PRAKTIKUM SISTEM BASIS DATA

Modul 5: SELECT Query

Tujuan dan Capaian Pembelajaran Praktikum:

1. Mahasiswa dapat menjelaskan konsep Data Query Language, yang berupa pernyataan SELECT terhadap isi dari suatu tabel.
2. Mahasiswa dapat memunculkan data dari suatu tabel dengan atau tanpa kondisi tertentu.

INSTRUKSI UMUM: Bacalah dahulu modul ini sebelum praktikum, sehingga ketika praktikum bisa langsung mencoba latihan dan menjelaskan variasi kasus yang mungkin muncul.

Alat dan Bahan:

1. Modul Praktikum
2. Komputer
3. Software MySQL **(phpMyAdmin)**
4. Viewer LCD

Materi dan Pembagian Waktu:

Materi dan pembagian waktu pelaksanaan praktikum meliputi:

No	Materi	Waktu
1	Pretest (sebelum kelas)	10 menit
2	SELECT Query	50 menit
5	Latihan Mandiri	30 menit
6	Postest	70 menit
	Total	150 menit

Materi Praktikum

Seperti yang sudah kita pelajari sebelumnya, CRUD memegang peranan penting dalam basis data. Apa yang akan kita pelajari di modul ini merupakan bentuk nyata dari CRUD yang diimplementasikan dalam bahasa SQL, terutama untuk Read data.

1. **Pengertian DQL SELECT**
   SELECT adalah statemen yang sangat berguna untuk mengambil (*retrieve*) baris yang dipilih dari satu tabel atau lebih. SELECT dapat digabungkan dengan sesama statemen SELECT lainnya dengan operator UNION dan dapat diterapkan secara bersarang/subquery.
   Sintaks statemen SELECT yang paling sederhana adalah sebagai berikut:

```
SELECT select_expr
FROM table_references
[WHERE where_condition];
```

```
SELECT * FROM table_references [WHERE where_condition];
```

```
SELECT column1, column2, columnN FROM table_references [WHERE where_condition];
```

Sedangkan sintaks lengkap dari statemen SELECT adalah:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Kita akan membahas dari yang bentuk sederhana terlebih dahulu. Perhatikan hal-hal berikut:

- Tiap `select_expr` menunjukkan kolom yang ingin diambil. Jika ingin menampilkan semua kolom bisa menggunakan karakter *. Minimal ada satu `select_expr` setiap statemen SELECT.
- `table_references` menunjukkan tabel mana saja yang akan diambil barisnya. Apabila mengambil lebih dari satu tabel maka perlu memperhatikan relasi antar tabel dan *join* (dibahas pada modul 6).
- Klausa WHERE menunjukkan kondisi apa saja yang harus dipenuhi oleh baris yang akan diambil. `where_condition` adalah ekspresi yang mengevaluasikan nilai *true* terhadap masing-masing baris yang akan diambil. Statemen SELECT akan menampilkan seluruh baris apabila klausa WHERE tidak dipakai.

Pada ekspresi WHERE, kita dapat menerapkan berbagai macam fungsi dan operator untuk mendukung pembatasan pengambilan baris.

SELECT juga dapat digunakan untuk mendapatkan hasil dari suatu perhitungan tanpa melibatkan tabel manapun. Contoh:

```
SELECT 1+1
FROM dual;
```

maka akan menghasilkan nilai 2. Pada beberapa DBMS (salah satunya Oracle), kita diminta menggunakan DUAL sebagai tabel *dummy* sebagai kondisi ketika tidak ada tabel yang diacu. DBMS MySQL sendiri tidak membutuhkan pencantuman FROM DUAL.

