

LAPORAN PRAKTIKUM
PRAKTIKUM 5:
“RESPONSI”



Disusun Oleh :

Oktaviana Sadama Nur Azizah
24060121130060

PRAKTIKUM MANAJEMEN BASIS DATA
LAB A2

DEPARTEMEN ILMU KOMPUTER / INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
SEMARANG
2023

BAB I

PENDAHULUAN

A. Rumusan Masalah

1. Di dalam database RESPONSI terdapat dua buah tabel, yaitu:
 - a. PRODUK (NoProduk, NamaProduk, Kategori, Warna, Stok, StokMin, HargaPokok, HargaJual).
 - b. JUAL (NoFaktur, NoProduk, TglFaktur, Jumlah).
2. Buat trigger untuk:
 - a. Mengupdate nilai harga jual menjadi sebesar harga pokok + 5% * harga pokok.
 - b. Mengupdate nilai stok pada tabel produk ketika produk tersebut berhasil dijual dan sekaligus menampilkan pesan ketika mengalami kondisi stok dibawah stok minimum.
3. Buat Store Procedure untuk:
 - a. Menampilkan periode waktu (bulan/ tahun), total pendapatan, dan total laba per bulan.
 - b. Menampilkan data produk terjual (NoProduk, NamaProduk, JumlahTerjual) per masing-masing produk berdasarkan inputan periode waktu tertentu.
4. Berdasarkan database di atas, kebutuhan sistem yang sedang dikembangkan antara lain meliputi:

Pada kedua tabel di atas sering dikenai perintah SQL seperti:

```
SELECT Produk.NoProduk, NamaProduk  
FROM Produk, Jual  
WHERE Produk.NoProduk = Jual.NoProduk  
AND Kategori = 'Alat Rumah Tangga' ..... (query 1)
```

Selain hal di atas pada tabel-tabel tersebut juga sering dikenakan perintah SQL untuk menampilkan data produk yang terjual untuk suatu periode waktu tertentu.

 - a. Tuliskan indeks-indeks yang telah dibuat dalam database RESPONSI! Jelaskan!
 - b. Tentukan indeks-indeks tambahan yang diperlukan, field yang digunakan sebagai search key, tipe indeks, dan jelaskan untuk masing-masing indeks tersebut!
 - c. Bagaimana query optimizer mengeksekusi perintah SQL yang disebutkan di atas (query 1), setelah ditambahkan indeks? Jelaskan!
 - d. Implementasikan indeks yang sudah anda desain pada soal b!
5. Gunakan fasilitas yang ada pada SQL Server untuk menyimpan:
 - a. Data trigger dan indeks yang dilampirkan pada tabel dengan nama ***“trigger_index.sql”***
 - b. Data stored procedure yang dibuat pada database anda dengan nama ***“stor_proc.sql”***

B. Tujuan

1. Mahasiswa mampu mengimplementasikan trigger di dalam database.
2. Mahasiswa mampu mengimplementasikan store procedure di dalam database.
3. Mahasiswa mampu mengimplementasikan indeksing di dalam database.

BAB II DASAR TEORI

A. Trigger

Dalam Database Management System (DBMS), trigger merupakan kumpulan script yang berhubungan dengan table, view, ataupun schema yang dijalankan secara otomatis ketika terdapat event yang dijalankan. Event tersebut meliputi operasi yang biasa dilakukan dalam mengolah database, seperti:

1. Data Manipulation Language (DML) yang meliputi DELETE, INSERT, atau UPDATE.
2. Data Definition Language (DDL) yang meliputi CREATE, ALTER, atau DROP.
3. Operasi database lainnya, seperti SERVERERROR, LOGON, LOGOFF, STARTUP, atau SHUTDOWN.

Secara sederhana, perintah untuk membuat trigger adalah sebagai berikut:

```
CREATE TRIGGER nama_trigger
  ON nama_table
  [ BEFORE | AFTER ] [ INSERT | UPDATE | DELETE ]
AS
BEGIN
  // trigger body
END;
```

Keterangan:

1. Nama_trigger
Nama trigger yang dibuat sesuai dengan karakteristik penamaan.
2. Nama_table
Menunjukkan table yang akan dilakukan trigger didalamnya.
3. [BEFORE | AFTER]
Menunjukkan waktu untuk mengeksekusi trigger secara otomatis, apakah sebelum atau sesudah perubahan pada row data table. Sehingga pilihannya adalah AFTER atau BEFORE.
4. [INSERT | UPDATE | DELETE]
Digunakan untuk menentukan event yang menyebabkan terjadinya trigger, pilihan event tersebut terdiri dari INSERT, UPDATE, dan DELETE.
5. Trigger_body
Menunjukkan statement perintah yang akan otomatis dijalankan jika event sedang aktif.

B. Store Procedure

Stored Procedure adalah sebuah fungsi yang berisi kode SQL yang dapat digunakan kembali dengan cara memanggil atau *execute* Stored Procedure yang telah dibuat. Dalam

Stored Procedure juga dapat dimasukkan parameter sehingga fungsi dapat digunakan lebih dinamis berdasarkan parameter tersebut.

