

Lab.Info®matika
Fakultas Teknik Univ. Dr. Soetomo
Surabaya

Praktikum Pemrograman Basis Data Client Server
(SQL Server 2000 & Visual Basic 6)

Bab I. Membangun Database

Bab II. SQL

Bab III. Transact-SQL

Bab IV. Stored Procedure & Trigger

Bab V. Pemrograman dengan ADO

Bab VI. Class yang berhubungan dengan Data (class data aware)

Bab VII. Crystal Report

Bab I

Membangun Database

1.1.Tujuan

- a. Mahasiswa mampu merancang database sederhana.
- b. Mahasiswa mampu mengaplikasikan Data Definition Language (DDL) secara visual.

1.2.Materi

- a. Database Management System
- b. Data Definition Language

1.3.Alat dan bahan

- a. PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- b. Microsoft SQL Server 2000

1.4.Teori

Database adalah sebuah object yang kompleks untuk menyimpan informasi yang terstruktur, yang diorganisir dan disimpan dalam suatu cara yang memungkinkan pemakainya dapat mengambil informasi dengan cepat dan efisien.

- DBMS menyediakan fasilitas untuk mendefinisikan struktur dari database dengan pernyataan SQL. Subbagian dari pernyataan SQL yang mendefinisikan dan mengedit struktur ini disebut dengan Data Definition Language (DDL). Saat ini hampir semua DBMS menggunakan antarmuka (user interface) visual untuk mendefinisikan dan mengedit struktur ini. Yang selanjutnya piranti ini menerjemahkan tindakan user ke dalam pernyataan DDL yang sesuai. Sebagai contoh, SQL Server menyediakan tool untuk membuat database dengan piranti visual seperti *enterprise manager*, tetapi ia juga menghasilkan pernyataan DDL yang sesuai dan menyimpannya ke dalam sebuah file khusus yang disebut dengan Script.
- DBMS terdapat fasilitas untuk memanipulasi informasi yang disimpan di dalam database dengan pernyataan SQL. Subbagian dari pernyataan SQL yang memanipulasi informasi ini disebut dengan Data Manipulation Language (DML). Operasi dasar untuk memanipulasi data seperti menambah (*insert*)

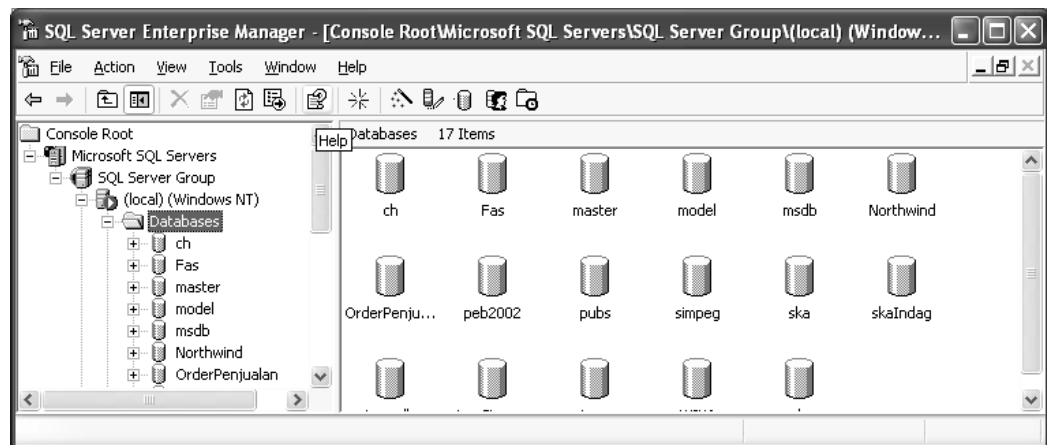
record baru, memodifikasi (*update*) dan menghapus (*delete*) record yang sudah ada, serta mengambil (*select*) record.

- DBMS melindungi integritas database dengan menerapkan aturan, yang dimasukkan ke dalam perancangan database tersebut. Kita bisa menentukan nilai default, tidak mengizinkan field tertentu kosong, melarang penghapusan record yang terhubung ke record lain, dan sebagainya.

