MODUL

SISTEM BASIS DATA

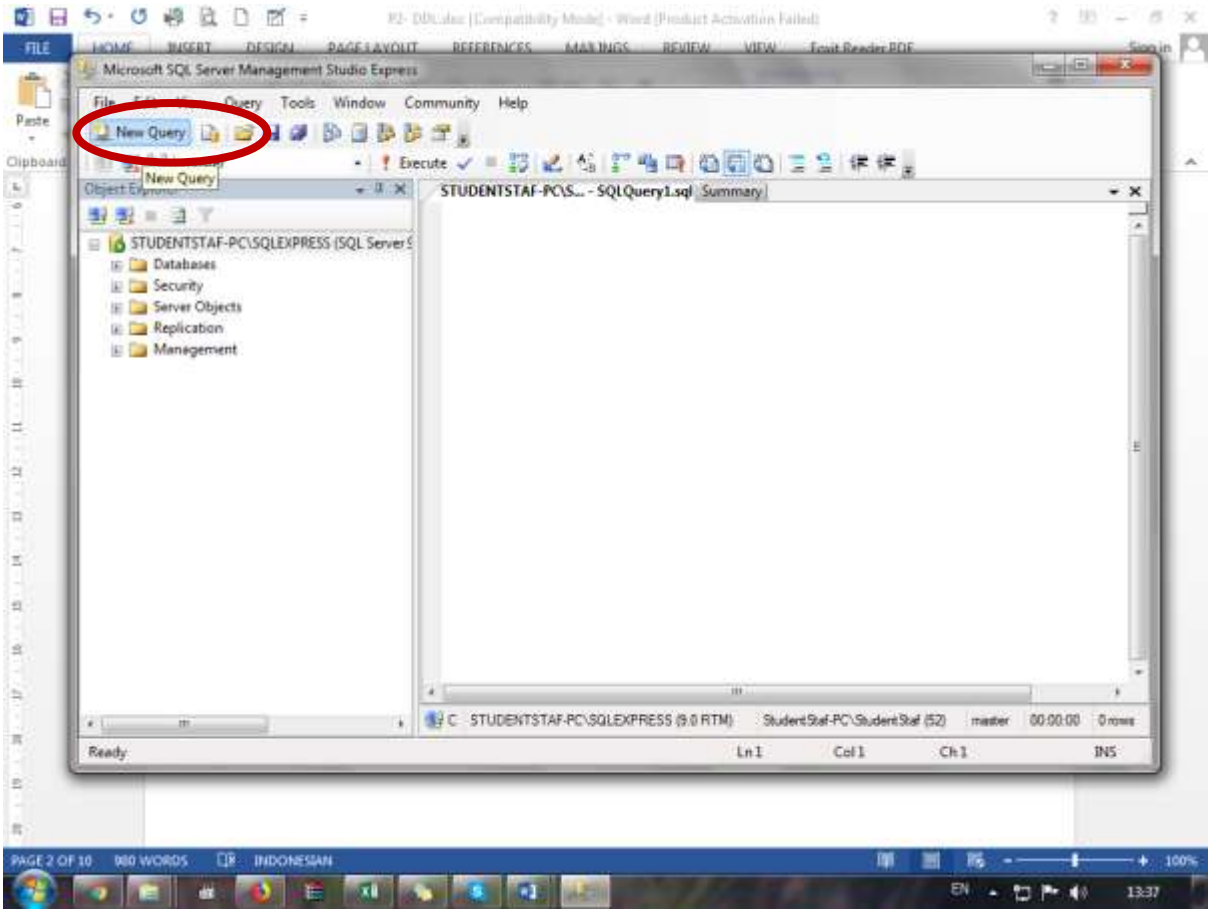
TUJUAN :

1. Mampu membuat membuat database menggukan perintah query.
2. Mampu membuat tabel dengan perintah query.
3. Mampu membuat constraint relasi antar tabel dan mendefinisikan primary key.
4. Mampu mengimplenetasi perintah-perintah Data Definition Language (DDL) seperti drop, alter, dan lain-lain.

