatika. Operator ini digunakan untuk menentukan kondisi dalam pernyataan SQL dan berfungsi sebagai konjungsi untuk beberapa kondisi dalam sebuah pernyataan.

Operator Aritmatika SQL, Contoh

```
SELECT 40 + 10;
```

Operator Bitwise Operator, contoh

& Bitwise AND

Operator Perbandingan SQL, Contoh

```
SELECT * FROM Produk WHERE Harga= 18;
```

Operator Logika SQL, contoh

SELECT * FROM Produk

WHERE Harga > ALL (SELECT Harga FROM Produk WHERE Harga > 500);

SQL Backup DB

Statement BACKUP DATABASE dapat digunakan di SQL Server untuk membuat cadangan penuh dari database SQL yang ada.

Syntax

BACKUP DATABASE databasename

TO DISK = 'filepath';

Statement BACKUP WITH DIFFERENTIAL SQL

Sebuah diferensial cadangan digunakan hanya untuk membuat cadangan bagian dari database yang telah berubah sejak cadangan database penuh terakhir.

Syntax

BACKUP DATABASE databasename

TO DISK = 'filepath'

WITH DIFFERENTIAL;

Contoh BACKUP DATABASE

Statement SQL berikut dapat membuat cadangan data penuh dari database “testDB” yang ada ke disk D:

Contoh

BACKUP DATABASE testDB

TO DISK = 'D:\backups\testDB.bak';

Tips: Selalu cadangkan database ke drive yang berbeda dari database sebenarnya. Kemudian, jika ingin mendapatkan disk crash, kita tidak akan kehilangan file cadangan bersama dengan database.

Contoh BACKUP DENGAN DIFERENSIAL

Statement SQL berikut membuat cadangan diferensial dari database “testDB”:

Contoh

BACKUP DATABASE testDB

TO DISK = 'D:\backups\testDB.bak'

WITH DIFFERENTIAL;

Tip: Pencadangan diferensial mengurangi waktu pencadangan (karena hanya data perubahan yang dicadangkan).

Tipe- tipe constraints :

Constraint

Keterangan

NOT NULL

Menentukan suatu kolom tidak boleh berisi NULL.

UNIQUE

Untuk mencegah suatu kolom memiliki 2 baris atau lebih berisi data yang sama.

PRIMARY KEY

Mengkombinasikan constraint NOT NULL dan UNIQUE dalam satu deklarasi.
Mengidentifikasikan secara unik setiap baris pada tabel.

FOREIGN KEY

Memaksakan nilai pada suatu tabel untuk bernilai sama dengan tabel lain.

CHECK

Menentukan suatu kondisi yang harus benar.

SQL NOT NULL

NOT NULL merupakan constraint yang digunakan untuk menjamin pengisian record ke sebuah tabel agar nilai record tersebut harus berisi data. Sebuah kolom jika di berikan constraint NOT NULL, maka kolom tersebut harus berisikan nilai untuk recordnya dan tidak boleh di kosongkan. NOT NULL hanya bisa di definisikan dalam sebuah tabel pada level kolom dan tidak bisa pada level tabel. Perbedaan antara NOT NULL dan PRIMARY KEY adalah bahwa setiap kolom yang dijadikan PRIMARY KEY pasti NOT NULL dan tidak berlaku sebaliknya.

– Berikut adalah contoh penggunaan constraint NOT NULL:

```
CREATE TABLE dosen (

nip INTEGER

CONSTRAINT PK_dosen_nip PRIMARY KEY NOT NULL ,

nama_dosen VARCHAR(45),

alamat_dosen VARCHAR(255)

);
```

SQL UNIQUE

Constraint UNIQUE merupakan sebuah constraint yang akan membatasi pengisian record yang sama kedalam sebuah kolom jika kolom tersebut di berikan constraint UNIQUE dalam sebuah tabel. Constraint ini hampir sama dengan PRIMARY KEY, yaitu menjamin bahwa setiap nilai record yang ada dalam sebuah kolom UNIQUE tidak boleh ada yang sama (unik satu sama lain).

UNIQUE dapat dibuat pada level kolom ataupun level tabel. Sama dengan constraint PRIMARY KEY, FOREIGN KEY, CHECK.

– Perintah dasar pada constraint unique :

```
CONSTRAINT U_(nama-tabel)_(nama-kolom) UNIQUE
```

Keterangan :

U = Singkatan dari UNIQUE

Nama-Tabel = Nama tabel tempat UNIQUE tersebut dibuat.

