**NAMA : MUH. MAULID**

**NIM : 200250502042**

**JURUSAN : TEKNIK INFORMATIKA**

**KELAS : B**

**1. SQL TUTORIAL**

**SQL Home Dan Intro**

Kedua SQL ini Merupakan Bahasa standar untuk mengakses dan memanipulasi database.

Cara menggunakan SQL disitus web, untuk membuat sebuah situs web menampilkan data dari database, kita memerlukan:

Program database RDBMS

Adalah dasar SQL, untuk semua sistem database modern seperti MSSQL server, IBM DB2, oracle, MYSQL, dan Microsoft acces. Dalam RDBMS disimpan dalam objek database yang biasa disebut dengan tabel. Setiap tabel terpecah menjadi intitas yang lebih kecil yang disebut bidang. Bidang pada tabel pelanggan terdiri dari IDpelanggan, nama pelanggan, nama kontak, alamat, kota, kode pos, dan negara. Bidang merupakan kolom dalam tabel yang dirancang untuk mempertahankan informasi spesifik setiap catatan dalam tabel.

**Contoh:**

SELECT \* FROM customers;

**SQL syntax**

Database paling sering berisi satu atau lebih tabel. Setiap tabel diidentifikasi dengan nama pelanggan atau pesanan. Tabel berisi record (baris) dengan data.

**Contoh:**

SELECT \* FROM Buyers;

**SQL Select**

Merupakan perintah dasar SQL yang digunakan untuk memilih data dari database, data yang dikembalikan disimpan dalam tabel yang disebut result-set.

**Contoh:**

sintaks

SELECT kolom 1, kolom 2, FROM nama\_tabel;

atau

SELECT\*FROM nama\_tabel;

Contoh: SELECT nis, nama, alamat FROM siswa.

Atau

SELECT \* FROM siawa;

Perintah pertama digunakan untuk menyeleksi kolom nis, nama, dan alamat dari tabel siswa. Sedangkan perintah kedua digunakan untuk menyeleksi semua semua kolom dari tabel siswa.

**SQL Select distinct**

Adalah perintah dasar SQL yang digunakan untuk mengembalikan hanya nilai yang berbeda dari dalam sebuah tabel dengan kata lain semua record duplikat (record dengan nilai yang sama) yang terdapat pada tabel akan dianggap sebagai satu record/nilai.

**Contoh:**

Sintaks

SELECT DISTINCT kolom 1, kolom 2, … FROM nama\_tabel;

Contoh

SELECT DISTINCT alamat FROM siswa;

Perintah diatas digunkan untuk menampilkan kolom alamat dari tabel siswa dengan mengabaikan nilai yang duplikat, misalnya terdapat 5 siswa dengan alamat “Kalimantan timur”, 11 orang dengan alamat “Sulawesi barat”, dan seterusnya. Result-set hanya menampilkan daftar alamat diantaranya “Kalimantan timur”, dan “Sulawesi barat” masing-masing 1 record

**SQL where**

Merupakan perintah yang digunakan untuk memfilter hasil SELECT dengan mengekstrakkan record yang memenuhi persyaratan tertentu.

**Contoh”**

Sintaks

SELECT kolom1, kolom2, … FROM nama\_tabel WHERE kondisi;

Sintaks

SELECT nis, nama, FROM siswa WHERE alamat=’sulawesi’;

Perintah diatas digunakan untuk mengekstraksikan data (nis dan nama) dari tabel siswa dengan kondisi “field alamat berisi nilai Sulawesi”. Perintah tersebut menggunakan operator sama dengan (=), untuk operator lain yang digunakan perintah WHERE bisa kita lihat dibawah ini:

= sama dengan

<> tidak sama dengan, pada beberapa versi SQL, operator yang digunakan adalah!=

> lebih besar dari

< lebih kecil dari

>= lebih besar sama dengan

<= lebih kecil sama dengan

BETWEEN antara rentang inklusif

LIKE cari pola yang ‘seperti’ parameter

IN menentukan kemungkinan nilai dari bebrapa kolom

**SQL and, or dan not**

Merupakan perintah dasar yang digunakan untuk mengkombinasikan dengan perintah WHERE. Ketiganya digunakan untuk memfilter record berdasar kan suatu kondisi, operator AND akan menampilkan record apabila kondisi bernilai TRUE, operator OR akan menampilkan record appabila salah satu kondidi 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=’sulawesi’ AND tahun\_lahir=’2000’

Sintaks OR

SELECT kolom1, kolom2, … FROM nama\_tabel WHERE kondisi1 OR kondisi2 OR kondisi3 …;

Contoh OR

SELECT nis, nama, FROM siswa WHERE alamat=’sulawesi’ OR alamat=’sulawesi’

Sintaks NOT

SELECT kolom1, kolom2, … FROM nama\_tabel WHERE NOT kondisi;

Contoh NOT

SELECT nis, nama, FROM siswa WHERE NOT alamat=’sulawesi’

**SQL order by**

Merupakan perintah yang digunakan untuk mengurtkan result-set dalam pengurutan ‘ascending’ secara default perintah order by menampilkan record dalam pengurutan ‘ascending’ (asc). Untuk mengurutkan menggunakan kata kunci “DESC”.

Sintaks

SELECT kolom1, kolom2, … FROM nama\_tabel ORDER BY colum DESC;

**Contoh:**

SELECT nis, nama FROM siswa ORDER BY tahun\_lahir DESC;

Perintah ini menampilkan result-set berupa filed nis dan nama dari tabel siswa yang diurutkan berdasarkan tahun\_lahir secara diending, yang berarti tahun lahir akan ditampilkan mulai dari yang terbesar hinggah yang terkecil.

**SQL insert into**

Merupakan bagian dari perintah untuk DML (data manipulation language). perintah insert into dapat kita gunakan untuk menambahkan record baru kedalam tabel.

Sintaks

INSERT INTO nama\_tabel VALUES (nilai1, nilai2, nilai3, …);

Contoh

INSERT INTO siswa VALUES (‘12345’, ‘abdul’, ‘semarang’);

Perintah ini digunakan untuk menabahkan ‘12345’, ‘abdul’, dan ‘semarang’ pada tabel siswa, pastikan urutan nilai (values) dalam urutan yang sama seperti kolom dalam tabel. Dan jika urutan nilai tidak sama dengan urutan kolom pada tabel maka sintaks yang digunakan sebagai berikut.

