LAPORAN RESMI

MODUL IV

(SQL)

Sistem Manajemen Basis Data

NAMA	**: Giraldo Stevanus**
N.R.P	**: 220441100064**
DOSEN	**: Achmad Yasid,S.Kom.,M.Kom.,MBA**
ASISTEN	**: Fanky Abdilqoyyim**
TGL PRAKTIKUM	**: 04-04-2024**

Disetujui..... 2022
Asisten

Fanky Abdilqoyyim
21.04.411.00054

LABORATORIUM BISNIS INTELIJEN SISTEM

PRODI SISTEM INFORMASI

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

SQL, atau Structured Query Language, adalah bahasa pemrograman yang digunakan dalam sistem manajemen basis data relasional (RDBMS) untuk mengelola data. Dalam SQL, terdapat beberapa subset yang penting, termasuk DDL (Data Definition Language) yang digunakan untuk mendefinisikan struktur dan skema database, DML (Data Manipulation Language) yang digunakan untuk mengelola data dalam database, dan DCL (Data Control Language) yang digunakan untuk mengatur hak akses dan izin pengguna terhadap objek database. Dengan menggunakan perintah-perintah DDL, DML, dan DCL, pengguna dapat efisien membuat, mengubah, menghapus objek database, memanipulasi data, dan mengontrol akses pengguna sesuai kebutuhan.

Di sisi lain, Data Manipulation Language (DML) adalah bagian dari SQL yang digunakan untuk mengelola data dalam database. Perintah DML memungkinkan pengguna untuk memasukkan, mengubah, menghapus, dan mengambil data dari tabel. Ini mencakup perintah seperti INSERT untuk menambahkan data baru, UPDATE untuk mengubah data yang ada, DELETE untuk menghapus data, dan SELECT untuk mengambil data dari tabel. Dengan menggunakan perintah-perintah DML, pengguna dapat memanipulasi data sesuai dengan kebutuhan aplikasi mereka. Selain itu, Data Control Language (DCL) adalah aspek penting dari SQL yang digunakan untuk mengatur hak akses dan izin pengguna terhadap data dalam database. Perintah DCL, seperti GRANT dan REVOKE, memungkinkan administrator database untuk memberikan atau mencabut hak akses pengguna ke objek database tertentu, memberikan tingkat kontrol yang lebih tinggi terhadap keamanan data. Dengan demikian, DCL membantu dalam menjaga keamanan data dan mencegah akses yang tidak sah atau tidak diinginkan.

1.2 Tujuan

- Mampu memahami pengembangan data
- Mampu memahami administrasi database

BAB II
DASAR TEORI

2.1 Pengertian Dasar

a. SQL

Secara umum perintah-perintah yang terdapat di dalam SQL, diklasifikasikanmenjadi tiga bagian, antara lain yaitu :

1. DDL (Data Definition Language)

Merupakan perintah SQL yang berkaitan dengan pendefinisian suatustruktur database, dalam hal ini database dan table.

Perintah DDL adalah: CREATE, ALTER, RENAME, DROP

2. DML (Data Manipulation Language)

Merupakan perintah SQL yang berkaitan dengan manipulasi atau pengolahan data atau record dalam table.

Perintah DML antara lain: SELECT, INSERT, UPDATE, DELETE

3. DCL (Data Control Language)

Merupakan perintah SQL yang berkaitan dengan manipulasi user danhak akses (priviledges).

Perintah SQL yang termasuk dalam DCL antara lain: GRANT, REVOKE.

b. SELECT

SELECT merupakan salah satu pondasi dalam SQL Programming. SELECT digunakanuntuk menampilkan data, terlebih untuk mencari informasi dalam kumpulan data.

