

BAB VII

SUB QUERY DAN INDEKS

7.1 Bahasan dan Sasaran

7.1.1 Bahasan

- Pada bab kali ini akan membahas tentang sub query dan indeks
- Selain hal itu akan dibahas juga mengenai cluster, kolom unik dan trim

7.1.2 Sasaran

1. Mahasiswa memahami dan menggunakan sub query dan indeks dalam pengelolaan database.
2. Mahasiswa memahami cara penggunaan cluster, kolom unik dan trim.

7.2 Materi

7.2.1 SELECT INTO STATEMENT

Query select into statement merupakan query SQL yang digunakan untuk mengopi informasi dari tabel ke tabel yang lain tanpa membuat tabel sebelumnya.

Berikut struktur query penggunaan select into statement untuk mengopi semua data dari tabel1 ke tabel yang baru :

```
SELECT * INTO newtable FROM table1;
```

Berikut struktur query penggunaan select into statement untuk mengopi data berdasarkan kolom tertentu dari tabel1 ke tabel yang baru :

```
SELECT column_name(s) INTO newtable [IN externaldb] FROM table1;
```

Contoh :

- mengopi semua data dari tabel mahasiswa ke tabel baru yang dinamakan 'identitas 1':

Select * into identitas1 from mahasiswa;

- mengopi data pada kolom nim dan nama dari tabel mahasiswa ke tabel baru yang dinamakan 'identitas 2':

Select nim_mhs, nama_mhs into identitas2 from mahasiswa;

7.2.2 SUB QUERY

Subquery atau query Nested merupakan bentuk query yang terdapat dalam query yang lain. *Subquery* dapat ditempatkan dalam klausa where, having, from bersama

dengan operator perbandingan seperti = untuk baris tunggal dan untuk baris berganda menggunakan in, not in atau <>, < any, >, >=,<=. Penggunaan sub query dapat diterapkan pada pernyataan SELECT, UPDATE, DELETE, dan INSERT. Bentuk penggunaannya sebagai berikut :

Select nama_kolom from nama_tabel where nama_kolom operator (subquery);

Berikut contoh dari subquery menggunakan data pegawai :

Id_peg	Nama_peg	Alamat_peg	Telp_peg	Jabatan_peg	Gaji_peg
1	Hendro	Solo	081223300	Teknisi	900000
2	Tika	Semarang	0897735357	Sekretaris	2000000
3	Wijaya	Jogjakarta	0865433225	Kepala	3000000
4	Dodi	Banyuwangi	076544677	Teknisi	1000000

- Mencari nama pegawai yang memiliki jabatan yang sama dengan pak hendro bisa menggunakan query sebagai berikut :

Select nama_peg,jabatan_peg from pegawai where jabatan_peg in (select jabatan_peg from pegawai where nama_peg='Hendro');

Hasil :

NAMA	JABATAN
Hendro	Teknisi
Dodi	Teknisi

- Mencari nama pegawai yang gajinya lebih besar dari pegawai dengan nama Dodi bisa menggunakan query sebagai berikut :

select nama_peg,gaji_peg from pegawai where gaji_peg > any (select gaji_peg from pegawai where nama_peg ='Dodi');

Hasil :

NAMA	Gaji
Tika	2000000
Wijaya	3000000

- Mencari nama pegawai yang gajinya lebih besar dari 950000 dan jabatannya bukan seperti jabatan pak hendro bisa menggunakan query sebagai berikut :

```
select nama_peg, jabatan_peg, gaji_peg from pegawai where gaji_peg >= 950000
and jabatan_peg <> (select jabatan_peg from pegawai where
nama_peg='Hendro');
```

Hasil :

NAMA	JABATAN	Gaji
Tika	Sekretaris	2000000
Wijaya	Kepala	3000000

7.2.3 INDEKS

Indeks disini berguna dalam suatu pencarian nilai atau data dalam database. Dalam suatu kasus ketika mengakses sebuah tabel biasanya DBMS akan membaca seluruh tabel baris perbaris hingga selesai. Ketika baris sangat banyak dan hasil dari query hanya sedikit, maka hal ini 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 buku maka kita cukup dengan membuka indeks, sehingga akan cepat dalam pencarian kata tersebut. PostgreSQL tidak bisa membuat indeks dengan otomatis, sehingga *user* dapat membuat indeks tersebut untuk sering kali digunakan kolom, biasanya dalam *clause* WHERE. Berikut struktur SQL :

```
CREATE INDEX nama_index ON nama_tabel (nama kolom);
```

Contoh :

- Pada tabel pegawai kita berikan index pada kolom gaji untuk query sebagai berikut :

```
Create index gaji_index on pegawai(gaji_peg);
```

