Author Yuri Rahmanto, M.Kom
yurirahmanto@teknokrat.ac.id

1 Membuat Objek Database Menggunakan SQL Server

Materi Pembahasan :

1. Bisnis Proses dan ERD
2. Membuat Database
3. Membuat Tabel
4. Membuat View
5. Membuat Stored Procedure
6. Membuat Trigger

3.1 Bisnis Proses dan Entity Relationship Diagram

Sistem informasi penjualan adalah sub sistem informasi bisnis yang mencakup kumpulan prosedur yang melaksanakan, mencatat, mengkalkulasi, membuat dokumen dan informasi penjualan untuk keperluan manajemen dan bagian lain yang berkepentingan, mulai dari diterimanya order penjualan sampai mencatat timbulnya tagihan/piutang dagang. Berkembangnya peran teknologi saat ini, proses pencatatan atau dokumentasi proses jual-beli dilakukan dengan bantuan aplikasi atau program yang dapat mempermudah bagian terkait dalam penginputan dan pelaporan data.

Dalam studi kasus ini, kita diminta untuk membuat aplikasi penjualan barang berbasis desktop. Perlu ditekankan, aplikasi yang dibuat **tidak membahas** bagaimana proses pemesanan dan pembelian barang melalui supplier, namun **hanya membahas** bagaimana proses penjualan barang yang sudah tersedia sampai ke pelanggan. Adapun ketentuan proses bisnisnya adalah :

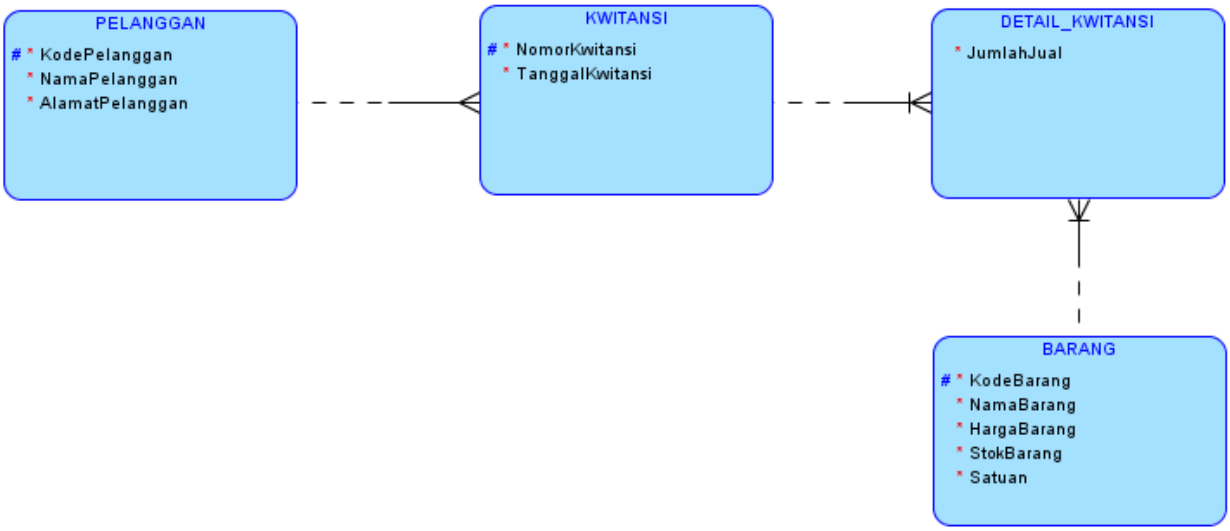
1. Toko ABC melakukan penjualan barang kepada pelanggan. Pelanggan yang bisa melakukan transaksi adalah pelanggan yang harus terdaftar terlebih dahulu. Pelanggan memiliki data berupa Kode pelanggan, nama pelanggan dan alamat pelanggan.
2. Pelanggan akan menerima kwitansi dengan data Nomor kwitansi, tanggal kwitansi, data barang dan jumlah barang yang dijual oleh toko.
3. Satu pelanggan boleh saja memiliki lebih dari satu nomor kwitansi.
4. Satu nomor kwitansi hanya boleh dimiliki oleh satu pelanggan.
5. Satu nomor kwitansi boleh saja terdiri dari beberapa barang dengan data berupa Kode, Nama, Harga, Stok, Satuan.
6. Satu barang boleh saja terdapat pada beberapa nomor kwitansi.