Sintaks

INSERT INTO nama\_tabel (kolom1, kolom2) VALUES (nilai1, nilai2);

**Contoh:**

INSERT INTO siswa (nim, nama) VALUES (‘12345’, ‘abdul’);

Perintah ini digunakan untuk menambahkan nilai ‘12345’ pada kolom nim, dan nilai ‘abdul’ pada kolom nama dengan mengabaikan kolom lain yang tidak diisi, misalnya kolom alamat.

**SQL null values**

Data null artinya no value, dan null artinya tidak sama dengan nol atau ‘spasi’. Null tidak dapat digunakan untuk aritmatika, karena berapapun jika di tambah atau dikurang atau dikali dengan null hasilnya akan null. Null juga tidak bisa digunakan dengan operator relational.

**Contoh:**

SELECT customername, Contactname,address FROM customer WHERE address IS NOT NULL;

**SQL Update**

Digunakan untuk mengubah catatan atau isi yang ada pada sebuah data dalam tabel. Saat memperbaharui record pada sebuah table, perhatIkan klausa WHERE dapat menentukan record mana yang harus kita perbaharui. Jika kita menghilangkan klausa WHERE maka seluruh data pada tabel akan di perbaharui.

**Contoh:**

UPDATE customers SET contactname = ‘alfred schmidt’, city= ‘frankfurt’ WHERE costumerID = 1;

**SQL Delete**

Digunakan untuk menghapus rakaman data yang ada dalam tabel.

**Contoh:**

DELETE FROM customers WHERE customername=’alfreds futterkister’;

**SQL Select top**

Untuk penggunaan penentuan jumlah record yang akan kita kembalikan. Klausa select top digunakan ribuan record mengembalikan sejumlah besar rekaman dapat mempengaruhi kinerja.

Sintaks

SELECT TOP number/percent colum\_name FROM table\_name WHERE condition;

**Contoh:**

SELECT TOP 3 \* FROM customers;

**SQL min dan max**

Fungsi Min mengembalikan nilai terkecil dari kolom yang kita pilih. Sedangkan fungsi max mengembalikan nilai terbesar dari kolom yang telah dipilih.

Sintaks min ()

SELECT moin (colum\_name) FROM table\_name WHERE condition;

Contoh min ()

SELECT min(price) AS smallestrprice FROM products;

Sintaks max ()

SELECT max (colum\_name) FROM table\_name WHERE condition;

Contoh max ()

SELECT max(price) AS largestprice FROM products;

**SQL count, avg, dan sum.**

Count digunakan untuk mengembalikan jumlah baris yang cocok dengan kriteria yang telah ditentukan. Avg, untuk mengembalikan nilai rata-rata kolom numerik. Sum, digunakan untuk mengembalikan total jumlah kolom numeriknya.

Sintaks count

SELECT COUNT (colum\_name) FROM table\_name WHERE condition;

Contoh count

SELECT COUNT (productid) FROM products;

Sintaks avg

SELECT avg (colum\_name) FROM (table\_name) WHERE condition;

Contoh avg

SELECT avg (price) FROM products;

Sintaks sum

SELECT sum (colum\_name) FROM table\_name WHERE condition;

Contoh sum

SELECT sum (Quantity) FROM orderdetails;

**SQL like**

Yang digunakan pada klausa where untuk mencari pola yang ditentukan dalam sebuah kolom. Terdapat dua karakter pengganti yang digunakan terkait penggunaan like.

>> % tanda persen mewakili nol, satu atau beberapa karakter.

>> \_ garis bawah mewakili satu karakter.

Sintaks like

SELECT colum1, colum2, … FROM table\_name WHERE colum LIKE pattern;

Contoh like

SELECT \* FROM customerrsmWHERE customersname LIKE ‘a%’;

**SQL wildcards**

Digunakan untuk mengganti karakter pada sebuah string. SQL wilcards digunakan dengan operator SQL like.

Salah satu contohnya yaitu:

**Contoh:**

SELECT \* FROM customers WHERE city LIKE ‘ber%’;

**SQL in**

Digunakan untuk beberapa nilai yang ada dalam sebuah klausa where.

**Contoh:**

Sintaks

SELECT colum\_nsme (S) FROM table\_name WHERE colum\_name IN (value1, value2, …);

Atau

SELECT colum\_name(S) FROM table\_name WHERE colum\_name IN (SELECT statement);

**SQL Between**

Dalam sebuah operator between itu kita dapat memilih sebuah nilai dalam waktu tertentu yang berupa nilai awal dan nilai akhir yang harus kita sertakan. Pada nilai yang kita pilih itu bsa berupa angka, teks atau bisa juga berupa tanggal.

**Contoh:**

* **Angka**

SELECT \* FROM Taper wer AND Baju berbie

WHERE Price BETWEEN 60 AND 20;

SELECT \* FROM Taper wer AND Baju berbie

WHERE Price NOT BETWEEN 60 AND 20;

* **Teks**

SELECT \* FROM Taper wer AND Baju berbie

WHERE ProductName BETWEEN “Taper wer” AND “Baju berbie”

ORDER BY Farmania;

SELECT \* FROM Taper wer AND Baju berbie

WHERE ProductName NOT BETWEEN “Taper wer” AND “Baju berbie”

ORDER BY Farmania;

* **Tanggal**

SELECT \* FROM Jaya Toko

WHERE OrdarDate BETWEEN # 3/11/2020 # AND # 4/14/2020 #;

SELECT \* FROM Jaya Toko

WHERE OrdarDate BETWEEN ‘2020/11/3’AND‘2020/14/4’ ;

**SQL Aliases**

Untuk SQL Aliases (SQL alias) itu digunakan pada bagian pemberian nama hanya sementara saja pada table, atau bisa juga pada bagian kolom table. Aliasesn pada tabel bisa juga berfungsi ketika pada tabel lebih dari satu yang berperan didalam kueri.

**Contoh:**

SELECT MK.011 AS MK.011, Buyer AS Buyer

FROM Farmania ;

**SQL Joins**

Untuk joins in dapat kita gunakan untuk menggabungkan baris yang dapat kita gunakan dua atau bisa saja lebih pada tabel, yang berdasarkan kolom yang terkait diantaranya. Pada SQL joins itu memiliki beberapa jenis yaitu diantaranya: (INNER) JOIN : mengembalikan rekaman yang mempunyai nilai yang sama dikedua tabel tersebut, LEFT (OUTER) JOIN : mengembalikan seluruh catatan yang di mulai dari tabel kiri dan catatan yang cocok pada bagian tabel kanan, RIGHT (OUTER) JOIN : Mengembalikan seluruh record mulai dari tabel kanan, dan record yang cocok pada bagian tabel kiri, FULL (OUTER ) JOIN : mengembalikan seluruh record karena adanya kecocokan antara tabel kiri ataupun kanan.