Hasil :

Column	Table "public.pegawai" Type	Modifiers
id	integer	not null
nama	character varying(20)	
alamat	text	
no_telp	character varying(20)	
jabatan	character varying(15)	
gaji	integer	
Indexes:		
"pegawai_pkey" PRIMARY KEY, btree (id)		
"gaji_index" btree (gaji)		

indeks sebaiknya jangan digunakan pada tabel atau kolom yang sangat jarang atau tidak pernah diakses. Selain untuk perintah SELECT Indeks juga bermanfaat untuk UPDATE dan DELETE yang menggunakan kondisi pencarian. Sedangkan *Unique index* mirip dengan indeks tetapi lebih digunakan untuk mencegah duplikasi nilai yang terdapat dalam tabel. Jadi dengan adanya *unique index* berarti pembaca tidak dapat meng-*insert* nilai yang sama dalam sebuah tabel. Berikut struktur SQL nya :

CREATE UNIQUE INDEX nama_index ON nama_tabel (nama_kolom);

Untuk menghapus index berikut strukturnya :

DROP INDEX Nama_index;

Contoh :

- Pada tabel pegawai kita berikan index yang bersifat unik pada kolom nama, untuk query sebagai berikut :

Create unique index unama_index on pegawai(nama_peg);

Hasil :

Column	Table "public.pegawai" Type	Modifiers
id	integer	not null
nama	character varying(20)	
alamat	text	
no_telp	character varying(20)	
jabatan	character varying(15)	
gaji	integer	
Indexes:		
"pegawai_pkey" PRIMARY KEY, btree (id)		
"unama_index" UNIQUE, btree (nama)		
"gaji_index" btree (gaji)		

Apabila kita memasukkan dengan nama yang sudah terdapat pada data terdahulu akan terdapat error.

7.2.4 KOLOM UNIK

Unique berfungsi untuk menjaga agar tidak terjadinya duplikasi nilai (kesamaan data) dalam sebuah kolom, hal ini dapat ditangani dengan membuat sebuah indeks unik atau fungsi unik sendiri pada kolom yang dimaksud. Unique ini sering digunakan dalam pembuatan bukan primary key namun membutuhkan cek dupikasi agar tidak ada yang sama, karena dalam primary key sudah otomatis mempunyai sifat unik. Berikut Struktur SQL saat pembuatan tabel baru :

CREATE TABLE nama_tabel (nama_kolom tipe_data unique);

Ketika tabel sudah ada kita bisa menggunakan cara seperti pada BAB. 2 berikut struktur SQL nya :

ALTER TABLE nama_tabel ADD UNIQUE (nama_kolom);

Untuk menghapus unique berikut caranya :

ALTER TABLE nama_table DROP CONSTRAINT NAMA_CONSTRAIN

Contoh :

```
postgres=# \d pegawai
Table "public.pegawai"
  Column      |      Type      | Modifiers
-----+-----+-----
 id_peg       | integer        | not null
 nama_peg     | character varying(20)
 alamat_peg   | text
 telp_peg     | character varying(15)
 jabatan_peg  | character varying(15)
 gaji_peg     | integer
Indexes:
    "pegawai_pkey" PRIMARY KEY, btree (id_peg)
    "gaji_index" btree (gaji_peg)

postgres=# alter table pegawai add unique (nama_peg);
ALTER TABLE
postgres=# \d pegawai
Table "public.pegawai"
  Column      |      Type      | Modifiers
-----+-----+-----
 id_peg       | integer        | not null
 nama_peg     | character varying(20)
 alamat_peg   | text
 telp_peg     | character varying(15)
 jabatan_peg  | character varying(15)
 gaji_peg     | integer
Indexes:
    "pegawai_pkey" PRIMARY KEY, btree (id_peg)
    "pegawai_nama_peg_key" UNIQUE CONSTRAINT, btree (nama_peg)
    "gaji_index" btree (gaji_peg)

postgres=# alter table pegawai drop constraint pegawai_nama_peg_key;
ALTER TABLE
postgres=# \d pegawai
Table "public.pegawai"
  Column      |      Type      | Modifiers
-----+-----+-----
 id_peg       | integer        | not null
 nama_peg     | character varying(20)
 alamat_peg   | text
 telp_peg     | character varying(15)
 jabatan_peg  | character varying(15)
 gaji_peg     | integer
Indexes:
    "pegawai_pkey" PRIMARY KEY, btree (id_peg)
    "gaji_index" btree (gaji_peg)
```

