SELECT TABEL

Query SELECT

Semua data dalam table dapat dimunculkan dengan perintah SELECT, dalam SQL semua kata (all) diartikan dengan bintang (*)

Contoh :
*SELECT * FROM penduduk;*

SELECT Kolom Tertentu

Syntax :

SELECT kol1, kol2, kol3, kol-n FROM nama_tabel;

Contoh:

SELECT nama, alamat, telp FROM pegawai;

SELECT Beberapa Baris

Syntax :

SELECT * FROM nama_tabel [Where Cond];

Contoh:

SELECT * FROM penduduk WHERE id > 10;

Operator Pembanding

Operator Perbandingan	Arti
=	Sama dengan
<	Lebih kecil dari
<=	Lebih kecil dari atau sama dengan
>	Lebih besar dari
>=	Lebih besar dari atau sama dengan
<>	Tidak sama dengan

SELECT Kolom & Baris Tertentu

Akan ditampilkan data kod,nama,stok,min_stok untuk produk yang mempunyai stok kurang dari 40

SELECT kode,nama,stok,min_stok FROM produk WHERE stok < 40;

Klausa AND , OR

Penggunaan klausa WHERE dapat mengandung beberapa kondisi yang dihubungan dengan AND atau OR.

Sebuah klausa mengandung AND berarti semua kondisi harus bernilai true untuk menampilkan kolom yang dipilih.

Pada kasus OR berarti paling tidak satu kondisi benar maka bernilai true untuk menampilkan kolom yang dipilih.

Contoh Klausa AND

Tampilkan seluruh dimana harga produk diatas 1000000 dan stok kurang dari 30

SELECT * FROM produk WHERE harga > 1000000 AND stok < 30;

Key & Non Key

Query berdasarkan dengan Key dan Non Key bertujuan untuk menggaransi bahwa sembarang baris pada table dapat dinyatakan unik.

Contoh Query dgn Key

Pada contoh ini, digunakan kolom kode untuk menampilkan satu baris data dikarenakan kode adalah unik untuk setiap instan dari produk dengan kode STR8

```
SELECT * FROM produk WHERE kode='STR8';
```

Contoh Query dgn Non Key

Query berdasarkan dengan kolom non primary key tidak menjamin untuk mendapatkan satu baris data, berikut contohnya :

Tampilkan data produk yang mempunyai minimum stok 4

```
SELECT * FROM produk WHERE min_stok=4;
```

Klausa IN & NOT IN

Klausa IN digunakan untuk menampilkan data berdasarkan himpunan dari suatu nilai.

Keyword IN selalu digunakan bersamaan dengan nama kolom. Semua baris yang mempunyai nilai yang sesuai dengan himpunan nilai yang dipilih akan ditampilkan

Contoh Klausa IN

Tampilkan data produk yang mempunyai kode produk KU02, STR8 atau MAG6

SELECT * FROM produk WHERE kode IN ('KU02','STR8','MAG6');

Contoh Klausa NOT IN

Tampilkan data produk yang mempunyai kode produk SELAIN KU02, STR8 atau MAG6

```
SELECT * FROM produk WHERE kode NOT IN
('KU02','STR8','MAG6');
```

Mengurutkan Data

Mengurutkan data sangat penting untuk mengubah atau memodifikasi data dalam suatu tabel untuk keperluan laporan.

Ada dua hal yang bisa diurutkan datanya berdasarkan :

- Mengurutkan berdasarkan kolom
- Mengurutkan berdasarkan baris

Mengurutkan Kolom

Urutan kolom dalam perintah SQL adalah urutan kolom yang akan ditampilkan dalam hasil perintah SQL.

Contoh :

```
SELECT nama,kode FROM produk WHERE min_stok =4;
```

Mengurutan Baris

Gunakan klausa ORDER BY untuk mengurutkan data secara terurut dalam sebuah laporan.

Secara default urutan perintah ORDER BY adalah ascending atau dari terkecil hingga terbesar (A sebelum B, 1 sebelum 2).

Sedangan untuk mengurutkan dari besar terkecil tambahkan klausa DESC setelah nama kolom yang akan diurutkan.

Klausa ORDER BY ASC

Urutkan data produk berdasarkan nama secara ascending dimana harga produk diatas 500000 dan urutkan nama secara alpabet.

SELECT * from produk WHERE harga > 500000 ORDER BY min_stok, nama;

Klausa ORDER BY DESC

Urutkan data produk berdasarkan harga secara descending atau dari harga yang paling mahal

SELECT * from produk ORDER BY harga DESC;

Klausa LIMIT & OFFSET

Klausa LIMIT digunakan untuk menampilkan jumlah baris data yang diinginkan.