**Contoh:**

SELECT Buyer.BuyerID,Shop.ShopName,Item.ItemCode

FROM Buyers

INNER JOIN ItemCode ON Buyers.ItemCode=ItemCode.

ItemCodeID

**SQL Inner Join**

Pada inner join itu kita harus memilih record yang mempunyai nilai yang tepat pada kedua tabel tersebut. Dalam inner join ada kata kunci yang harus kita perhatikan yaitu memilih seluruh baris pada kedua tabel pada saat kedua tabel tersebut memiliki kesamaan diantara kolom.

**Contoh:**

SELECT Novel.Novel POR-0000019. Farmania.Customer Farmania

FROM Novel

INNER JOIN Farmania ON Novel.MK.011 = Farmania. MK.011

Pernyataan SQL berikut memilih semua pesanan dengan informasi pelanggan dan pengirim

SELECT Novel.Novel POR-0000019, Farmania.CustomerFarmania, Toko Jaya Cahaya.ShipperTokoJayaCahaya

FROM ((Novel

INNER JOIN Farmania ON Novel.FarmaniaMK.011 = Farmania. MK.011)

INNER JOIN Toko Jaya Cahaya ON Novel.MK.005 = Toko Jaya Cahaya.MK.005);

**SQL Left Join**

Yang harus kita perhatikan dalam kata kunci pada left join yaitu mengembalikan seluruh record mulai dari tabel kiri (table pertama), dan record yang sama pada tabel kanan (tabel kedua). Yaitu hasilnya adalah NULL yang dari sisi kanan, jika pada saat itu tidak ada yang cocok.

**Contoh**:

SELECT Farmania.MK.011

FROM Farmania

LEFT JOIN Boneka ON Farmania.MK.011 = Boneka.TOP-000004 =

ORDER BY Farmania.Mania;

**SQL Right Join**

Untuk SQL Right Join sedikit memiliki kesamaan hanya saja yang membedakan adalah kata kunci Right Join adalah mengembalikan seluruh record mulai dari tabel kanan (tabel pertama), dan record yang sama pada tabel kiri (tabel kedua ). Dan hasilnya adalah NULL yang dari sisi kiri, jika pada saat itu tidak ada yang cocok.

**Contoh:**

SELECT Baju berbie.NOV-0000067, Store.Address, Store.StoreName

FROM Baju berbie

RIGHT JOIN Instansi ON AC split ¼ PK.8796321 = Instansi.MK,111

ORDER BY Baju berbie.NOV-0000067;

**SQL Full Join**

FULL JOIN itu akan mengembalikan seluruh rekaman yang sama atau cocok pada kedua tabel apakah kedua tabel itu sama atau tidak. FULL JOIN juga sangat berpotensi untuk mengembalikan semua data yang cukup besar pada kedua tabel; FULL JOIN dan FULL OUTER JOIN adalah sama. Yang harus kita ketahui juga adalah FULL OUTER JOIN yang sering disingkat dengan kata full join yaitu dia akan mengembalikan seluruh baris pada kedua tabel yang yaitu ON yang termasuk data-data yang bernilai NULL.

**Contoh:**

SELECT ID\_Toko, Kode Barang

FROM ID Toko

FULL OUTER JOIN Kode Barang

ON (tabel 1. ID\_Toko = (tabel 2).Kode Barang.ID Barang

**SQL Self Join**

Self Join adalah gabungan regular, akan tetapi pada tabel tersebut dia bergabung dengan sendirinya dan dengan cara ini sangat membantu pada saat memodelkan hierark. Self Join juga sangat membantu dalam perbandingan suatu tabel.

**Contoh:**

SELECT itemCode AS ItemCode1, B.ItemCode AS ItemCode2, A.Price

FROM ItemCode.A,ItemCode.B

WHERE A.ItemCode <> B.ItemCode

AND A.Date = B.Date

WHERE condition;

**SQL Union**

UNION itu menggabungkan seluruh gabungan dari hasil pada kedua query yang berbeda. Akan tetapi Pada tipe data kolom di dua query itu harus cocok atau sama. Pada union itu juga harus menggabungkan mulai dari hasil dengan posisi kolom dari pada nama kolom. Yang harus kita perhatikan juga dalam SQL Union adalah operator Union yang kita gunakan untuk menggabungkan seluruh hasil dari yang kedua atau pun lebih pernyataan SELECT.

 Yang pertama adalah pada setiap pernyataan SELECT pada Union itu harus mempunyai hasil atau jumlah kolom yang sama

 Yang kedua adalah pada setiap kolom juga harus mempunyai tipe data yang sama

 Dan yang terakhir adalah pada kolom disetiap pernyataan SELECT itu juga harus dalam urutan yang tepat.

**Contoh:**

SELECT ShopID FROM ItemCode(Tabel 1)

UNION

SELECT ShopID FROM NameOf Goods(Tabel 2);

**SQL Group By**

Pernyataan Goup By pada SQL itu digunakan untuk mengatur semua data yang identik yang dimasukkan kedalam kelompok dengan beberapa fungsi yaitu seperti jika kolom tersebut memiliki nilai yang sama pada baris yang berbeda maka dari itu kita akan mengatur baris-baris yang ada didalam grup. Poin penting yang harus kita perhatikan juga adalah seperti:

* Clause Group By digunakan pada pernyataan SELECT.
* Pada query, clause Group By harus menempatkan setelah clause WHERE
* Pada query, clause Group By harus ditempatkan sebelum clause Order By misalnya digunakan.

Pada pernyataan Group By sering menggunakan dengan fungsi yang agrerat seperti (COUNT, MAX, MIN, SUM, AVG) untuk mengelompokkan jumlah atau hasil yang ditetapkan oleh satu atau lebih kolom.

Pernyataan ini adalah merupakan jumlah yang dicantumkan dalam jumlah tersebut.

SELECT ID Instansi, Jumlah

FROM Instansi

GROUB BY Jumlah;

**SQL Having**

Pada Clause HAVING itu digunakan pada pernyataan SELECT guna untuk menentukan kondisi filter untuk sekelompok baris atau pun agrerat. Clause Having itu sering kali digunakan dengan clause Groub By itu dikarenakan karena untuk memfilter grub berdasarkan kondisi tertentu. Pada saat clause Having dhilangkan maka clause akan sama seperti clause WHERE.