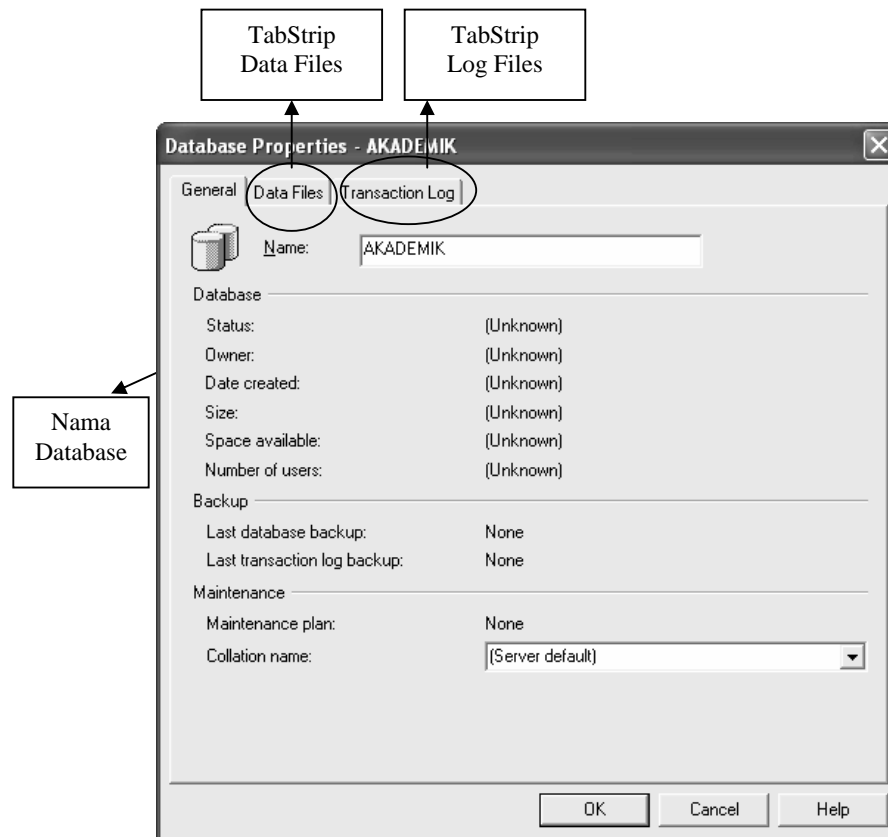
1.5. Latihan

1.5.1. Membuat Database

1. Jalankan SQL Server Enterprise Manager
2. Click SQL Server Group
3. Click Pada Sever yang aktif
4. Click Database hingga tampak seperti gambar berikut:



5. Pilih Menu Action dan New Database
6. Masukkan nama Database pada textbox nama, masukkan nama AKADEMIK. Untuk melihat atau merubah dimana file database (ada dua file Data File .MDF dan Log File .LDF) disimpan click Tabstrip Data files atau Transaction Log. Di kedua Tabstrip ini anda juga dapat menentukan tingkat pertumbuhan masing-masing file dengan persen atau dalam ukuran MegaByte, juga dapat menentukan maksimum ukuran file yang anda kehendaki.

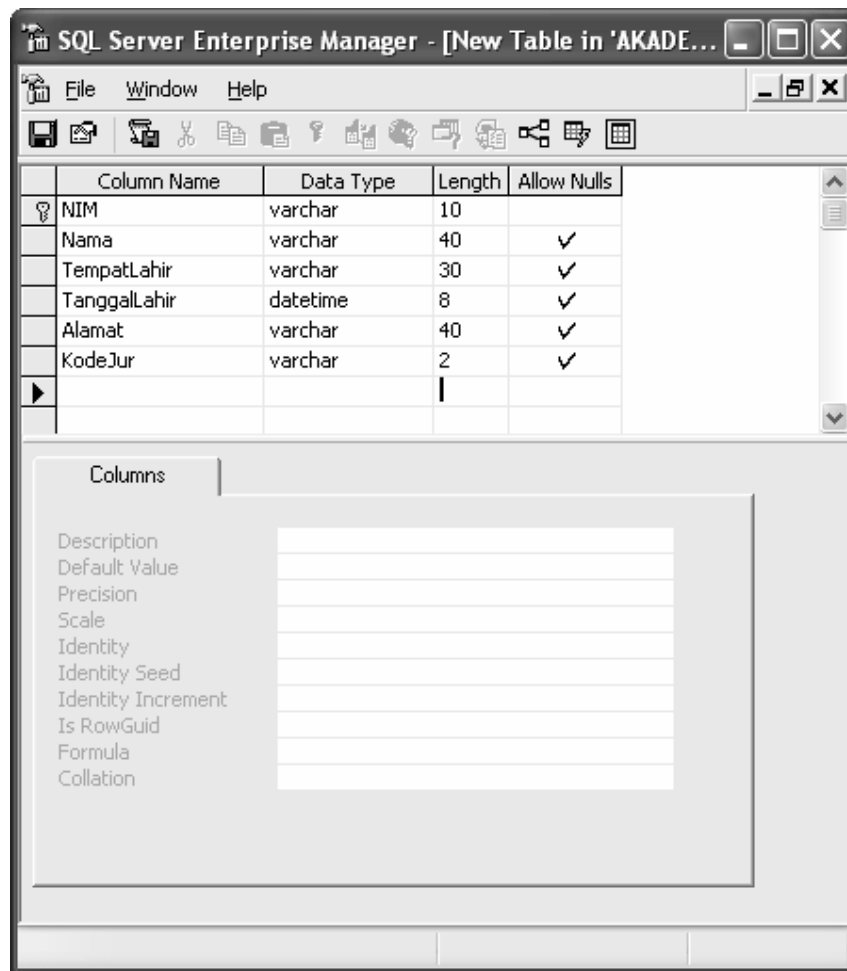



7. Click OK

8. Lihat pada SQL Server Enterprise Manager, Treeview dibawah Database akan muncul nama database baru yang anda buat.

1.5.2. Membuat Tabel


1. Click object table dibawah database AKADEMIK
2. Pilih menu Action dan New Table hingga tampak seperti gambar di bawah ini
3. Isi table seperti pada gambar di bawah ini.



4. Tentukan primary key dengan arahkan kursor pada nim dan click tanda kunci  pada toolbar di atas kolom. Amati perubahannya, penunjuk baris pada kiri NIM akan berubah menjadi tanda kunci dan kolom Allow Null Check dihilangkan. Artinya NIM menjadi Primary Key dan tidak boleh Null (harus diisi).
5. Click toolbar save (disket) masukkan nama table (MAHASISWA)
6. Close window table, lihat di window sebelah kanan aka terlihat nama table MAHASISWA berada di bawah table-table system SQL-Server.