Klausa OFFSET digunakan untuk memilih awal mulai baris data ditampilkan dengan data pada baris pertama dimulai dari 0

Contoh Klausa LIMIT

Tampilkan 2 baris pertama data pegawai

SELECT id,nip,nama,tgl_lahir,telpon FROM pegawai LIMIT 2;

Contoh Klausa OFFSET

Tampilkan 2 baris data mulai dari data ke 4 (baris data dimulai dari 0)

```
SELECT id,nip,nama,tgl_lahir,telpon FROM pegawai LIMIT 2 OFFSET 3;
```

Fungsi Aggregate

SQL mempunyai fungsi built-in yang dapat digunakan untuk menampilkan total data dari kolom tertentu. Berikut diantaranya fungsi-fungsi aggregate :

- AVG
- SUM
- MIN
- MAX
- COUNT

Fungsi Aggregate

SQL mempunyai fungsi built-in yang dapat digunakan untuk menampilkan total data dari kolom tertentu. Berikut diantaranya fungsi-fungsi aggregate :

- AVG
- SUM
- MIN
- MAX
- COUNT

Fungsi Aggregate

SQL mempunyai fungsi built-in yang dapat digunakan untuk menampilkan total data dari kolom tertentu. Berikut diantaranya fungsi-fungsi aggregate :

- AVG
- SUM
- MIN
- MAX
- COUNT

Contoh 1 Fungsi Aggregate

COUNT digunakan untuk menghitung jumlah baris dalam suatu table. Jumlah baris dihitung walaupun mengandung nilai null. COUNT dapat digunakan dengan klausa WHERE untuk menampilkan data yang spesifik.

Berapa jumlah produk yang ada ?

```
SELECT COUNT(*) AS jumlah_produk FROM produk;
```

Contoh 2 Fungsi Aggregate

Berapa jumlah produk yang harganya dibawah 1juta ?

SELECT COUNT(*) AS jumlah_produk FROM produk WHERE harga < 1000000;

SUB QUERY

SUB Query adalah query yang menjadi bagian dari query lain, lebih tepatnya menjadi bagian dari statement SQL lainnya (karena subquery tidak hanya menjadi bagian dari statement select saja tetapi dapat juga menjadi bagian dari statement insert, update dan delete).

SUB query terdiri dari dua jenis yaitu :

- uncorrelated subquery
- correlated subquery

uncorrelated subquery

Contoh Anda ingin menampilkan baris pada tabel purchase_order_items yang memiliki jumlah kuantitas barang paling besar.

SELECT * FROM purchase_order_items WHERE qty=(SELECT MAX(qty) FROM purchase_order_items)

correlated subquery

Adalah subquery yang menggunakan ekspresi nilai yang mengacu pada ekspresi nilai dari query diluarnya.
Contoh Anda ingin menampilkan nama barang yang jumlah kuantitasnya dalam purchase_order_items lebih besar dari 30 dengan tanggal PO nya dari tanggal 9 April 2011 sampai 29 Mei 2011.

```
SELECT * FROM barang b WHERE 30 < (SELECT SUM(qty) FROM purchase_order_items i , purchase_order o WHERE i.id_bar=b.id and i.id_po=o.id AND o.tgl_po BETWEEN '2011-04-09' AND '2011-05-29');
```

KLAUSA LIKE

Klausa LIKE digunakan untuk pencarian data dengan pola kata kunci pencarian yang diinginkan. Klausa LIKE menggunakan dua simbol untuk menentukan pola pencarian :

- Tanda persen (%) adalah simbol untuk sembarang karakter atau bukan karakter.
- Tanda garis bawah (_) adalah simbol untuk satu karakter

CONTOH KLAUSA LIKE

Mencari nama produk yang mempunyai nama awalan huruf K
SELECT nama FROM produk WHERE nama LIKE 'K%';

Mencari nama produk yang mengandung tulisan TA
SELECT nama FROM produk WHERE nama LIKE '%TA%';

Tampilkan nama produk yang karakter ke tiga adalah huruf L
SELECT nama FROM produk WHERE nama LIKE '__L%';

Tampilkan nama produk yang tidak mengandung huruf S
SELECT nama FROM produk WHERE nama NOT LIKE '%S%';

KLAUSA DISTINCT

Klausa DISTINCT digunakan untuk meniadakan duplikasi baris data.

Klausa DISTINCT dapat digunakan sebelum nama kolom, ketika menggunakan nama kolom akan meniadakan duplikasi nilai yang ditampilkan

CONTOH KLAUSA DISTINCT

Tampilkan HARGA produk yang berbeda.
SELECT DISTINCT harga FROM produk;

Cari jumlah dari produk yang mempunyai minimum stok yang berbeda.
SELECT COUNT(DISTINCT min_stok) AS 'Jumlah data Minimum Stok yang Berbeda' FROM produk;

KLAUSA CASE

Klausa CASE digunakan untuk seleksi pada statement query.