Sintak

SELECT dibagi kedalam 6 komponen, antara lain:

a) SELECT. Diikuti oleh **<select_list>,** dapat berupa

literal_value ataucolumn_list atau asterisk (*).

b) FROM. Diikuti oleh **<table_name>** sesuai dengan column_list. Jadi jika ada

data yang diambil dari kolom tertentu, harus diketahui kolom tersebut diambil dari tabel mana. Tabel pada FROM dapat diikuti dengan alias untuk mempermudah penulisan khususnya ketika join dan subquery.

c) WHERE. Diikuti oleh kondisi secara umum.

d) GROUP BY. Diikuti oleh **<select_list>.** Bagian ini muncul ketika ada fungsi-fungsi agregasi.

e) HAVING. Diikuti oleh kondisi hanya untuk fungsi -fungsi agregasi.

f) ORDER BY. Diikuti oleh <select_list>

Perintahnya :

```
SELECT     <select_list>
[FROM      <table_name>]
[WHERE     <kondisi1> [AND/OR <kondisi2>]]
[GROUP BY  <select_list>]
[HAVING    <kondisi1> [AND/OR <kondisi2>]]
[ORDER BY  <select_list>]
```

Keterangan :

SELECT, INTO, FROM, WHERE, GROUP BY, HAVING DAN ORDER BY

Kata kunci:

(keyword) yang harus disertakan jika kita membutuhkannya di dalam pengolahan data.

select_list, table_source, search_condition, group_by_expression, order_expression

isian yang bisa kita ubah berdasarkan kebutuhan kita.

Kurung kotak [ ] bagian tersebut boleh disertakan atau tidak, tergantung dari kebutuhan

c. Menyaring data

Tidak semua data yang ada pada tabel, ingin ditampilkan. Terlebih ketika tabelterbagi kedalam banyak kolom dengan jumlah data yang sangat besar.

Operator Pembanding :

Operator	Keterangan
=	Sama dengan
>	Lebih besar dari
>=	Lebih besar sama dengan
<	Kurang dari
<=	Kurang dari sama dengan
<> atau !=	Tidak sama dengan
BETWEEN .. AND ..	Diantara 2 nilai
IN (set)	Cocok dengan salah satu diantara daftar nilai
LIKE	Cocok dengan pola karakter
IS NULL	Sama dengan NULL

Perintahnya :

Select * From Nama_Table Where Nama_Field [Operator Relasional] Ketentuan;

Contoh :

select * from customer where cust_id = ‘10003’ or cust_name =’wascals;

d. Pengurutan data (ACS, DESC, ORDER BY)

1. Mengurutkan Data

Untuk mengurutkan tampilan data dari suatu table, digunakan klausa OrderBy. Klausa Order By, dapat digunakan untuk mengurutkan data :

- Asc (Ascending) : Untuk mengurutkan data dari kecil ke besar
- Desc (Descending) : Untuk mengurutkan data dari besar ke kecil Perintahnya :

Select * From Nama_Table Order By Nama_Field_Key Asc/Desc;

Contoh :

Select * From products Order By prod_name Asc;

e. OPERATOR BETWEEN, IN, LIKE

1. Operator Between

Operator Between merupakan operator yang digunakan untuk menangani operasi jangkauan.

Perintahnya :

Select * From Nama_Table Where Nama_Field_ketentuan Between 'Ketentuan_1' And 'Ketentuan_2';

Contoh :

Select * From orderitems Where quantity Between '1' And '10';

2. Operator In

Operator In merupakan operator yang digunakan untuk mencocokan suatu nilai.

Perintahnya :

Select Nama_Field From Nama_Table Where Nama_Field_Pencocok In ('Isi_Field_1','Isi_Field_2');

Contoh :

Menampilkan nama customer, alamat dan email customer tertentu.

Select cust_name,cust_address,cust_email From customers Where cust_id In ('10002','10005');

3. Operator Like

Operator Like merupakan operator yang digunakan untuk mencari suatu data (*search)*.