Nama-Kolom = Nama Kolom yang akan dijadikan UNIQUE

UNIQUE = Jenis constraint-nya, yaitu UNIQUE

– Berikut adalah contoh penggunaan UNIQUE dan PRIMARY KEY secara in-line:

```
CREATE TABLE dosen (
nip INTEGER
CONSTRAINT pk_dosen_nip PRIMARY KEY ,
nama_dosen VARCHAR(45) CONSTRAINT UQ_dosen_nama
UNIQUE ,
alamat_dosen VARCHAR(255)
);
```

SQL PRIMARY KEY

PRIMARY KEY atau Kunci Utama dalam sebuah tabel merupakan kunci yang akan membatasi pengisian record dalam sebuah tabel agar tidak duplikat (redundant). Syarat sebuah kolom/ field dijadikan PRIMARY KEY dalam sebuah tabel adalah unik dan tidak boleh kosong (NOT NULL). Artinya bahwa nilai record-record dalam kolom yang dijadikan PRIMARY KEY haruslah unik satu sama lain dan nilainya tidak boleh di kosongkan. Jika sebuah kolom di beri constraint PRIMARY KEY maka sudah pasti kolom tersebut nilainya tidak boleh kosong atau harus berisi data. Berikut adalah perintah dasar pembuatan constraint PRIMARY KEY:

– Perintah dasar pada constraint primary key :

CONSTRAINT pk_(nama-tabel)_(nama-kolom) [jenis_constraint]

Keterangan:

PK = Singkatan dari jenis constraint yaitu PRIMARY KEY

Nama-tabel = Nama tabel tempat constraint tersebut di buat

Nama-kolom = Nama kolom yang akan di beri constraint

Jenis_constraint = Jenis-jenis constraint yang akan dibuat (PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK.)

– Berikut merupakan contoh penggunaan PRIMARY KEY secara in-line:

```
CREATE TABLE dosen (
nip INTEGER PRIMARY KEY ,
nama_dosen VARCHAR(45),
alamat_dosen VARCHAR(255)
);
```

– Berikut merupakan contoh penggunaan PRIMARY KEY secara out-of-line:

```
CREATE TABLE dosen (
nip INTEGER,
nama_dosen VARCHAR(45),
alamat_dosen VARCHAR(255)
CONSTRAINT pk_dosen_nip PRIMARY KEY
);
```

SQL FOREIGN KEY

Foreign key disebut juga sebagai contraint Referential Integrity, constraint ini memastikan bahwa data pada suatu tabel didefinisikan pada tabel lainnya dan mengikat kedua tabel tersebut dalam hubungan parent/child atau referenced/dependent.

Saat kita menggunakan constraint FORIGN KEY, maka kita mengindetifikasikan bahwa suatu kolom pada suatu tabel harus ada pada kolom primary key atau unique pada tabel lain. Tabel dengan kolom primary key atau unique sebagai parent atau referenced, sedangkan tabel dengan foreign key sebagai child atau dependent.

– Perintah dasar pada constraint foreign key :

CONSTRAINT fk_(nama-tabel)_(nama-kolom) FOREIGN KEY REFERENCES tabel-referensi(kolom-referensi)

Keterangan:

FK = Singkatan dari FOREIGN KEY

Nama-Tabel = Nama tabel tempat FOREIGN KEY tersebut dibuat.

Nama-Kolom = Nama Kolom yang akan dijadikan FOREIGN KEY

FOREIGN KEY = Jenis constraint-nya, yaitu FOREIGN KEY

REFERENCES = Kata kunci untuk merelasikan tabel ini ke tabel utamanya (tabel

yang memiliki PRIMAR KEY)

Tabel-Referensi = Tabel yang akan dijadikan referensi (tabel utama).

Kolom-Referensi = Kolom yang menjadi referensi dari tabel utama.