Stored Procedure Syntax

```
CREATE PROCEDURE nama_prosedur  
AS  
BEGIN  
    SQL_statement  
END ;
```

Pemanggilan Stored Procedure

```
EXEC nama_prosedur;
```

Adapun kelebihan menggunakan Stored Procedure adalah sebagai berikut:

1. Berbagi logik dengan aplikasi lainnya
Stored Procedure merangkum fungsionalitas untuk memastikan akses data dan manipulasi koheren antara aplikasi yang berbeda.
2. Keamanan
Stored Procedure mengisolasi pengguna dari tabel data. Fitur ini memberi kemampuan untuk memberikan akses ke Stored Procedure yang memanipulasi data namun tidak secara langsung ke tabel.
3. Performa
Stored Procedure dikompilasi yang selanjutnya dicache dan akan digunakan kembali. Dengan begitu waktu respon dan performa menjadi lebih cepat karena Stored Procedure yang sama dieksekusi kembali.

Selain kelebihan Stored Procedure yang sudah disebutkan diatas, terdapat pula kekurangan dari Stored Procedure sebagai berikut:

1. Peningkatan beban pada database server. Sebagian besar pekerjaan dilakukan di sisi server dan kurang di sisi client.
2. Pengulangan logika aplikasi di dua tempat yang berbeda: kode bahasa pemrograman dan kode Stored Procedure membuat pemeliharaan aplikasi menjadi lebih sulit. Migrasi ke sistem manajemen basis data yang berbeda berpotensi menjadi lebih sulit. Tidak membutuhkan ruang disk tambahan.

C. Indeks

Indeks adalah kunci yang dibuat dari satu atau beberapa kolom dalam database yang berguna untuk mempercepat pengambilan baris dalam tabel atau tampilan. Kunci ini membantu database untuk menemukan baris yang terkait dengan nilai kunci dengan cepat.

Adapun dua tipe indeks dalam database adalah sebagai berikut.

1. Clustered Indeks

Clustered indeks atau yang biasa disebut indeks berkelompok atau berkerumun adalah jenis indeks yang mengurutkan baris data dalam tabel berdasarkan nilai kuncinya. Di database, hanya diperbolehkan terdapat satu clustered indeks per tabel. Adapun ciri dari clustered indeks sebagai berikut.

- Menentukan urutan penyimpanan baris dalam tabel secara keseluruhan.
- Hanya diperbolehkan ada satu clustered indeks dalam satu tabel.
- Akses data lebih cepat.
- Tidak membutuhkan ruang disk tambahan.

2. Non Clustered Indeks

Non clustered indeks menyimpan data di satu lokasi dan indeks di lokasi lain. Indeks ini berisi pointer ke lokasi data tersebut. Satu tabel dapat memiliki banyak non clustered indeks karena indeks disimpan di tempat yang berbeda. Adapun ciri dari non clustered indeks sebagai berikut.

- Menentukan urutan penyimpanan baris dalam tabel dengan bantuan struktur fisik yang terpisah.
- Diperbolehkan terdapat lebih dari satu non clustered indeks dalam satu tabel.
- Akses data lebih lambat dibandingkan dengan clustered indeks.
- Memerlukan ruang disk tambahan untuk menyimpan indeks secara terpisah.

Berikut adalah syntax yang digunakan untuk mengimplementasikan indeks dalam SQL Server.

```
USE database_name

CREATE index_type INDEX index_name
ON table_name (column_name)
```

BAB III PEMBAHASAN

1. Di dalam database RESPONSI terdapat dua buah tabel, yaitu:

- a. PRODUK (NoProduk, NamaProduk, Kategori, Warna, Stok, StokMin, HargaPokok, HargaJual).

	Column Name	Data Type	Allow Nulls
PK	NoProduk	int	<input type="checkbox"/>
	NamaProduk	varchar(50)	<input checked="" type="checkbox"/>
	Kategori	varchar(50)	<input checked="" type="checkbox"/>
	Warna	varchar(50)	<input checked="" type="checkbox"/>
	Stok	int	<input checked="" type="checkbox"/>
	StokMin	int	<input checked="" type="checkbox"/>
	HargaPokok	int	<input checked="" type="checkbox"/>
	HargaJual	int	<input checked="" type="checkbox"/>

Tabel PRODUK dibuat dimana kolom NoProduk diberi tipe data INT dan di set sebagai Primary Key, kolom NamaProduk diberi tipe data VARCHAR(50), kolom Kategori diberi tipe data VARCHAR(50), kolom Warna diberi tipe data VARCHAR(50), kolom Stok diberi tipe data INT, kolom StokMin diberi tipe data INT, tipe HargaPokok diberi tipe data INT, sedangkan kolom HargaJual diberi tipe data INT. Berikut tabel PRODUK yang sudah isi data.

Results Messages								
	NoProduk	NamaProduk	Kategori	Warna	Stok	StokMin	HargaPokok	HargaJual
1	1	Kaos Polos	Pakaian	Putih	100	10	49000	79999
2	2	Sweater Rajut	Pakaian	Lilac	120	10	79000	129999
3	3	Celana Jeans Slim Fit	Celana	Denim	50	5	120000	299999
4	4	Sepatu Sneakers	Sepatu	Hitam	20	5	195000	349999
5	5	Topi Baseball	Aksesoris	Abu-abu	50	5	29000	59999

- b. JUAL (NoFaktur, NoProduk, TglFaktur, Jumlah).

	Column Name	Data Type	Allow Nulls
PK	NoFaktur	char(3)	<input type="checkbox"/>
	NoProduk	int	<input checked="" type="checkbox"/>
	TglFaktur	date	<input checked="" type="checkbox"/>
	Jumlah	int	<input checked="" type="checkbox"/>

Tabel JUAL dibuat dimana kolom NoFaktur diberi tipe data CHAR(3) dan di set sebagai Primary Key, kolom NoProduk diberi tipe data INT, kolom TglFaktur diberi tipe data DATE, sedangkan kolom Jumlah diberi tipe data INT. Berikut tabel JUAL yang sudah isi data.