Dari hasil analisis bisnis proses di atas, didapat beberapa entitas yaitu :

a) Pelanggan,
b) Barang, dan
c) Kwitansi

Sehingga, rancangan Entity Relationship Diagram (ERD) dapat dilihat pada gambar 1.1 berikut :

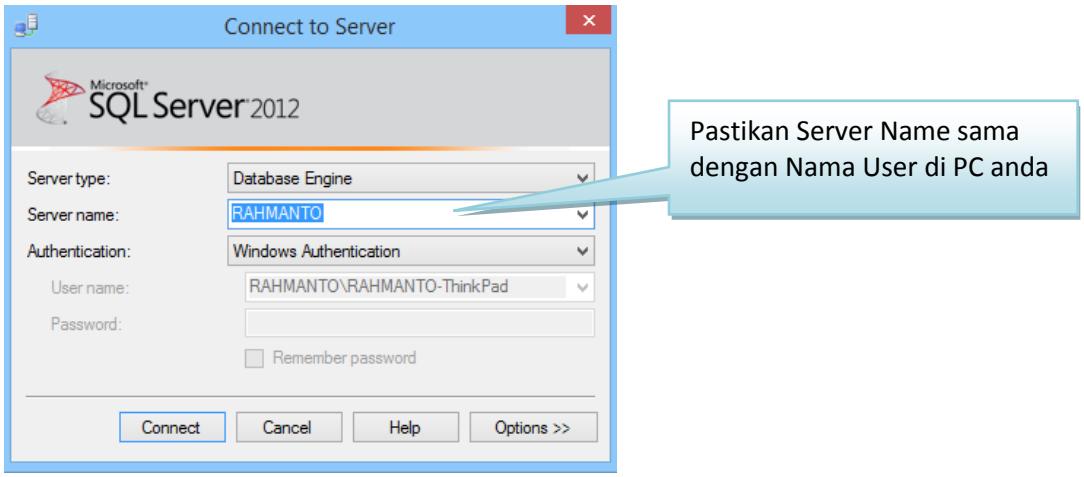
Author Yuri Rahmanto, M.Kom
yurirahmanto@teknokrat.ac.id



Gambar 1.1. ERD Proses Bisnis Penjualan Barang

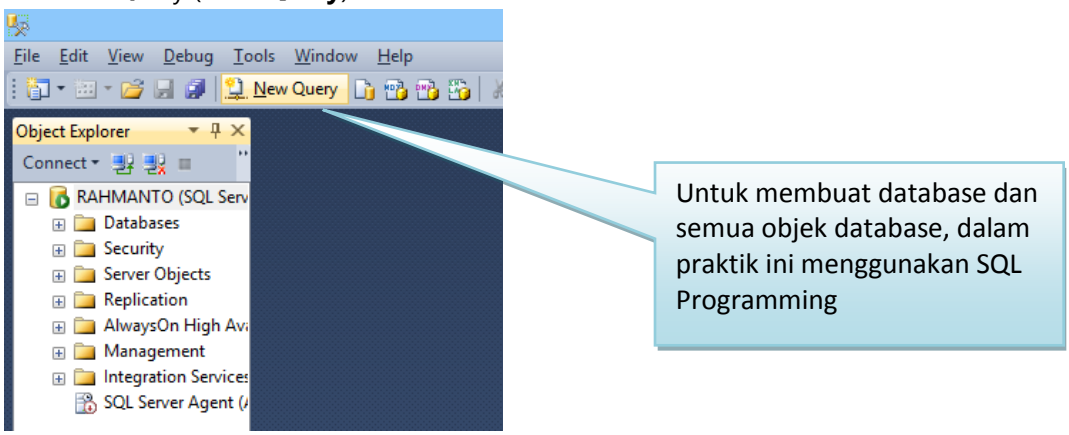
3.2 Membuat Database

1. Buka aplikasi SQL Server Management Studio
2. Klik **button connect**



Gambar 1.2. Connect to SQL Server

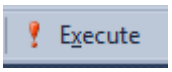
3. Buka tab Query (**New Query**)