Perintahnya :

Select * From Nama_Table Where Nama_Field_Dicari Like '%Key';

Contoh :

Select * From Products Where prod_name Like '%s';

Query yang pertama menampilkan produk dengan nama produk diawali huruf dan pada query yang kedua nama produk diakhiri huruf s.

f. Ekspresi Query

Ekspresi Query dapat digunakan untuk melakukan perubahan terhadap fieldkolom keluaran, menambah baris teks field keluaran.

1. Mengganti Nama Fieldkeluaran

Perintahnya :

Select Nama_Field_Asal As 'Nama_Field_Pengganti' From Nama_Table; Contoh :

Select Kode_Mtkul As 'Kode Matakuliah', Nama_Mtkul As 'Matakuliah' From Mtkul;

2. Menambahkan Baris Teks FieldKeluaran

Perintahnya :

Select 'Nama Field Tambahan', Nama_Field_Asal From Nama_Table; Contoh :

Select vend_name,’diproduksi di’, vend_city From vendors;

3. Ekspresi Kondisi

Perintahnya :

Select Nama_Field_1 Case Nama_Field_2 When 'Nilai_field_2' Then 'Keterangan_1' Else 'Keterangan_2' End As Nilai_field_2 From Nama_Table; Contoh :

Select Kode_Mtkul, Nama_Mtkul, Case Sks When '1' Then 'Praktikum' Else 'Matakuliah' End As Sks From Mtkul;

g. Agregasi dan Pengelompokan Data

Dalam pemrosesan data mentah menjadi data statistik, diperlukan fungsi-fungsi yang dapat meng-agregasi data-data tersebut. Fungsi-fungsi ini meliputi SUM, MIN, MAX, COUNT, dan AVG.

Fungsi	Keterangan
AVG	Menghitung rata-rata
COUNT	Menghitung cacah data /jumlah baris

MAX	Memperoleh nilai terbesar
MIN	Memperoleh nilai terkecil
SUM	Memperoleh jumlahan data

a. Multiple-table Query

Data-data yang tersimpan dalam basis data, tersebar kedalam beberapa tabel. Tabel-tabel ini dihubungkan dengan yang namanya referential constraint, yaitu hubungan antara foreign key dan primary key.

Karena itulah, untuk mendapatkan informasi yang tersebar, dibutuhkan metode untuk menggabungkan property tabel-tabel tersebut. Metode yang digunakan ada 2 macam, yaitu join dan subquery.

Perbedaannya sederhana, join menggunakan satu SELECT, sedangkan subquery menggunakan dua atau lebih SELECT (umumnya dikatakan sebagai SELECT withina SELECT).

Join

Bentuk joinpertama kali adalah menggunakan kata kunci WHERE untuk melakukan penggabungan tabel.

```
SELECT  <select_list>
FROM    <table1>, <table2> [, ...]
WHERE   <table1.PK = table2.FK> [AND ...]
```

```
SELECT  <select_list>
FROM    <table1> JOIN <table2>
        ON < table1.PK = table2.FK> [[AND ...]
        JOIN ...];
```

Perkembangan SQL ANSI

Tipe Join

Ada 2 tipe join, yaitu inner join yang lebih menekankan pada keberadaan datayang sama, dan outer join.

```
SELECT  <select_list>
FROM    <tabel1 sebagai kiri>
        <LEFT/RIGHT> [OUTER] JOIN
        <tabel2 sebagai kanan>
        ON <table1.PK = table2.FK> [AND ...];
```

Pada INNER JOIN atau CROSS JOIN *output*/hasil yang ditampilkan adalah data- data dari semua table yang terlibat dimana baris yang tampil hanya yang memiliki kondisi kesamaan data. Kesamaan data berdasarkan relasinya (kesamaan data *foreign key* dengan *primary key* tabel yang diacu). Berikutadalah bentuk umum INNER JOIN yang umumnya hanya disebut sebagai JOIN:

SELECT nm_tabel1.nm_kolom1,
nm_tabel1.nm_kolom2,
nm_tabel2.nm_kolom1, nm_tabel2.nm_kolom2 FROM tabel1, tabel2 WHERE tabel1.nama_kolom1

(primary key)=tabel2.nama_kolom1(foreign key yg mengacu ke tabel1)

Contoh penggunaan Join, kita lihat kembali skema order entry dibawah ini.
Menampilkan prod_name, vend_name dari table vendors dan products.