– Berikut adalah contoh penggunaan FOREIGN KEY secara in-line dan out-of-line:

```
CREATE TABLE mahasiswa(
nim INTEGER CONSTRAINT pk_mahasiswa_nim PRIMARY
KEY,
nip INTEGER
CONSTRAINT fk_mahasiswa_nim FOREIGN KEY REFERENCES dosen(nip),
nama_mhs VARCHAR(45),
alamat_mhs VARCHAR(255)
);
CREATE TABLE mahasiswa(
nim INTEGER CONSTRAINT pk_mahasiswa_nim PRIMARY
KEY,
nip INTEGER
nama_mhs VARCHAR(45),
alamat_mhs VARCHAR(255),
CONSTRAINT fk_mahasiswa_nim FOREIGN KEY REFERENCES dosen(nip)
);
```

Secara default, constraint foreign key mencegah penghapusan data pada tabel parent, apabila data tersebut di referensi oleh tabel child. Tetapi, jika kita tidak menginginkan hal demikian, kita dapat mengatur database agar secara otomatis menjaga referensial integrity dengan 2 cara, yaitu dengan menghapus data pada tabel child menggunakan klausa ON DELETE CASCADE dan mengubah nilai referensi pada tabel chile menjadi NULL menggunakan klausa ON DELETE SET NULL.

– Contoh penggunaan ON DELETE CASCADE

```
CREATE TABLE mahasiswa(
```

```
nim INTEGER CONSTRAINT pk_mahasiswa_nim PRIMARY

KEY,

nip INTEGER,

CONSTRAINT fk_mahasiswa_nim FOREIGN KEY REFERENCES dosen(nip) ON DELETE CASCADE,

nama_mhs VARCHAR(45),

alamat_mhs VARCHAR(255)

);

– Contoh penggunaan ON DELETE SET NULL

CREATE TABLE mahasiswa(

nim INTEGER CONSTRAINT pk_mahasiswa_nim PRIMARY

KEY,

nip INTEGER,

CONSTRAINT fk_mahasiswa_nimFOREIGN KEY REFERENCES dosen(nip) ON DELETE SET NULL,

nama_mhs VARCHAR(45),

alamat_mhs VARCHAR(255)

);
```

FOREIGN KEY YANG MEREFERENSIKAN DIRINYA SENDIRI

Tabel parent dan child tidak terbatas pada dua tabel yang berbeda, namun bisa juga pada dua kolom yang berbeda pada tabel yang sama, ini disebut dengan sel-referencing foreign key. Contohnya adalah tabel Pegawai yang membutuhkan data atasan dari tiap-tiap pegawai, dan atasan tersebut juga merupakan seorang pegawai. Dengan konfigurasi ini hierarki dalam tabel bisa dibuat lebih fleksibel.

5. CHECK

Constraint CHECK fungsinya untuk pengecekan apakah sebuah kolom memenuhi sebuah kondisi spesifik yang dievaluasi dengan nilai boolean. Jika evaluasi menghasilkan nilai FLASE, maka oracle akan mengeluarkan eksepsi, dan klausa insert atau update akan gagal.

```
– Berikut adalah contoh penggunaan constraint CHECK dalam sebuah tabel:

CREATE TABLE rekening (
```

No_rekening CHAR(15) CONSTRAINT

pk_rekening_no_rekening NOT NULL,

Kode_cabang CHAR(5)

CONSTRAINT fk_rekening_kode_cabang FOREIGN KEY REFERENCES cabang_bank(kode_cabang),

Pin CHAR(6),

Saldo MONEY CONSTRAINT chk_rekening_saldo CHECK(saldo > 50000)

SQL DEFAULT

digunakan untuk dapat memberikan nilai default untuk kolom.

Nilai default akan ditambahkan ke semua record baru jika tidak ada nilai lain yang ditentukan.

SQL berikut menetapkan nilai DEFAULT untuk kolom “Kota” saat tabel “Persons” dibuat:

My SQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    Kota varchar(255) DEFAULT 'London'
);
```

Constrait DEFAULT juga dapat digunakan untuk memasukkan nilai sistem, dengan menggunakan fungsi seperti GETDATE():

```
CREATE TABLE Orders (
    TanggalPesanan date DEFAULT GETDATE()
);
```

SQL DEFAULT di ALTER TABLE

Untuk membuat batasan DEFAULT pada kolom “Kota” saat tabel sudah dibuat, gunakan SQL berikut:

MySQL:

```
ALTER TABLE Persons
ALTER Kota SET DEFAULT 'London';
```

SQL Server:

ALTER TABLE Persons

ADD CONSTRAINT df_Kota

DEFAULT 'London' FOR Kota;

MS Access:

ALTER TABLE Persons

ALTER COLUMN Kota SET DEFAULT 'London';

Oracle:

ALTER TABLE Persons

MODIFY Kota DEFAULT 'London';

DROP Constrait DEFAULT

Untuk menghapus constrait DEFAULT, gunakan SQL berikut ini:

MySQL:

ALTER TABLE Persons

ALTER Kota DROP DEFAULT;

Akses SQL Server / Oracle / MS:

ALTER TABLE Persons

ALTER COLUMN Kota DROP DEFAULT;

SQL Auto-increment

dapat digunakan untuk nomor unik yang dibuat secara otomatis saat record baru dimasukkan ke dalam tabel.

Syntax untuk MySQL

Pernyataan SQL berikut mendefinisikan kolom “Personid” jadi auto-increment di field primary key pada tabel “Person”:

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
```

```
    Usia int,
    PRIMARY KEY (Personid)
);
```

MySQL menggunakan keyword AUTO_INCREMENT untuk melakukan fitur auto-increment.

Secara default, nilai awal untuk AUTO_INCREMENT adalah 1 dan akan bertambah 1 untuk setiap record baru.

Untuk membiarkan urutan AUTO_INCREMENT dimulai dengan nilai lain, gunakan pernyataan SQL berikut ini:

```
ALTER TABLE Persons AUTO_INCREMENT=100;
```

Untuk memasukkan record baru ke dalam tabel “Persons”, kita TIDAK perlu menentukan nilai untuk kolom “Personid” (nilai unik akan ditambahkan secara otomatis):

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Larry','Monsen');
```

Pernyataan SQL di atas akan memasukkan record baru ke dalam tabel “Persons”. Kolom “Personid” akan diberi nilai unik. Kolom “FirstName” akan disetel ke “Larry” dan kolom “LastName” akan disetel ke “Monsen”.

Syntax untuk SQL Server

```
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Usia int);
```

Syntax untuk Access

