LAPORAN RESMI

MODUL II

VIEW

NAMA	**:**	**ANISYAFAAH**
N.R.P	**:**	**220441100105**
DOSEN	**:**	**FITRI DAMAYANTI, S.Kom., M.Kom.**
ASISTEN	**:**	**AFFAN MAULANA ZULKARNAIN**
TGL PRAKTIKUM	**:**	**29 MARET 2024**

Disetujui : April 2024
Asisten

AFFAN MAULANA ZULKARNAIN
20.04.411.00052

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin berkembang, penggunaan database telah menjadi suatu keharusan bagi berbagai organisasi dan perusahaan. Pertumbuhan data yang cepat dan kompleksitas informasi yang semakin tinggi membuat pentingnya pengelolaan data secara efisien. Pertama, database memungkinkan penyimpanan data dalam format terstruktur, yang memudahkan pengelolaan dan pengambilan informasi. Dengan menggunakan database, informasi dapat disimpan secara teratur dalam tabel dan kolom, memfasilitasi proses pencarian dan analisis data dengan lebih mudah dan cepat.

Selain itu, implementasi database juga memberikan keuntungan dalam hal konsistensi dan integritas data. Dengan aturan yang ditetapkan dalam basis data, seperti constraint dan relasi antartabel, kesalahan dalam input data dapat diminimalkan. Hal ini penting untuk menjaga keakuratan informasi dan mencegah terjadinya inkonsistensi yang dapat merugikan dalam pengambilan keputusan. Dengan demikian, penggunaan database membantu organisasi untuk menjaga kualitas dan keandalan data mereka.

Dalam menggunakan sebuah database, kita pasti perlu menggunakan view. Penggunaan view dalam database memberikan fleksibilitas dalam pengelolaan dan presentasi data. View memungkinkan pengguna untuk membuat tampilan virtual dari satu atau beberapa tabel, yang dapat disesuaikan dengan kebutuhan pengguna tertentu. Dengan menggunakan view, pengguna dapat mengakses dan menganalisis data tanpa harus mengetahui struktur tabel yang kompleks di baliknya.

1.2 Tujuan

- Mampu memahami konsep dasar view di dalam basis data
- Mampu memahami penerapan view
- Mampu menyelesaikan pengambilan data dengan menggunakan pendekatan view

BAB II
DASAR TEORI

2.1 View

View adalah tabel virtual yang terbuat dari suatu query terhadap satu tabel atau beberapa tabel. View tidak ada secara nyata dalam database. View hanya digunakan untuk menyederhanakan dan mempermudah persepsi pengguna dalam database. Tidak seperti pada umumnya tabel di dalam basis data relasional, view bukanlah bagian dari skema fisik. View bersifat dinamis, ia mengandung data dari tabel yang direpresentasikannya. Dengan demikian, ketika tabel yang menjadi sumber datanya berubah, data di view juga akan berubah.

Kegunaan view antara lain:

1. Memfokuskan pada data tertentu
2. Penyederhanaan manipulasi data
3. Menyesuaikan data dengan kebutuhan user
4. Export dan impor data
5. Mengkombinasikan data terpartisi

Syntax

```
CREATE VIEW view_name [(column[,..n])] [with encryption]
AS select_statement [with check option]
```

Contoh : Buatlah view untuk membuat daftar seluruh nama kota yang ada dalam tabel PLAYERS!

```
CREATE VIEW TOWNS (TOWN) AS
SELECT DISTINCT TOWN
FROM PLAYERS
```

2.2 Updatable View

View dapat berisi read-only atau updatable. Kondisi ini sangat dipengaruhi oleh adanya pendefinisian view itu sendiri. Bagaimanapun, untuk menciptakan updatable view, pernyataan SELECT yang didefinisikan di view harus mengikuti aturan-aturan sebagai berikut:

- Pernyataan SELECT tidak boleh merujuk ke lebih dari satu tabel.

- Pernyataan SELECT tidak boleh menggunakan klausa GROUP BY atau HAVING.
- Pernyataan SELECT harus tidak menggunakan DISTINCT.
- Pernyataan SELECT harus tidak merujuk ke view lain yang tidak updatable.
- Pernyataan SELECT tidak boleh mengandung ekspresi apa pun, misalnya fungsi agregat.

