PRAKTIKUM SISTEM BASIS DATA

Modul 11: Himpunan dan Subquery

Tujuan Praktikum:

1. Mahasiswa dapat menjelaskan berbagai macam operasi himpunan dan subquery yang bisa dilakukan antar terhadap beberapa tabel dengan komposisi kolom tertentu.
2. Mahasiswa dapat menuliskan sintaks himpunan dan subquery dengan benar.
3. Mahasiswa dapat menerapkan operasi himpunan atau subquery yang tepat untuk kebutuhan analisis data.

Alat dan Bahan:

1. Modul Praktikum
2. Komputer
3. MySQL *phpMyAdmin* create database **`[nama]_p11`**
    a) Sub bab HIMPUNAN menggunakan file **`prak_sbd_11_himpunan.sql`**
    b) Sub bab SUBQUERY menggunakan file **`prak_sbd_11_subquery_mhs.sql`** dan **`prak_sbd_11_subquery_emp.sql`**
4. Viewer LCD

Materi Dan Pembagian Waktu:

Materi dan pembagian waktu pelaksanaan praktikum meliputi:

No	Materi	Waktu
1	Pretest	15 menit
2	Operasi Himpunan	45 menit
3	Operasi Subquery	45 menit
4	Posttest	45 menit
	Total	150 menit

Materi Praktikum

A. OPERASI HIMPUNAN

Operasi himpunan adalah operasi untuk menggabungkan dua tabel atau lebih sesuai prinsip matematika. Ada tiga jenis operasi himpunan yang akan dibahas, yaitu operasi union, minus, dan intersect.

A.1. UNION

Union digunakan untuk menggabungkan hasil dari beberapa statemen SELECT menjadi satu set hasil. Nama kolom dari statemen SELECT yang pertama digunakan sebagai nama kolom dari hasil query. Pada SELECT berikutnya harus memuat kolom yang identik dengan kolom dari statemen SELECT yang pertama.

Konsep penggabungan sesuai dengan prinsip matematika berikut:

Prima = { 2, 3, 5, 7 } dan Genap = { 2, 4, 6, 8, 10 } maka Prima $\cup$ Genap = { 2, 3, 5, 7, 4, 6, 8, 10 }

Operasi union sendiri memiliki 2 macam, yaitu **UNION** dan **UNION ALL**. Bedanya, UNION ALL memperbolehkan data yang sama dari 2 tabel muncul 2 kali. Menggunakan contoh yang tadi, hasil UNION ALL adalah {**2**, 3, 5, 7, **2**, 4, 6, 8, 10}

Sintaks dari UNION adalah:

```
SELECT ... UNION [ALL] SELECT ...
```

Contoh UNION:

```
SELECT * FROM prima UNION SELECT * FROM genap
```

Contoh UNION ALL:

```
SELECT * FROM prima UNION ALL SELECT * FROM genap
```

A.2. MINUS

Minus digunakan untuk "memotong" himpunan dari hasil SELECT pertama oleh SELECT kedua. Konsep pemotongan adalah sebagai berikut:

Prima = { 2, 3, 5, 7 } dan Genap = { 2, 4, 5, 6, 7, 8, 10 } maka Prima – Genap = {3, 5, 7}

Operator minus tidak didukung oleh MySQL, namun ada di Oracle dengan sintaks:

```
SELECT ... MINUS SELECT ...
```

Alternatifnya, kita bisa menggunakan statemen berikut:

```
SELECT [kolom] FROM tabel1 LEFT JOIN tabel2 ON [penghubung antar tabel]
WHERE tabel2.id IS NULL
```

Contoh:

```
SELECT * FROM prima LEFT JOIN genap ON prima.bilangan = genap.bilangan WHERE
genap.bilangan IS NULL
```

A.3. INTERSECT

Intersect digunakan untuk menunjukkan irisan dari himpunan hasil SELECT pertama yang juga ada di hasil SELECT kedua. Konsep irisan sesuai konsep matematika berikut:

Prima = { 2, 3, 5, 7 } dan Genap = { 2, 4, 6, 8, 10 } maka Prima $\cap$ Genap = { 2 }

Operator intersect didukung oleh MariaDB MySQL, dengan sintaks:

```
SELECT ... INTERSECT SELECT ...
```

Alternatifnya, kita bisa menggunakan statemen berikut:

```
SELECT DISTINCT [kolom] FROM tabel1 INNER JOIN tabel2 USING(id)
```