Klausa CASE sama seperti logika IF, jika persyaratan statement query benar maka akan menghasilkan nilai TRUE, dan sebaliknya akan menghasilkan nilai FALSE

Penggunaan Klausa CASE akan membentuk output tersendiri dengan terbentuknya kolom baru

CONTOH KLAUSA CASE

Pada tabel nilai_siswa terdapat nilai –nilai siswa. Siswa dinyatakan lulus ketika nilainya minimal 60. Maka Querynya adalah sebagai berikut :

```
SELECT nis,nama,nilai,
    CASE
    WHEN nilai >= 60 THEN 'Lulus'
    ELSE ‘Gagal'
    END AS keterangan
FROM nilai_siswa;
```

TABEL VIEW

Tabel view adalah tabel bayangan untuk perintah SELECT yang panjang. Dengan adanya tabel bayangan ini memudahkan user untuk query SELECT.

Syntax :
CREATE VIEW nama_tabel_view AS [query SELECT];

Running Tabel View :
SELECT * FROM nama_tabel_view;

KLAUSA GROUP BY

Klausa ini digunakan untuk mengelompokkan data.

Contoh :
Mengelompokkan jumlah penganut masing-masing Agama di tabel penduduk

Syntax :
SELECT agama, COUNT(agama) AS jml_penganut FROM penduduk GROUP BY agama;

KLAUSA HAVING

Klausa Having ini digunakan untuk menyaring data yang sudah dikelompokkan.

Contoh :
Menyaring Data Penganut Agama > 10 yang dikelompokkan berdasarkan agama.

Syntax :
SELECT agama, COUNT(agama) AS jml_penganut FROM penduduk GROUP BY agama HAVING count(agama) > 10;

INDEX(1)

Ketika mengakses sebuah tabel biasanya PostgreSQL akan membaca seluruh tabel baris per baris sampai selesai. Ketika jumlah row sangat banyak sedangkan hasil dari query hanya sedikit, maka hal tersebut sangat tidak efisien.

Seperti halnya ketika kita membaca sebuah buku, dan ingin mencari kata atau istilah tertentu dalam buku maka biasanya akan di cari dengan membuka setiap halaman dari awal sampai akhir. Dengan adanya indeks pada buku maka kita cukup membuka indeks, sehingga dengan cepat bisa mengetahui posisi halaman dimana kata yang dicari berada.

INDEX(2)

Dalam database juga demikian karena dengan indeks dapat dengan cepat dan spesifik menemukan nilai dalam indeks, dan langsung mencocokkan dengan row yang dimaksud. Sebagai contoh, perhatikan query berikut ini SELECT * FROM customer WHERE col = 26.

Tanpa indeks, PostgreSQL harus mengamati atau meninjau seluruh tabel untuk mencari di mana row col yang sama dengan 26. Jika menggunakan indeks maka PostgreSQL akan langsung menuju ke row yang sama dengan 26.

Untuk sebuah tabel yang besar dan luas,PostgreSQLdapat mengecek setiap row dalam menit sedangkan jika menggunakan indeks untuk menemukan spesifik row waktu yang diperlukan hanya sedikit (dalam hitungan detik).

Lembaga Pendidikan & Pengembangan Profesi Terpadu

LP3T NURUL FIKRI

IT Training - IT Education

LP3T-NF

PENAMAAN INDEX

Primary Key di suatu tabel otomatis sudah dianggap index. Penamaan indeks dapat diberikan dengan bebas, akan tetapi sangat disarankan jika penamaanya mewakili nama yang di-indeks sehingga memudahkan perawatan. Selain itu indeks sebaiknya jangan digunakan pada tabel yang sangat jarang atau tidak pernah diakses.

Sekali sebuah indeks dibuat, maka tidak diperlukan lagi interfensi user karena sistem akan secara otomatis melakukan update indeks ketika terjadi perubahan dalam tabel. Selain untuk perintah SELECT Indeks juga bermanfaat untuk UPDATE dan DELETE yang menggunakan kondisi pencarian.

SYNTAX INDEX

CREATE INDEX nm_index_idx ON nm_tabel(field);

Testing Index :

- Sebelum index dibuat test suatu query dengan syntax sbb :

  EXPLAIN [query …];

  Query ini akan memberikan informasi
- Lalu buat index, contoh :

  CREATE INDEX nik_idx ON penduduk(nik);
- Explain kembali query yang sama, perhatikan apakah ada perbedaan antara query sebelum dan sesudah di index

Lembaga Pendidikan & Pengembangan Profesi Terpadu
LP3T NURUL FIKRI
IT Training - IT Education
LP3T-NF

THANK YOU

OUR QUALITY
YOUR TRUST

www.nurulfikri.com

PT. Nurul Fikri Cipta Inovasi
Jl. Margonda Raya No. 522
Depok Jawa Barat
Telp / Fax : (021) 7874224 / 7874225
Website: www.nurulfikri.com