1.5.3. Relasi antar Tabel

Sebelum kita memulai Latihan ini, buat satu table seperti langkah membuat table di atas dengan nama JURUSAN dengan kamus data sebagai berikut:

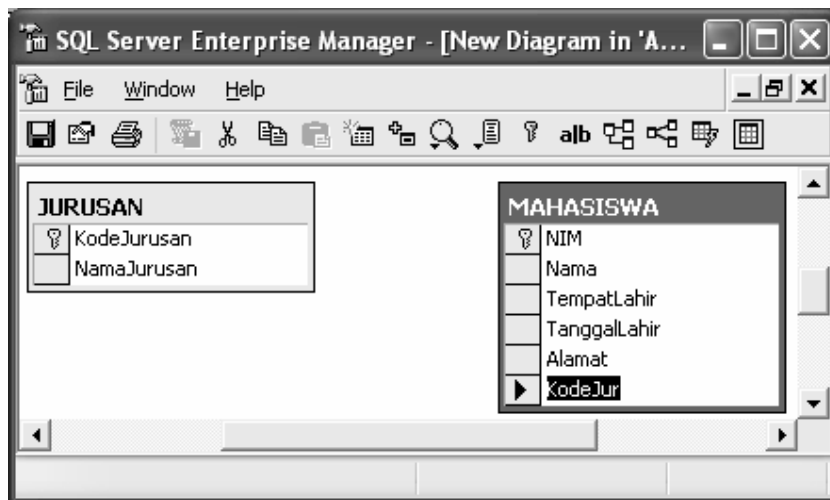
	<i>KodeJurusan</i>	<i>Varchar</i>	<i>2</i>
	<i>NamaJurusan</i>	<i>Varchar</i>	<i>30</i>

Selanjutnya ikuti langkah berikut untuk merelasikan kedua table di atas:

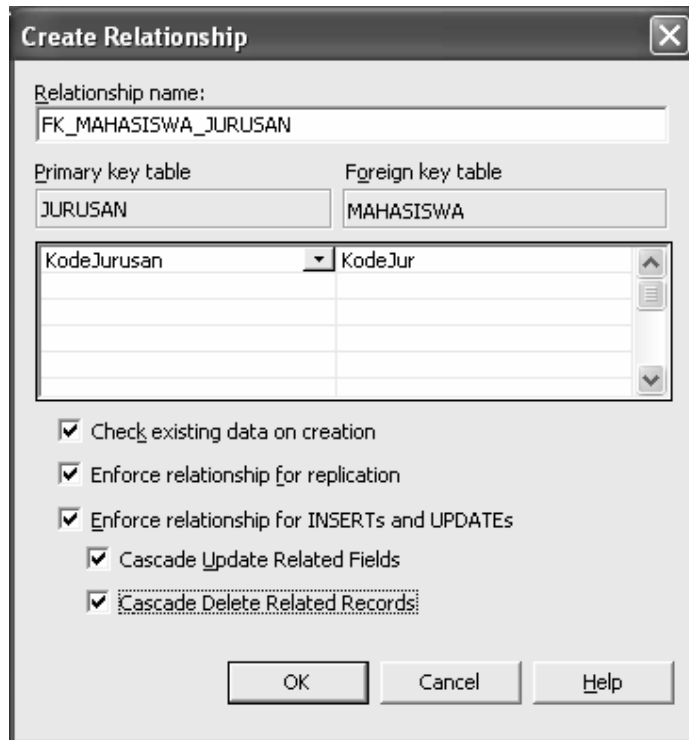
1. Click object Diagrams di bawah database AKADEMIK
2. Pilih menu Action dan New Database Diagrams.
3. Akan muncul dialog Create Database Diagram Wizard, click Next
4. Pilih table yang akan direlasikan, pilih JURUSAN click Add, Pilih MAHASISWA click Add. Lanjutkan langkah berikut dengan Click Next



5. Click Finish
6. Akan ditampilkan dialog new diagram seperti gambar berikut:



7. Pilih field KodeJur pada table MAHASISWA Drag dan drop pada tabel JURUSAN. Selanjutnya akan ditampilkan dialog



8. Aktifkan option Cascade Update related Fields dan Cascade Delete Related Records. Jika Cascade Update related Fields diaktifkan maka pada saat KodeJurusan pada table JURUSAN diganti secara otomatis KodeJur pada table MAHASISWA akan berubah. Jika Cascade Delete Related Records diaktifkan maka pada saat record JURUSAN dihapus semua record pada table MAHASISWA yang berelasi dengan record tersebut akan terhapus semua. Hati-hati penggunaan opsi ini, karena dapat menghapus data secara otomatis.
9. Click OK
10. Simpan dengan nama erAKADEMIK.
11. Akan muncul message box tentang table-tabel yang direlasikan, click Yes
12. Keluar dari dialog diagram, amati pada object diagram akan terdapat satu object dengan nama erAKADEMIK. Untuk satu database boleh dibuat lebih dari satu diagram, apabila jumlah table sangat banyak.

1.6. Tugas

1.6.1. Tambahkan Tabel

- Buat table – table berikut seperti langkah di atas

Table	FAKULTAS		
No	Nama	Tipe Data	Panjang
1	<u>KodeFakultas</u>	varchar	2
2	NamaFakultas	varchar	30

Table	MataKuliah		
No	Nama	Tipe Data	Panjang
1	<u>KodeMK</u>	varchar	6
2	NamaMK	varchar	40
3	SKS	tinyint	1

Table	KRS		
No	Nama	Tipe Data	Panjang
1	<u>NIM</u>	Varchar	10
2	<u>KodeMK</u>	Varchar	6
3	NilaiUTS	Decimal	9
4	NilaiUAS	Decimal	9

*Catatan : Nama field bergaris bawah berarti **primary key***

1.6.2. Relasi antar Tabel

- Modifikasi table jurusan tambahkan satu field (KodeFakultas , varchar, 2)
- Relasikan table JURUSAN dengan Fakultas
- Relasikan table KRS dengan MAHASISWA dan MATAKULIAH

1.6.3. Manipulasi data dan Amati perubahan data berdasarkan opsi Cascade

Update dan Cascade Delete

Isi data sebanyak dan sevariatif mungkin, lakukan penghapusan pada record yang tertentu amati perubahan record pada table yang berelasi. Demikian juga untuk perubahan data terutama perubahan data yang sebagai primary key, amati table lain yang berelasi apakah bila kodejurusan pada table jurusan dirubah maka kodejur pada mahasiswa juga berubah? Apakah bila salah satu record pada table jurusan dihapus record-record pada table MAHASISWA juga akan terhapus?

