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 :

```
bab3=# select * from fakultas;
 id_fk |  nama_fk
-------+-----------
     1 | saintek
     2 | psikologi
     3 | tarbiyah
     4 | ekonomi
(4 rows)


bab3=# select id_fk as kode_fakultas, nama_fk as nama_fakultas from fakultas;
 kode_fakultas | nama_fakultas
---------------+---------------
             1 | saintek
             2 | psikologi
             3 | tarbiyah
             4 | ekonomi
(4 rows)
```

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 :

```
bab4=# select * from pegawai order by nip_peg;
 nip_peg | nama_peg |  kota_peg   | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-------------+----------+------------+------------
       1 | budi     | nganjuk     |  2000000 | 2000-02-20 | L
       2 | retno    | ponorogo    |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan     |   700000 | 1999-05-01 | P
       4 | sugeng   | cengkareng  |  1700000 | 1993-12-15 | L
       5 | binti    | purworejo   |  2900000 | 1991-10-13 | P
(5 rows)


bab4=# select * from pegawai where gender_peg='P' and ttl_peg >'1993-10-10';
 nip_peg | nama_peg | kota_peg | gaji_peg |  ttl_peg   | gender_peg
---------+----------+----------+----------+------------+------------
       2 | retno    | ponorogo |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan  |   700000 | 1999-05-01 | P
(2 rows)


bab4=# select * from pegawai where gender_peg='P' or ttl_peg >'1993-10-10';
 nip_peg | nama_peg |  kota_peg   | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-------------+----------+------------+------------
       1 | budi     | nganjuk     |  2000000 | 2000-02-20 | L
       4 | sugeng   | cengkareng  |  1700000 | 1993-12-15 | L
       2 | retno    | ponorogo    |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan     |   700000 | 1999-05-01 | P
       5 | binti    | purworejo   |  2900000 | 1991-10-13 | P
(5 rows)
```

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.

```
bab4=# select * from pegawai;
 nip_peg | nama_peg |  kota_peg  | gaji_peg |  ttl_peg   | gender_peg
---------+----------+------------+----------+------------+------------
       1 | budi     | nganjuk    |  2000000 | 2000-02-20 | L
       4 | sugeng   | cengkareng |  1700000 | 1993-12-15 | L
       2 | retno    | ponorogo   |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan    |   700000 | 1999-05-01 | P
       5 | binti    | purworejo  |  2900000 | 1991-10-13 | P
(5 rows)


bab4=# select * from pegawai where ttl_peg between '1998-9-9' and '2001-1-1';
 nip_peg | nama_peg | kota_peg | gaji_peg |  ttl_peg   | gender_peg
---------+----------+----------+----------+------------+------------
       1 | budi     | nganjuk  |  2000000 | 2000-02-20 | L
       3 | rona     | pacitan  |   700000 | 1999-05-01 | P
(2 rows)


bab4=# select * from pegawai where ttl_peg >='1998-9-9' and ttl_peg <='2001-1-1'
;
 nip_peg | nama_peg | kota_peg | gaji_peg |  ttl_peg   | gender_peg
---------+----------+----------+----------+------------+------------
       1 | budi     | nganjuk  |  2000000 | 2000-02-20 | L
       3 | rona     | pacitan  |   700000 | 1999-05-01 | P
(2 rows)
```

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

```
bab4=# select nip_peg, nama_peg from pegawai where nip_peg = 2 or nip_peg = 3 or
 nip_peg = 1;
 nip_peg | nama_peg
---------+----------
       1 | budi
       2 | retno
       3 | rona
(3 rows)


bab4=# select nip_peg, nama_peg from pegawai where nip_peg in (2,3,1);
 nip_peg | nama_peg
---------+----------
       1 | budi
       2 | retno
       3 | rona
(3 rows)
```

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.

```
bab4=# select * from pegawai;
 nip_peg | nama_peg |  kota_peg  | gaji_peg |  ttl_peg   | gender_peg
---------+----------+------------+----------+------------+------------
       1 | budi     | nganjuk    |  2000000 | 2000-02-20 | L
       4 | sugeng   | cengkareng |  1700000 | 1993-12-15 | L
       2 | retno    | ponorogo   |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan    |   700000 | 1999-05-01 | P
       5 | binti    | purworejo  |  2900000 | 1991-10-13 | P
(5 rows)


bab4=# select * from pegawai where nama_peg like 'r%';
 nip_peg | nama_peg | kota_peg | gaji_peg |  ttl_peg   | gender_peg
---------+----------+----------+----------+------------+------------
       2 | retno    | ponorogo |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan  |   700000 | 1999-05-01 | P
(2 rows)
```