2. Query pada Relasi/Tabel Tunggal

Kita menggunakan contoh tabel *matakuliah* seperti yang dibahas di kelas teori. Anda bisa membuat dan mengisi datanya dengan mencopas SQL berikut.

```
CREATE DATABASE [nama]_p5;
USE [nama]_p5;
CREATE TABLE matakuliah (
  idMtk VARCHAR(8) PRIMARY KEY,
  namaMtk TEXT NOT NULL,
  namaProdi VARCHAR(50) NOT NULL,
  sks INT(1) NOT NULL
);

INSERT INTO matakuliah VALUES
('BIO-101', 'Pengantar Biologi', 'Biologi', 4),
('BIO-301', 'Genetika', 'Biologi', 4),
('BIO-399', 'Biologi Komputasional', 'Biologi', 3),
('TI-101', 'Teknologi Komputer', 'Informatika', 4),
('TI-190', 'Game Design', 'Informatika', 4),
('TI-315', 'Sistem Basis Data', 'Informatika', 3),
('TI-319', 'Pemrograman Web', 'Informatika', 3),
('DP-322', 'Estetika Produk', 'Desain Produk', 3),
('AK-181', 'Perpajakan', 'Akuntansi', 3),
('AR-351', 'Analisis Struktur I', 'Arsitektur', 3),
('KD-200', 'Anatomi I', 'Kedokteran', 3),
('PB-255', 'Grammar', 'Pendidikan Bahasa', 4);
```

Untuk sementara kita mengabaikan relationship dalam kasus *matakuliah* dengan relasi *prodi*.

Tabel *matakuliah*

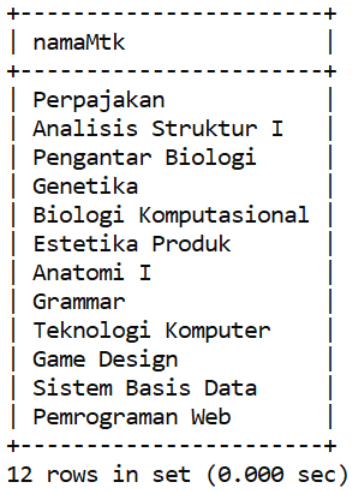
idMtk	*namaMtk*	*namaProdi*	*sks*
BIO-101	Pengantar Biologi	Biologi	4
BIO-301	Genetika	Biologi	4
BIO-399	Biologi Komputasional	Biologi	3
TI-101	Teknologi Komputer	Informatika	4
TI-190	Game Design	Informatika	4
TI-315	Sistem Basis Data	Informatika	3
TI-319	Pemrograman Web	Informatika	3
DP-322	Estetika Produk	Desain Produk	3
AK-181	Perpajakan	Akuntansi	3
AR-351	Analisis Struktur I	Arsitektur	3
KD-200	Anatomi I	Kedokteran	3
PB-255	Grammar	Pendidikan Bahasa	4

Gambar 1. Tabel *matakuliah*

Kita menjawab pertanyaan munculkan seluruh nama mata kuliah yang ada. Nama mata kuliah berada pada tabel *matakuliah* sehingga nama tabel ini kita sebutkan di klausa FROM. Nama mata kuliah sendiri merupakan atribut/kolom di tabel *matakuliah* dengan nama kolom *namaMtk* sehingga nama kolom ini kita sebutkan di klausa SELECT. Statemen lengkapnya adalah sebagai berikut:

```
SELECT namaMtk
FROM matakuliah;
```

Hasilnya adalah sebuah relasi yang berisi atribut tunggal dengan nama *namaMtk*. Gambar 2 menunjukkan tabel hasil querynya.



```
+----------------------+
| namaMtk              |
+----------------------+
| Perpajakan           |
| Analisis Struktur I  |
| Pengantar Biologi    |
| Genetika             |
| Biologi Komputasional |
| Estetika Produk      |
| Anatomi I            |
| Grammar              |
| Teknologi Komputer   |
| Game Design          |
| Sistem Basis Data    |
| Pemrograman Web      |
+----------------------+
12 rows in set (0.000 sec)
```

Gambar 2. Tabel hasil query

Kita dapat mengurutkan tampilan di atas dengan menambahkan klausa ORDER BY:

```
SELECT namaMtk
FROM matakuliah
ORDER BY namaMtk;
```

Kemudian jika kita ingin memunculkan kolom *namaProdi* dapat dilakukan dengan statemen:

```
SELECT namaProdi
FROM matakuliah;
```

Dari statemen di atas didapat beberapa nama prodi yang berulang. Jika kita hanya ingin menampilkan nama prodi yang sama sebanyak satu kali saja, kita dapat menggunakan keyword DISTINCT.

```
SELECT DISTINCT namaProdi
FROM matakuliah;
```