	NoFaktur	NoProduk	TglFaktur	Jumlah
1	A01	1	2023-05-30	50
2	A02	2	2023-05-30	30
3	B01	3	2023-05-30	10
4	C01	4	2023-05-30	10
5	D01	5	2023-05-30	25

2. Buat trigger untuk:

- Mengupdate nilai harga jual menjadi sebesar harga pokok + 5% * harga pokok.

```
-- =====
-- Author      : Oktaviana Sadama Nur Azizah
-- Create date  : 2023-05-30
-- Description  : Mengupdate nilai harga jual menjadi sebesar harga produk + 5% harga pokok
-- =====
CREATE TRIGGER [dbo].[update_harga_jual]
ON [dbo].[PRODUK]
AFTER UPDATE
AS
BEGIN
    UPDATE PRODUK
    SET HargaJual = HargaPokok + (0.05 * HargaPokok)
END;
```

Trigger dibuat dengan perintah CREATE TRIGGER dengan nama trigger 'update_harga_jual' dan menggunakan perintah AFTER UPDATE. Untuk mengupdate nilai kolom HargaJual di tabel produk sebesar harga pokok + 5% * harga pokok, digunakan perintah UPDATE PRODUK dan SET HargaJual = HargaPokok + (0.05 * HargaPokok). Berikut adalah tabel produk sebelum dan sesudah dilakukan update harga jual.

Sebelum Trigger

	NoProduk	NamaProduk	Kategori	Warna	Stok	StokMin	HargaPokok	HargaJual
1	1	Kaos Polos	Pakaian	Putih	100	10	49000	79999
2	2	Sweater Rajut	Pakaian	Lilac	120	10	79000	129999
3	3	Celana Jeans Slim Fit	Celana	Denim	50	5	120000	299999
4	4	Sepatu Sneakers	Sepatu	Hitam	20	5	195000	349999
5	5	Topi Baseball	Aksesoris	Abu-abu	50	5	29000	59999

Setelah Trigger

	NoProduk	NamaProduk	Kategori	Warna	Stok	StokMin	HargaPokok	HargaJual
1	1	Kaos Polos	Pakaian	Putih	100	10	49000	51450
2	2	Sweater Rajut	Pakaian	Lilac	120	10	79000	82950
3	3	Celana Jeans Slim Fit	Celana	Denim	50	5	120000	126000
4	4	Sepatu Sneakers	Sepatu	Hitam	20	5	195000	204750
5	5	Topi Baseball	Aksesoris	Abu-abu	50	5	29000	30450

- Mengupdate nilai stok pada tabel produk ketika produk tersebut berhasil dijual dan sekaligus menampilkan pesan ketika mengalami kondisi stok dibawah stok minimum.

```

-- =====
-- Author      : Oktaviana Sadama Nur Azizah
-- Create date : 2023-05-30
-- Description  : Mengupdate stok ketika produk berhasil dijual dan menampilkan
--               pesan jika kondisi stok dibawah stok minimum
-- =====
CREATE TRIGGER [dbo].[update_stok_produk]
ON [dbo].[JUAL]
AFTER INSERT
AS
BEGIN
    DECLARE @Stok INT = (SELECT TOP 1 Stok FROM PRODUK p
    INNER JOIN inserted i ON p.NoProduk = i.NoProduk
    ORDER BY i.NoFaktur DESC)

    DECLARE @Jumlah INT = (SELECT Jumlah FROM inserted)

    DECLARE @StokMin INT = (SELECT TOP 1 p.StokMin FROM PRODUK p
    INNER JOIN inserted i ON p.NoProduk = i.NoProduk
    ORDER BY NoFaktur DESC)

    IF @Stok - @Jumlah > @StokMin
        UPDATE PRODUK set Stok = Stok - @Jumlah
        FROM PRODUK p INNER JOIN inserted i ON p.NoProduk = i.NoProduk
        WHERE p.NoProduk = i.NoProduk
    ELSE
        RAISERROR('[!WARNING: STOK DIBAWAH MINIMUM!]', 16, 1)
END;

```

Trigger dibuat dengan perintah CREATE TRIGGER dengan nama trigger 'update_stok_produk' dan menggunakan perintah AFTER INSERT. Variabel @Stok digunakan untuk menyimpan nilai stok terbaru dari produk yang terlibat dalam operasi INSERT terakhir di tabel JUAL. Variabel ini mendapatkan nilai stok dengan mengambil stok terakhir dari tabel PRODUK yang terhubung dengan baris yang baru di insert menggunakan pernyataan JOIN.

Variabel @Jumlah menyimpan nilai jumlah produk yang diinsert ke dalam tabel JUAL. Variabel ini mendapatkan nilai jumlah dari baris yang baru diinsert menggunakan pernyataan SELECT. Sedangkan variabel @StokMin menyimpan nilai stok minimum terbaru dari produk yang terlibat dalam operasi INSERT terakhir di tabel JUAL. Variabel ini mendapatkan nilai stok minimum terakhir dari tabel PRODUK yang terhubung dengan baris yang baru diinsert menggunakan pernyataan JOIN.