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 :

```
bab4=# select * from pegawai;
 nip_peg | nama_peg |  kota_peg   | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-------------+----------+------------+------------
       1 | budi     | nganjuk     |  2000000 | 2000-02-20 | L
       4 | sugeng   | cengkareng  |  1700000 | 1993-12-15 | L
       2 | retno    | ponorogo    |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan     |   700000 | 1999-05-01 | P
       5 | binti    | purworejo   |  2900000 | 1991-10-13 | P
(5 rows)


bab4=# select * from pegawai where nama_peg ~ 'bi|bu';
 nip_peg | nama_peg | kota_peg  | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-----------+----------+------------+------------
       1 | budi     | nganjuk   |  2000000 | 2000-02-20 | L
       5 | binti    | purworejo |  2900000 | 1991-10-13 | P
(2 rows)


bab4=# select * from pegawai where nama_peg ~ 'b[uis]';
 nip_peg | nama_peg | kota_peg  | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-----------+----------+------------+------------
       1 | budi     | nganjuk   |  2000000 | 2000-02-20 | L
       5 | binti    | purworejo |  2900000 | 1991-10-13 | P
(2 rows)


bab4=# select * from pegawai where nama_peg ~ 'ro?';
 nip_peg | nama_peg | kota_peg | gaji_peg |  ttl_peg   | gender_peg
---------+----------+----------+----------+------------+------------
       2 | retno    | ponorogo |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan  |   700000 | 1999-05-01 | P
(2 rows)


bab4=# select * from pegawai where nama_peg ~ 're..o';
 nip_peg | nama_peg | kota_peg | gaji_peg |  ttl_peg   | gender_peg
---------+----------+----------+----------+------------+------------
       2 | retno    | ponorogo |  1000000 | 2002-11-22 | P
(1 row)
```

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 :

```
bab4=# select gender_peg from pegawai;
 gender_peg
------------
 L
 L
 P
 P
 P
(5 rows)


bab4=# select distinct gender_peg from pegawai;
 gender_peg
------------
 L
 P
(2 rows)
```

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 :

```
bab4=# select * from pegawai;
 nip_peg | nama_peg |  kota_peg   | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-------------+----------+------------+------------
       1 | budi     | nganjuk     |  2000000 | 2000-02-20 | L
       4 | sugeng   | cengkareng  |  1700000 | 1993-12-15 | L
       2 | retno    | ponorogo    |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan     |   700000 | 1999-05-01 | P
       5 | binti    | purworejo   |  2900000 | 1991-10-13 | P
(5 rows)


bab4=# select * from pegawai limit 2;
 nip_peg | nama_peg |  kota_peg   | gaji_peg |  ttl_peg   | gender_peg
---------+----------+-------------+----------+------------+------------
       1 | budi     | nganjuk     |  2000000 | 2000-02-20 | L
       4 | sugeng   | cengkareng  |  1700000 | 1993-12-15 | L
(2 rows)


bab4=# select * from pegawai offset 2;
 nip_peg | nama_peg | kota_peg   | gaji_peg |  ttl_peg   | gender_peg
---------+----------+------------+----------+------------+------------
       2 | retno    | ponorogo   |  1000000 | 2002-11-22 | P
       3 | rona     | pacitan    |   700000 | 1999-05-01 | P
       5 | binti    | purworejo  |  2900000 | 1991-10-13 | P
(3 rows)
```

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;

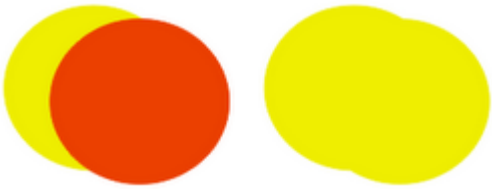
EXCEPT digunakan untuk menampilkan hanya query pertama saja, sedangkan hasil query kedua dan yang sama dengan hasil query kedua tidak akan ditampilkan. Berikut struktur query penggunaan EXCEPT :

select nama_kolom from nama_tabel except select nama_kolom from nama_tabel;