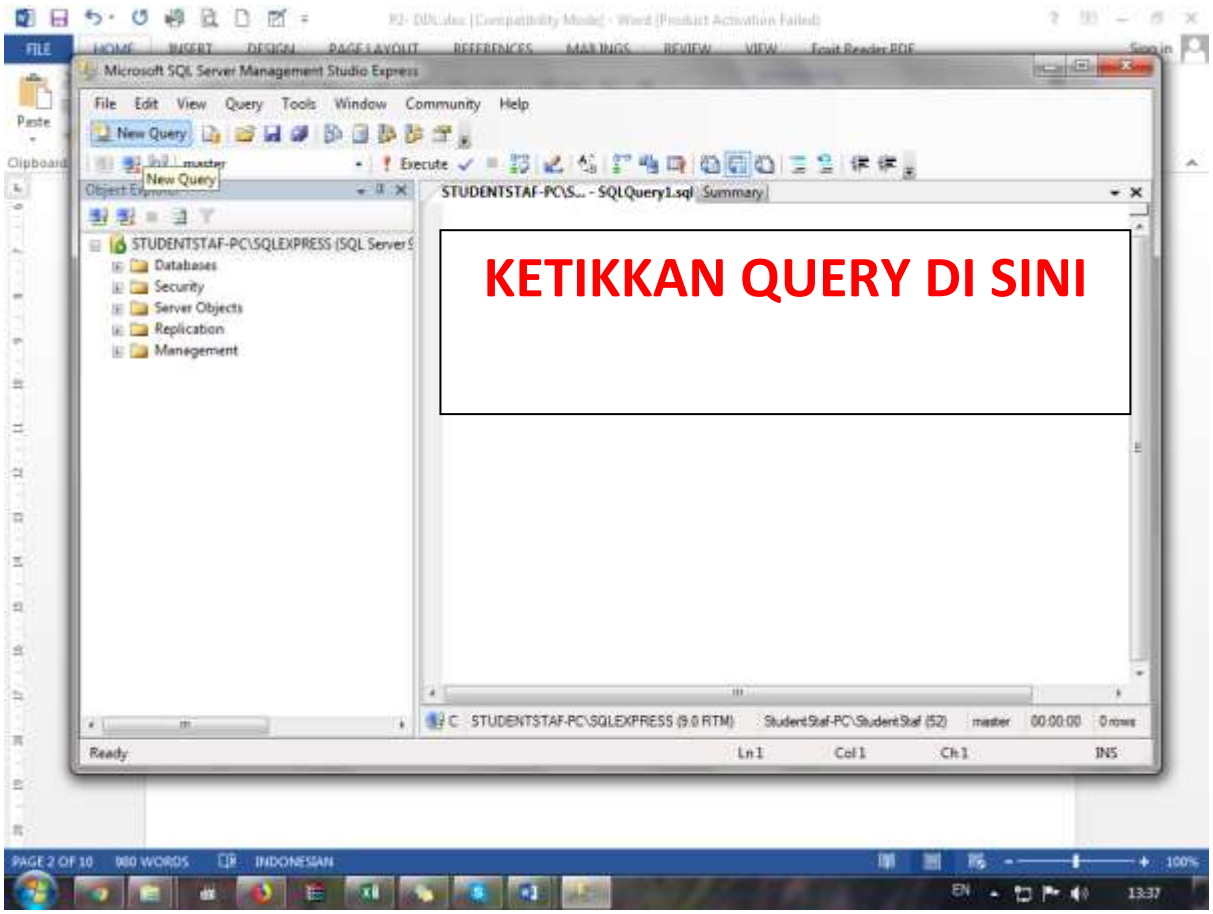
1. PENGERTIAN DATA DEFINITION LANGUAGE (DDL)

Data Definition Language (DDL) merupakan perintah SQL yang digunakan untuk mendefinisikan dan mendeklarasikan suatu objek database, menciptakan objek database atau menghapus objek database. Objek database tersebut dapat berupa database dan tabel. DDL dapat juga digunakan untuk membuat koneksi antar tabel dalam database beserta batasannya dengan menentukan kolom indeks sebagai kuncinya. DDL yang biasa digunakan adalah CREATE, DROP dan ALTER.

Dalam praktikum kali ini, kita menggunakan program MS SQL Server 2005. Untuk menjalankan perintah SQL kita akan menggunakan Query Analyzer yang telah disediakan oleh SQL Server 2005. Cara menggunakan fitur query analyzer pada Ms. SQL Server 2005 adalah dengan mengklik fitur new query seperti gambar di bawah ini.