Kemudian digunakan pernyataan kondisional IF @Stok - @Jumlah > @StokMin, apabila terpenuhi maka dilakukan UPDATE jumlah produk. Sedangkan apabila tidak terpenuhi maka akan mengeluarkan teks error [!WARNING: STOK DIBAWAH MINIMUM!].

Inserted Data

Results Messages

	NoFaktur	NoProduk	TglFaktur	Jumlah
1	A01	1	2023-05-30	50
2	A02	2	2023-05-30	30
3	B01	3	2023-05-30	10
4	B02	3	2023-05-30	15
5	C01	4	2023-05-30	10
6	D01	5	2023-05-30	25

Sebelum Trigger

	NoProduk	NamaProduk	Kategori	Warna	Stok	StokMin	HargaPokok	HargaJual
1	1	Kaos Polos	Pakaian	Putih	100	10	49000	51450
2	2	Sweater Rajut	Pakaian	Lilac	120	10	79000	82950
3	3	Celana Jeans Slim Fit	Celana	Denim	50	5	120000	126000
4	4	Sepatu Sneakers	Sepatu	Hitam	20	5	195000	204750
5	5	Topi Baseball	Aksesoris	Abu-abu	50	5	29000	30450

Setelah Trigger

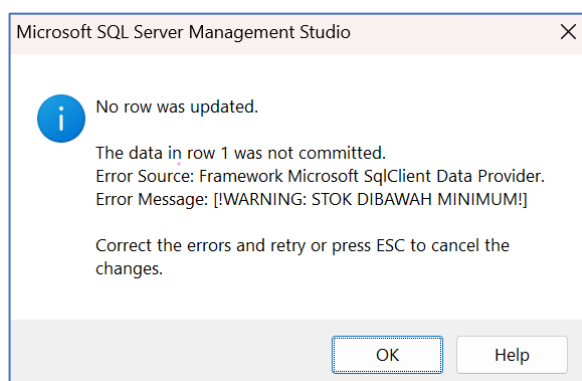
	NoProduk	NamaProduk	Kategori	Warna	Stok	StokMin	HargaPokok	HargaJual
1	1	Kaos Polos	Pakaian	Putih	100	10	49000	51450
2	2	Sweater Rajut	Pakaian	Lilac	120	10	79000	82950
3	3	Celana Jeans Slim Fit	Celana	Denim	35	5	120000	126000
4	4	Sepatu Sneakers	Sepatu	Hitam	20	5	195000	204750
5	5	Topi Baseball	Aksesoris	Abu-abu	50	5	29000	30450

Dilakukan inserting data pada tabel JUAL ('B02', 3, '2023-05-30', 15), maka stok barang yang memiliki NoProduk = 3, yaitu Celana Jeans Slim Fit akan mengalami pengurangan jumlah stok menjadi $50 - 15 = 35$.

Teks Error

```
SQLQuery13.sql -...N1VB6\Sadama (64))* X
INSERT INTO JUAL VALUES (
    'C02', '4', '2023-05-30', '17')
100 %
Messages
Msg 50000, Level 16, State 1, Procedure update_stok_produk, Line 27 [Batch Start Line 7]
[!WARNING: STOK DIBAWAH MINIMUM!]
```

Kemudian dilakukan lagi inserting data pada tabel JUAL seperti gambar diatas. Inserting data dilakukan pada kolom yang memiliki NoProduk = 4, yaitu Sepatu Sneakers yang merefer pada kolom PRODUK. Apabila inserting kolom Jumlah = 17 pada tabel JUAL dilakukan maka sisa stok sepatu menjadi $20 - 17 = 3$. Hal ini tidak diperbolehkan karena stok minimum sepatu sneakers yang diperbolehkan adalah 5. Maka error teks akan muncul seperti gambar diatas ataupun seperti gambar dibawah ini.



3. Buat Store Procedure untuk:

- a. Menampilkan periode waktu (bulan/ tahun), total pendapatan, dan total laba per bulan.

	NoFaktur	NoProduk	TglFaktur	Jumlah
1	A01	1	2023-05-30	50
2	A02	2	2023-05-30	30
3	A03	1	2023-06-01	80
4	A04	2	2023-06-01	100
5	B01	3	2023-05-30	10
6	B02	3	2023-05-30	15
7	B03	3	2023-06-01	10
8	C01	4	2023-05-30	10
9	C02	4	2023-06-01	3
10	D01	5	2023-05-30	25
11	D02	5	2023-06-01	20

	NoProduk	NamaProduk	Kategori	Warna	Stok	StokMin	HargaPokok	HargaJual
1	1	Kaos Polos	Pakaian	Putih	20	10	49000	51450
2	2	Sweater Rajut	Pakaian	Lilac	20	10	79000	82950
3	3	Celana Jeans Slim Fit	Celana	Denim	25	5	120000	126000
4	4	Sepatu Sneakers	Sepatu	Hitam	17	5	195000	204750
5	5	Topi Baseball	Aksesoris	Abu-abu	30	5	29000	30450

Sebelum membuat Store Procedure, dilakukan inserting data seperti gambar diatas.

```

-- =====
-- Author       : Oktaviana Sadama Nur Azizah
-- Create date  : 2023-05-30
-- Description   : Menampilkan periode waktu, total pendapatan, dan total laba per bulan
-- =====
CREATE PROCEDURE [dbo].[sp_showSummary]
AS
BEGIN
    SELECT MONTH(j.TglFaktur) AS 'Bulan', YEAR(j.TglFaktur) AS 'Tahun',
           SUM(j.Jumlah * p.HargaJual) AS 'Total Pendapatan',
           SUM((j.Jumlah * p.HargaJual) - (j.Jumlah * p.HargaPokok)) AS 'Total Laba'
    FROM JUAL j
    INNER JOIN PRODUK p ON j.NoProduk = p.NoProduk

    GROUP BY MONTH(j.TglFaktur), YEAR(j.TglFaktur)
    ORDER BY YEAR(j.TglFaktur), MONTH(j.TglFaktur)
END;

EXEC sp_showSummary