Pada clause Having seperti WHERE akan tetapi dia beroperasi pada rekaman yang dikelompokkan dan dikembalikan oleh Group By. Dalam Having ini berperan untuk catatan grub yang diringkas, sedangkan WHERE itu berperan untuk catatan individu.

**Contoh:**

Pada pernyataan ini adalah pernyataan yang menentukan jumlah pada jumlah pesanan barang yang dihitung pada jumlah ID toko tersebut.

SELECT (ShopID),Price

FROM ShopID

WHERE Condition

GROUP BY Price

WHERE Condition

HAVING COUNT (ShopID ) > 4;

**SQL Exists**

EXISTS itu untuk mengembalikan sebuah nilai true

Pada operator Exists itu digunakan untuk menguji rekaman yang ada pada di subquery. Pada operator exists ada yang mengembalikan nilai yang benar ketika subquery mengembalikan satu atau pun lebih pada rekaman.

**Contoh:**

SELECT Buyer’sName

FROM Buyer

WHERE EXISTS (SELECT NameOfGoods FROM Goods WHERE Goods.Buyer = Buyer’s.Buyer’s AND Price < 50);

**SQL Any, All**

ANY dan ALL dalam kata kunci itu digunakan dengan WHERE atau bisa juga dengan HAVING, Yang pada kedua operasi itu mereka beroperasi pada subquery yang dapat mengembalikan banyak nilai. Pada ANY mengembalikan true ketika pada salah satu nilai subquery itu dapat memenuhi kondisi. Dan ALL mengembalikan nilai true jika semua nilai subquery memenuhi kondisinya.

**Contoh:**

SELECT NameOfGoods

FROM Goods

WHERE GoodsID operator ANY

(SELECT GoodsID FROM Orders WHERE Total = 5);

SELECT NameOfGoods

FROM Goods

WHERE GoodsID operator ALL

SELECT GoodsID FROM Orders WHERE Total = (5);

**SQL SELECT INTO**

SELECT INTO dapat menyalin data dari satu tabel ke tabel yang baru. Pada operasi ini kita membuat tabel baru pada grub file default.

**Contoh:**

* Pada pernyataan SQL berikut ini adalah untuk membuat cadangan untuk para pembeli:

SELECT \* INTO BuyersBackup(2020)

FROM Buyers;

* Pada pernyataan yang berikut ini adalah menggunakan clause IN yang untuk digunakan untuk menyalin dari tabel ke tabel yang baru di data base lainnya.

SELECT \* INTO BuyersBackup(2020) IN ‘Backup.mdb’

FROM Buyers;

* Pada pernyataan berikut ini adalah hanya dengan menyalin beberapa kolom ke dalam tabel yang baru:

SELECT NameOfGoods,ItemCode INTO BuyersBackup(2020)

FROM Buyers;

* Pada pernyataan yang berikut ini adalah untuk menyalin data dari satu tabael ke dalam tabel yang baru:

SELECT ShopID FROM ItemCode(Tabel 1)

UNION

SELECT ShopID FROM NameOf Goods(Tabel 2);

**SQL INSERT INTO SELECT**

Pernyataan INSERT INTO SELECT dapat menyalin data dari satu tabel dan menyimpannya ke tabel yang lain. Pada operasi ini mengharuskan jenis tipe data di tabel sumber dan target yang cocok. Catatan yang ada pada tabel tabel tidak terpengaruh.

**Contoh:**

Untuk pernyataan berikut ini adalah menyalin kedua tabel yang dimana “penjual” ke “pembeli” telah kita buat sebelumnya (isi semua kolom):

INSERT INTO (ShopID, Address, ShopName, TransactionNumber, Period, ItemCode, TheAmountOfGoods, NameOfGoods, Price)

SELECT Buyer’sName, Address, ShopName, TransactionNumber, Period, ItemCode, TheAmountOfGoods, NameOfGoods, Price FROM Buyer’sName;

**SQL CASE**

Pernyataan CASE melewati kondisi dan mengembalikan nilai pada saat kondisi pertama yang terpenuhi (seperti pada pernyataan IF-THEN-ELSE). Jadi, begitulah suatu kondisi pada saat benar, itu akan berhenti membaca dan mengembalikan hasilnya. Jika pada saat itu tidak kondisi yang benar, Ini akan mengembalikan nilai dalam clause ELSE. Jika tidak ada bagian ELSE dan tidak ada kondisi yang benar, dia mengembalikan NULL.

**Contoh:**

SELECT TransactionName,Price,

CASE

WHEN Price > 30.000 THEN ‘The price is greater than 30.000’

WHEN Price = 30.000 THEN ‘The price is 30.000’

ELSE ‘The price is under 30.000’

END AS Price

FROM ShopID;

**SQL Null functions**

Pada bagian ini adalah merupakan fungsi NULL yang didefinisikan oleh SQL.ISNULL (): pada fungsi ISNULL mempunyai kegunaan yang tidak sama atau bisa dikatakan berbeda pada SQL server dan My SQL. Pada SQL server fungsi dari ISNULL () adalah digunakan untuk menggantikan nilai NULL.

**Contoh:**

SELECT ItemName,TotalPrice \* (TheNumberOfGoods + OrderQuantity)

FROM Item;

**SQL Stored procedures**

SQL stored prosedures (prosedur tersimpan SQL) merupakan suatu kumpulan pernyataan pada SQL dan logika perintah SQL. Yang telah dikomplikasi dan kita simpan pada database. Prosedur tersimpan pada SQL itu dapat memunkinkan kita untuk membuat query SQL untuk dismpan dan dijalankan pada server. Prosedur yang telah kita simpan itu juga dapat di-cache dan digunakan untuk kembali. Tujuan utama pada prosedur tersimpan adalah untuk menyembunyikan query SQL yang langsung dari kode dan meningkatkan kinerja operasi databease seperti memiliih, memperbaharui, dan menghapus data.

**Contoh:**

CREATE PROCEDURE selectAllBuyer’s

AS

SELECT \* FROM Buyer

GO;

**SQL comments**

Seperti pada bahasa pemrograman lainnya, komentar sangat penting dalam SQL, itu dikarenakan dalam SQL komentar adalah bagian terpenting dalam bahasa apapun. Komentar SQL menunjukkan tujuan kode dan pembuat kode yang akan mendapatkan ide tentang mengapa kode ini ditulis dan akan mendapatkan ide tentang tujuan kode. Komentar juga dapat membuat aplikasi kita lebih muda dibaca dan dipelihara. Misalnya kita menyertakan komentar dalam pernyataan yang menjelaskan tujuan pernyataan dalam aplikasi kita. Dengan pengecualian petunjuk, komentar dalam pernyataan SQL tidak memengaruhi eksekusi pada pernyataan. Komentar dapat kita tulis dalam format: komentar saatu baris, komentar multi baris, dan dalam komentar baris.