Contoh:

```
SELECT DISTINCT * FROM prima INNER JOIN genap USING(bilangan)
```

Atau:

```
SELECT * FROM prima JOIN genap ON prima.bilangan = genap.bilangan
```

Atau menggunakan subquery:

```
SELECT * FROM prima WHERE EXISTS(SELECT * FROM genap WHERE prima.bilangan = genap.bilangan)
```

Latihan A

1. Impor file prak_sbd_11_himp.sql di sini. Kemudian tampilkan seluruh mahasiswa baik yang PJJ maupun yang reguler. Data kembar cukup ditampilkan satu kali saja.
2. Tampilkan nama mahasiswa baik yang PJJ maupun yang reguler. Urutkan berdasarkan nama namun tidak menggunakan klausa ORDER BY.

nama
Amadeus
Angela
Cahyono
Erika
Joshua
Kevin
Michael
Patricia
Amadeus
Elisabeth
Erika
Joshua
Lucas

3. Tampilkan mahasiswa hanya mahasiswa yang merangkap PJJ dan reguler. Mahasiswa yang PJJ saja atau reguler saja tidak ditampilkan.
4. Tampilkan mahasiswa PJJ yang tidak merangkap reguler.
5. Tampilkan mahasiswa reguler yang tidak merangkap PJJ.

B. SUBQUERY

Subquery adalah statemen SELECT di dalam statemen lainnya. Berikut ini adalah contoh dari subquery:

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

Pada contoh di atas, **`SELECT * FROM t1 WHERE column1 =`** disebut sebagai *outer query* dan **`(SELECT column1 FROM t2)`** disebut sebagai *inner query* atau *subquery.* Outer query bisa berupa statemen SELECT, INSERT, UPDATE, DELETE, SET atau DO.

Kita bisa mengatakan bahwa subquery *nested/bersarang* di dalam outer query, dan sangat dimungkinkan untuk menyarangkan subquery lagi di dalam subquery, sampai batas kedalaman yang wajar. Subquery harus selalu berada di dalam tanda kurung (parentheses).

Keuntungan menggunakan subquery adalah:

- Menyediakan cara alternatif yang relatif lebih mudah untuk mendapatkan hasil yang diinginkan tanpa menggunakan join atau union yang kompleks.
- Membuat struktur penulisan query lebih mudah dibaca dan dibayangkan dibandingkan menggunakan join atau union.

Aturan Subquery:

- Klausa ORDER BY tidak boleh digunakan di subquery, namun ORDER BY dapat digunakan, di outer query (query induk).
- Ketika subquery diletakkan di klausa SELECT, maka hasil subquery semestinya adalah nama kolom. Kecuali untuk subquery-subquery menggunakan kata kunci EXIST.
- Ketika subquery diletakkan di klausa FROM, maka hasil subquery semestinya berupa tabel.
- Ketika subquery diletakkan di klausa WHERE, maka hasil subquery semestinya selaras melengkapi kondisi yang diberikan.
- Saat subquery adalah salah satu dari dua operan yang dilibatkan di pembandingan, subquery harus muncul di sisi kanan pembandingan.

Subquery bisa menghasilkan satu nilai skalar tunggal, satu kolom, satu baris, atau satu tabel. Outer query menggunakan operator yang berbeda-beda tergantung pada hasil subquery.

Tabel 1. Operator Pembanding

Operator	Deskripsi	Row yang dihasilkan Subquery
<> atau !=	Tidak sama dengan	Single row
> / <	Lebih besar/lebih kecil dari	
>= / <=	Lebih besar/lebih kecil atau sama dengan	
ANY atau SOME, IN	Membandingkan sebuah nilai dengan SETIAP nilai dari hasil subquery. Penggunaannya harus diawali dengan operator =, !=, >, <, >=, atau <=. Query akan mengembalikan nilai FALSE jika tidak ada row yang dihasilkan. Operator "= ANY" ekuivalen dengan IN.	Multiple rows Multiple columns
ALL	Membandingkan sebuah nilai dengan SEMUA nilai dari hasil subquery. Penggunaannya harus diawali dengan operator =, !=, >, <, >=, atau <=. Query akan mengembalikan nilai TRUE jika tidak ada row yang dihasilkan.	Multiple rows Multiple columns
EXISTS	Operator logikal jika digunakan untuk melihat apakah nilai yang dihasilkan query utama BERADA dalam subquery. TRUE, jika mengembalikan row.	
NOT EXISTS	Operator logikal jika digunakan untuk melihat apakah nilai yang dihasilkan query utama TIDAK BERADA dalam subquery. TRUE, jika tidak mengembalikan row.	