Select vendors.vend_name,products.prod_name from vendors, products
Where vendors.vend_id = products.vend-id

Clausa Join On Alias

SELECT a.nm_kolom1, b.nm_kolom2, a.nm_kolom3

FROM tabel1 aJOIN tabel2 b

ON a.nama_kolom1(primary key)=b.nama_kolom1(foreign key yg mengacu ke tabel1)
WHERE kondisi;

a. JOIN 3 TABLE ATAU LEBIH

Pada prinsipnya sama, hanya jumlah tabel ditambah dan sintaks disesuaikan. Contoh penerapan join dua tabel atau lebih untuk menampilkan nama customer, tgl order dan total jumlah order.

select a.cust_name,b.order_date,c.quantity from customers a join orders b on a.cust_id=b.cust_id join orderitems c on b.order_num=c.order_num;

a. OUTER JOIN

Pada OUTER JOIN hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data yang tidak memiliki kesamaan data berdasarkan relasinya dari salah satu tabel. Terdapat dua tipe OUTER JOIN, yaitu:

1. LEFT OUTER JOIN atau biasa disebut left join
2. RIGHT OUTER JOIN atau biasa disebut righ join

a) LEFT JOIN

Pada LEFT JOIN output/hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data-data yang tidak memilikikesamaan data berdasarkan relasinya dari tabel sebelah kiri dari klausa LEFT JOIN. Berikut adalah bentuk umum:

SELECT nm_tabel1.nm_kolom1, nm_tabel1.nm_kolom2, nm_tabel2.nm_kolom1, nm_tabel2.nm_kolom2 FROM tabel1

LEFT JOIN tabel2

ON tabel1.nama_kolom1(primary key)=tabel2.nama_kolom1(foreignkey yg mengacu ke tabel1)

WHERE kondisi;

Contoh :

select a.cust_name,b.order_date
from customers a left join orders b on a.cust_id=b.cust_id

RIGHT JOIN Pada RIGHT JOIN output/hasil yang ditampilkan adalah data-data dari semuatabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan databerdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data-data yang tidak memiliki kesamaan databerdasarkan relasinya dari tabel sebelah kanan dari klausa RIGHT JOIN.

SELECT nm_tabel1.nm_kolom1, nm_tabel1.nm_kolom2, nm_tabel2.nm_kolom1, nm_tabel2.nm_kolom2
FROM tabel1 RIGHT
JOIN tabel2
ON tabel1.nama_kolom1(primary key)=tabel2.nama_kolom1(foreignkey yg mengacu ke tabel1)
WHERE kondisi;

Contoh :

select a.cust_name,b.order_date from customers a right join orders b on a.cust_id=b.cust_id

b) SELF JOIN

Self join adalah melakukan join dengan dirinya sendiri. Atau join dengantable yang sama.

Sintak nya sbb :

select nama alias1_table.kolom1, nama alias2_table.kolom2,
fromtable alias1 inner join table alias2 on alias1.kolom3=alias2.kolom3

contoh

select a.vend_name,b.vend_state, 'negaranya' ,b.vend_country
fromvendors a inner join vendors b on a.vend_id=b.vend_id

b. Subquery

Subquery merupakan query didalam query. Umumnya, subquery ini dipakai untuk mencari data yang belum diketahui. Penggunaan query didalam query ini umumnya menjadi bagian dari kondisi.