```

	Bulan	Tahun	Total Pendapatan	Total Laba
1	5	2023	11019750	524750
2	6	2023	14894250	709250

Stored Procedure dibuat dengan perintah CREATE PROCEDURE dengan nama 'sp_showSummary'. Store Procedure diatas menampilkan kolom periode waktu (bulan dan tahun) dengan perintah MONTH(j.TglFaktur) dan YEAR(j.TglFaktur), kolom Total Pendapatan dengan rumus SUM(j.Jumlah * p.HargaJual), dan kolom Total Laba dengan rumus SUM((j.Jumlah * p.HargaJual) - (j.Jumlah * p.HargaPokok)).

Untuk menghubungkan tabel PRODUK dan JUAL maka dilakukan INNER JOIN dengan menghubungkan `j.NoProduk = p.NoProduk`. Kemudian dilakukan GROUPING dan ORDER BY ASC. Store Procedure dipanggil dengan perintah EXEC `sp_showSummary`.

- b. Menampilkan data produk terjual (`NoProduk`, `NamaProduk`, `JumlahTerjual`) per masing-masing produk berdasarkan inputan periode waktu tertentu.

```
-- Author      : Oktaviana Sadama Nur Azizah
-- Create date  : 2023-05-30
-- Description  : Menampilkan data produk terjual per masing-masing produk
--               berdasarkan inputan periode waktu tertentu
-- =====
CREATE PROCEDURE [dbo].[sp_produkTerjual]
    @StartDate DATE, @EndDate DATE
AS
BEGIN
    SELECT p.NoProduk, p>NamaProduk, SUM(j.Jumlah) AS JumlahTerjual
    FROM PRODUK p
    INNER JOIN JUAL j ON p.NoProduk = j.NoProduk
    WHERE j.TglFaktur BETWEEN @StartDate AND @EndDate

    GROUP BY p.NoProduk, p>NamaProduk
    ORDER BY p.NoProduk

END;

EXEC sp_produkTerjual '2023-05-01', '2023-05-31';
```

	NoProduk	NamaProduk	JumlahTerjual
1	1	Kaos Polos	50
2	2	Sweater Rajut	30
3	3	Celana Jeans Slim Fit	25
4	4	Sepatu Sneakers	10
5	5	Topi Baseball	25

Stored Procedure dibuat dengan perintah CREATE PROCEDURE dengan nama `'sp_produkTerjual'` menggunakan parameter `@StartDate` dan `@EndDate` dengan tipe data DATE. Parameter ini digunakan untuk menginputkan periode waktu tertentu. Perintah SELECT digunakan untuk menampilkan `NoProduk`, `NamaProduk` dan `JumlahTerjual` dengan rumus `SUM(j.Jumlah)`.

Perintah INNER JOIN digunakan untuk menghubungkan tabel PRODUK dan JUAL dimana `p.NoProduk = j.NoProduk` dan dilakukan seleksi dengan perintah WHERE dimana `j.TglFaktur BETWEEN @StartDate AND @EndDate`. Kemudian dilakukan GROUPING dan ORDER BY ASC. Store Procedure dipanggil dengan perintah EXEC `sp_produkTerjual '2023-05-01', '2023-05-31'`.

4. Berdasarkan database di atas, kebutuhan sistem yang sedang dikembangkan antara lain meliputi:

Pada kedua tabel di atas sering dikenai perintah SQL seperti:

```
SELECT Produk.NoProduk, NamaProduk
FROM Produk, Jual
WHERE Produk.NoProduk = Jual.NoProduk
```