Pada hakikatnya, jika sistem database mampu menentukan pemetaan balik dari skema view ke skema tabel dasar, maka view memungkinkan untuk di update. Dalam kondisi ini, operasi-operasi INSERT, UPDATE dan DELETE dapat diterapkan pada view.

2.3 Hapus View

Suatu view dari query selalu menampilkan data yang terbaru, sebagai contoh bila kita memodifikasi sebagian tupel dalam tabel dasarnya dimana view tersebut didefinisikan maka secara otomatis akan berpengaruh pada view di query. Jika tidak membutuhkan view lagi, kita bisa menggunakan perintah :

Syntax

```
DROP VIEW view_name
```

BAB III
TUGAS PENDAHULUAN

3.1 Soal

1. Apa itu view dalam konteks database dan apa kegunaannya?
2. Jelaskan query pembuatan view dalam sebuah database?
3. Apa perbedaan antara view dan tabel dalam database?
4. Mengapa penggunaan view dapat membantu dalam manajemen data dan pengembangan aplikasi database?
5. Apa saja keuntungan dan kerugian menggunakan view dalam pengembangan aplikasi database?

3.2 Jawab

1. View adalah tabel virtual yang dibuat dari suatu query terhadap satu atau beberapa tabel. View tidak ada nyata dalam database dan hanya digunakan untuk menyederhanakan dan mempermudah persepsi pengguna dalam database. Kegunaan view antara lain:
   a) Memfokuskan pada data tertentu
   b) Penyederhanaan manipulasi data
   c) Mengkombinasikan data terpartisi
2. Untuk membuat view, secara syntax sama seperti SELECT statement, namun hanya ditambahkan fungsi CREATE VIEW pada awal syntax. Untuk penulisan syntaxnya yaitu:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
```

3. Perbedaan View dan tabel dalam database
   a) View
      View merupakan representasi atau tabel virtual yang terbentuk dari hasil query dan digunakan untuk menyajikan data - data dari tabel yang sudah ada.
   b) Tabel
      Tabel merupakan struktur dasar dalam database yang

digunakan untuk menyimpan data. Data - data yang disimpan merupakan milik tabel tersebut dan bukan sebuah representasi atau tabel virtual.

4. View merupakan tampilan virtual dari subset data yang ada. Dengan menggunakan view, pengembang dapat menyembunyikan kompleksitas struktur database yang mendasarinya, menyediakan tingkat abstraksi yang lebih tinggi, dan memudahkan proses pengambilan data yang konsisten dan efisien.

5. Keuntungan dan kerugian View dalam pengembangan aplikasi database

+ a) Membatasi akses data
  Dengan view, pengguna dapat mengatur izin akses untuk kolom tertentu.
+ b) Mengoptimalkan kinerja
  View dapat membantu dalam mengoptimalkan kinerja database dengan menyediakan cara untuk menyimpan query yang kompleks.
+ c) Proses penampilan data yang lebih cepat
  Dengan menggunakan view, kita tidak perlu run dengan block tabel, namun cukup memanggil fungsi view untuk melihat data.
- d) Performa yang agak kurang
  Query terhadap view dapat menjadi lambat jika view tersebut dibuat menggunakan view yang lain.
- e) Tidak cocok digunakan pada query tunggal
- f) Table dependecies
  Jika tabel berubah, maka kita juga harus mengubah viewnya.

BAB IV

IMPLEMENTASI

4.1 Source Code

4.1.1 Mengisi Data Setiap Tabel

- **Tabel Pelanggan**

a) Source Code

```
CREATE DATABASE db_penjualan;

USE db_penjualan;

CREATE TABLE Pelanggan (
   ID_Pelanggan INT (11) PRIMARY KEY,
    Nama_Pelanggan VARCHAR (100),
    Email VARCHAR (50),
    Alamat VARCHAR (255)
    );