**Contoh:**

Berikut ini adalah contoh yang menggunakan komentar multi-baris sebagai penjelasan berikut dibawah ini:

/\* select all the columns

Of all the records

In the Buyer table :\*/

SELECT \* FROM Buyer;

**SQL Operators**

Kita ketahui bahwa yang dimaksud operator dalam SQL adalah kata atau karakter yang khusus yang sering digunakan terutama dalam clause WHERE pada pernyataan SQL untuk melakukan operasi, misalnya seperti pada perbandingan dan operasi aritmatika. Pada operator ini biasanya digunakan untuk menentukan kondisi pada pernayataan SQL dan berfungsi sebagai konjungsi untuk beberapa kondisi pada sebuah pernyataan. Karakter yang sering digunakan dalam operator adalah operator aritmatika, operator perbandingan, operator logika, operator biasa.

**Contoh :**

* Berikut ini adalah contoh penggunaan pada symbol (+)

INPUT OUTPUT

SQL SELECT ITEM> PRODUCE, WHOLESALE, WHOLESALE + 0,5 ECERAN DARI HARGA;

Dibawah ini kita akan mengganti nama kolomnya.

WHOLESALE PRODUCE ECERAN

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

PENSIL 0,34 0,84

BUKU 0,10 0,6

PENGGARIS 0,20 0,7

SEPATU 0,25 0,75

**2. SQL DATABASE**

**SQL Database**

Apa yang dimaksud dengan database? Database adalah tempat menyimpan data. Sistem database relasional (RDMS) menyimpan sebuah data kedalam tabel. Pada setiap tabel itu tersebut harus memiliki baris kolom, seperti spreadsheet. Setiap baris juga harus mempunyai satu atau lebih kolom untuk digunakan dalam menyimpan nilai data.

**SQL create DB**

Pada pernayataan SQL Create DataBase itu kita gunakan untuk membuat database yang baru.

**Contoh:**

Pada pernyataan untuk CREATE DATABASE biasanya digunakan untuk membuat databaseSQL yang baru.

CREATE DATABASE databasename;

Selanjutnya adalah SQL yang membuat database yang disebut dengan “testDB”:

Misalanya:

CREATE DATABASE testDB

**SQL Drob DB**

Fungsi statement DROP DATABASE adalah untuk menghapus database. Satu hal yang perlu diperhatikan ialah, MySQL tidak akan memberi konfirmasi penghapusan database jika anda menggunakan statement ini. Oleh karenanya anda harus berhati-hati menggunakan statement ini karena semua record, tabel dan objek yang ada dalam database anda akan ikut hilang bersama database yang anda hapus.

**Contoh :**

DROP DATABASE databasename;

Berikut Ini adalah contoh dari backup database

DROP DATABASE testDB;

**SQL Backup DB**

Untuk Backup DB sendiri dapat kita gunakan dalam membuat cadangan penuh dari database SQL yang ada. Peraturannya:

**Contoh:**

BACKUP DATABASE databasename

TO DISK = ‘FilePath’

WITH DIFFERENTAL;

Berikut ini adalah contoh dari backup database:

BACKUP DATABASE testDB

TO DISK = ‘D:/backup/testDB.bak’

WITH DIFFERENTAL;

**SQL CREATE TABLE**

Untuk Create Table sendiri dapat kita gunakan dalam membuat table baru didalam database.

**Contoh:**

pada pernyataan berikut ini kita akan membuat tabel yang bernama “Shop Name” yang pada nama tabel tersebut berisi lima nama kolom yaitu ShopID, Address, ItemCode, NameOfGoods, dan Price:

CREATE TABLE Shop Name (

ShopID int,

Address Varchar(10),

ItemCode Varchar(10),

NameOfGoods Varchar(10),

Price Varchar (10)

);

**SQL DROP TABLE**

Untuk Perintah drop table dapat digunakan untuk menghapus tabel dan semua baris dalam tabel. Untuk menghapus seluruh tabel termasuk semua barisnya, kita jalankan perintah drop table diikuti dengan tablename. drop table berbeda dengan menghapus semua record dalam tabel.

**Contoh:**

DROP TABLE\_Name;

SQL> DESC TRANSFER;

5 rows in set

Jadi berikut ini adalah tabel yang berisi tentang pengiriman barang yang ada di data base, jadi disini kita akan menjatuhkan seperti yang dibawah ini.

SQL> DROB TABLE TRANSFER;

QUERY OK, 0 rows affected (o,o1 total)

**SQL Alter Table**

Pada pernyataan untuk ALTER TABLE biasanya digunakan untuk menambahkan, mengghapus, atau bisa juga dengan mengubah kolom yang ada didalam tabel yang sudah ada. Untuk pernyataan ALTER TABLE biasanya digunakan untuk menambahkan dan menghapus dari berbagai batasan pada tabel yang telah ada.

**Contoh:**

Berikut ini adalah bagian untuk menembahkan kolom pada tabel, berikut ini dengan menggunakan sintaks:

ALTER TABLE table\_name

ADD column\_name datatype;

SQL berikut ini adalah untuk menambahkan “price” ke dalam tabel “buyer”, seperti dibawah ini:

ALTER TABLE buyer’s

ADD Price varchar (255);

**SQL CONTRAINTS**

SQL contraints dapat digunakan dalam menentukan aturan untuk data dalam tabel. Contrains sendiri digunakan untuk membatasi jenis data yang bisa masuk ke dalam tabel.

Contraints dapat berupa tingkat kolom atau tingkat tabel. Contraints tingkat kolom berlaku untuk kolom, dan contrains tingkat tabel berlaku untuk seluruh tabel.

**Contoh:**

CREATE TABLE table\_name (

ShopID datatype (constraint),

NameOfGoods datatype (constraint),

Price datatype (constraint),

);

**SQL not NULL**

SQL Not Null yaitu bidang tanpa nilai. Jika bidang dalam tabel bersifat opsional, dimungkinkan untuk menyisipkan catatan baru atau memperbarui catatan tanpa menambahkan nilai ke bidang ini. Kemudian, field akan disimpan dengan nilai NULL.