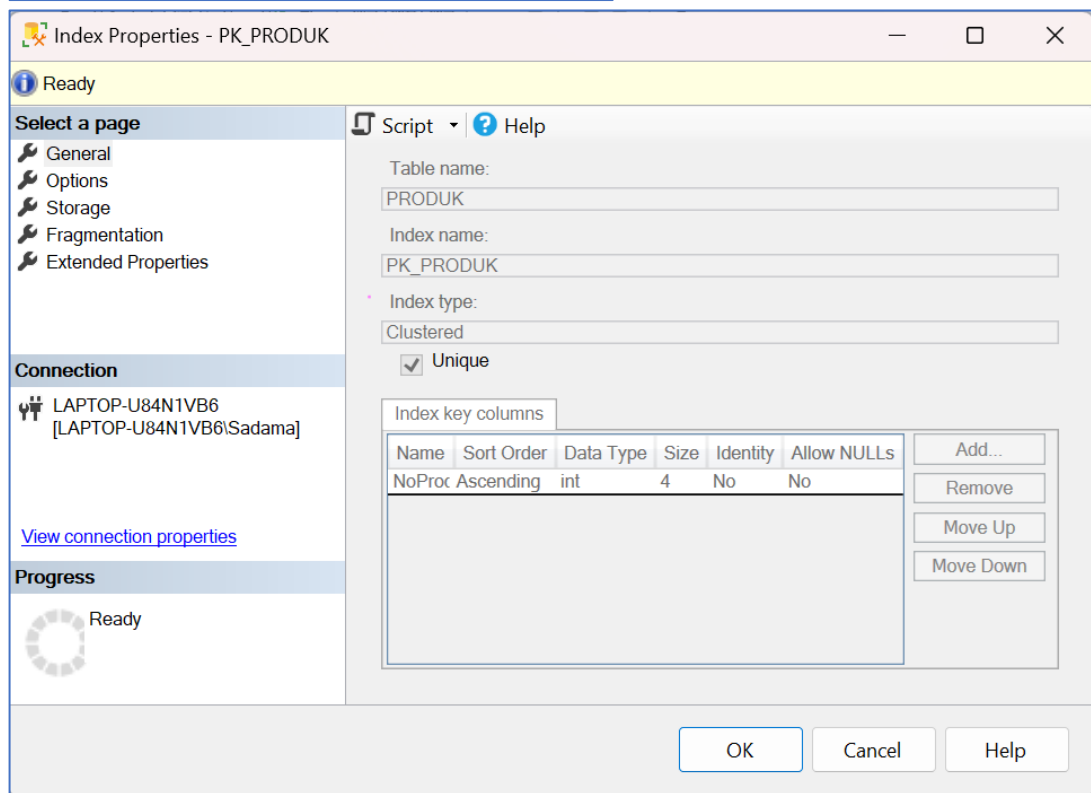
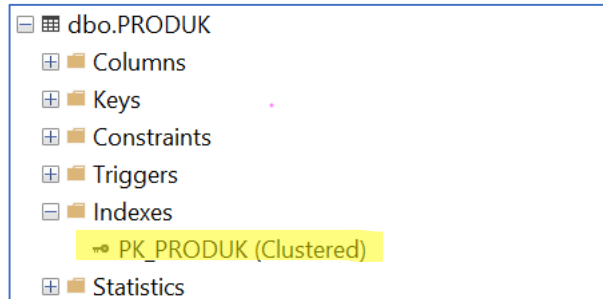
AND Kategori = 'Alat Rumah Tangga' (query 1)

Selain hal di atas pada tabel-tabel tersebut juga sering dikenakan perintah SQL untuk menampilkan data produk yang terjual untuk suatu periode waktu tertentu.

a. Tuliskan indeks-indeks yang telah dibuat dalam database RESPONSI! Jelaskan!

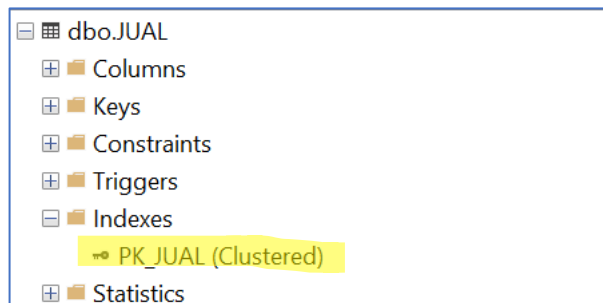
Penjelasan:

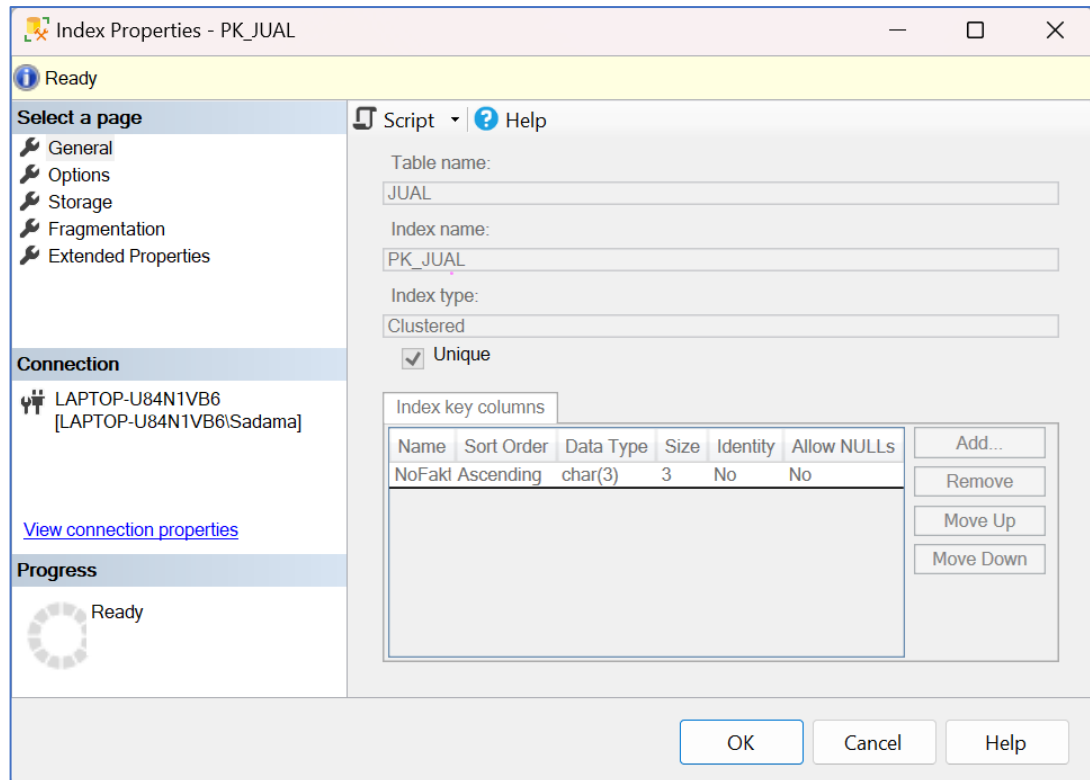
Clustered Indeks Tabel Produk



Clustered indeks otomatis terbuat pada tabel PRODUK saat mendefinisikan Primary Key. Field yang digunakan sebagai search key adalah NoProduk.