INSERT INTO Pelanggan VALUES (01, "Anisyafaah", "anisyafaah@gmail.com", "Bangkalan");
INSERT INTO Pelanggan VALUES (02, "Herdiyanti Fifin Purwaningrum", "herdiyantififin@gmail.com", "Gresik");
INSERT INTO Pelanggan VALUES (03, "Anisah Nuril Izzati", "anisahnuril@gmail.com", "Bangkalan");
INSERT INTO Pelanggan VALUES (04, "Adhelia Kusumawati", "adheliakusumawati@gmail.com", "Gresik");
INSERT INTO Pelanggan VALUES (05, "Firdausi Putri Cahyani", "firdausiputri@gmail.com", "Surabaya");
INSERT INTO Pelanggan VALUES (06, "Rayhanza Nadhif Athala", "rayhanzanadhif@gmail.com", "Bangkalan");
INSERT INTO Pelanggan VALUES (07, "Birrur Rijaal", "birrurrijaal@gmail.com", "Gresik");
INSERT INTO Pelanggan VALUES (08, "Juanzha Nanda Pratama", "juanzhananda@gmail.com", "Gresik");
INSERT INTO Pelanggan VALUES (09, "Erick Firmansyah", "erickfirmansyah@gmail.com", "Surabaya");
INSERT INTO Pelanggan VALUES (010, "Syahrul Ramadhani", "syahrulramadhani@gmail.com", "Surabaya");
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel pelanggan dan mengisi data sebanyak 10 data. Untuk membuat tabel pelanggan menggunakan perintah CREATE TABLE Pelanggan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id_pelanggan sebagai primary key, nama pelanggan, email pelanggan, dan alamat pelanggan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

- **Tabel Pesanan**

a) Source Code

```
CREATE TABLE Pesanan (
   ID_Pesanan INT (11) PRIMARY KEY,
    ID_Pelanggan INT (11),
    Tanggal_Pesanan DATE,
    Total INT (11),
    FOREIGN KEY (ID_Pelanggan) REFERENCES Pelanggan (ID_Pelanggan)
);

INSERT INTO Pesanan VALUES (1001, 01, "2024-04-04", 15000);
INSERT INTO Pesanan VALUES (1002, 02, "2024-04-03", 15000);
INSERT INTO Pesanan VALUES (1003, 03, "2024-04-02", 18000);
INSERT INTO Pesanan VALUES (1004, 04, "2024-04-01", 4000);
INSERT INTO Pesanan VALUES (1005, 05, "2024-03-31", 42000);
INSERT INTO Pesanan VALUES (1006, 06, "2024-03-30", 8000);
INSERT INTO Pesanan VALUES (1007, 07, "2024-03-29", 8000);
INSERT INTO Pesanan VALUES (1008, 08, "2024-03-28", 24000);
INSERT INTO Pesanan VALUES (1009, 09, "2024-03-27", 25000);
INSERT INTO Pesanan VALUES (1010, 010, "2024-03-26", 30000);
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel pesanan dan mengisi data sebanyak 10 data. Untuk membuat tabel pesanan menggunakan perintah CREATE TABLE Pesanan (namaKolom1,

namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id_pesanan sebagai primary key, id_pelanggan yang terhubung dengan tabel pelanggan menggunakan perintah FOREIGN KEY, tanggal pesanan, dan total harga pesanan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO**.**

- **Tabel Produk**

a) Source Code

```
CREATE TABLE Produk (
        ID_Produk INT (11) PRIMARY KEY,
   Nama_Produk VARCHAR (100),
   Harga INT (11),
   Stok INT (11)
   );

INSERT INTO Produk VALUES (111, "Susu Ultramilk", 5000, 4);
INSERT INTO Produk VALUES (112, "Richeese Nabati", 3000, 10);
INSERT INTO Produk VALUES (113, "Wafer Tango", 6000, 8);
INSERT INTO Produk VALUES (114, "Nabati Rolls", 2000, 15);
INSERT INTO Produk VALUES (115, "Susu Indomilk", 7000, 3);
INSERT INTO Produk VALUES (116, "Sosis So Nice", 4000, 6);
INSERT INTO Produk VALUES (117, "Richoco", 2000, 11);
INSERT INTO Produk VALUES (118, "Roma Malkist", 8000, 7);
INSERT INTO Produk VALUES (119, "Qtela Singkong", 8000, 10);
INSERT INTO Produk VALUES (120, "Sari Gandum", 5000, 13);
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel produk dan mengisi data sebanyak 10 data. Untuk membuat tabel produk menggunakan perintah CREATE TABLE Produk (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id_produk sebagai primary key, nama produk, harga produk, dan stok produk. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO**.**