Gambar 1.3. Membuka Halaman Query

4. Ketik Query berikut :

```
--membuat database
create database pbd_ti15
--menggunakan database
use pbd_ti15
```

5. Eksekusi tiap query dengan cara blok query, lalu klik button **Execute** 

3.3 Membuat Tabel

1. Setelah membuat database, lalu menggunakan database, objek yang pertama dibuat adalah tabel
2. Ketik Query berikut, dan eksekusi tiap query

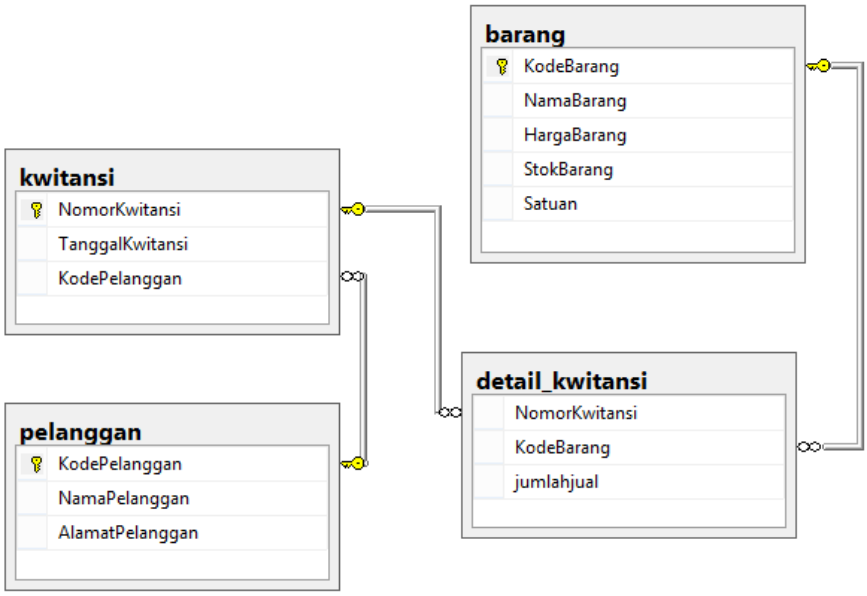
```
--membuat tabel pelanggan
create table pelanggan
(
KodePelanggan char(5) primary key,
NamaPelanggan varchar(25) not null,
AlamatPelanggan varchar(50) not null
)
```

```
--membuat tabel barang
create table barang
(
KodeBarang char(5) primary key,
NamaBarang varchar(25) not null,
HargaBarang int not null,
StokBarang int not null,
Satuan varchar(15) not null
)
```

```
--membuat tabel kwitansi
create table kwitansi
(
NomorKwitansi char(5) primary key,
TanggalKwitansi date not null,
KodePelanggan char(5) not null foreign key references
pelanggan(KodePelanggan)
)
```

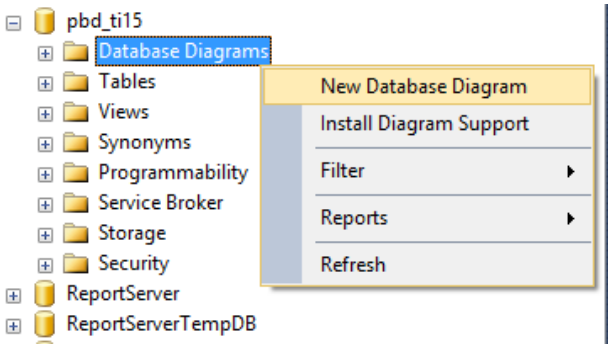
```
--membuat tabel detail_kwitansi
create table detail_kwitansi
(
NomorKwitansi char(5) not null foreign key references
kwitansi(NomorKwitansi),
KodeBarang char(5) not null foreign key references
barang(KodeBarang),
jumlahjual int not null
)
```

3. Skema diagram dari tabel yang dibuat adalah sebagai berikut :



Gambar 1.4. Skema Database Diagrams

4. Untuk melihat skema database diagrams, Klik kanan pada Database Diagrams – New Database Diagram, lalu add 4 tabel yang ada