Nilai null berbeda dari nilai nol atau bidang yang berisi spasi. Bidang dengan nilai null adalah bidang yang dibiarkan kosong selama pembuatan catatan. **Contoh:**

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255) NOT NULL,

Total int

);

Ketika kita membuat batasan NOT NULL pada kolom “total” ketika pada tabel “Goods Shop” yang sudah dibuat, gunakan SQL berikut ini:

ALTER TABLE ShopGoods

MODIFY Total int NOT NULL;

**SQL Unique**

SQL UNIQUE yaitu memastikan bahwa semua nilai dalam kolom berbeda. Baik kendala UNIQUE dan PRIMARY KEY memberikan jaminan keunikan untuk kolom atau kumpulan kolom. Batasan PRIMARY KEY secara otomatis memiliki batasan UNIK. Namun, Anda dapat memiliki banyak batasan UNIK per tabel, tetapi hanya satu batasan PRIMARY KEY per tabel.

**Contoh:**

pada bagian berikut ini adalah untuk membuat batasan yang unik pada kolom “Shop ID” pada saat tabel “Goods Shop” yang telah dibuat:

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255) NOT NULL,

Total int

);

**MySQL:**

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255) NOT NULL,

Total int

UNIQUE (ID)

);

Untuk pemberian penamaan batasan yang UNIK, dan untuk menentukan batasan yang UNIK pada beberapa kolom, berikut ini adalah gunakan sintaks SQL berikut:

CREATE TABLE Goods Shop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255) NOT NULL,

Total int

CONSTRAINT UC\_Goods Shop UNIQUE (ShopID, NameOfGoods)

)

**SQL PRIMARY KEY**

Batasan primary key secara unik mengidentifikasi setiap record dalam table. Kunci utama harus berisi nilai UNIK, dan tidak boleh berisi nilai NULL.

Sebuah tabel hanya dapat memiliki SATU kunci utama; dan dalam tabel, kunci utama ini dapat terdiri dari satu atau beberapa kolom (bidang).

**contoh:**

pada bagian berikut ini adalah untuk membuat batasan yang unik pada kolom “Shop ID” pada saat tabel “Goods Shop” yang telah dibuat:

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255)

Total int,

PRIMARY KEY (SHOP ID)

);

Akses SQL Server / Oracle / MS:

CREATE TABLE GoodsShop (

ShopID int NOT NULL PRIMARY KEY,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255)

Total int,

);

Untuk memperbolehkan penamaan pada batasan PRIMARY KEY, daan untuk menentukan batasan PRIMARY KEY pada beberapa kolom, kita harus menggunakan sintaks SQL berikut ini:

CREATE TABLE Goods Shop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255)

Total int,

CONSTRAINT PK\_GoodShop UNIQUE (ShopID, NameOfGoods)

);

Untuk contoh diatas hanya ada SATU KATA KUNCI UTAMA (PK\_GoodsShop). Akan tetapi, NILAI kunci yang paling utama terdiri dari DUA KOLOM yaitu seperti (ShopID + Nama Barang). Ketika kita membuat batasan PRIMARY KEY pada kolom “ShopID” pada saat tabel sudah dibuat, kita gunakan SQL berikut ini:

ALTER TABLE Goods Shop

ADD PRIMARY KEY (ID);

Untuk memperbolehkan penamaan batasan PRIMARY KEY, dan untuk menentukan batasan PRIMARY KEY pada beberapa kolom, kita menggunakan sintaks SQL berikut ini:

ALTER TABLE Goods Shop

ADD CONSTRAINT PK\_GoodsShop PRIMARY KEY (ShopID,NameOfGoods);

Ketika kita menggunakan pada pernyataan ALTER TABLE untuk menanmbahkan kunci utama, kolom kunci utama itu harus sudah dinyatakan tidak berisi nilai NULL (pada saat tabel pertama kali kita buat).

Untuk menjatuhkan batasan PRIMARY KEY, kita menggunakan SQL berikut ini:

ALTER TABLE Goods Shop

DROB PRIMARY KEY;

Akses SQL Server / Oracle / MS:

ALTER TABLE Goods Shop

DROB CONSTRAINT PK\_GoodsShop;

**SQL Foreign key**

FOREIGN KEY adalah kunci yang digunakan untuk menghubungkan dua tabel bersama-sama.

FOREIGN KEY adalah bidang (atau kumpulan bidang) dalam satu tabel yang merujuk ke PRIMARY KEY di table lain. Tabel yang berisi kunci asing disebut tabel anak, dan tabel yang berisi kunci kandidat disebut tabel referensi atau induk.

**Contoh:**

CREATE TABLE Buyer(

BuyerID int NOT NULL,

BuyerNumber int NOT NULL,

GoodsShopID int,

PRIMARY KEY (BuyerID),

FOREIGN KEY (GoodsShopID)REFERENCES GoodsShop(GoodsShopID)

);

SQL SERVER / Oracle / MS Access:

CREATE TABLE Buyer (

BuyerID int NOT NULL PRIMARY KEY,

BuyerNumber int NOT NULL,

GoodsShopID int FOREIGN KEY REFERENCES GoodsShop(GoodsShopID)

);

Untuk memperbolehkan pada penamaan batasan FOREIGN KEY, dan untuk menentukan batasan FOREIGN KEY pada beberapa kolom, kita bisa menggunakan sintaks berikut ini:

CREATE TABLE

BuyerID int NOT NULL,

BuyerNumber int NOT NULL,

GoodsShopID int,

PRIMARY KEY (BuyerID),

CONSTRAINT FK\_GoodsShop FOREIGN KEY (GoodsShopID)

REFFERENCES GoodsShop(GoodsShopID)

);

**SQL check**

Batas periksa digunakan untuk membatasi jumlah nilai yang dapat ditempatkan pada kolom, jika kita menentukan Batasan periksa pada suatu kolom itu hanya akan memungkinkan milai-nilai tertentu pada kolom tersebut. Dan jika kita menentukan Batasan periksa pada tabel hal itu dapat membatasi nilai pada kolom tersebut berdasarkan nilai pada kolom lai pada baris.

**Contoh:**

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255),

Total int,

CHECK (Total>=18)

Akses SQL Server / Oracle / MS:

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255),

Total int, CHECK (Total>=18)

);

Pada penamaan batasan PERIKSA, dan untuk batasan PERIKSA, pada beberapa kolom, dan menggunakan sintaks SQL berikut ini.

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255),

Total int,

Address Varchar(255),

CONSTRAINT CHK\_GoodsShop CHECK (Total>=18 AND = ‘jln.cendrawesi 200’)

);

**SQL default**

Digunakan untuk memberikan nilai default untuyk sebuah kolom. Dan nilai default digunakan untuk menambahkan keseluruh rekaman yang baru jika tidak memiliki nilai yang lain untuk ditentukan.