- **Tabel Detail Pesanan**

a) Source Code

```
CREATE TABLE Detail_Pesanan (
   ID_Detail INT (11) PRIMARY KEY,
    ID_Pesanan INT (11),
    ID_Produk INT (11),
    Jumlah INT (11),
    FOREIGN KEY (ID_Pesanan) REFERENCES Pesanan (ID_Pesanan),
    FOREIGN KEY (ID_Produk) REFERENCES Produk (ID_Produk)
    );


INSERT INTO Detail_Pesanan VALUES (121, 1001, 111, 3);
INSERT INTO Detail_Pesanan VALUES (122, 1002, 112, 5);
INSERT INTO Detail_Pesanan VALUES (123, 1003, 113, 3);
INSERT INTO Detail_Pesanan VALUES (124, 1004, 114, 2);
INSERT INTO Detail_Pesanan VALUES (125, 1005, 115, 2);
INSERT INTO Detail_Pesanan VALUES (126, 1006, 116, 2);
INSERT INTO Detail_Pesanan VALUES (127, 1007, 117, 4);
INSERT INTO Detail_Pesanan VALUES (128, 1008, 118, 3);
INSERT INTO Detail_Pesanan VALUES (129, 1009, 119, 5);
INSERT INTO Detail_Pesanan VALUES (130, 1010, 120, 6);
```

b) Penjelasan

Kode di atas digunakan untuk membuat tabel detail pesanan dan mengisi data sebanyak 10 data. Untuk membuat tabel pesanan menggunakan perintah CREATE TABLE Detail_Pesanan (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id_detail sebagai primary key, id_pesanan yang terhubung dengan tabel pesanan menggunakan perintah FOREIGN KEY, id_produk yang terhubung dengan tabel produk menggunakan perintah FOREIGN KEY, dan jumlah pesanan. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO**.**

4.1.2 View Data Pesanan yang Lebih dari Rata-Rata

a) Source Code

```
-- Nomor 1
SELECT AVG(Total) FROM Pesanan;
CREATE VIEW Pemesanan AS
SELECT Pelanggan.Nama_Pelanggan, Pesanan.Total AS
Total_Harga_Pesanan, Pesanan.Tanggal_Pesanan
FROM Pelanggan JOIN Pesanan ON Pelanggan.ID_Pelanggan =
Pesanan.ID_Pelanggan
WHERE Pesanan.Total > (SELECT AVG(Total) FROM Pesanan);


SELECT * FROM Pemesanan;
```

b) Penjelasan

Kode di atas digunakan untuk membuat view. Karena yang ingin ditampilkan adalah pesanan yang lebih dari rata-rata, maka membutuhkan perintah AVG (Average) dari kolom total pada tabel pesanan. Selanjutnya membuat view dengan perintah CREATE VIEW nama_view AS statement SELECTnya. View ini membutuhkan dua tabel yaitu pelanggan (nama pelanggan) dan pesanan (total dan tanggal pesanan), sehingga perlu menggunakan JOIN ON pada kedua tabel yang saling berhubungan dimana WHERE (kondisinya) berupa perintah AVG yang sebelumnya.

4.1.3 View Data Produk yang Terjual Setiap Pesanan

a) Source Code

```
-- Nomor 2
CREATE VIEW Total_Pendapatan_Produk AS
SELECT Produk.Nama_Produk, Produk.Harga AS Harga_Satuan,
Detail_Pesanan.Jumlah AS Jumlah_Produk_Terjual, (Produk.Harga *
Detail_Pesanan.Jumlah) AS Total_Pendapatan
FROM Produk JOIN Detail_Pesanan ON Produk.ID_Produk =
Detail_Pesanan.ID_Produk;


SELECT * FROM Total_Pendapatan_Produk;
```

b) Penjelasan

Kode view di atas digunakan untuk menampilkan total pendapatan pada keseluruhan produk. Kode ini dimulai dengan perintah membuat view yaitu CREATE VIEW nama_view AS statement SELECTnya. View ini membutuhkan dua tabel yaitu produk (nama produk dan harga produk) dan tabel detail pesanan (jumlah), sehingga perlu menggunakan JOIN ON pada kedua tabel yang saling berhubungan. Karena ingin menampilkan total pendapatan setiap produk, maka perlu rumus total harga pada tabel pesanan * jumlah pesanan pada tabel detail pesanan. Rumus ini dianggap sebagai total pendapatannya (AS total_pendapatan).