```
CREATE TABLE Persons (
    Personid AUTOINCREMENT PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Usia int
```

);

Syntax untuk Oracle

CREATE SEQUENCE seq_person

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 10;

SQL Dates

Hal yang tidak mudah saat menggunakan fungsi Date pada SQL adalah memastikan bahwa format tanggal yand dimasukkan bisa cocok dengan format kolom tanggal pada database.

Tipe Data Date SQL

MySQL memiliki tipe data berikut untuk menyimpan date atau nilai tanggal / waktu dalam database:

DATE – format YYYY-MM-DD

DATETIME – format: YYYY-MM-DD HH:MI:SS

TIMESTAMP – format: YYYY-MM-DD HH:MI:SS

YEAR – format YYYY or YY

SQL Injection

Penyerang yang ingin mengeksekusi SQL Injection memanipulasi kueri SQL standar untuk mengeksploitasi kerentanan input yang tidak divalidasi dalam database. Ada banyak cara agar serangan ini dapat dieksekusi. Misalnya, masukan yang disebutkan di atas, yang menarik informasi untuk produk tertentu, dapat diubah menjadi http://www.eniaga.com/items/items.asp?itemid=999 atau 1 = 1 .Hasilnya, kueri SQL terkait terlihat seperti ini:

SELECT ItemName, ItemDescription

FROM Items

WHERE ItemNumber = 999 OR 1=1

Dan karena pernyataan 1 = 1 selalu benar, kueri mengembalikan semua nama dan deskripsi produk dalam database, bahkan yang mungkin tidak memenuhi syarat untuk kalian akses.

Penyerang juga dapat memanfaatkan karakter yang disaring secara tidak tepat untuk mengubah perintah SQL, termasuk menggunakan titik koma untuk memisahkan dua bidang.

Misalnya, masukan ini http://www.eniaga.com/items/iteams.asp?itemid=999; DROP TABLE USERS akan membuat kueri SQL berikut:

SELECT ItemName, ItemDescription

FROM Items

WHERE ItemNumber = 999; DROP TABLE USERS

Akibatnya, seluruh database pengguna bisa terhapus.

Cara lain query SQL dapat dimanipulasi adalah dengan pernyataan UNION SELECT. Ini menggabungkan dua kueri SELECT yang tidak terkait untuk mengambil data dari tabel database yang berbeda.

Misalnya, input http://www.eniaga.com/items/items.asp?itemid=999 UNION SELECT user-name, password FROM USERS, menghasilkan kueri SQL berikut:

SELECT ItemName, ItemDescription

FROM Items

WHERE ItemID = '999' UNION SELECT Username, Password FROM Users;

Menggunakan pernyataan UNION SELECT, kueri ini menggabungkan permintaan untuk nama item 999 dan deskripsi dengan yang lain yang menarik nama dan kata sandi untuk setiap pengguna dalam database.

SQL Data Types

data digunakan untuk mendefinisikan suatu field atau kolom. Setiap kolom yang dibuat harus didefinisikan terlebih dahulu. Jenis – jenis tipe data ada bermacam – macam. Bisa numerik yang digunakan untuk angka dan proses perhitungan, bisa karakter / teks, tanggal atau Biner. Berikut ini Macam – macam tipe data yang digunakan di SQL :

1. Tipe Numerik

Tipe data numerik digunakan untuk menyimpan data numeric (angka). Ciri utama data numeric adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe numerik:

TINYINT

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

Jangkauan : -128 s/d 127

Ukuran : 1 byte (8 bit)

SMALLINT

IT-Jurnal.com

Home database Tipe Data Pada Database SQL

Tipe Data Pada Database SQL

By Dwiky Andika 0

Tipe data digunakan untuk mendefinisikan suatu field atau kolom. Setiap kolom yang dibuat harus didefinisikan terlebih dahulu. Jenis – jenis tipe data ada bermacam – macam. Bisa numerik yang digunakan untuk angka dan proses perhitungan, bisa karakter / teks, tanggal atau Biner. Berikut ini Macam – macam tipe data yang digunakan di SQL :

Tipe data numerik digunakan untuk menyimpan data numeric (angka). Ciri utama data numeric adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe numerik:

TINYINT

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

Jangkauan : -128 s/d 127

Ukuran : 1 byte (8 bit)

SMALLINT

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif..

Jangkauan : -32.768 s/d 32.767

Ukuran : 2 byte (16 bit).

MEDIUMINT

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

Jangkauan : -8.388.608 s/d 8.388.607

Ukuran : 3 byte (24 bit)

INT

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

Jangkauan : -2.147.483.648 s/d 2.147.483.647

Ukuran : 4 byte (32 bit).

BIGINT

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif.

Jangkauan : ± 9,22 x 1018

Ukuran : 8 byte (64 bit)

FLOAT

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi tunggal.

Jangkauan : 3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38.

Ukuran : 4 byte (32 bit).

DOUBLE / REAL

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi ganda.

Jangkauan : -1.79…E+308 s/d -2.22…E-308, 0, dan 2.22…E-308 s/d 1.79…E+308.

Ukuran : 8 byte (64 bit).

DECIMAL / NUMERIC

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif dan negatif.

Jangkauan : -1.79…E+308 s/d -2.22…E-308, 0, dan 2.22…E-308 s/d 1.79…E+308.

Ukuran : 8 byte (64 bit).

2. Tipe Date dan Time

Tipe data date dan time digunakan untuk menyimpan data tanggal dan waktu. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe date dan time:

DATE

Penggunaan : digunakan untuk menyimpan data tanggal.

Jangkauan : 1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)Ukuran : 3 byte.

TIME

Penggunaan : digunakan untuk menyimpan data waktu.

Jangkauan : -838:59:59 s/d +838:59:59 (HH:MM:SS)Ukuran : 3 byte.

DATETIME

Penggunaan : digunakan untuk menyimpan data tanggal dan waktu.

Jangkauan : '1000-01-01 00:00:00' s/d '9999-12-31 23:59:59'Ukuran : 8 byte.

YEAR

Penggunaan : digunakan untuk menyimpan data tahun dari tanggal.

Jangkauan : 1900 s/d 2155

Ukuran : 1 byte.

3. Tipe String (Text)

Tipe data string digunakan untuk menyimpan data string (text). Ciri utama data string adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti pertambahan, pengurangan, perkalian dan pembagian. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe string:

CHAR

Penggunaan : digunakan untuk menyimpan data string ukuran tetap.

Jangkauan : 0 s/d 255 karakter

VARCHAR

Penggunaan : digunakan untuk menyimpan data string ukuran dinamis.

Jangkauan : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

TINYTEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

TEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d 65.535 (216 – 1) karakter

MEDIUMTEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d 224 – 1 karakter

LONGTEXT

Penggunaan : digunakan untuk menyimpan data text.

Jangkauan : 0 s/d 232 – 1 karakter

4.Tipe BLOB (Biner)

Tipe data blob digunakan untuk menyimpan data biner. Tipe ini biasanya digunakan untuk menyimpan kode-kode biner dari suatu file atau object. BLOB merupakan singkatan dari Binary Large Object. Berikut ini tipe field (kolom) di MySQL yang termasuk ke dalam kelompok tipe blob:

BIT (sejak versi 5.0.3)

Penggunaan : digunakan untuk menyimpan data biner.

Jangkauan : 64 digit biner

TINYBLOB

Penggunaan : digunakan untuk menyimpan data biner.

Jangkauan : 255 byte

BLOB

Penggunaan : digunakan untuk menyimpan data biner.

Jangkauan : 216 – 1 byte

MEDIUMBLOB

Penggunaan : digunakan untuk menyimpan data biner.

Jangkauan : 224 – 1 byte

LONGBLOB

Penggunaan : digunakan untuk menyimpan data biner.

Jangkauan : 232 – 1 byte

5. Tipe Data yang Lain

Selain tipe data di atas, MySQL juga menyediakan tipe data yang lain. Tipe data di MySQL mungkin akan terus bertambah seiring dengan perkembangan versi MySQL. Berikut ini beberapa tipe data tambahan MySQL:

ENUM

Penggunaan : Enumerasi (kumpulan data).

Jangkauan : Sampai dengan 65535 string.

SET

Pengunaan : Combination (himpunan data).

Jangkauan : Sampai dengan 255 string anggotas.

SQL FOREIGN KEY

adalah kunci yang digunakan untuk menghubungkan dua tabel bersama-sama. Constraint FOREIGN KEY adalah field(atau kumpulan field) dalam satu tabel yang merujuk ke PRIMARY KEY di tabel lain. SQL berikut membuat FOREIGN KEY di kolom “PersonID” ketika tabel “Order” dibuat:

MySQL:

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   NomorPesanan int NOT NULL,
   PersonID int,
   PRIMARY KEY (OrderID),
   FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

Akses SQL Server / Oracle / MS:

```
CREATE TABLE Orders (
   OrderID int NOT NULL PRIMARY KEY,
   NomorPesanan int NOT NULL,
   PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

Untuk dapat melakukan penamaan constraint FOREIGN KEY dan untuk menentukan constraint FOREIGN KEY pada beberapa kolom, gunakan sintaks SQL berikut:

Akses MySQL / SQL Server / Oracle / MS:

```
CREATE TABLE Orders (
  OrderID int NOT NULL,
  NomorPesanan int NOT NULL,
  PersonID int,
  PRIMARY KEY (OrderID),
  CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
  REFERENCES Persons(PersonID)
);
```

SQL References

dapat digunakan untuk menentukan beberapa nilai dalam klausa WHERE.

SQL berikut memilih semua pelanggan yang berada di “Jerman”, “Prancis”, dan “Inggris Raya”:

Contoh

```
SELECT * FROM Customers
WHERE Negara IN ('Jerman', 'Prancis', 'Inggris Raya');
```

SQL berikut memilih semua pelanggan yang TIDAK berada di “Jerman”, “Prancis”, atau “Inggris Raya”:

Contoh

```
SELECT * FROM Customers
WHERE Negara NOT IN ('Jerman', 'Prancis', 'Inggris Raya');
```

SQL berikut memilih semua pelanggan yang berasal dari negara yang sama dengan Suppliers:

Contoh

```
SELECT * FROM Customers
WHERE Negara IN (SELECT Negara FROM Suppliers);
```

MySQL Functions

Fungsi (function) atau kata lainnya method merupakan suatu sub program yang diperuntukan untuk mengerjakan suatu perintah tertentu sesuai dengan fungsi method itu sendiri. Didalam mysql sudah terdapat beberapa fungsi default yang dapat digunakan untuk mengerjakan tugas tertentu. Misalnya untuk mencari nilai minimal, menghitung rata-rata dan menjumlahkan suatu nilai

SQL Server Functions

Function memiliki 3 jenis parameter yaitu sbb :

Input

Adalah parameter yang wajib diisi ketika function dipanggil. Contoh :

CREATE FUNCTION nama (@parameter1 int)

Optional

Adalah parameter yang boleh dikosongkan ketika function dipanggil. Contoh :