Bab II

Structured Query Language (SQL)

2.1.Tujuan

- a. Mahasiswa mengerti bahasa dibalik DBMS.
- b. Mahasiswa dapat mengaplikasikan perintah-perintah Data Manipulation Language (DML).

2.2.Materi

- a. Selection Query
- b. Action Query

2.3.Alat dan bahan

- a. PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- b. Microsoft SQL Server 2000

2.4.Teori dan Latihan

Pernyataan pada DML dari bahasa SQL juga dikenal dengan istilah *query*, terdapat dua jenis query yaitu selection query dan action query. Selection query mengambil informasi dari database dan tidak memodifikasinya. Semua selection query diawali dengan pernyataan SELECT. Action Query memodifikasi data pada table-table database dan diawali dengan pernyataan INSERT, UPDATE atau DELETE.

Untuk mencoba SQL statement di bawah ini jalankan dulu bagian program dari SQL Server Query Analyzer dan aktifkan database AKADEMIK anda.

Selection Query

Pernyataan umum SELECT adalah sbb:

```
SELECT select_list
[ INTO new_table ]
FROM table_source
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

Select_list

Adalah kumpulan dari field-field yang dipisahkan dengan tanda koma atau * untuk menampilkan semua field yang ada pada suatu table atau query.

Contoh:

Berikut adalah contoh perintah select yang paling sederhana untuk menampilkan semua field dan semua record MAHASISWA

```
Select * from MAHASISWA
```

Jika kita hanya ingin menampilkan field NIM dan Nama saja maka:

```
Select NIM, NAMA from MAHASISWA
```

New_Table

INTO *new_table* merupakan klausa optional apabila kita ingin mengkopikan hasil dari select pada table yang baru (*new_table*). Apabila klausa ini kita sertakan maka akan secara otomatis membuat table baru dengan isi yang sama persis dengan hasil select.

Contoh:

Berikut adalah contoh perintah select yang akan mengkopikan semua record MAHASISWA ke tabel baru yang bernama *CopyOfMHS*

```
Select * INTO CopyOfMHS from MAHASISWA
```

Jika kita hanya ingin menampilkan field NIM dan Nama saja maka:

```
Select NIM, NAMA from MAHASISWA
```

Table_Source

Tabel sumber dari perintah select, yaitu nama-nama tabel, view, atau bahkan nama alias dari suatu query dapat kita gunakan disini.

WHERE *search_condition*

Klausa ini juga optional, yang digunakan untuk memfilter atau mencari record dengan kondisi tertentu.

Contoh:

Untuk mencari mahasiswa jurusan informatika (kode jurusan = 42) gunakanlah kondisi berikut:

```
Select * from MAHASISWA WHERE KodeJur='42'
```

Untuk menampilkan Nim, Nama dan jurusan dapat digunakan klausa where seperti berikut:

```
Select Nim, Nama, NamaJurusan from MAHASISWA, JURUSAN  
WHERE MAHASISWA.KodeJur=JURUSAN.KodeJurusan
```

Keyword TOP

Keyword ini digunakan untuk mengambil beberapa record teratas sesuai dengan nilai yang kita masukkan setelah keyword TOP

Contoh:

Untuk menampilkan Mahasiswa 10 baris teratas adalah sbb:

```
SELECT TOP 10 * from MAHASISWA
```

Operator LIKE

Seringkali kita menginginkan hasil dari suatu query dengan hasil yang tidak benar-benar tepat misalnya untuk menampilkan semua mahasiswa yang angkatan tahun 2003 dapat kita gunakan operator like sbb:

```
SELECT * from MAHASISWA
WHERE NIM like '2003%'
```

Atau kita ingin menampilkan semua mahasiswa yang namanya mengandung kata ADI dan beralamat di SURABAYA dapat digunakan operator like sbb:

```
SELECT * from MAHASISWA
WHERE NAMA like '%adi%' and ALAMAT like '%SURABAYA%'
```

Tanda % pada query diatas disebut dengan karakter *wildcard*, pada SQL Server terdapat beberapa macam karakter wildcard seperti tabel di bawah ini:

WildCard	Fungsi
%	Mencocokkan semua karakter. Seperti '2003%' akan mencari NIM 2003420121, 2003411234, dsr. Perhatikan bahwa % dapat digunakan sebagai karakter pertama, terakhir atau pertama dan terakhir seperti pada %adi% akan mencari string yang mengandung kata adi seperti adi, adi handoko, lumadya adi, dimas hadinata dst.
_	Garis Bawah mencocokkan satu karakter alfabet. Pola b_s akan mencari kata bos, bis, bus , tetapi bukan business
[]	Mencocokkan dengan salah satu karakter di dalam tanda kurung, pola Pol[yi]gon , akan mencari kata-kata Polygon dan Poligon
[^]	Mencocokkan setiap karakter yang tidak terdapat dalam tanda kurung. Pola Pol[^i]gon akan mencari kata yang mengandung Pol dan gon yang tidak mengandung karakter i, hasilnya Polygon, Polugon, Polkgon dst yang bukan kata Poligon
[-]	Mencocokkan karakter pada range tertentu. Pola %[a-c] akan mencari kata yang berakhiran a, b atau c saja .
#	Mencocokkan satu karakter numerik. Pola Rp. ##.000 akan mencari Rp. 10.000, Rp. 00.000, Rp. 99.000 dst

NULL value

Nilai Null sering ada dalam suatu field, null tidak sama dengan string kosong (") atau nol (0), untuk membandingkan suatu field atau variabel bernilai null atau bukan kita tidak dapat menggunakan sama dengan (=) tetapi dengan operator IS.

Contoh:

Untuk mencari Mahasiswa yang alamatnya kosong atau null

```
SELECT * from MAHASISWA
WHERE ALAMAT IS NULL
```

Sedangkan untuk mencari MAHASISWA yang alamatnya terisi