**Contoh:**

CREATE TABLE GoodsShop (

ShopID int NOT NULL,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255),

Total int,

Address Varchar(255)DEFAULT ‘jln.cendrawesi 200’

);

Batasan DEFAULT itu biasanya dapat juga kita gunakan untuk memasukkan nilai sistem, dengan menggunakan fungsi sepsrti GETDATE ():

CREATE TABLE Buyer(

BuyerID int NOT NULL,

BuyerNumber int NOT NULL,

BuyerDate date DEFAULT GETDATE ()

);

**SQL INDEX**

SQL Index adalah sebuah objek dalam sistem database yang dapat mempercepat proses pencarian (query) data. Hal ini dikarenakan resource CPU banyak digunakan untuk pencarian data atau pengaksesan query SQL dengan metode table-scan. Index membuat pencarian data akan lebih cepat dan tidak banyak menghabiskan resource CPU

Fungsi dari SQL Indeks dapat digunakan untuk secara efisien menemukan semua baris yang cocok dengan beberapa kolom dalam kueri Anda dan kemudian hanya menelusuri sebagian dari tabel untuk menemukan kecocokan yang tepat. Jika Anda tidak memiliki indeks pada kolom apa pun dalam WHERE klausa, SQLserver harus menelusuri seluruh tabel dan memeriksa setiap baris untuk melihat apakah cocok, yang mungkin merupakan operasi lambat pada tabel besar.

**Contoh:**

CREATE INDEX idx\_NameOfGoods

ON GoodsShop (NameOfGoods);

Pada saat kita menggunakan ingin membuat indeks pada kombinasi kolom, kita bisa membuat daftar nama kolom dalam tanda kurung, dipisahkan dengan koma:

CREATE INDEX idx\_name

ON GoodsShop (NameOfGoods, ItemCode);

**SQL AUTO INCREMENT**

Auto Increment merupakan fitur pada database yang memungkinkan pengguna membuat angka unik yang dihasilkan secara otomatis dan berurut untuk setiap rekor / catatan baru yang dimasukkan / di input kedalam table database tersebut. Dan kebanyakan orang menggunakan angka unik Auto Increment ini sebagai Primary Key / Kunci Utama pada tabel database tersebut.

Fungsi dari SQL Auto increment adalah membuat angka berurut dari mulai terkecil sampai terbesar dan tidak perlu diisi secara manual ketika memasukkan data ke table, Jadi, misalnya ketika memasukkan data ke table contohTutorial kita cukup memasukkan data nama, alamat dan tanggalLahir saja biarkan kolom id dikosongkan karena id akan secara otomatis membuat angka unik menyusun dari angka sebelumnya. Tapi jika ingin menentukannya secara manual juga bisa, cukup masukkan angka di kolom id dengan angka yang diinginkan. Angka tersebut tetap akan menjadi id di kolom id.

**Contoh:**

CREATE TABLE GoodsShop (

ShopID int NOT NULL, AUTO\_INCREMENT,

NameOfGoods Varchar(255) NOT NULL,

Price Varchar(255),

Total int,

PRIMARY KEY (Personid)

);

Secara default, nilai pertama untuk auto AUTO\_INCREMENT adalah 1, dan akan bertambah 1 pada setiap record baru. Kita membiarkan urutan AUTO\_INCREMENT dimulai dengan nilai yang lain, kita menggunakan pernyataan SQL berikut ini:

ALTER TABLE GoodsShop AUTO\_INCREMENT=100;

Untuk menambahkan record yang baru kedalam tabel “GoodsShop”, kita TIDAK perlu menentukan nilai untuk untuk kolom “personid” (nilai yang unik untuk ditambahkan secara otomatis)

INSERT INTO GoodsShop (NameOfGoods, ItemCode)

VALUES (Novel,POR-0000019)

**SQL DATES**

Fungsi SQLDATES untuk menghasilkan hanya bagian tanggal dari tanggal dan jam. Fungsi ADDDATE () untuk menghasilkan tanggal beberapa HARI atau BULAN kemudian. Memperoleh umur pegawai dengan membandingkan tanggal lahir dan tanggal sekarang kemudian dibagi dengan 365 (1 tahun = 365 hari), dan Fungsi dates dalam melakukan manipulasi data dengan bahasa query merupakan fungsi yang diperuntukkan bagi data, variable atau field yang berkaitan dengan data tanggal dan dapat pula diterapkan untuk data dari tanggal sistem komputer yang sedang aktif. Fungsi tanggal dalam SQL (Structured Query Language) meliputi fungsi DateAdd, DateIff, DatePart, DateName dan GetDate. Data Manipulation Language (DML) merupakan bahasa basis data yang berfungsi untuk melakukan penelusuran data pada suatu basis data didalam DBMS (Database Management System) diantaranya melakukan insert data, delete data, update data dan penelurusan data (query). SQL (Structured Query Language) terdiri dari sintaks sederhana dalam bentuk instruksi-instruksi dalam melakukan manipulasi data, instruksi tersebut sering disebut dengan query.

**Contoh:**

SELECT \* FROM Buyer WHERE BuyerDate=’2020-12-25’

**SQL VIEWS**

SQL VIEWS adalah perintah query yang disimpan pada database dengan suatu nama tertentu, sehingga bisa digunakan setiap saat untuk melihat data tanpa menuliskan ulang query tersebut dan dapat didefenisikan sebagai **‘tabel virtual’**. Tabel ini bisa berasal dari tabel lain, atau gabungan dari beberapa table.

Tujuan dari pembuatan **VIEWS** adalah untuk kenyamanan (mempermudah penulisan query), untuk keamanan (menyembunyikan beberapa kolom yang bersifat rahasia), atau dalam beberapa kasus bisa digunakan untuk mempercepat proses menampilkan data (terutama jika kita akan menjalankan query tersebut secara berulang).

**Contoh:**

CREATE VIEW Syntax

CREATE VIEW view\_name AS

SELECT COLUMN1, COLUMN2...