7.2.5 Check

Check berfungsi untuk melakukan pembatasan nilai masukan dalam sebuah kolom, sebagai contoh misalkan kita ingin agar kolom *gender* yang terdiri dari satu karakter hanya memiliki dua pilihan karakter yaitu **M** (*male*) atau **F** (*Female*) ini dapat kita seting dengan menggunakan CHECK. Dengan menggunakan CHECK maka sebuah kolom hanya bisa diisi dengan data yang memenuhi kriteria dalam CHECK. Berikut query contoh penggunaan check :

```
db_contoh=> CREATE TABLE pelanggan (
db_contoh(> nama varchar(35),
```

```

db_contoh(> kode_area CHAR(10) CHECK
            (length(trim(kode_area)) = 2) ,
db_contoh(> umur INTEGER CHECK (umur >= 0) ,
db_contoh(> gender CHAR(1) CHECK (gender IN ('L', 'P')) ,
db_contoh(> ttl DATE CHECK (ttl BETWEEN '1998-01-01' AND
            CURRENT_DATE) ,
db_contoh(> CHECK (upper(trim(nama)) != 'nita' OR
db_contoh(> upper(trim(nama)) != 'jeki')
db_contoh(> ) ;
CREATE

```

7.2.6 Penggunaan TRIM

Suatu ketika pasti akan memiliki data yang di dalamnya terdapat spasi kosong yang tidak diperlukan, misalnya spasi ganda. Jika ada masalah seperti ini, kita dapat membersihkan spasi-spasi kosong yang tidak diperlukan menggunakan fungsi TRIM, RTRIM, dan LTRIM. Ketiga fungsi ini memiliki bentuk penggunaan sebagai berikut :

- RTRIM : digunakan untuk membersihkan spasi kosong yang ada di bagian kanan (Right) String.
- LTRIM : digunakan untuk membersihkan spasi kosong yang ada di bagian kiri (Left) String.
- TRIM : digunakan untuk membersihkan spasi kosong yang ada di bagian kiri, kanan, maupun tengah String

Berikut Struktur SQL nya :

Select trim(nama_kolom) from nama_tabel;

Dalam penggunaannya, fungsi TRIM memiliki tiga opsi. Ketiga opsi ini dapat digunakan untuk menentukan karakter apa yang akan dihapus dari suatu String. Jadi, fungsi TRIM juga dapat menghilangkan karakter tertentu (bukan spasi kosong saja) dari suatu string. Opsinya sebagai berikut :

- LEADING : merupakan opsi untuk menghilangkan karakter terpilih yang ada di sebelah kiri. Parameter Leading diartikan sebagai sufik dari karakter yang ada.
- TRAILING : merupakan opsi untuk menghilangkan karakter terpilih yang ada di sebelah kanan String. Parameter Trailing diartikan sebagai sufik dari karakter yang ada.
- BOTH : merupakan opsi yang dapat menangani parameter Leading maupun Trailing.

Berikut Struktur SQL nya :

Select trim(LEADING 'karakter, misal : -' from nama_kolom) from nama_tabel;

Tugas Praktikum

Tugas praktikum kali ini masih menggunakan tabel pada praktikum sebelumnya :

1. Tampilkan nama fakultas dan jumlah mahasiswa yang mempunyai ketentuan nama fakultas yang dimunculkan dengan jumlah mahasiswanya terkecil!
2. Tampilkan nama mahasiswa, nama fakultas, alamat dengan syarat nama fakultas sama dengan edi dan alamatnya tidak sama dengan luki!
3. Buatlah index di tabel mahasiswa(alamat). Kemudian buat lagi index yang bersifat unik pada tabel fakultas(fak_nama) kemudian amati perbedaannya ketika memasukkan data yang sama!
4. Buat kolom nama di mahasiswa menjadi unik dan inputkan 2 data yang sama. Kemudian amati perbedaannya !
5. Pindahkan data dari tabel mahasiswa, fakultas ambil kolom nim, nama mahasiswa, alamat, nama fakultas ke tabel baru yang dinamai 'tabel identitas'.
6. Buatlah contoh penggunaan check pada sub bab pembahasan CHECK. Kemudian masukkan beberapa data baik yang sesuai dengan criteria check maupun yang bukan dan amati perbedaannya.
7. Inputkan data di tabel mahasiswa dimana pada kolom nama sebelum inputkan karakter dahulukan dengan spasi dan di akhiri dengan tanda "+" seperti berikut : " andi cahyono++++". kemudian munculkan seluruh data dan hilangkan spasi didepan!
8. Munculkan data mahasiswa dengan hilangkan karakter "+" di akhir data dan karakter "a" di awal kata pada kolom nama!

Tugas Rumah

1. Buat laporan praktikum menggunakan DBMS mysql untuk mengerjakan tugas praktikum 1-8 dan letakkan di blog.