Clustered Indeks Tabel Jual





Clustered indeks otomatis terbuat pada tabel JUAL saat mendefinisikan Primary Key. Field yang digunakan sebagai search key adalah NoFaktur.

- b. Tentukan indeks-indeks tambahan yang diperlukan, field yang digunakan sebagai search key, tipe indeks, dan jelaskan untuk masing-masing indeks tersebut!

Penjelasan:

Berdasarkan query 1, indeks tambahan yang diperlukan adalah non clustered indeks sebagai berikut.

Non Clustered Indeks	
Tabel	Field Search Key
PRODUK	NoProduk NamaProduk Kategori
JUAL	NoProduk

Non clustered indeks pada tabel PRODUK dan JUAL dibuat untuk mempercepat waktu eksekusi query saat pencarian data.

- c. Bagaimana query optimizer mengeksekusi perintah SQL yang disebutkan di atas (query 1), setelah ditambahkan indeks? Jelaskan!

Penjelasan:

Sebelum ditambahkan non clustered indeks, query 1 membutuhkan elapsed time sebesar 00:00:00.0456403 ditunjukkan seperti gambar berikut.

SQLQuery19.sql -...N1VB6\Sadama (55))*

```

SELECT Produk.NoProduk, NamaProduk
FROM Produk, Jual
WHERE Produk.NoProduk = Jual.NoProduk
AND Kategori = 'Pakaian'

```

Results

NoProduk	NamaProduk
1	Kaos Polos
2	Sweater Rajut
1	Kaos Polos
2	Sweater Rajut

LAPTOP-U84N1VB6\Sadama... RESPONSI 00:00:00 4 rows

Properties

Current connection parameters

Aggregate Status

Connection failures	
Elapsed time	00:00:00.0456403
Finish time	31/05/2023 15:11:09
Name	LAPTOP-U84N1VB6
Rows returned	4
Start time	31/05/2023 15:11:09
State	Open

Connection

Connection name	LAPTOP-U84N1VB6 (LAPTOP)
-----------------	--------------------------

Connection Details

Connection elapsed time	00:00:00.0456403
Connection encryption	Not encrypted
Connection finish time	31/05/2023 15:11:09
Connection rows returned	4
Connection start time	31/05/2023 15:11:09
Connection state	Open
Display name	LAPTOP-U84N1VB6
Login name	LAPTOP-U84N1VB6\Sadama
Server name	LAPTOP-U84N1VB6
Server version	16.0.1000
Session Tracing ID	

Name

The name of the connection.

Setelah ditambahkan non clustered indeks, query 1 hanya membutuhkan elapsed time sebesar 00:00:00.0454909 ditunjukkan seperti gambar berikut.

SQLQuery19.sql -...N1VB6\Sadama (55))*

```

SELECT Produk.NoProduk, NamaProduk
FROM Produk, Jual
WHERE Produk.NoProduk = Jual.NoProduk
AND Kategori = 'Pakaian'

```

Results

NoProduk	NamaProduk
1	Kaos Polos
2	Sweater Rajut
1	Kaos Polos
2	Sweater Rajut

(16.0 RTM) LAPTOP-U84N1VB6\Sadama... RESPONSI 00:00:00 4 rows

Properties

Current connection parameters

Aggregate Status

Connection failures	
Elapsed time	00:00:00.0454909
Finish time	31/05/2023 16:52:29
Name	LAPTOP-U84N1VB6
Rows returned	4
Start time	31/05/2023 16:52:29
State	Open

Connection

Connection name	LAPTOP-U84N1VB6 (LA
-----------------	---------------------

Connection Details

Connection elapsed time	00:00:00.0454909
Connection encryption	Not encrypted
Connection finish time	31/05/2023 16:52:29
Connection rows returned	4
Connection start time	31/05/2023 16:52:29
Connection state	Open
Display name	LAPTOP-U84N1VB6
Login name	LAPTOP-U84N1VB6\Sac
Server name	LAPTOP-U84N1VB6
Server version	16.0.1000
Session Tracing ID	

Name

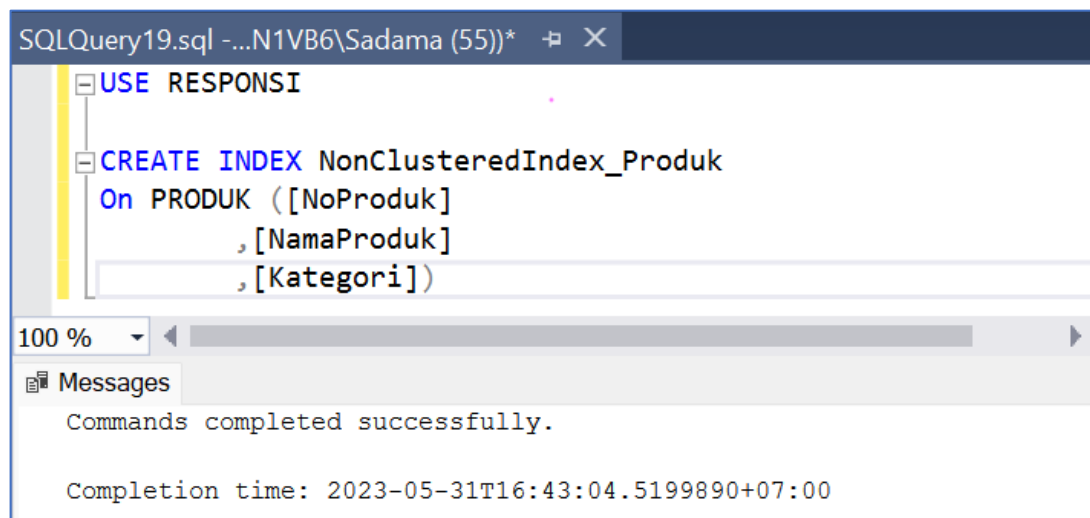
The name of the connection.