Sintak subquery adalah sebagai berikut:

```
SELECT  <select_list>
FROM    <tabel>
WHERE   <column> =
           (SELECT <single_column>
           FROM    <tabel>
           WHERE   <kondisi>);
```

BAB IV

IMPLEMENTASI

4.1 Source Code

```sql
-- Menampilkan data di urutkan berdasarkan ASC, DESC, ORDER BY
SELECT * FROM karyawan ORDER BY nama ASC;

SELECT * FROM departemen ORDER BY nama_departemen DESC;

SELECT * FROM proyek ORDER BY id_proyek ASC;

SELECT * FROM proyek ORDER BY id_proyek ASC;

SELECT * FROM evaluasikinerja ORDER BY id_karyawan ASC;



-- Soal 2 SQL menggunakan klausa Order By, Group By, dan Having
SELECT id_departemen, COUNT(id_karyawan) AS jumlah_karyawan
FROM departemen
JOIN karyawan  ON id_departemen = id_departemen
GROUP BY id_departemen
HAVING COUNT(id_karyawan) > 1
ORDER BY jumlah_karyawan DESC;



-- Soal 3 Tampilkan data berdasarkan rentang tanggal
SELECT *
FROM absensi
WHERE tanggal BETWEEN '2024-03-01' AND '2024-03-10';


-- soal 4 Tampilkan “produk/barang/jasa” atau data sejenis yang paling banyak muncul pada studi kasus
SELECT nama_departemen, COUNT(*) AS jumlah_kemunculan
FROM departemen
GROUP BY nama_departemen
ORDER BY jumlah_kemunculan DESC
LIMIT 1;



-- Soal 5 Tampilkan“produk/barang/jasa” atau data sejenis yang paling sedikit muncul pada studi kasus
SELECT kota, COUNT(*) AS jumlah_karyawan
FROM karyawan
GROUP BY kota
ORDER BY jumlah_karyawan ASC
LIMIT 1;



-- Soal 6 Tampilkan“produk/barang/jasa” atau data sejenis berdasarkan studi kasus yang dipilih yang
memiliki huruf “a” didepan katanya
SELECT *
FROM karyawan
WHERE nama LIKE 'A%'



-- Soal 7 Cari “konsumen/user” atau data sejenis yang yang paling banyak melakukan transaksi
SELECT id_karyawan, COUNT(*) AS jumlah_absensi
FROM absensi
GROUP BY id_karyawan
ORDER BY jumlah_absensi DESC
LIMIT 1;



-- Soal 8 Cari “konsumen/user” atau data sejenis yang melakukan transaksi lebih dari 3
SELECT id_karyawan, COUNT(*) AS jumlah_absensi
FROM absensi
GROUP BY id_karyawan
HAVING COUNT(*) > 3;


-- Soal 9 Tampilkan data seluruh “konsumen/user” atau data sejenis mulai dari data diri, dan data
melakukan transaksi apa saja
SELECT k.id_karyawan, k.nama, k.jabatan, a.tanggal, a.jam_masuk, a.jam_pulang
FROM karyawan k
LEFT JOIN absensi a ON k.id_karyawan = a.id_karyawan;
```

4.1 Hasil

Rangkaian perintah SQL tersebut mengilustrasikan berbagai penggunaan klausa-klausa seperti ORDER BY, GROUP BY, HAVING, dan JOIN dalam pemrosesan data:

Pertama-tama, ada serangkaian perintah yang digunakan untuk menampilkan data yang diurutkan berdasarkan kolom tertentu menggunakan klausa ORDER BY. Misalnya, perintah SELECT * FROM karyawan ORDER BY nama ASC; akan menampilkan data dari tabel "karyawan" yang diurutkan berdasarkan kolom "nama" secara ascending (ASC).

Selanjutnya, terdapat contoh penggunaan klausa ORDER BY, GROUP BY, dan HAVING dalam sebuah perintah. Perintah tersebut menggabungkan data dari tabel "departemen" dan "karyawan" menggunakan JOIN, kemudian mengelompokkan data berdasarkan id_departemen menggunakan GROUP BY, dan menyaring hasil dengan jumlah karyawan lebih dari 1 menggunakan HAVING. Hasilnya kemudian diurutkan berdasarkan jumlah karyawan dalam urutan menurun (DESC).