CREATE FUNCTION nama (@parameter1 int = NULL)

Default

Adalah parameter yang akan berisi nilai default jika parameter tidak disi ketika function dipanggil. Contoh :

CREATE FUNCTION nama (@parameter1 int = 100)

Berikut adalah contoh syntax function :

```
USE inventory1;
GO
   -- membuat function
CREATE FUNCTION umur_faktur(@tgl_faktur DATE)
RETURNS INT
AS
BEGIN
   DECLARE @umur INT
   SELECT @umur = DATEDIFF(YEAR,@tgl_faktur,GETDATE())
   RETURN @umur
```

```
END
GO
   -- mengubah function
ALTER FUNCTION umur_faktur(@tgl_faktur DATE)
RETURNS INT
AS
BEGIN
   DECLARE @umur INT
   SELECT @umur = DATEDIFF(DAY,@tgl_faktur,GETDATE())
   RETURN @umur
END
GO
   -- menghapus function
DROP FUNCTION umur_faktur;
GO
   -- menggunakan function
SELECT kd_jual, tgl_jual, dbo.umur_faktur(tgl_jual) AS 'Umur Faktur' FROM t_jual_h;
GO.
```

MS Access Functions

Microsoft Access adalah program aplikasi keluaran Microsoft yang berguna untuk membuat, mengolah, dan mengelola database (basis data). Database (basis data) yaitu kumpulan arsip data berbentuk tabel yang saling relasi atau berhubungan sehingga menghasilkan informasi. Untuk menghasilkan sebuah informasi, diperlukan adanya DATA untuk dijadikan sebagai masukan.

Fungsi/kegunaan Ms.Access yaitu :

- Untuk membuat basis data (database).
- Untuk membuat program aplikasi jumlah peserta didik.
- Untuk membuat program aplikasi gaji karyawan.
- Untuk membuat program aplikasi penyimpan buku perpustakaan.

- Untuk membuat program aplikasi absensi.
- Untuk membuat program aplikasi persediaan barang.
- Dan lain-lain.

SQS Quick Referensi

Quick Reference dan Penelusuran Informasi Sederhana Arief Wicaksono Sarasehan Pustakawan Perpustakaan Nasional RI Jakarta, 16 Oktober 2017. Materi Unsur, Sub Unsur, Butir Kegiatan Layanan referensi, Era Digital, Pengetahuan dan Keterampilan Quick reference Penelusuran informasi sederhana Perbedaan Praktek Contoh Pertanyaan Unsur 1. Pendidikan 2. Pengelolaan perpustakaan 3. Pelayanan perpustakaan 4. Pengembangan sistem kepustakawanan 5. Pengembangan profesi 6. Penunjang tugas kepustakawanan Juknis Pustakawan 2015. Sub Unsur 1. Pelayanan teknis 2. Pelayanan pemustaka a. Pelayanan sirkulasi b. Pelayanan referensi Juknis Pustakawan 2015 PP Pelaksanaan UU Perpus Butir Kegiatan Pelayanan Pemustaka Terdapat 10 butir kegiatan Melakukan layanan referensi cepat (quick reference) Melakukan layanan penelusuran informasi sederhana Juknis Pustakawan 2015 Layanan Referensi kegiatan memenuhi kebutuhan informasi pemustaka: 1) menjawab pertanyaan substantif 2) mengajarkan dan melakukan penelusuran 3) merujuk sumber daya dalam dan luar perpustakaan 4) membantu evaluasi informasi 5) membuat statistik referensi 6) mengembangan koleksi referensi SKKNI Bidang Perpustakaan Layanan Referensi di Era Digital-1 Mampu menggunakan perpustakaan secara efektif? Mampu melakukan penelusuran sendiri? Layanan Referensi di Era Digital-2 Sulit dibayangkan jika suatu perpustakaan, bahkan perpustakaan digital sekalipun, tidak memberkan layanan referensi (Cassell & Hiremath, 2012; Bopp & Smith, 2011) Keterampilan yang diperlukan Menggunakan TIK Komunikasi interpersonal dan melakukan wawancara. Menggunakan sumber referensi Teknik penelusuran informasi SKKNI Bidang Perpustakaan. Quick Reference kegiatan memberi jawaban langsung atas permintaan informasi dari pemustaka melalui atau tanpa pemanfaatan sumber referensi Juknis Pustakawan 2015