FROM Table\_name

WHERE condition;

Pada SQL berikut ini membuat tampilan yang memperlihatkan seluruh buyers dari Indonesia

CREATE VIEW (Indonesia Buyer) AS

SELECT Buyer’sName, ItemCode

FROM Buyer

WHERE Country = ‘indonesia’;

Pada SQL berikut ini adalah untuk membuat tampilan yang memilih pada setiap barang dalam tabel “Goods” dengan harga yang besar dari harga yang rata-rata:

CREATE VIEW (Goods Above Average Price)AS

SELECT NameOfGoods, Price

FROM Goods

WHERE Price > (SELECT AVG(Price) FROM Goods);

Berikut ini kita dapat query view di atas sebagai berikut dibawah ini:

SELECT \* FROM (Goods Above Average Price

**SQL INJECTION**

SQL injection adalah sebuah aksi hacking yang dilakukan di aplikasi client dengan cara memodifikasi perintah SQL yang ada di memori aplikasi client. SQLInjection merupakan teknik mengeksploitasi web aplikasi yang didalamnya menggunakan database untuk penyimpanan data dan SQL injection adalah teknik hacking untuk mendapatkan akses pada sistem database yang berbasis SQL. SQL merupakan singkatan dari Structured Query Language yaitu bahasa yang digunakan untuk membuat serta mengolah database. Dalam melakukan teknik SQL injection, para hacker akan memanfaatkan celah keamanan pada web atau aplikasi.

Mereka akan memasukkan perintah-perintah SQL ke dalam database mesin server sehingga mereka dapat masuk ke dalam sistem tanpa harus memiliki username dan password administrator. SQL injection ini dapat terjadi karena beberapa hal seperti kurangnya penanganan terhadap karakter-karakter seperti tanda petik satu atau karakter double minus yang dapat menyebabkan suatu aplikasi dapat disisipi peretas dengan perintah SQL.

Tujuan dari SQL INJECTION

* Mengidentifikasi parameter injeksi Penyerang ingin menyelidiki Aplikasi Web untuk menemukan mana parameter dan bidang-input pengguna rentan terhadap SQLIA.
* Menambah dan memodifikasi data tujuan dari serangan ini adalah untuk menambah atau mengubah informasi dalam database
* Menggali data pada sebuah web jenis-jenis serangan menggunakan teknik yang akan mengekstrak nilai data dari database. Tergantung pada jenis dari aplikasi Web, informasi ini bisa menjadi sensitif dan sangat diinginkan untuk penyerang. Serangan dengan maksud ini adalah Jenis yang paling umum dari SQLIA.
* Mengeksekusi perintah Jarak jauh jenis serangan berusaha untuk mengeksekusi perintah sewenang-wenang pada database. Perintah-perintah ini dapat disimpan prosedur atau fungsi yang tersedia bagi pengguna database.

**Contoh:**

txtUserId = getRequestString(“UserId);

txtSQL = “SELECT \* FROM User WHERE UserId = “ + txtUserId;

SQL Injectoion berdasarkan 1=1 selalu benar

SELECT \* FROM User WHERE UserId = 110 OR 1=1;

**SQL HOSTING**

Hosting mempunyai peran penting dalam kebutuhan untuk suatu website. Jika beberapa waktu yang lalu kita sudah membahas tentang mengenai pengertian diomain dan fungsinya, kali ini tim IDCloudHost mengajak teman-teman untuk memahami sedikit mengenai Hostingan dan Fungsinya untuk website dan aplikasi. **Hostingan** atau sering juga disebut dengan WebHosting merupakan suatu infrastruktur yang digunakan untuk menyimpan data dari website atau aplikasi sehingga bisa diakses menggunakan Internet. Singkatnya jika Smartpohone teman-teman membutuhkan kartu memori untuk menyimpan data seperti gambar, file, dan lainnya. Nah, fungsi hostingan mempunyai peran yang sama seperti kartu memori yang digunakan pada smartphone.

Hostingan sendiri mempunyai kapasitas yang bisa digunakan sesuai dengan kebutuhan suatu website atau aplikasi. Misalnya Hostingan digunakan oleh website universitas untuk menyimpan data para mahasiswa, atau bisa juga menyimpan informasi seperti gambar, artikel, file dokumen dan lainnya di database yang ada pada hostingan.

**SQL DATA TYPES**

Pengetahuan tentang type data sangat penting diketahui oleh setiap programmer yang menggunakan database sebagai gudang data. Secara umum, setiap Database Management System (DBMS) menggunakan type data yang hampir sama. Baik itu mysql, oracle, mariaDB, mongoDB, SQLite, SQL Server, dan lainnya.

Tipe data adalah skema pengkodean rinci yang dikendalikan oleh DBMS untuk mempresentasikan data yang terorganisasi. Setiap data memiliki tipe data, beberapa tipe data sebagai berikut:

* INT merupakan type data angka (Integer) yang berukuran normal. Jangkauan nilainya adalah dari -21474883648 hingga 214783647.
* BIGINT Merupakan type data Integer (angka) yang berukuran besar. Jangkauan nilainya adalah dari -9223372036854775808 hingga 9223372036854775807.
* SMALLINT Merupakan type data Integer berukuran kecil. Jangkauannya dari-32768 hingga 32767.

**3. SQL REFERENCE**

**SQL Reference**

Merupakan kumpulan pernyataan yang kita gunakan untuk semua program atau pengguna untuk mengakses semua data dalam database oracle, program aplikasi dan alat oracle sering memungkinkan pengguna mengakses semua database tanpa menggunakan SQL secara langsung tetapi aplikasi ini memiliki giliran yang harus menggunakn SQL, saat kita menjalankan permintaan penggunaan.

**SQL keywords**

Memiliki kata khusus dalam opersai database, kata kunci cadangan adalah bagian dari tata Bahasa SQL server.

SQL server telah mengakui kata masa kini dan masa depan. Katta kunci dapat kita gunakan untuuk pengenal den gan menempatkan diantaranya [] (tanda kurung siku).

**MYSQL functions**

Merupakan program yang tersimpan dimana kita dapat memasukkan parameter ke dalamnya dan mengembalikan nilai.

**SQL server functions**

Memiliki banyak fungsi bawaan, referensi ini memiliki isi string, numerik, tanggal, dan beberapa fungsi lanjutan pada mysql.

**MS access functions**

Merupakan prosedur VBA untuk melakukan tugas, perhitungan, dan mengembalikan hasil. Fungsi umum dapat kita gunakan dalam aplikasi, tetapi ada tempat lain yang dapat menggunakan fungsi.

**SQL Quick Ref**

**Contoh :**

**AND / OR**

SELECT column\_name(s)

FROM table\_name

WHERE condition

AND|OR condition

**DAFTAR PUSTAKA**

W3school.com, 2020 SQL Tutorial <https://www.w3schools.com/sql/default.asp> ,

Dofactory, Subquery EXISTS-whit Examples-Dofactory <https://www.dofactory.com/sql/where-exists>

<https://etapetap.blogspot.com/2009/11/create-alter-dan-drop-database.html>