Setelah di klik makan akan muncul tampilan awal dari Query Analyzer adalah sbb :



Untuk mengeksekusi perintah SQL menggunakan tombol **F5** atau mengklik tombol **Execute**.

1. Membuat Database dan Tabel

2.1. Membuat Database

CREATE digunakan untuk membuat objek database baru. Sebagai contoh adalah untuk membuat database baru, tabel baru, dan lain-lain.

```
CREATE DATABASE Nama_database

Contoh:
```

Sedangkan untuk membuat basis data menggunakan parameter querynya adalah sebagai berikut.

```
CREATE DATABASE Nama_Database
ON
(
NAME = 'Nama_File_Primer',
```

```
FILENAME = 'Lokasi_File_Primer',
SIZE = Ukuran,
MAXSIZE = Ukuran Maksimal,
FILEGROWTH = Pertambahan File
)
LOG ON
(
NAME = 'Nama_File_log',
FILENAME = 'Lokasi_File_Log',
SIZE = Ukuran,
MAXSIZE = Ukuran Maksimal,
FILEGROWTH = Pertambahan File
)
```

Contoh lain dengan tambahan parameter, membuat database Penjualan :

```
CREATE DATABASE PENJUALAN
ON
( NAME = PENJUALAN_dat,
   FILENAME = 'D:\PENJUALANdat.mdf',
   SIZE = 10,
   MAXSIZE = 50,
   FILEGROWTH = 5 )
LOG ON
( NAME = 'PENJUALAN_log',
   FILENAME = 'D:\PENJUALANlog.ldf',
   SIZE = 5MB,
   MAXSIZE = 25MB,
   FILEGROWTH = 5MB )
```

2.2. Mengaktifkan Database

Sebelum membuat tabel maka anda harus mengaktifkan basis data yang akan dibuatkan tabelnya dengan menggunakan perintah **USE.**

Sintaks :

```
USE DatabaseName
```

Contoh :

```
USE Penjualan
```

2.2. Membuat Tabel

Membuat tabel baru

```
CREATE TABLE nama_table(
Nama_field1 tipedata1 [ket1],
Nama_field2 tipedata2 [ket2],
```

```
.                                                    .    .
                                                     .    .
                                                     .
Nama_fieldN tipedataN [ketN])
```

Catatan:

- Sebelum membuat tabel maka anda harus mengaktifkan database yang akan dibuat tabelnya dengan perintah **USE**.

  Contoh:

```
USE PBD
```

- Tanda [ ] berarti bahwa perintah tersebut bersifat optional, artinya boleh diikutsertakan atau boleh juga diabaikan, tergantung dari keperluan. Misalnya, jika kita ingin membuat primary key pada table yang kita buat, maka kita harus menyertakan kata kunci primary key.

Contoh :

```
CREATE TABLE dokter(
kd_dokter varchar(5) primary key not null,
nama varchar(30),
gender char(1) check(gender='L' or gender='P'),
alamat varchar(30),
gaji numeric
)
```

Catatan :

- Null tidak sama dengan nol, tetapi memiliki arti bahwa kolom atau field tersebut tidak ada data yang dimasukkan.
- Not null menyatakan bahwa data pada kolom tersebut tidak boleh kosong dan harus diisi.

Contoh:

```
CREATE TABLE resep(
kd_resep int identity(1,1) primary key not null,
hari varchar(10) CHECK
(hari IN
('senin','selasa','rabu','kamis', 'jumat','sabtu')),
tanggal datetime,
kd_pasien varchar(5)
```

```
foreign key references pasien(kd_pasien),
kd_penyakit varchar(5)
foreign key references penyakit(kd_penyakit),
kd_obat varchar(5)
foreign key references obat(kd_obat),
kd_dokter varchar(5)
foreign key references dokter(kd_dokter)
)
```

Identity atau yang biasa disebut dengan istilah *autonumber* adalah nilai yang dihasilkan oleh sistem secara otomatis dan terurut sesuai dengan nilai urutan yang dimasukkan. Contoh di atas kolom kode resep (**kd_resep int identity(1,1)**) akan terurut mulai dari angka 1 dengan pertambahan pengurutan 1 nilai.

Bentuk umum dari identity adalah:

```
Nama_kolom tipe_data identity (m,n) [ket]
```

Dengan m adalah nilai awal sedangkan n adalah pertambahan nilainya.