Gambar 1.5. New Database Diagram

5. Kembali ke tab query, isi 30 baris data ke dalam tabel pelanggan menggunakan perintah berikut :

```
--memasukkan data pelanggan
insert into pelanggan values
('P0001','aditya','Bandar Lampung'),
('P0002','nandi','Lampung Barat'),
('P0003','indrajat','Lampung Selatan'),
('P0004','ali','Bandar Lampung'),
('P0005','agung p','Lampung Selatan'),
('P0006','dedi','Lampung Selatan'),
('P0007','rhendy','Lampung Selatan'),
('P0008','lulus','Lampung Timur'),
('P0009','alipi','Bandar Lampung'),
('P0010','indra','Bandar Lampung')
```

6. Buka aplikasi Ms. Office Excel, lalu buat data seperti berikut :

	A	B	C	D	E
1	KodeBarang	NamaBarang	HargaBarang	StokBarang	Satuan
2	B0001	Xiaomi Redmi 5A	1210000	127	pcs
3	B0002	Xiaomi Redmi 4X	1650000	100	pcs
4	B0003	HP Pavilion 14	3925000	23	pcs
5	B0004	Acer Aspire One	3420000	10	pcs
6	B0005	Canon EOS 700D	6200000	5	pcs
7	B0006	ADATA Premiere	651000	7	pcs
8	B0007	RJ45	50000	100	pack

7. Simpan file Excel tersebut dengan nama data_barang.csv
   ubah ekstensi file sebelum disimpan menjadi* *.csv (comma delimited)***
8. Kembali ke tab query SQL Server, ketik perintah berikut untuk import data barang yang berasal dari file .csv ke dalam tabel barang

```
--memasukkan data barang
BULK INSERT barang
FROM 'D:\DATA\2018\PBD 2018\Pertemuan 2 (praktik)\data_barang.csv'
WITH
(
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)
GO
```

Alamat penyimpanan file .csv

9. Gunakan perintah **SELECT** untuk melihat data pada tabel pelanggan dan tabel barang

3.4 Membuat View

Setelah membuat tabel, objek berikutnya yang akan dibuat adalah view. View merupakan virtual tabel, view berisi query yang dapat dipanggil berulang kali pada saat dibutuhkan (Forta, 2007). View yang akan dibuat akan berisi query join untuk menggabungkan 4 tabel pada database (pelanggan, barang, kwitansi dan detail_kwitansi).

1. Ketik lalu eksekusi query berikut untuk membuat view_kwitansi :

```
--membuat view kwitansi
CREATE VIEW View_Kwitansi AS
SELECT detail_kwitansi.NomorKwitansi, kwitansi.TanggalKwitansi,
pelanggan.KodePelanggan, pelanggan.NamaPelanggan,
detail_kwitansi.KodeBarang, barang.NamaBarang, barang.HargaBarang,
barang.Satuan, detail_kwitansi.jumlahjual
FROM detail_kwitansi
JOIN kwitansi
ON kwitansi.NomorKwitansi = detail_kwitansi.NomorKwitansi
JOIN barang
ON barang.KodeBarang = detail_kwitansi.KodeBarang
JOIN pelanggan
ON pelanggan.KodePelanggan= kwitansi.KodePelanggan
```

2. Sebelum menampilkan view_kwitansi, tambahkan terlebih dahulu data ke dalam tabel kwitansi dan detail_kwitansi. Ketik dan eksekusi query berikut :

```
--memasukkan data kwitansi
INSERT INTO kwitansi VALUES
('K0001','2018/03/01','P0001'),
('K0002','2018/03/05','P0002'),
('K0003','2018/03/07','P0003'),
('K0004','2018/03/07','P0004')
```

```
--memasukkan data detail kwitansi
INSERT INTO detail_kwitansi VALUES
('K0001','B0001',2),
('K0001','B0002',1),
('K0001','B0004',1),
('K0002','B0002',1),
('K0003','B0001',2),
('K0003','B0002',1),
('K0003','B0003',1),
('K0003','B0004',3),
('K0004','B0001',1),
('K0004','B0004',1)
```

3. Untuk menampilkan view_kwitansi, ketik lalu eksekusi query berikut :

```
--memanggil view
SELECT * FROM View_Kwitansi;
```

