**BAB IV**

**PENGENALAN OPERATOR DASAR**

4.1 Bahasan dan Sasaran

4.1.1 Bahasan

- Operator dasar dalam standar query sql

- Operator dasar yang berhubungan dengan manipulasi data.

4.1.1 Sasaran

1. Mahasiswa memahami operator dasar dalam standar query sql.

2. Mahasiswa memahami cara penggunaan operator dasar dalam hal manipulasi data.

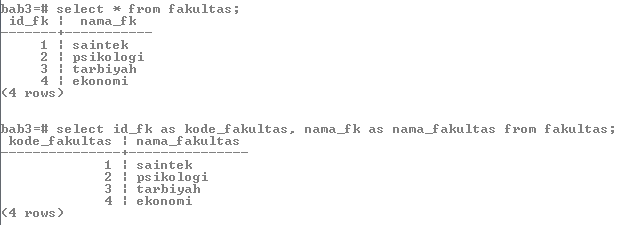
4.2 Materi

4.2.1 AS

AS biasa digunakan untuk menampilkan label kolom dengan nama lain sehingga yang akan muncul dalam hasil query bukan nama asli kolom, tetapi nama yang mungkin lebih sesuai dan mudah dimengerti. AS digunakan setelah nama kolom yang akan diganti yang kemudian diikuti dengan nama penggantinya. Berikut struktur querynya :

**select namakolom as namakolompengganti from namatabel;**

**Contoh :**

****

4.2.2 AND dan OR

Pada bagian sebelumnya kita menggunakan anak kalimat atau sintaks WHERE hanya pada konteks yang sederhana. Berikut ini kita akan mencoba menggunakan *WHERE* untuk konteks yang lebih kompleks lagi, anak kalimat *where* yang kompleks akan bekerja dengan baik dengan menggunakan kata *AND* dan *OR*. SQL menggunakan standar logika *boolean three-valued* seperti pada tabel berikut ;

|  |  |  |  |
| --- | --- | --- | --- |
| Kondisi A | Kondisi B | A AND B | A OR B |
| True | True | True | True |
| True | False | False | True |
| True | Null | Null | True |
| False | False | False | False |
| False | Null | False | Null |
| Null | Null | Null | Null |

|  |  |
| --- | --- |
| A | NOT A |
| True | False |
| False | True |
| Null | Null |

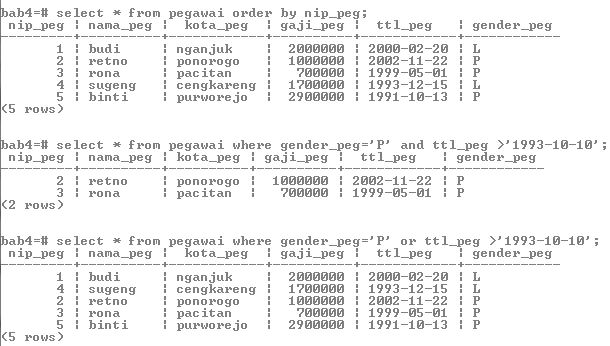
Berikut struktur SQL untuk penampilan data :

**select \* from namatabel where namakolom = ‘pencarian 1’ and namakolom = ‘pencarian 2’;**

Atau

**select \* from namatabel where namakolom = ‘pencarian 1’ or namakolom = ‘pencarian 2’;**

Contoh :



4.2.3 BETWEEN

*BETWEEN* digunakan untuk menentukan lebar nilai dari nilai terendah dan nilai tertinggi. Pada *BETWEEN* menggunakan operator pembanding seperti pada tabel berikut;

|  |  |
| --- | --- |
| Operator | Penjelasan |
| < | Kurang dari |
| > | Lebih dari |
| <= | Kurang dari atau sama dengan |
| >= | Lebih dari atau sama dengan |
| = | Sama dengan |
| < > atau != | Tidak sama dengan |

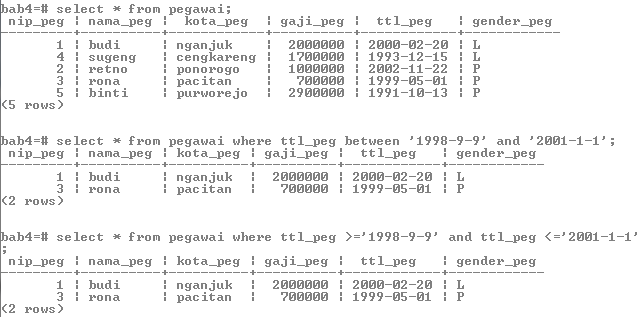
Logika operator BETWEEN sebagai berikut :

* Nilai a dalam formula “a **BETWEEN x AND y**” indentik dengan “**a >= x AND a <= y**“
* “**a NOT BETWEEN x AND y** “ identik dengan “**a < x OR a > y**”

Berikut Struktur yang digunakan :

**select \* from nama\_tabel where nama\_kolom between ‘nilai\_awal’ and ‘nilai\_akhir’;**

Contoh : Memunculkan data pegawai dimana tanggal lahirnya antara tanggal 9-9-1999 sampai 1-1-2001.