Kita dapat melibatkan operasi aritmatik untuk mendapatkan perhitungan yang bersifat sementara. Untuk membuat hasil perhitungan menjadi tersimpan secara permanen sudah dipelajari di bagian UPDATE.

```
SELECT namaMtk, namaProdi, sks * 2
FROM matakuliah;
```

Kita dapat mengubah atau memberikan alias dari suatu kolom hasil SELECT, yaitu dengan menambahkan keyword AS diikuti nama kolom baru.

```
SELECT namaMtk, namaProdi, sks * 2 AS harga
FROM matakuliah;
```

Hasil query di atas ditunjukkan pada Gambar 3.

```
+----------------------+-------------------+-------+
| namaMtk              | namaProdi         | harga |
+----------------------+-------------------+-------+
| Perpajakan           | Akuntansi         |     6 |
| Analisis Struktur I  | Arsitektur        |     6 |
| Pengantar Biologi    | Biologi           |     8 |
| Genetika             | Biologi           |     8 |
| Biologi Komputasional| Biologi           |     6 |
| Estetika Produk      | Desain Produk     |     6 |
| Anatomi I            | Kedokteran        |     6 |
| Grammar              | Pendidikan Bahasa |     8 |
| Teknologi Komputer   | Informatika       |     8 |
| Game Design          | Informatika       |     8 |
| Sistem Basis Data    | Informatika       |     6 |
| Pemrograman Web      | Informatika       |     6 |
+----------------------+-------------------+-------+
12 rows in set (0.003 sec)
```

Gambar 3. Pemberian alias pada suatu kolom

3. Penerapan Filter/Syarat hasil Query

Seringkali kita membutuhkan pembatasan, penerapan filter atau pemberian syarat terhadap baris-baris yang akan kita tampilkan. Hal ini ditangani dengan memanfaatkan klausa **WHERE**. Contohnya, jika kita ingin menampilkan mata kuliah dari Prodi Informatika yang bobot SKSnya 3 seperti yang ditunjukkan pada Gambar 4, kita bisa menuliskan statemen berikut:

```
SELECT namaMtk
FROM matakuliah
WHERE namaProdi = 'Informatika' AND sks = 3;
```

```
+--------------------+
| namaMtk            |
+--------------------+
| Sistem Basis Data  |
| Pemrograman Web    |
+--------------------+
2 rows in set (0.004 sec)
```

Gambar 4. Hasil penerapan filter

SQL mengizinkan penggunaan operator logika AND, OR dan NOT untuk menggabungkan beberapa filter sekaligus. SQL juga memiliki berbagai cara untuk membandingkan nilai yang digunakan sebagai syarat, seperti misalnya <, >, >=, <= untuk pembandingan nilai bilangan, dan LIKE untuk pembandingan nilai string. Contoh query berikut memberikan hasil yang sama dengan Gambar 4:

```
SELECT namaMtk
FROM matakuliah
WHERE namaProdi LIKE 'Infor%' AND sks < 4;
```

Penggunaan karakter wildcard '%' berarti akan menganggap cocok seluruh string sisanya. Jika hanya menghendaki satu karakter saja yang dianggap cocok, kita dapat menggunakan karakter '_'. Kita akan mempelajari operator dan fungsi pembandingan data ini di bagian/pertemuan lain.

List Operator yang dapat digunakan dalam klausa WHERE

<table>
<tr><th>Operator</th><th>Deskripsi</th><th>Format Penggunaan dan Contoh</th></tr>
<tr><td>></td><td>Lebih Besar dari</td><td>SELECT * FROM Emp1 WHERE Age>24;

Artinya kita akan mencari baris data dalam tabel Emp1, dimana umur pegawainya lebih besar dari 24</td></tr>
<tr><td>>=</td><td>Lebih Besar dari atau Sama</td><td>SELECT * FROM Emp1 WHERE Age>=24;

Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya berumur 24 tahun ke atas</td></tr>
<tr><td><</td><td>Kurang dari</td><td>SELECT * FROM Emp1 WHERE Age<24;

Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya belum berumur 24.</td></tr>
<tr><td><=</td><td>Kurang dari atau Sama</td><td>SELECT * FROM Emp1 WHERE Age<=24;

Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya yang berumur 24 ke bawah.</td></tr>
<tr><td>=</td><td>Sama</td><td>SELECT * FROM Emp1 WHERE Age=24;

Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya yang berumur 24 saja.</td></tr>
<tr><td><></td><td>Tidak Sama</td><td>SELECT * FROM Emp1 WHERE Age<>24;

Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya yang tidak berumur 24.</td></tr>
<tr><td>BETWEEN</td><td>Antara rentang inklusif</td><td>Sintax: expression BETWEEN value1 AND value2;

SELECT * FROM Emp1 WHERE AGE BETWEEN 24 AND 27;

Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya yang berumur dari 24 sampai dengan 27.</td></tr>
<tr><td>LIKE</td><td>Mencari pola/pattern</td><td>Bisa menggunakan % atau _</td></tr>
<tr><td colspan="3">
<table>
<tr><th>LIKE Operator</th><th>Description</th></tr>
<tr><td>WHERE CustomerName LIKE 'a%'</td><td>Finds any values that start with "a"</td></tr>
<tr><td>WHERE CustomerName LIKE '%a'</td><td>Finds any values that end with "a"</td></tr>
<tr><td>WHERE CustomerName LIKE '%or%'</td><td>Finds any values that have "or" in any position</td></tr>
<tr><td>WHERE CustomerName LIKE '_r%'</td><td>Finds any values that have "r" in the second position</td></tr>
<tr><td>WHERE CustomerName LIKE 'a_%'</td><td>Finds any values that start with "a" and are at least 2 characters in length</td></tr>
<tr><td>WHERE CustomerName LIKE 'a__%'</td><td>Finds any values that start with "a" and are at least 3 characters in length</td></tr>
<tr><td>WHERE ContactName LIKE 'a%o'</td><td>Finds any values that start with "a" and ends with "o"</td></tr>
</table>
</td></tr>
<tr><td>IN</td><td>Untuk menentukan beberapa nilai yang mungkin untuk kolom</td><td>SELECT * FROM Emp1 WHERE Age in (23,24,25);</td></tr>
</table>

		Artinya kita akan mencari baris data dalam tabel Emp1, yang pegawainya yang berumur 23, 24, dan 25.

AND and OR operators in SQL

Operator **AND** menampilkan record jika kedua kondisi pertama dan kondisi kedua adalah benar. Operator **OR** menampilkan record jika salah satu kondisi pertama ATAU kondisi kedua benar.

```
SELECT * FROM table_name WHERE condition1 AND condition2 AND …conditionN
```

Contoh: **`SELECT * FROM Student WHERE Age = 18 AND ADDRESS = 'Delhi';`**
Artinya kita akan mencari baris data dalam tabel Student, yang siswanya berusia 18 tahun dan tinggal di Delhi.

```
SELECT * FROM table_name WHERE condition1 OR condition2 OR …conditionN
```

Contoh: **`SELECT * FROM Student WHERE Age = 18 AND ADDRESS = 'Delhi';`**
Artinya kita akan mencari baris data dalam tabel Student, yang siswanya berusia 18 tahun atau seorang siswa yang tinggal di Delhi.

`SELECT * FROM Student WHERE Age = 18 AND (NAME = 'Ram' OR NAME = 'RAMESH');`
Apa arti query tersebut?

4. Penggabungan Kolom Hasil Query

Untuk menggabungkan beberapa value/data dari berbagai kolom menjadi satu kolom, kita bisa menggunakan fungsi CONCAT(). Berikut contohnya jika kita hendak menampilkan nama mata kuliah menjadi Sistem Basis Data (Prodi Informatika) seperti pada Gambar 5, kita menuliskan statemen:

```
SELECT CONCAT(namaMtk,' (Prodi ',namaProdi,')')
FROM matakuliah;
```

```
+-----------------------------------------+
| CONCAT(namaMtk,' (Prodi ',namaProdi,')') |
+-----------------------------------------+
| Perpajakan (Prodi Akuntansi)            |
| Analisis Struktur I (Prodi Arsitektur)  |
| Pengantar Biologi (Prodi Biologi)       |
| Genetika (Prodi Biologi)                |
| Biologi Komputasional (Prodi Biologi)   |
| Estetika Produk (Prodi Desain Produk)   |
| Anatomi I (Prodi Kedokteran)            |
| Grammar (Prodi Pendidikan Bahasa)       |
| Teknologi Komputer (Prodi Informatika)  |
| Game Design (Prodi Informatika)         |
| Sistem Basis Data (Prodi Informatika)   |
| Pemrograman Web (Prodi Informatika)     |
+-----------------------------------------+
12 rows in set (0.004 sec)
```