```
SELECT * from MAHASISWA
WHERE ALAMAT IS NOT NULL
```

Operator UNION

Operator UNION berfungsi untuk menggabungkan hasil dari dua query atau lebih menjadi satu.

Contoh:

Untuk mencari semua orang yang pernah dan sedang kuliah kita harus menggabungkan tabel alumni dan tabel mahasiswa sbb:

```
SELECT NIM, NAMA, 'MHS' as STATUS from MAHASISWA
UNION
SELECT NIM, NAMA, 'ALUMNI' as STATUS from ALUMNI
```

Keyword DISTINCT

Keyword DISTINCT digunakan untuk menghilangkan baris yang sama persis dari suatu query.

Contoh:

Untuk melihat matakuliah apa saja yang diambil mahasiswa dari tabel KRS, dapat kita buat dengan keyword DISTINCT, jika tanpa DISTINCT maka akan muncul banyak nama mata kuliah.

```
SELECT DISTINCT KRS.KodeMK, NamaMK
FROM KRS, MATAKULIAH
WHERE KRS.KodeMK = MATAKULIAH.KodeMK
```

Keyword ORDER BY

Digunakan untuk mengurutkan baris sesuai dengan kehendak kita.

Contoh:

Kita akan menampilkan semua mahasiswa yang diurutkan berdasarkan KodeJur kemudian NIM, sbb:

```
SELECT * FROM MAHASISWA
ORDER BY KodeJur, NIM
```

Kolom Hasil perhitungan

Seringkali kita membutuhkan perhitungan-perhitungan untuk baris-baris tertentu, misalnya untuk menghitung nilai akhir dari nilai UTS dan UAS dengan prosentase 40% * UTS ditambah 60% * UAS, maka dalam query harus kita buat sbb:

```
SELECT KRS.NIM, MAHASISWA>Nama, KRS.KodeMK, NamaMK, NilaiUTS,
NilaiUAS,
(NilaiUTS*0.4 + NilaiUAS * 0.6) as [Nilai Akhir]
from KRS,MAHASISWA, MATAKULIAH
where KRS.NIM=MAHASISWA.NIM and KRS.KodeMk=MATAKULIAH.KodeMK
```

Selain perhitungan sederhana seperti di atas kita juga dapat melakukan perhitungan-perhitungan khusus dengan *aggregate function*, seperti tabel berikut:

Agregate Function	Result
SUM([ALL DISTINCT] <i>expression</i>)	Menjumlahkan nilai dari suatu ekspresi numeric.
AVG([ALL DISTINCT] <i>expression</i>)	Nilai rata-rata dari suatu ekspresi numeric.
COUNT([ALL DISTINCT] <i>expression</i>)	Jumlah dari suatu ekspresi
COUNT(*)	Jumlah dari record yang dipilih
MAX(<i>expression</i>)	Nilai tertinggi dari ekspresi yg ditentukan
MIN(<i>expression</i>)	Nilai terendah dari ekspresi yg ditentukan

SUM, AVG, COUNT, MAX, dan MIN ignore null values; COUNT(*) tidak.

Klausula GROUP BY dan HAVING

Group By digunakan untuk mengelompokkan record berdasarkan kolom (field) setelah klausa tersebut.

Contoh:

Misalnya kita ingin menampilkan jumlah mahasiswa dikelompokkan per jurusan dapat kita gunakan query sbb:

```
SELECT NamaJurusan, Count(NIM) as [Jumlah MHS]
from MAHASISWA, JURUSAN
WHERE MAHASISWA.KodeJur = JURUSAN.KodeJurusan
Group By NamaJurusan
Order By NamaJurusan
```

Having digunakan untuk memberikan batasan pada klausa group by, misalnya kita hanya ingin menampilkan nama jurusan yang jumlah mahasiswa per angkatan lebih dari 10, maka:

```
SELECT NamaJurusan, left(NIM,4) as Angkatan, Count(NIM) as [Jumlah MHS]
from MAHASISWA, JURUSAN
WHERE MAHASISWA.KodeJur = JURUSAN.KodeJurusan
Group By NamaJurusan, left(NIM,4)
HAVING Count(NIM) > 10
```

Keyword IN dan NOT IN

Keyword IN dan NOT IN digunakan bersama klausa WHERE untuk menentukan daftar nilai yang sesuai atau (tidak sesuai) dengan kolom tertentu. Kedua klausa ini sama halnya dengan notasi singkat untuk banyak operator OR.

Contoh:

Kita ingin menampilkan daftar mahasiswa yang jurusannya sipil (41) atau kode jurusannya informatika saja maka querynya adalah:

```
SELECT * from MAHASISWA where KodeJur = '41' or KodeJur='42'
```

Query diatas dapat kita singkat dengan IN menjadi

```
SELECT * from MAHASISWA where KodeJur IN ('41', '42')
```

Atau bila kita menginginkan sebaliknya, kita akan menampilkan mahasiswa yang bukan kedua jurusan diatas kita hanya tinggal menambahkan operator NOT di depan IN, sbb:

```
SELECT * from MAHASISWA where KodeJur NOT IN ('41', '42')
```

Subquery

Subquery adalah sebuah query biasa yang cursor hasilnya digunakan sebagai bagian dari query yang lain.

Contoh:

Misalnya kita ingin menampilkan semua Mahasiswa yang tidak melakukan KRS, maka kita harus mencari mahasiswa yang NIM-nya tidak ada di tabel KRS, sbb:

```
SELECT * from MAHASISWA
WHERE NIM NOT IN
(SELECT NIM from KRS)
```

Keyword BETWEEN

BETWEEN berfungsi untuk menentukan nilai pada suatu range tertentu dan membatasi baris-baris yang sesuai dengan nilai pada range tersebut. Between merupakan notasi singkat dari ekspresi:

Field >= MinValue AND Field <= MaxValue

Contoh:

Misalnya kita ingin tahu mahasiswa yang lahir mulai tanggal 01/01/1960 sampai tanggal 31/12/1980, maka