Selain itu, terdapat perintah yang menggunakan klausa WHERE untuk menyaring data berdasarkan rentang tanggal tertentu. Misalnya, perintah SELECT * FROM absensi WHERE tanggal BETWEEN '2024-03-01' AND '2024-03-10'; akan menampilkan data absensi yang tanggalnya berada dalam rentang antara 1 Maret 2024 dan 10 Maret 2024.

Kemudian, ada perintah yang menghitung jumlah kemunculan data tertentu dalam sebuah tabel menggunakan GROUP BY, dan kemudian mengurutkan hasilnya untuk menemukan data yang paling banyak muncul. Sebaliknya, terdapat juga perintah yang mencari data yang paling sedikit muncul dengan mengurutkan hasil secara menaik (ASC) dan membatasi jumlah hasil yang ditampilkan dengan LIMIT 1.

Selain itu, ada juga contoh penggunaan klausa WHERE untuk mencari data yang

memenuhi suatu pola tertentu. Misalnya, perintah SELECT * FROM karyawan WHERE nama LIKE 'A%'; akan menampilkan data karyawan di mana nama mereka dimulai dengan huruf 'A'.

Terakhir, ada perintah yang menggabungkan data dari dua tabel menggunakan LEFT JOIN dan menampilkan data karyawan beserta data absensi yang sesuai. Ini memberikan gambaran menyeluruh tentang bagaimana klausa-klausa SQL dapat digunakan untuk memanipulasi dan menganalisis data dari berbagai sudut pandang.

BAB V

PENUTUP

5.1 Analisa

Dari hasil praktikum yang telah dilakukan, praktikan berhasil menganalisis bahwa implementasi konsep dan teknologi dalam Sistem Manajemen Basis Data (SMBD) memiliki dampak yang signifikan terhadap efisiensi dan efektivitas pengelolaan data dalam suatu organisasi. Berdasarkan praktikum tersebut, beberapa temuan kunci telah diidentifikasi:

- **Peningkatan Efisiensi Operasional**: Penggunaan SMBD memungkinkan organisasi untuk menyimpan, mengambil, dan memperbarui data dengan lebih cepat dan efisien. Hal ini membantu dalam meningkatkan produktivitas dan mengurangi biaya operasional terkait pengelolaan data.

- **Ketepatan dan Konsistensi Data**: Dengan adanya kontrol keamanan dan integritas data yang kuat dalam SMBD, praktikan menemukan bahwa data yang disimpan dalam basis data lebih akurat dan konsisten. Ini memberikan kepercayaan yang lebih tinggi terhadap informasi yang dihasilkan dari sistem.

- **Fleksibilitas dan Skalabilitas**: SMBD memungkinkan organisasi untuk menyesuaikan basis data mereka dengan kebutuhan yang berkembang, serta dapat dengan mudah memperluas kapasitas penyimpanan saat diperlukan. Hal ini menjadi penting dalam menghadapi pertumbuhan volume data yang cepat.

5.2 Kesimpulan

Pentingnya Konsep dan Teknologi SMBD: Mata kuliah ini telah membuka wawasan mahasiswa tentang pentingnya SMBD dalam konteks bisnis dan teknologi saat ini. Pemahaman mendalam tentang konsep seperti model data, bahasa SQL, dan teknik pengelolaan transaksi merupakan fondasi yang kuat untuk karier di berbagai bidang terkait teknologi informasi.

Dampak Positif Implementasi SMBD: Implementasi SMBD dapat memberikan dampak positif yang signifikan terhadap efisiensi, keakuratan, dan fleksibilitas dalam pengelolaan data suatu organisasi. Hal ini menggarisbawahi pentingnya investasi dalam infrastruktur data yang baik.