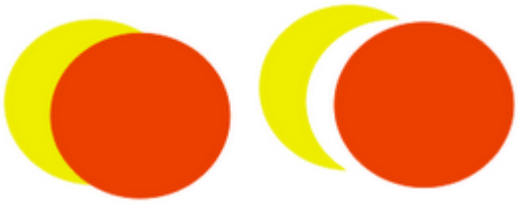
Perintah INTERSECT hanya akan menampilkan seluruh isi dari data yangmemiliki kesaman diantara hasil kedua query tersebut. Berikut struktur query penggunaan INTERSECT :

select nama_kolom from nama_tabel except select nama_kolom from nama_tabel;

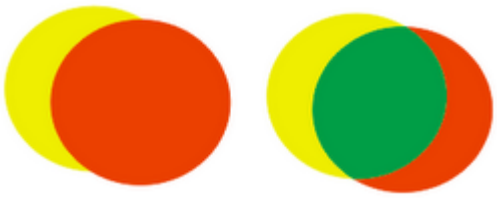
Dari uraian materi tentang UNION, EXCEPT dan INTERSECT dapat digambarkan sebagai berikut :



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 :

```
bab4=# select*from pelajaran_biologi;
 nama_pbi
----------
 genetika
 anatomi
 komputer
(3 rows)


bab4=# select*from pelajaran_informatika;
  nama_pbi
-------------
 komputer
 pemrograman
 logika
(3 rows)


bab4=# select*from pelajaran_biologi union select * from pelajaran_informatika;
  nama_pbi
-------------
 anatomi
 logika
 pemrograman
 komputer
 genetika
(5 rows)


bab4=# select*from pelajaran_biologi except select * from pelajaran_informatika;

 nama_pbi
----------
 anatomi
 genetika
(2 rows)


bab4=# select*from pelajaran_biologi intersect select * from pelajaran_informati
ka;
 nama_pbi
----------
 komputer
(1 row)
```

Tugas Praktikum

Persiapan praktek : Gunakan tabel mahasiswa dan fakultas pada pertemuan 3. Tambahkan kolom gender di tabel mahasiswa kemudian update data mahasiswa. Insert data mahasiswa hingga 7 baris.

1. Berdasarkan tabel sebelumnya yaitu bab 3, tampilkan data mahasiswa dengan nama dari kolomnya berubah menjadi nomor_identitas, nama, alamat,nama_fakultas tanpa harus merubah nama kolom secara permanen.
2. Tampilkan nama mahasiswa fakultas saintek yang berjenis kelamin laki-laki. Kemudian nama mahasiswa fakultas saintek atau berjenis kelamin laki-laki
3. Suatu tabel mahasiswa terdapat 7 baris data mahasiswa. Tampilkan data mahasiswa dari nomor 3 sampai dengan 5, kemudian tampilkan juga data mahasiswa yang bukan dari nomor 3 sampai dengan 5 menggunakan between dan tanpa menggunakan between.
4. Tampilkan mahasiswa dengan nomor 2,3,5 menggunakan operator IN.
5. Tampilkan semua data mahasiswa yang namanya mempunyai awalan huruf "a".
6. Tampilkan semua data mahasiswa yang namanya mempunyai akhiran "a" dan huruf ke-3 dari nama tersebut terdapat huruf "n".
7. Tampilkan nomor fakultas di tabel mahasiswa dengan nomor harus tidak ada yang sama dari baris pertama sampai akhir.
8. Tampilkan data mahasiswa 3 baris saja.
9. Tampilkan data mahasiswa dengan ciri-ciri memiliki nama awalan a kemudian kata selanjutnya boleh r atau boleh t atau boleh d. misalkan yang memenuhi kriteria ciri tersebut seperti adi, atik, arif, adam dll.

   *Boleh dengan awalan dan ciri yang lain namun karakteristiknya harus sesuai soal
10. Buatlah tabel dengan nama organ_dalam dan organ_luar sebagai berikut :
    a. Tabel organ_dalam

Nomor	Nama
1	Jantung
2	Paru-paru
3	Tenggorokan

    b. Tabel organ_luar

Nomor	Nama
1	Hidung

2	Mata
3	Tenggorokan

Tampilkan data dari kedua tabel tersebut menggunakan UNION, EXCEPT dan INTERSECT

Tugas Rumah

1. Buat laporan, Tampilkan perintah-perintah SQL menggunakan DBMS MYSQL beserta hasilnya di blog anda. Perintahnya sesuai dengan tugas praktikum 1-10.
2. Jelaskan evaluasi dari DBMS MySQL dan PostgreSQL dalam blog anda berdasarkan tugas praktikum 1-10. (evaluasi meliputi perbedaan atau kesamaan, kelebihan atau kekurangan, tambahan dari penulis)