Penambahan non clustered indeks pada kolom NoProduk, NamaProduk, dan Kategori terbukti mengoptimalkan query. Waktu eksekusi yang lebih singkat meningkatkan efektifitas dan kinerja sistem secara keseluruhan.

Note: Kategori disesuaikan isi data milik pribadi, disini saya menggunakan Kategori = 'Pakaian'.

- d. Implementasikan indeks yang sudah anda desain pada soal b!
Non Clustered Indeks Prodok

Berikut adalah pembuatan non clustered indeks untuk tabel PRODUK dengan perintah `CREATE INDEX [nama_indeks]`. Dalam contoh dibawah ini dibuat non clustered indeks dengan nama `NonClusteredIndex_Produk` di kolom `NoProduk`, `NamaProduk`, dan `Kategori`.



```
SQLQuery19.sql -...N1VB6\Sadama (55))* X
USE RESPONSI
CREATE INDEX NonClusteredIndex_Produk
On PRODUK ([NoProduk]
           ,[NamaProduk]
           ,[Kategori])
```

100 %

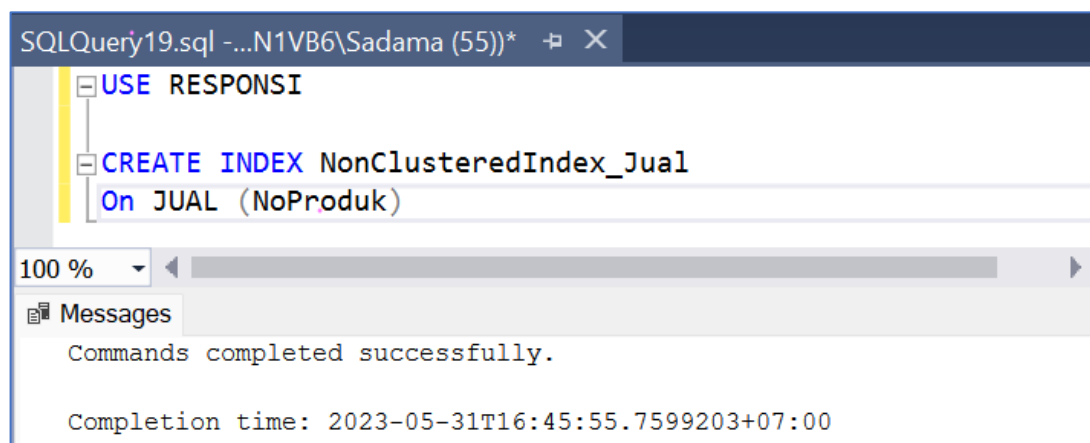
Messages

Commands completed successfully.

Completion time: 2023-05-31T16:43:04.5199890+07:00

Non Clustered Indeks Jual

Berikut adalah pembuatan non clustered indeks untuk tabel JUAL dengan perintah `CREATE INDEX [nama_indeks]`. Dalam contoh dibawah ini dibuat non clustered indeks dengan nama `NonClusteredIndex_Jual` di kolom `NoProduk`.



```
SQLQuery19.sql -...N1VB6\Sadama (55))* X
USE RESPONSI
CREATE INDEX NonClusteredIndex_Jual
On JUAL (NoProduk)
```

100 %

Messages

Commands completed successfully.

Completion time: 2023-05-31T16:45:55.7599203+07:00

5. Gunakan fasilitas yang ada pada SQL Server untuk menyimpan:
- Data trigger dan indeks yang dilampirkan pada tabel dengan nama ***“trigger_index.sql”***
 - Data stored procedure yang dibuat pada database anda dengan nama ***“stor_proc.sql”*** (No.5 terlampir dalam RAR file. Terimakasih 😊)

BAB IV

PENUTUP

Kesimpulan

Berdasarkan responsi yang telah dilakukan, didapati beberapa kesimpulan sebagai berikut:

1. Trigger merupakan kumpulan script yang berhubungan dengan table, view, ataupun schema yang dijalankan secara otomatis ketika terdapat event yang dijalankan.
2. Stored Procedure adalah sebuah fungsi yang berisi kode SQL yang dapat digunakan kembali dengan cara memanggil Stored Procedure yang telah dibuat dengan perintah EXEC. Dalam Stored Procedure juga dapat dimasukkan parameter sehingga fungsi dapat digunakan lebih dinamis berdasarkan parameter tersebut.
3. Indeks adalah kunci yang dibuat dari satu atau beberapa kolom dalam database yang berguna untuk mempercepat pengambilan baris dalam tabel. Indeks terdiri dari dua tipe, yaitu clustered indeks dan non clustered indeks.