2.3. Menghapus Object

DROP merupakan perintah SQL yang digunakan untuk menghapus objek database.

```
DROP Nama_objek
```

Contoh:

```
DROP DATABASE PBD
DROP TABLE dokter
```

2.4. Mengubah Object

ALTER digunakan untuk menambah kolom, mengubah kolom atau menghapus kolom dari sebuah tabel. ALTER juga bisa digunakan untuk mengubah sebuah database. Sintaks untuk mengubah tabel menggunakan perintah ALTER adalah:

```
ALTER TABLE nama_table <ACTION>
```

<ACTION> bisa berupa :

1. Manambah field

   ADD nama_field tipe_data

2. Menghapus field

   DROP nama_field

3. Memodifikasi field

   ALTER nama_field tipe_data

4. Menambah constraint

   ADD CONSTRAINT nama_constraint definisi_constraint

5. Menghapus constraint

   DROP CONSTRAINT nama_constraint

Contoh:

Menambah kolom agama pada tabel dokter dengan type data varchar(10),

```
ALTER TABLE dokter ADD agama varchar(10)
```

Jika hanya ada 6 agama yang boleh dimasukkan ke dalam kolom agama, maka pada tabel dokter maka kita bisa memakai perintah :

```
ALTER TABLE dokter
ADD CONSTRAINT cek_agama
CHECK (agama IN
('islam','protestan','katolik','hindu','budha','konghuchu'))
```

Perintah IN mempunyai arti semua data dalam kurung adalah benar. IN merupakan alternatif untuk mengganti perintah OR yang terlalu banyak. Seperti contoh yang sama di bawah ini :

```
ALTER TABLE dokter
ADD CONSTRAINT cek_agama
CHECK (agama ='islam'
OR agama='protestan'
OR agama='katolik'
OR agama='hindu'
OR agama='budha'
OR agama='konghuchu')
```

Menghapus constraint cek_agama

```
ALTER TABLE dokter
DROP CONSTRAINT cek_agama
```

2. Praktikum

1. Buatlah database dengan nama SI_rumahsakit_xxxx (4 digit terakhir NIM anda) dan simpan file .mdf dan .ldf pada folder anda masing-masing.
2. Kemudian buat tabel pada database menggukan query dengan ketentuan sebagai berikut: (khusus untuk nomor 2 ini tidak perlu dituliskan pada answer sheet)

Tabel dokter

Kolom	Tipe Data	Keterangan
Kd_dokter	Varchar(5)	Primary key not null
Nama	Varchar(30)	Not null
Alamat	Varchar(30)	
Gender	Char(1)	Check(L/P)
Gaji	Numeric	

Tabel pasien

Kolom	Tipe Data	Keterangan
Kd_pasien	Varchar(5)	Primary key not null
Nama	Varchar(30)	Not null
Alamat	Varchar(30)	
Gender	Char(1)	Check(L/P)
Periksa	Numeric	

Tabel penyakit

Kolom	Tipe Data	Keterangan
Kd_penyakit	Varchar(5)	Primary key not null
Penyakit	Varchar(30)	Not null

Tabel obat

Kolom	Tipe Data	Keterangan
Kd_obat	Varchar(5)	Primary key not null
Obat	Varchar(30)	Not null

Tabel jadwal_dokter

Kolom	Tipe data	Keterangan
Kd_jadwal	Varchar(5)	Primary key not null
Hari	Varchar(10)	Check (senin,selasa,rabu,kamis,jumat,sabtu)
Shift	Varchar(10)	Check(pagi/sore)
Kd_dokter	Varchar(5)	Foreign key (dokter.kd_dokter)

Tabel resep

Kolom	Tipe Data	Keterangan
Kd_resep	Int	Identity (1,1) Primary key not null
Hari	Varchar(10)	Check (senin,selasa,rabu,kamis,jumat,sabtu)
Tanggal	Datetime	
Kd_pasien	Varchar(5)	Foreign key (pasien.kd_pasien)
Kd_penyakit	Varchar(5)	Foreign key (penyakit.kd_penyakit)
Kd_obat	Varchar(5)	Foreign key (obat.kd_obat)
Kd_dokter	Varchar(5)	Foreign key (dokter.kd_dokter)

3. Ubahlah nama atribut **Tanggal** pada **tabel resep** menjadi "**tgl_resep**" dengan tipe data yang sama.
4. Berikan fungsi check pada tabel paseie atribut gender dengan nilai hanya P dan L.