Berikut ini contoh subquery menggunakan tabel-tabel dalam file **prak_sbd_11_subquery_emp.sql.**

Contoh penggunaan operator single row:

- Menampilkan nama mahasiswa yang memiliki nilai_angka = 90.

```
SELECT nama FROM mahasiswa WHERE nim = (SELECT nim FROM registrasi WHERE
nilai_angka = 90)
```

- Menampilkan nama mahasiswa yang memiliki berat badan lebih berat daripada LUKE SKYWALKER.

```
SELECT nama, beratbadan FROM mahasiswa WHERE beratbadan > (SELECT
beratbadan FROM mahasiswa WHERE nama LIKE "LUKE%")
```

- Menampilkan nama mahasiswa yang memiliki berat badan di bawah berat badan rata-rata mahasiswa.

```
SELECT nama, beratbadan FROM mahasiswa WHERE beratbadan < (SELECT AVG(beratbadan) FROM mahasiswa)
```

Contoh penggunaan operator multiple rows/columns:

- Menampilkan nama mahasiswa dari hasil pencarian nim dan berat badan dari mahasiswa yang memiliki berat badan di bawah 55 kg.

```
SELECT nama FROM mahasiswa WHERE (nim, beratbadan) = ANY (SELECT nim, beratbadan FROM mahasiswa WHERE beratbadan < 55)
```

ekuivalen dengan

```
SELECT nama FROM mahasiswa WHERE (nim, beratbadan) IN (SELECT nim, beratbadan FROM mahasiswa WHERE beratbadan < 55)
```

- Menampilkan nim dan nilai angka dari registrasi dimana nilai angka lebih besar dari setiap nilai angka antara 70 hingga 80.

```
SELECT nim, nilai_angka FROM registrasi WHERE nilai_angka > ANY (SELECT nilai_angka FROM registrasi WHERE nilai_angka BETWEEN 70 AND 80) ORDER BY 1
```

- Menampilkan nim dan nilai angka dari registrasi dimana nilai angka lebih besar dari semua nilai angka antara 70 hingga 80.

```
SELECT nim, nilai_angka FROM registrasi WHERE nilai_angka > ALL (SELECT nilai_angka FROM registrasi WHERE nilai_angka BETWEEN 70 AND 80) ORDER BY 1
```

- Bila ada mahasiswa yang nilai angkanya lebih besar dari 90, maka tampilkan seluruh isi tabel registrasi.

```
SELECT * FROM registrasi WHERE EXISTS (SELECT nilai_angka FROM registrasi WHERE nilai_angka > 90)
```

- EXISTS juga bisa digunakan sebagai operator intersect.

```
SELECT * FROM a WHERE EXISTS(SELECT * FROM b WHERE a.data=b.data)
```

Latihan B (menggunakan file **`prak_sbd_11_subquery_emp.sql`**)
Pengerjaan menggunakan SQL Aggregation yang sudah pernah dibahas sebelumnya. Klik di sini untuk membuka modulnya.

id_karyawan	nama_depan	nama_belakang	gaji	id_departemen
110	Emil	Artanti	4000000	1
111	Rino	Cheros	3000000	4
112	Regina	Bella	5000000	1
113	Richard	Adinata	3500000	3
114	Angeline	Novi	3500000	2
115	Tomi	Sudiro	2000000	1
116	Setiawan	Setiawan	1500000	2

id_departemen	nama_departemen	region
1	Finance	2
2	IT	2
3	Marketing	1
4	HRD	1

1. Tampilkan data karyawan yang memiliki departemen yang sama dengan Emil.
2. Tampilkan nama depan karyawan yang memiliki gaji lebih besar daripada Rino Cheros.
3. Tampilkan data karyawan yang memiliki gaji di bawah gaji rata-rata karyawan.
4. Tampilkan data karyawan yang bekerja pada departemen Finance atau pada Region 2.
5. Tampilkan nama departemen yang karyawannya memiliki gaji di atas 3,5 juta.
6. Tampilkan karyawan yang tidak bekerja di departemen Region 1.
7. Tampilkan karyawan yang gajinya lebih rendah daripada semua karyawan yang bekerja di departemen Finance.

REFERENSI:

MySQL Intersect Tutorial - http://www.mysqltutorial.org/mysql-intersect/

MySQL Minus Tutorial - http://www.mysqltutorial.org/mysql-minus/

MySQL Subquery Syntax - https://dev.mysql.com/doc/refman/5.7/en/subqueries.html

MySQL Union Syntax - https://dev.mysql.com/doc/refman/5.7/en/union.html