```
SELECT * FROM MAHASISWA
```

WHERE TanggalLahir BETWEEN '01/01/1960' AND '31/12/1980'

Menggabungkan table dengan JOIN

Jika pada latihan sebelumnya jika kita ingin menampilkan query yang melibatkan banyak tabel kita gunakan klausa where, tetapi klausa WHERE hanyalah untuk mengekspresikan batasan yang melibatkan satu atau lebih tabel. Metode yang tepat untuk menghubungkan tabel adalah dengan operasi JOIN. Mengapa harus dengan join? Coba kita perhatikan query berikut:

```
Select Nim, Nama, NamaJurusan from MAHASISWA, JURUSAN  
WHERE MAHASISWA.KodeJur=JURUSAN.KodeJurusan
```

Untuk dapat memahami keempat jenis join berikut, sebelumnya tambahkan satu baris pada tabel MAHASISWA dan kosongi KODEJUR-nya, dan tambahkan satu baris pada tabel JURUSAN yang KODEJURUSAN-nya tidak digunakan pada tabel MAHASISWA. Dan asumsikan bahwa tabel MAHASISWA adalah tabel sebelah KIRI sedangkan Tabel JURUSAN tabel sebelah KANAN

Query tersebut tidak akan dapat menampilkan mahasiswa yang KodeJur-nya NULL, karena nilai NULL tidak akan pernah ketemu pada tabel JURUSAN.

Penggabungan atau JOIN dapat diklasifikasikan menjadi:

INNER JOIN, penggabungan ini akan menghasilkan pasangan-pasangan baris yang cocok pada kedua tabel dan membuang baris-baris yang tidak ada pasangannya pada kedua tabel.

Contoh:

```
select NIM, NAMA, NAMAJURUSAN  
from MAHASISWA INNER JOIN JURUSAN  
ON MAHASISWA.KODEJUR= JURUSAN.KODEJURUSAN
```

LEFT JOIN, penggabungan LEFT JOIN ini menghasilkan pasangan-pasangan baris yang cocok pada kedua tabel ditambah dengan baris dari tabel sebelah kiri yang tidak mempunyai pasangan pada tabel sebelah kanan.

Contoh:

```
select NIM, NAMA, NAMAJURUSAN  
from MAHASISWA LEFT JOIN JURUSAN  
ON MAHASISWA.KODEJUR= JURUSAN.KODEJURUSAN
```

RIGHT JOIN, penggabungan LEFT JOIN ini menghasilkan pasangan-pasangan baris yang cocok pada kedua tabel ditambah dengan baris dari tabel sebelah Kanan yang tidak mempunyai pasangan pada tabel sebelah kiri.

Contoh:

```
select NIM, NAMA, NAMAJURUSAN  
from MAHASISWA RIGHT JOIN JURUSAN  
ON MAHASISWA.KODEJUR= JURUSAN.KODEJURUSAN
```

FULL JOIN, penggabungan LEFT JOIN ini menghasilkan pasangan-pasangan baris yang cocok pada kedua tabel ditambah dengan baris yang tidak mempunyai pasangan pada kedua tabel.

Contoh:

```
select NIM, NAMA, NAMAJURUSAN  
from MAHASISWA FULL JOIN JURUSAN  
ON MAHASISWA.KODEJUR= JURUSAN.KODEJURUSAN
```

Action Query

Action query digunakan untuk memanipulasi database yang terdiri dari tiga jenis tindakan yaitu *insert* untuk menambah, *update* untuk merubah, dan *delete* untuk menghapus record atau baris. Untuk lebih memahami ketiga perilaku dari tindakan tersebut, usahakan setelah anda melakukan query-query dibawah ini jalankan query select untuk mengetahui perubahannya.

Pernyataan umum **INSERT** adalah sbb:

```
INSERT [ INTO]
  { table_name }
  { [ ( column_list ) ]
    { VALUES ( [ ,...n] ) }
  }
```

Pernyataan INSERT berfungsi untuk menyisipkan atau menambah sebuah baris atau record baru ke dalam tabel.

Table_name

Adalah nama tabel yang akan kita tambah barisnya.

column_list

Adalah field dari tabel yang akan kita tambah barisnya, bila tidak disertakan berarti semua field.

*VALUES ([,...*n*])*

Adalah Nilai dari field yang akan kita tambahkan.

Contoh:

Berikut adalah contoh perintah INSERT yang akan menambahkan satu baris pada tabel JURUSAN.

```
INSERT JURUSAN (KODEJURUSAN, NAMA JURUSAN)
VALUES ('30', 'Studi Pembangunan')
```

Pernyataan umum **UPDATE** adalah sbb:

```
UPDATE
  { table_name }
  SET
  { column_name = { expression }
  [ WHERE
    < search_condition > ] }
```

Update digunakan untuk meng-update atau merubah kolom-kolom pada baris tertentu.

Table_name

Adalah nama tabel yang akan kita Update.

column_name

Adalah field atau kolom dari tabel yang akan kita rubah isinya dengan *expression*.

Search_condition

Kondisi untuk memilih record-record mana saja yang akan diupdate.

Contoh:

Berikut adalah contoh perintah UPDATE yang akan merubah isi kolom KODEJURUSAN dan NAMA JURUSAN pada tabel JURUSAN.