4. View akan menampilkan data layaknya tabel

	NomorKwitansi	TanggalKwitansi	KodePelanggan	NamaPelanggan	KodeBarang	NamaBarang	HargaBarang	Satuan	jumlahjual
1	K0001	2018-03-01	P0001	aditya	B0001	Xiaomi Redmi 5A	1210000	pcs	2
2	K0001	2018-03-01	P0001	aditya	B0002	Xiaomi Redmi 4X	1650000	pcs	1
3	K0001	2018-03-01	P0001	aditya	B0004	Acer Aspire One	3420000	pcs	1
4	K0002	2018-03-05	P0002	nandi	B0002	Xiaomi Redmi 4X	1650000	pcs	1
5	K0003	2018-03-07	P0003	indrajat	B0001	Xiaomi Redmi 5A	1210000	pcs	2
6	K0003	2018-03-07	P0003	indrajat	B0002	Xiaomi Redmi 4X	1650000	pcs	1
7	K0003	2018-03-07	P0003	indrajat	B0003	HP Pavilion 14	3925000	pcs	1

Gambar 1.6. Tampilan Data View_Kwitansi

5. View_kwitansi yang dibuat nantinya akan digunakan untuk menampilkan/mencetak kwitansi setelah selesai melakukan transaksi penjualan. Selanjutnya buat satu view lagi, beri nama view_detailkwitansi yang berguna untuk menampilkan detail penjualan barang per nomor kwitansi pada saat transaksi penjualan.
   Ketik dan eksekusi query berikut :

```
--membuat VIEW DETAIL KWITANSI
CREATE VIEW view_detailkwitansi AS
SELECT  detail_kwitansi.NomorKwitansi, detail_kwitansi.KodeBarang,
barang.NamaBarang, barang.HargaBarang, detail_kwitansi.JumlahJual,
barang.HargaBarang * detail_kwitansi.JumlahJual AS JUMLAHBAYAR
FROM barang INNER JOIN
detail_kwitansi ON barang.KodeBarang = detail_kwitansi.KodeBarang;
```

3.5 Membuat Stored Procedure

Stored Procedure merupakan kumpulan dari satu atau lebih perintah SQL yang dapat dieksekusi lebih cepat (Forta, 2007). Berikut langkah-langkah untuk membuat Stored Procedure :

1. Ketik lalu eksekusi perintah berikut pada tab query :

```
--prosedur sp_view_kwitansi
CREATE PROCEDURE SP_View_Kwitansi AS
BEGIN
    select * from VIEW_KWITANSI;
END;
```

2. Untuk menggunakan Stored Procedure yang telah dibuat, gunakan perintah **EXECUTE** diikuti nama Stored Procedure

```
--eksekusi proc sp_view_kwitansi
EXECUTE SP_View_Kwitansi;
```

**Stored Procedure SP_View_Kwitansi akan menampilkan view_kwitansi*

3. Membuat Stored Procedure simpan_pelanggan

```
--membuat prosedur simpan_pelanggan
CREATE PROCEDURE simpan_pelanggan
@kdpelanggan char(5),
@nmpelanggan varchar(25),
@almtpelanggan varchar(50)
AS
BEGIN
    INSERT INTO pelanggan VALUES
    (@kdpelanggan, @nmpelanggan, @almtpelanggan);
END;
```

*****@kdpelanggan*** *merupakan variabel buatan yang berfungsi untuk menampung data pada saat Procedure simpan_pelanggan dieksekusi, variabel buatan diikuti tipe data sesuai dengan tipe data pada kolom yang bersangkutan di tabel tersebut (pelanggan).*

4. Eksekusi Procedure simpan_pelanggan

```
--eksekusi prosedur simpan_pelanggan
EXECUTE simpan_pelanggan 'P0035', 'Alzi', 'Bandar Lampung';
SELECT * FROM pelanggan;
```

5. Membuat Stored Procedure hapus_pelanggan

```
--prosedur hapus_pelanggan
CREATE PROCEDURE hapus_pelanggan
@kdpelanggan char(5)
AS
BEGIN
    DELETE FROM pelanggan WHERE KodePelanggan = @kdpelanggan;
END;
```