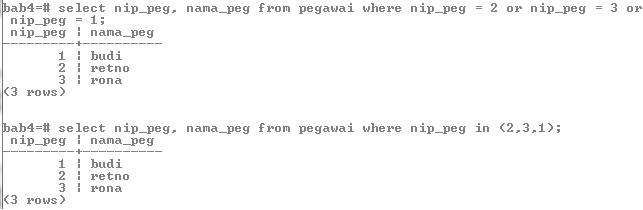


4.2.4 IN dan NOT IN

Operator IN berguna melakukan pencocokan dengan salah satu yang ada pada suatu daftar nilai. Berikut Struktur yang digunakan

**Select \* from nama\_tabel where nama\_kolom in (kata\_kunci1, kata\_kunci2, kata\_kunci13, kata\_kunci14);**

Contoh : menampilkan data pegawai yang memiliki ip 1,2, dan 3



4.2.5 LIKE

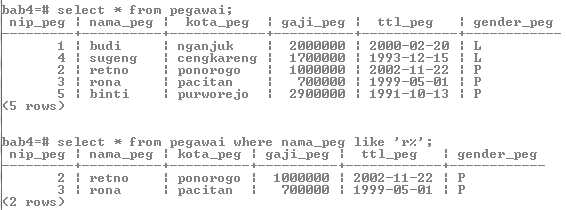
LIKE digunakan ketika kita ingin mencari sebuah data yang hanya diwakili oleh salah satu atau lebih hurufnya saja. Misalkan kita ingin mencari nama yang huruf awalnya *R*, maka kita harus menggunakan *LIKE.* Berikut adalah tabel contoh kasus penggunaan LIKE :

|  |  |
| --- | --- |
| Kasus | Operator |
| Diawali dengan huruf A | LIKE ‘A%’ |
| Diakhiri dengan huruf A | LIKE ‘%D’ |
| Huruf A pada posisi ke dua | LIKE ‘\_A%’ |
| Diawali dengan huruf A dan mengandung huruf I | LIKE ‘A%I%’ |
| Tidak diawali dengan huruf A | NOT LIKE ‘A%’ |

Berikut struktur SQL untuk pencarian data menggunakan LIKE :

**select \* from nama\_tabel where nama\_kolom like ‘operator’;**

Contoh : Mencari data pegawai yang mempunyai nama dengan huruf awalan r.

****

4.2.6 REGEXP

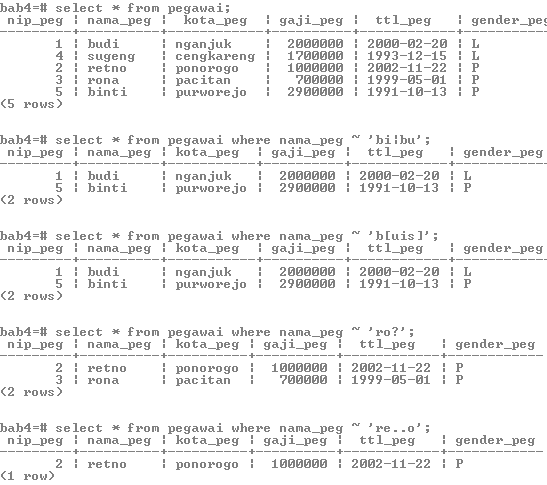
Regular Expression atau Regex, merupakan sebuah teknik atau cara untuk mencari persamaan - presamaan string atau data dan memanipulasinya. Biasanya lebih sering digunakan untuk string. Didalam PostgreSQL disimbolkan dengan “~”. Berikut tabel daftar simbol pada reguler expression :

|  |  |
| --- | --- |
| Pola | Penjelasan |
| [ ] | Ekspresi kurung. cocok dengan satu karakter yang berada dalam kurung, misal: pattern "a[bcd]i" cocok dengan string "abi", "aci", dan "adi". Penggunaan range huruf dalam kurung diperbolehkan, misal : pattern "[a-z]" cocok dengan salah satu karakter diantara string "a" sampai "z". pattern [0-9] cocok dengan salah satu angka. jika ingin mencari karakter "-" juga, karakter tersebut harus diletakkan di depan atau di belakang kelompok, misal: "[abc-]". |
| [^ ] | Cocok dengan sebuah karakter yang tidak ada dalam kurung, berlawanan dengan yang diatas. misal: pattern "[^abc]" cocok dengan satu karakter apa saja kecuali "a", "b", "c". |
| ? | Cocok dengan nol atau satu karakter sebelumnya. misal: pattern "died?" cocok dengan string "die" dan "died". |
| + | Cocok dengan satu atau lebih karakter sebelumnya. misal: "yu+k" cocok dengan "yuk", "yuuk", "yuuuk", dan seterusnya. |
| \* | Cocok dengan nol atau lebih karakter sebelumnya. misal: pattern "hu\*p" cocok dengan string "hp", "hup", "huup" dan seterusnya. |
| {x} | Cocok dengan karakter sebelumnya sejumlah x karakter. misal: pattern "[0-9]{3}" cocok dengan bilangan berapa saja yang berukuran 3 digit. |
| {x,y} | Cocok dengan karakter sebelumnya sejumlah x hingga y karakter. misal: pattern "[a-z]{3,5}" cocok dengan semua susunan huruf kecil yang terdiri dari 3 sampai 5 huruf. |
| ! | Jika diletakkan di depan pattern, maka berarti "bukan". misal pattern "!a.u" cocok dengan string apa saja kecuali string "alu", "anu", "abu", "asu", "aiu", dan seterusnya . |
| ^ | Jika diletakkan di depan pattern, akan cocok dengan awal sebuah string. |
| ( ) | Gruping. digunakan untuk mengelompokkan karakter - karakter menjadi single unit. string yang cocok dalan pattern yang berada dalam tanda kurung dapat digunakan pada operasi berikutnya. semacam variable. |
| $ | Jika diletakkan di belakang pattern, akan cocok dengan akhir sebuah string. |