Gambar 5. Hasil penggabungan kolom

Untuk membuat hasil penggabungan kolom yang lebih bagus dan dapat dimengerti bisa menggunakan keyword **AS** untuk mengubah nama kolom dalam query select.

```
SELECT CONCAT(namaMtk,' (Prodi ',namaProdi,')') AS 'Matakuliah (Prodi)'
FROM matakuliah;
```

Latihan Mandiri

Berikut ini adalah tabel *registrasi*.

nim	*idMtk*	*grup*	*semester*	*tahun*	*nilaiAngka*
1234	TI-101	A	Gasal	2022	86.55
1212	TI-190	B	Gasal	2022	84.03
2211	SI-201	A	Genap	2022	72.76
1414	SI-290	A	Genap	2022	62.56
4321	TI-101	B	Genap	2022	65.93
2222	TI-190	B	Gasal	2022	90.00
1212	TI-101	A	Gasal	2022	80.47
1234	TI-190	A	Gasal	2022	78.20

Data dictionary:

Kolom	Tipe Data	Lebar	Konstrain
nim	INT	4	PK
idMtk	VARCHAR	7	PK
grup	CHAR	1	PK
semester	VARCHAR	5	PK
tahun	VARCHAR	4	PK
nilaiAngka	FLOAT	4,2	

1. Tampilkan NIM mahasiswa yang mengambil mata kuliah TI-190 Semester Gasal 2022!
2. Tampilkan kode mata kuliah yang ditawarkan pada Semester Genap 2022!
3. Tampilkan NIM, kode mata kuliah dan nilai dalam satu kolom dengan nama kolom KHS, dengan delimiter/pemisah berupa karakter ' - '!
4. Tampilkan data registrasi dari Semester Gasal 2022 diurutkan nilai angka terbesar menuju terkecil!
5. Tampilkan data registrasi untuk mata kuliah selain SI-201!
6. Tampilkan NIM, kode mata kuliah, grup dan nilai angka bagi mahasiswa yang nilainya antara 70 hingga 80!
7. Tampilkan nim, matakuliah yang diambil oleh mahasiswa yang karakter ke-3 dari nim adalah 1.
8. Tampilkan matakuliah yang berasal dari prodi TI dan hanya sekali muncul (tidak boleh menampilkan data yang sama).
9. Dengan menggunakan logika keyword IF seperti pada Excel atau Google Sheet, buatlah tampilan sebagai berikut:

```
+------+--------+------+----------+-------+------------+----------+
| nim  | idMtk  | grup | semester | tahun | nilaiAngka | predikat |
+------+--------+------+----------+-------+------------+----------+
| 1212 | TI-101 | A    | Gasal    | 2022  |      80.47 | Cool!    |
| 1212 | TI-190 | B    | Gasal    | 2022  |      84.03 | Cool!    |
| 1234 | TI-101 | A    | Gasal    | 2022  |      86.55 | Cool!    |
| 1234 | TI-190 | A    | Gasal    | 2022  |      78.20 | Uncool!  |
| 1414 | SI-290 | A    | Genap    | 2022  |      62.56 | Uncool!  |
| 2211 | SI-201 | A    | Genap    | 2022  |      72.76 | Uncool!  |
| 2222 | TI-190 | B    | Gasal    | 2022  |      90.00 | Cool!    |
| 4321 | TI-101 | B    | Genap    | 2022  |      65.93 | Uncool!  |
+------+--------+------+----------+-------+------------+----------+
```

10. Agar sistem dapat memunculkan informasi nama mahasiswa, apa yang harus dilakukan?
11. Agar sistem dapat memunculkan informasi SKS setiap mata kuliahnya, apa yang harus dilakukan?

REFERENSI:

MySQL SELECT Statement - https://dev.mysql.com/doc/refman/5.7/en/select.html

Lynn Beighley, “Head First SQL”, 2007, O’Reilly. ISBN: 978-0-596-52684-9

https://www.geeksforgeeks.org/sql-and-and-or-operators/?ref=lbp

https://www.w3schools.com/mysql/mysql_like.asp