6. Eksekusi Procedure hapus_pelanggan

```
--eksekusi prosedur hapus_pelanggan
EXECUTE hapus_pelanggan 'P0035';
SELECT * FROM pelanggan;
```

7. Membuat Stored Procedure ubah_pelanggan

```
--prosedur ubah_pelanggan
CREATE PROCEDURE ubah_pelanggan
@kdpelanggan char(5),
@nmpelanggan varchar(25),
@almtpelanggan varchar(50)
AS
BEGIN
    UPDATE pelanggan set NamaPelanggan = @nmpelanggan,
    AlamatPelanggan =  @almtpelanggan
    WHERE KodePelanggan = @kdpelanggan ;
END;
```

8. Eksekusi Procedure ubah_pelanggan

```
--eksekusi prosedur ubah_pelanggan
EXECUTE ubah_pelanggan 'P0030', 'anita', 'Lampung Barat';
SELECT * FROM pelanggan;
```

9. Membuat Stored Procedure cari_pelanggan

```
--prosedur cari_pelanggan
CREATE PROCEDURE cari_pelanggan
@nmpelanggan varchar(25)
AS
BEGIN
    SELECT * FROM pelanggan
    WHERE NamaPelanggan LIKE '%'+@nmpelanggan+'%';
END;
```

10. Eksekusi Procedure cari_pelanggan

```
--mencari nama pelanggan yang mengandung huruf E
EXECUTE cari_pelanggan 'E';
```

11. Membuat Stored Procedure simpan_barang

```
--prosedur simpan_barang
CREATE PROCEDURE simpan_barang
@kdbarang char(5),
@nmbarang varchar(25),
@hrgbarang int,
@stok int,
@satuan varchar(15)
AS
BEGIN
    INSERT INTO barang VALUES
    (@kdbarang,@nmbarang,@hrgbarang,@stok,@satuan);
END;
```

12. Eksekusi Procedure simpan_barang

```
--eksekusi proc simpan_barang
EXECUTE simpan_barang 'B0008','Mouse Log',650000,49,'pcs';
EXECUTE simpan_barang 'B0009','Keyboard Razor',750000,50,'pcs';
SELECT * FROM barang;
```

13. Membuat Stored Procedure hapus_barang

```
--prosedur Hapus_Barang
CREATE PROCEDURE hapus_barang
@kodebarang char(5)
AS
BEGIN
    DELETE FROM barang WHERE KodeBarang = @kodebarang;
END;
```

14. Eksekusi Procedure hapus_barang

```
--eksekusi proc hapus_barang
EXECUTE hapus_barang 'B0009';
SELECT * FROM BARANG;
```

15. Membuat Stored Procedure ubah_barang

```
--prosedur Ubah_Barang
CREATE PROCEDURE ubah_barang
@nama varchar(25), @harga int, @stok int, @satuan varchar(15),
@kode char(5)
AS
BEGIN
    UPDATE barang set NamaBarang = @nama, HargaBarang = @harga,
    StokBarang = @stok, Satuan = @satuan WHERE KodeBarang = @kode;
END;
```

16. Eksekusi Procedure ubah_barang

```
--eksekusi proc ubah_barang
EXECUTE ubah_barang 'Handphone X5',4500000,100,'pcs','B0008';
SELECT * FROM barang;
```

17. Membuat Stored Procedure cari_barang

```
--prosedur Cari_Barang
CREATE PROCEDURE cari_barang
@namabarang varchar(25)
AS
BEGIN
    SELECT * FROM barang
    where NamaBarang LIKE '%'+@namabarang+'%';
END;
```

18. Eksekusi Procedure cari_barang

```
--mencari nama barang yang mengandung huruf J
EXECUTE cari_barang 'J';
```

19. Membuat Stored Procedure simpan_kwitansi

```
--prosedur Simpan_Kwitansi
CREATE PROCEDURE simpan_kwitansi
@nokwitansi char(5),
@tanggal date,
@kodepelanggan char(5)
AS
BEGIN
    INSERT INTO kwitansi VALUES(@nokwitansi, @tanggal,
    @kodepelanggan);
END;
```