Berikut struktur SQL untuk pencarian data menggunakan REGEXP :

**select \* from nama\_tabel where nama\_kolom ~ ‘operator\_regexp dan patern’;**

Contoh :

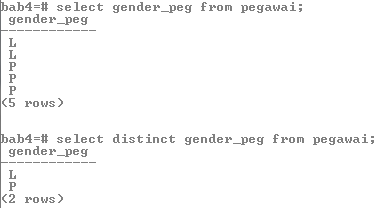


4.2.7 DISTINCT

DISTINCT sering kali diperlukan untuk mengembalikan hasil dari sebuah *query* dengan tidak terdapat duplikasi, artinya pada hasil outputnya tidak terjadi kesamaan data meskipun pada data sesungguhnya sangat mungkin banyak duplikasi.. Hal ini juga berlaku jika data yang ada dalam bentuk angka. Berikut struktur perintah dasar SQL :

**select distinct nama\_kolom from nama\_tabel;**

Contoh :



4.2.7 LIMIT

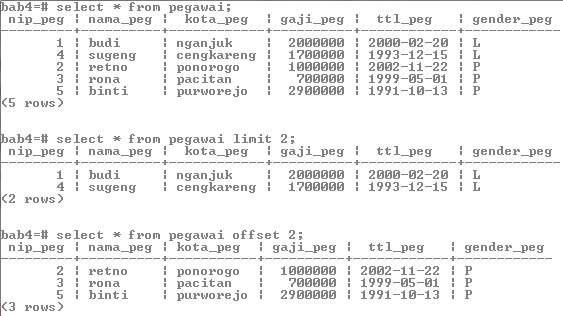
LIMIT dan OFFSET digunakan untuk membatasi jumlah output dari query berdasarkan jumlah row bukan karena kondisi seperti WHERE. Bisa dikatakan LIMIT adalah untuk menentukan jumlah baris yang akan ditampilkan yang dihitung dari baris pertama, sedangkan OFFSET digunakan untuk menghilangkan baris sesuai dengan jumlah yang diberikan pada OFFSET. Berikut struktur perintah dasar SQL :

**select \* from nama\_tabel limit 2;**

Atau

**select \* from nama\_tabel offset 2;**

Contoh :



4.2.8 UNION, EXCEPT dan INTERSECT

Hasil dari dua buah query dapat dikombinasikan dengan menggunakan UNION, EXCEPT atau INTERSECT. UNION digunakan untuk menggabungkan hasil dua buah query menjadi satu kolom. Berikut struktur query untuk UNION :

**select nama\_kolom from nama\_tabel union select nama\_kolom from nama\_tabel;**

atau

**select nama\_kolom from nama\_tabel union all select nama\_kolom from nama\_tabel;**

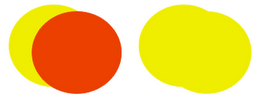
EXCEPT digunakan untuk menampilkan hanya query pertama saja, sedangkan hasil query kedua dan yang sama dengan hasil query kedua tidak akan ditampilkan. Berikut struktur query penggunaan EXCEPT :

**select nama\_kolom from nama\_tabel except select nama\_kolom from nama\_tabel;**

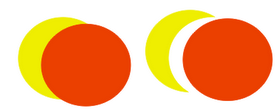
Perintah INTERSECT hanya akan menampilkan seluruh isi dari data yangmemiliki kesaman diantara hasil kedua query tersebut. Berikut struktur query penggunaan INTERSECT :

**select nama\_kolom from nama\_tabel except select nama\_kolom from nama\_tabel;**

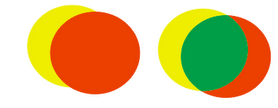
Dari uraian materi tentang UNION, EXCEPT dan INTERSECT dapat digambarkan sebagai berikut :



Gambar diatas menyatakan UNION yaitu gabungan dari kedua buah object.



Sedangan gambar kedua menyatakan EXCEPT dimana object yang diambil hanya yang kuning. Sehingga object yang merah atau object kuning yang sama dengan object merah tidak diambil.



Gambar ke tiga menyatakan INTERSECT yaitu mengambil object yang berwarna hijau atau object memiliki kandungan dari object kuning dan merah.

Contoh :