4.1.4 View Data Stok yang Kurang dari 5

a) Source Code

```
-- Nomor 3
CREATE VIEW Sisa_Stok AS
SELECT Produk.Nama_Produk, Produk.Stok AS Jumlah_Stok FROM Produk
WHERE Stok < 5;


SELECT * FROM Sisa_Stok;
```

b) Penjelasan

Kode view di atas digunakan untuk menampilkan produk-produk dengan stok yang kurang dari 5. Untuk membuatnya maka memerlukan perintah CREATE VIEW nama_view AS statement SELECTnya. View ini membutuhkan satu tabel saja yaitu tabel produk dari kolom nama produk dan stok dimana WHERE (kondisinya) diatur < 5 (kurang dari 5).

4.1.5 View Data Total Pesanan Sebulan Terakhir

a) Source Code

```
-- Nomor 4
CREATE VIEW Jumlah_Pesanan_Sebulan AS
SELECT Pelanggan.Nama_Pelanggan, COUNT(Pesanan.ID_Pesanan) AS
Total_pesanan
FROM Pelanggan
JOIN Pesanan ON Pelanggan.ID_Pelanggan = Pesanan.ID_Pelanggan
```

```
WHERE Pesanan.Tanggal_Pesanan BETWEEN
DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AND
CURRENT_DATE()
GROUP BY Pelanggan.ID_Pelanggan;

SELECT * FROM Jumlah_Pesanan_Sebulan;
```

b) Penjelasan

Kode view di atas digunakan untuk menampilkan jumlah pesanan setiap pelanggan selama periode satu bulan terakhir. Kode ini dimulai dengan perintah membuat view yaitu CREATE VIEW nama_view AS statement SELECTnya. View ini membutuhkan perintah COUNT (jumlah) id pesanan pada tabel pesanan sebagai total pesanannya. View ini juga membutuhkan dua tabel yaitu pelanggan (nama pelanggan), dan tabel pesanan (jumlah id pesanan), sehingga perlu menggunakan JOIN ON pada kedua tabel yang saling berhubungan. Karena ingin menampilkan total pesanan setiap pelanggan selama satu bulan terakhir, maka perlu sebuah kondisi (WHERE) Pesanan.Tanggal_Pesanan BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AND CURRENT_DATE(). Perintah ini akan mengambil pesanan yang memiliki tanggal pemesanan antara satu bulan sebelum tanggal saat ini dan tanggal saat ini. Kemudian kode GROUP BY Pelanggan.ID_Pelanggan untuk mengelompokkan hasil berdasarkan ID_Pelanggan untuk menghitung jumlah pesanan yang dilakukan oleh setiap pelanggan secara terpisah.

4.2 Hasil

4.2.1 Data Setiap Tabel

- **Tabel Pelanggan**

ID_Pelanggan	Nama_Pelanggan	Email	Alamat
1	Anisyafaah	anisyafaah@gmail.com	Bangkalan
2	Herdiyanti Fifin Purwaningrum	herdiyantififin@gmail.com	Gresik
3	Anisah Nuril Izzati	anisahnuril@gmail.com	Bangkalan
4	Adhelia Kusumawati	adheliakusumawati@gmail.com	Gresik
5	Firdausi Putri Cahyani	firdausiputri@gmail.com	Surabaya
6	Rayhanza Nadhif Athala	rayhanzanadhif@gmail.com	Bangkalan
7	Birrur Rijaal	birrurrijaal@gmail.com	Gresik
8	Juanzha Nanda Pratama	juanzhananda@gmail.com	Gresik
9	Erick Firmansyah	erickfirmansyah@gmail.com	Surabaya
10	Syahrul Ramadhani	syahrulramadhani@gmail.com	Surabaya
NULL	NULL	NULL	NULL

- **Tabel Pesanan**

ID_Pesanan	ID_Pelanggan	Tanggal_Pesanan	Total
1001	1	2024-04-01	15000
1002	2	2024-03-26	15000
1003	3	2024-03-20	18000
1004	4	2024-03-18	4000
1005	5	2024-03-15	42000
1006	6	2024-03-12	8000
1007	7	2024-03-09	8000
1008	8	2024-03-07	24000
1009	9	2024-03-04	25000
1010	10	2024-03-01	30000
NULL	NULL	NULL	NULL