20. Eksekusi Procedure simpan_kwitansi

```
--eksekusi proc simpan_kwitansi
EXECUTE simpan_kwitansi 'K0005','2018/03/20','P0001';
SELECT * FROM kwitansi;
```

21. Membuat Stored Procedure simpan_detailkwitansi

```
--prosedur Simpan_DetailKwitansi
CREATE PROCEDURE simpan_detailkwitansi
@nokwitansi char(5),
@kodebarang char(5),
@jumlah int
AS
BEGIN
    INSERT INTO detail_kwitansi VALUES
    (@nokwitansi, @kodebarang, @jumlah);
END;
```

22. Eksekusi Procedure simpan_detailkwitansi

```
--eksekusi proc simpan_detailkwitansi
EXECUTE simpan_detailkwitansi 'K0005','B0001',2;
SELECT * FROM detail_kwitansi;
```

23. Membuat Stored Procedure hapus_detailkwitansi

```
--prosedur Hapus_DetailKwitansi
CREATE PROCEDURE hapus_detailkwitansi
@nokwitansi char(5),
@kodebarang char(5)
AS
BEGIN
    DELETE FROM detail_kwitansi WHERE
    NomorKwitansi = @nokwitansi AND
    KodeBarang = @kodebarang;
END;
```

24. Eksekusi Procedure hapus_detailkwitansi

```
--eksekusi proc hapus_detailkwitansi
EXECUTE hapus_detailkwitansi 'K0005','B0001';
SELECT * FROM detail_kwitansi;
```

3.6 Membuat Trigger

Trigger merupakan kumpulan pernyataan SQL yang berada di antara BEGIN dan END yang secara otomatis akan dijalankan oleh SQL SERVER sebagai respon dari kondisi AFTER INSERT, DELETE dan UPDATE (Forta, 2007).

1. Membuat **Trigger AfterInsert_DetailKwitansi**, trigger ini akan aktif pada saat ada penambahan data **(AFTER INSERT)** di tabel detail_kwitansi. Aksi yang secara otomatis akan dijalankan oleh SQL Server adalah merubah stok pada tabel barang (stok berkurang).
   Ketik lalu eksekusi query berikut :

```
--membuat Trigger AfterInsert_DetailKwitansi
CREATE TRIGGER AfterInsert_DetailKwitansi
ON detail_kwitansi
AFTER INSERT
AS
BEGIN
    DECLARE
            @jumlah int, @kode char(5)
    Select  @jumlah = jumlahjual,
            @kode = KodeBarang from inserted

    Update barang set barang.StokBarang =
    barang.StokBarang - @jumlah
    Where barang.KodeBarang = @kode;
END;
```

2. Untuk mencoba trigger, eksekusi query berikut :

```
--menyimpan data ke dalam tabel detail_kwitansi
EXECUTE simpan_detailkwitansi 'K0005','B0002',9;
--cek stok barang
select * from barang;
```

Cek stok pada tabel barang dengan kode B0002, seharusnya jumlah stok berkurang 9 pcs

3. Membuat **Trigger AfterDelete_DetailKwitansi**, trigger ini akan aktif pada saat ada penghapusan data **(AFTER DELETE)** pada tabel detail_kwitansi. Aksi yang secara otomatis akan dijalankan oleh SQL Server adalah merubah stok pada tabel barang (stok bertambah kembali).
   Ketik lalu eksekusi query berikut :

```
--membuat Trigger AfterDelete_DetailKwitansi
CREATE TRIGGER AfterDelete_DetailKwitansi
ON detail_kwitansi
AFTER DELETE
AS
BEGIN
    DECLARE
            @jumlah int, @kode char(5)
    Select  @jumlah = jumlahjual,
            @kode = KodeBarang from deleted
    Update barang set barang.StokBarang =
    barang.StokBarang + @jumlah
    Where barang.KodeBarang = @kode;
END;
```

4. Untuk mencoba trigger, eksekusi query berikut :

```
--menghapus data dari tabel detail_kwitansi
EXECUTE hapus_detailkwitansi 'K0005','B0002';
--cek stok barang
select * from barang;
```