- **Tabel Produk**

ID_Produk	Nama_Produk	Harga	Stok
111	Susu Ultramilk	5000	4
112	Richeese Nabati	3000	10
113	Wafer Tango	6000	8
114	Nabati Rolls	2000	15
115	Susu Indomilk	7000	3
116	Sosis So Nice	4000	6
117	Richoco	2000	11
118	Roma Malkist	8000	7
119	Qtela Singkong	8000	10
120	Sari Gandum	5000	13
NULL	NULL	NULL	NULL

- **Tabel Detail Pesanan**

ID_Detail	ID_Pesanan	ID_Produk	Jumlah
121	1001	111	3
122	1002	112	5
123	1003	113	3
124	1004	114	2
125	1005	115	2
126	1006	116	2
127	1007	117	4
128	1008	118	3
129	1009	119	5
130	1010	120	6
NULL	NULL	NULL	NULL

4.2.2 Data Pesanan yang Lebih dari Rata-Rata

Nama_Pelanggan	Total_Harga_Pesanan	Tanggal_Pesanan
Firdausi Putri Cahyani	42000	2024-03-31
Juanzha Nanda Pratama	24000	2024-03-28
Erick Firmansyah	25000	2024-03-27
Syahrul Ramadhani	30000	2024-03-26

4.2.3 Data Produk yang Terjual Setiap Pesanan

Nama_Produk	Harga_Satuan	Jumlah_Produk_Terjual	Total_Pendapatan
Susu Ultramilk	5000	3	15000
Richeese Nabati	3000	5	15000
Wafer Tango	6000	3	18000
Nabati Rolls	2000	2	4000
Susu Indomilk	7000	2	14000
Sosis So Nice	4000	2	8000
Richoco	2000	4	8000
Roma Malkist	8000	3	24000
Qtela Singkong	8000	5	40000
Sari Gandum	5000	6	30000

4.2.4 Data Stok yang Kurang dari 5

Nama_Produk	Jumlah_Stok
Susu Ultramilk	4
Susu Indomilk	3

4.1.6 Data Total Pesanan Sebulan Terakhir

	Nama_Pelanggan	Total_pesanan
▶	Anisyafaah	1
	Herdiyanti Fifin Purwaningrum	1
	Anisah Nuril Izzati	1
	Adhelia Kusumawati	1
	Firdausi Putri Cahyani	1
	Rayhanza Nadhif Athala	1
	Birrur Rijaal	1
	Juanzha Nanda Pratama	1
	Erick Firmansyah	1
	Syahrul Ramadhani	1

BAB V
PENUTUP

5.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan view dalam sebuah database memiliki beragam manfaat yang signifikan. Pertama, view memungkinkan pengguna untuk mengakses dan menganalisis data dengan lebih efisien. Dengan menggunakan view, pengguna dapat membuat tampilan virtual dari data yang relevan, tanpa harus mengetahui struktur tabel yang kompleks di baliknya. Hal ini memudahkan pengguna dalam melakukan query dan analisis data, karena mereka dapat fokus pada informasi yang benar-benar mereka perlukan. Selain itu, view juga dapat membantu dalam melindungi keamanan data dengan membatasi akses pengguna hanya pada informasi yang relevan.

Kedua, penggunaan view juga meningkatkan fleksibilitas dalam pengelolaan dan presentasi data. Sebuah view dapat dibuat dari satu atau beberapa tabel, serta dapat mencakup kolom-kolom tertentu atau menyertakan kriteria tertentu dalam pemilihan data. Hal ini memungkinkan pengguna untuk menyesuaikan tampilan data sesuai dengan kebutuhan mereka tanpa mengganggu struktur data asli dalam database. Dengan demikian, implementasi view tidak hanya meningkatkan efisiensi dalam pengambilan keputusan, tetapi juga memungkinkan pengguna untuk menyesuaikan tampilan data sesuai dengan kebutuhan spesifik mereka.

5.2 Kesimpulan

1. View adalah tabel virtual yang terbuat dari suatu query terhadap satu tabel atau beberapa tabel yang digunakan untuk menyederhanakan dan mempermudah persepsi pengguna dalam database.
2. Jika sistem database mampu menentukan pemetaan balik dari skema view ke skema tabel dasar, maka view memungkinkan untuk di update. Dalam kondisi ini, operasi-operasi INSERT, UPDATE dan DELETE dapat diterapkan pada view.
3. View dapat dihapus jika sudah tidak dibutuhkan lagi dengan cara menggunakan perintah DROP VIEW view_name.