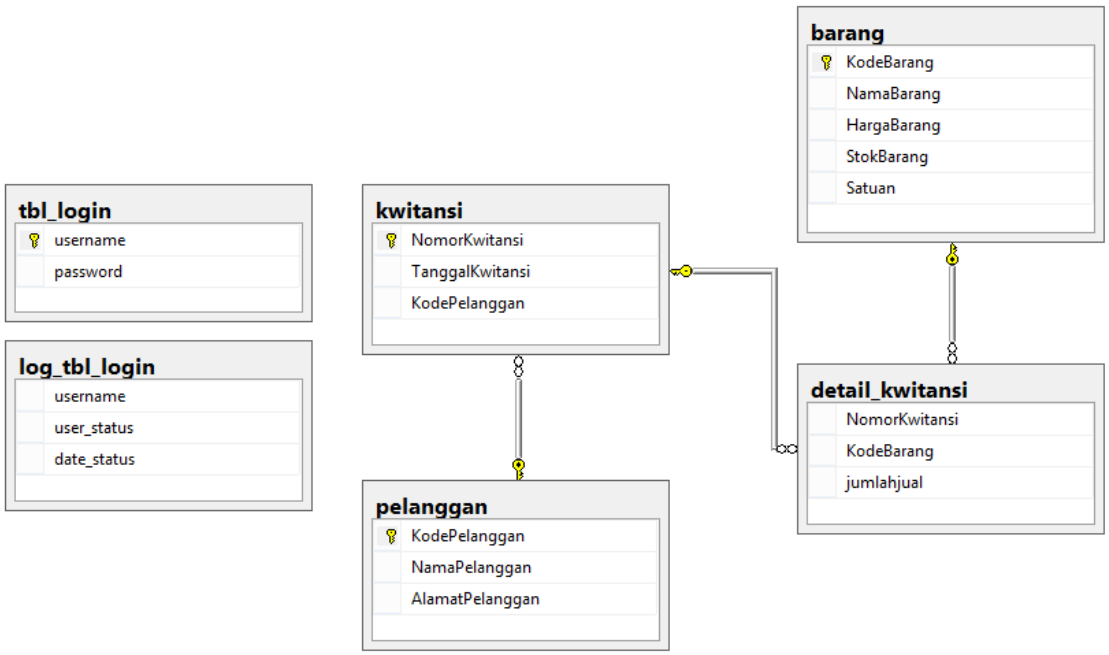
Cek stok pada tabel barang dengan kode B0002, seharusnya jumlah stok kembali bertambah 9 pcs

5. Trigger berikutnya yang akan dibuat adalah trigger yang aktif setelah ada penyimpanan data di tbl_login, aksi yang secara otomatis akan dijalankan oleh SQL Server adalah menyimpan data ke dalam tabel log_tbl_login dengan keterangan user_status "registered". Namun karena tbl_login dan log_tbl_login belum dibuat, maka sebelumnya ketik dan eksekusi query berikut untuk membuat kedua tabel.

```
--MEMBUAT TABEL LOGIN
CREATE TABLE tbl_login (username varchar(25) PRIMARY KEY,
password varchar(25) NOT NULL);
```

```
--MEMBUAT TABEL LOG LOGIN
CREATE TABLE log_tbl_login (username varchar(25) NOT NULL,
user_status varchar(10) NOT NULL,
date_status datetime DEFAULT GETDATE());
```

6. Sehingga tampilan database diagram yang baru akan menjadi seperti gambar 1.7 berikut :



Gambar 1.7. Skema Database Diagram dengan Tabel Login

7. Membuat trigger **insert_login** :

```
CREATE TRIGGER insert_login
ON tbl_login
AFTER INSERT
AS
BEGIN
    DECLARE @user varchar(25)
    SELECT @user = username FROM inserted
    INSERT INTO log_tbl_login (username,user_status)
    VALUES (@user, 'REGISTERED');
END;
```

8. Untuk mencoba trigger, eksekusi query berikut :

```
--simpan 1 user ke tbl_login
INSERT INTO tbl_login VALUES ('IF16B','123');
--cek data di tabel log_tbl_login
SELECT * FROM log_tbl_login;
```

9. Tampilan data pada log_tbl_login

	username	user_status	date_status
1	IF16B	REGISTERED	2018-03-24 10:14:10.163

Gambar 1.8. Tampilan Data log_tbl_login