```
UPDATE JURUSAN
SET KODEJURUSAN='31', NAMA JURUSAN= 'Ekonomi Pembangunan'
WHERE KODEJURUSAN='30'
```

Pernyataan umum DELETE adalah sbb:

```
DELETE
[ FROM ]
  { table_name }
[ WHERE
  { < search_condition > } ]
```

Pernyataan DELETE berfungsi untuk menghapus baris atau record dari sebuah tabel.

Table_name

Adalah nama tabel yang akan dihapus barisnya.

Search_condition

Kondisi untuk memilih record-record mana saja yang akan dihapus.

Contoh:

Berikut adalah contoh perintah DELETE yang akan menghapus baris pada tabel JURUSAN yang kodejurusan-nya bernilai '31'.

```
DELETE JURUSAN
WHERE KODEJURUSAN='31'
```

2.5. Tugas

1. Dengan selection query diatas coba anda cari:
 - o Mata Kuliah yang belum diprogram oleh mahasiswa.
 - o Nilai akhir Tertinggi (MAX), Nilai akhir Terendah (MIN) dan Nilai akhir rata-rata (AVG) dari masing-masing matakuliah.Dengan ketentuan $[\text{Nilai Akhir}] = 0.4 * \text{Nilai UTS} + 0.6 * \text{Nilai UAS}$
2. Lakukan penghapusan pada semua baris di tabel Matakuliah yang matakuliahnya tidak diprogram mahasiswa.

Bab III

Transact - SQL

3.1.Tujuan

Mahasiswa mampu memanipulasi database lebih rumit yang tidak dapat ditangani dengan SQL dan tanpa menggunakan bahasa pemrograman tertentu.

3.2.Materi

- Variable T-SQL
- Pernyataan Alur Kendali
- Fungsi

3.3.Alat dan bahan

- a. PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- b. Microsoft SQL Server 2000

3.4.Teori

Sebagian besar system manajemen database mengandung eksistensi yang dapat meningkatkan SQL dan menjadikannya lebih seperti bahasa pemrograman. SQL Server menyediakan sekumpulan pernyataan yang dikenal dengan **Transact-SQL (T-SQL)**. T-SQL mengenali pernyataan-pernyataan yang mengambil baris-baris dari satu atau lebih table, pernyataan alur kendali seperti IF...ELSE dan WHILE, serta berbagai fungsi yang dapat digunakan untuk memanipulasi string, numeric dan tanggal, yang hampir sama dengan fungsi-fungsi Visual Basic. Dengan T-SQL kita bisa melakukan semua yang bisa dilakukan dengan SQL, serta memprogram operasi tersebut.

Variabel T-SQL

Variabel Lokal dan Tipe Data

Variabel lokal dideklarasikan dengan pernyataan DECLARE, dan namanya harus diawali dengan karakter @. Syntax dari pendeklarasian variabel ini adalah:

```
DECLARE @var_name var_Datatype
```

Contoh:

```
Declare @myName varchar(30)
Set @myName = 'Namaku Siapa'
PRINT @myName
```

Untuk tipe data silahkan lihat lampiran.

Variabel Global

Selain variabel lokal yang bisa dideklarasikan di dalam T-SQL, T-SQL juga mendukung sejumlah variabel global yang namanya diawali dengan @@. Nilai-nilai ini dipelihara oleh sistem dan kita hanya tinggal mengambil tanpa harus mendeklarasikannya.

Sebagai contoh anda dapat langsung mengambil nilai-nilai berikut:

```
print @@MAX_CONNECTIONS -- Maximum koneksi sekaligus yang dapat dilakukan
print @@SERVERNAME - - Nama server yang aktif
```

Masih banyak variabel-variabel global lain yang ada di SQL-Server.

Pernyataan alur kendali

T-SQL mendukung pernyataan alur kendali bisa digunakan untuk memilih mengeksekusi blok-blok pernyataan tertentu berdasarkan hasil dari suatu perbandingan. Dan alur kendali ini mirip dengan kebanyakan bahasa pemrograman lainnya.

IF...ELSE

Pernyataan ini mengeksekusi blok pernyataan secara bersyarat, dengan syntax untuk satu statement:

```
IF condition
    {statement}
ELSE
    {statement}
```

Sedangkan untuk lebih dari satu statement harus diawali dengan BEGIN dan diakhiri dengan END untuk setiap bloknya, dengan syntax sbb:

```
IF condition
    BEGIN
        {statement}
        ...
        {statement}
    END
ELSE
    BEGIN
        {statement}
        ...
        {statement}
    END
```

Contoh:

```
IF (SELECT COUNT(*) from MAHASISWA) > 0
    BEGIN
        SELECT * FROM MAHASISWA
    END
ELSE
    BEGIN
        PRINT 'Tabel Mahasiswa belum diisi'
    END
```

CASE

Pernyataan CASE berfungsi membandingkan nilai variabel atau field dengan beberapa nilai lain dan menjalankan block pernyataan, bergantung apakah hasil dari perbandingan tersebut bernilai TRUE.

Syntax:

```
Case var_name
    WHEN value1 THEN {statemen1}
    WHEN value2 THEN {statemen2}
    ...
    WHEN value_n THEN {statemen_n}
    ELSE {statemen_n+1}
End
atau
Case
    WHEN {Expression_1} THEN {statemen1}
    WHEN {Expression_2} THEN {statemen2}
    ...
    WHEN {Expression_n} THEN {statemen_n}
    ELSE {statemen_n+1}
End
```

Contoh:

Untuk menentukan nilai mahasiswa menjadi A,B,C,D atau E dari nilai UTS dan UAS kita bisa menggunakan pernyataan CASE, sbb:

```
Select KRS.NIM, Nama, KRS.KodeMK, NAMAMK,
NilaiUTS*0.4+NilaiUAS*0.6 as NA,
case
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 76 then 'A'
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 66 then 'B'
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 56 then 'C'
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 36 then 'D'
    else 'E'
end as NH
from MAHASISWA inner JOIN KRS on MAHASISWA.NIM = KRS.NIM
INNER JOIN MATAKULIAH on KRS.KodeMK = MATAKULIAH.KodeMk
```

WHILE, BREAK & CONTINUE

Pernyataan WHILE digunakan untuk perulangan selama sesuai dengan kondisi yang ada setelah pernyataan WHILE. BREAK digunakan untuk keluar dari Loop dan mengabaikan pernyataan dibawahnya. Sedangkan CONTINUE digunakan untuk merestart Loop dan mengabaikan statement dibawahnya.

Syntax:

```
WHILE Boolean_expression
{ sql_statement | statement_block }
[ BREAK ]
{ sql_statement | statement_block }
[ CONTINUE ]
```

Contoh:

Kita akan mencetak bilangan ganjil 1 s/d 20, perhatikan meskipun pada while variabel @i harus diulang sampai kurang dari seratus tetapi, tetap akan keluar pada saat @i =21 (dengan pernyataan BREAK) sbb:

```
declare @i as integer
set @i = 0
WHILE @i < 100
begin
    set @i = @i+1
    if @i = 21 BREAK
    if @i%2 = 0 CONTINUE
    print @i
end
```

GOTO

GOTO digunakan untuk melompat ke baris yang sudah ditentukan labelnya.

Seperti contoh berikut mempunyai hasil yang sama dengan contoh while di atas:

```
declare @i as integer
set @i = 0
RepeatLoop:
    set @i = @i+1
    print @i
    if @i < 20 goto RepeatLoop
```

Cursor T-SQL

Operasi yang dilakukan pada database relational menggunakan query selalu menghasilkan himpunan baris-baris. Pada sebagian aplikasi, hal seperti itu masih kurang, karena seringkali kita perlu mengambil baris demi baris untuk dimanipulasi, bukan hanya sekumpulan baris. Cursor menyediakan mekanisme untuk bekerja baris per baris atau blok-blok yang lebih kecil dari pada hasil suatu query.

Syntax umum dari cursor sbb:

```
DECLARE cursor_name CURSOR  
[ LOCAL | GLOBAL ]  
[ FORWARD_ONLY | SCROLL ]  
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]  
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]  
FOR select_statement  
[ FOR UPDATE [ OF column_name [ ,...n ] ] ]
```

Cursor_Name

Nama dari cursor yang dideklarasikan.

LOCAL

Variabel cursor yang dideklarasikan bersifat lokal, artinya hanya berlaku dalam satu scope saja.

GLOBAL

Variabel cursor yang dideklarasikan bersifat global, artinya berlaku pada semua koneksi database yang ada.

FORWARD_ONLY

Cursor yang dideklarasikan pointernya hanya dapat digeser dari atas ke bawah saja, tidak bisa kembali ke atas lagi, yaitu hanya berlaku FETCH NEXT

SCROLL

Cursor yang dideklarasikan pointernya berlaku untuk semua model FETCH (FIRST, LAST, NEXT, PRIOR, RELATIVE, ABSOLUTE)

STATIC

Cursor akan mengkopikan semua record yang sesuai pada temporary table **tempdb**, artinya perubahan-perubahan pada tabel tidak akan terlihat, meskipun ada penambahan, perubahan atau penghapusan record.

KEYSET

Pada saat cursor ini dibuat SQL Server akan mencatat pointer yang menunjuk pada baris yang dipilih. Sehingga bila terjadi perubahan atau user lain meng-update record hasilnya dapat dilihat pada tipe cursor ini. Tetapi bila ada user lain menambahkan record baru tidak akan kelihatan, karena pointer record baru tidak tercatat pada cursor. Sedangkan pada saat user lain menghapus, pointer record masih ada tetapi isi dari record untuk masing-masing field menjadi NULL dan nilai variabel global @@FETCH_STATUS bukan 0 bila pointer mengarah pada record yang dihapus, melainkan bernilai -2.

DYNAMIC

Cursor dinamis merupakan kursor yang paling flexible dan mahal karena cursor ini akan memantau terus record-record yang dirubah, ditambah dan dihapus. Setiap kali pointer diarahkan pada record sistem selalu mengambil dari database. FETCH ABSOLUTE tidak berlaku untuk kursor ini.

/READ_ONLY/SCROLL_LOCKS/OPTIMISTIC/

Kata kunci ini menentukan strategi penguncian. Cursor READ_ONLY tidak dapat di-update. Cursor SCROLL_LOCKS akan mengunci setiap baris bila sedang dibaca dan akan melepas kunci ketika pointer dipindah ke baris lain. Cursor OPTIMISTIC tidak mengunci baris pada saat dibaca, tetapi akan melakukan penguncian bila baris setelah dibaca dan terjadi perubahan pada baris tersebut.

FOR select_statement

Di sini kita harus menentukan SELECT statement yang membaca baris-baris yang diinginkan ke dalam kursor.

FOR UPDATE

Klausa ini menentukan kolom-kolom tertentu saja yang bisa diupdate, jika kolom tidak ditentukan maka dianggap semua kolom dapat diupdate kecuali jika kita menggunakan tipe penguncian `READ_ONLY`.

Contoh:

use AKADEMIK

```
DECLARE curMHS cursor  
SCROLL  
DYNAMIC  
READ_ONLY  
For SELECT NIM, NAMA from MAHASISWA
```

open curMHS

```
declare @NIM varchar(10)  
declare @NAMA varchar(40)
```

```
FETCH FIRST from curMHS into @NIM,@NAMA  
WHILE @@FETCH_STATUS = 0  
BEGIN  
    print @NIM + ': ' + @NAMA  
    FETCH NEXT from curMHS into @NIM,@NAMA  
END
```

```
close curMHS  
deallocate curMHS
```

3.5.Tugas

- Buat query daftar IPK untuk masing-masing mahasiswa dengan ketentuan sbb:
Jika $(\text{nilaiUTS} \times 0.4 + \text{nilaiUAS} \times 0.6 \text{ jika}) \geq 76$ maka $N = 4$
Jika $(\text{nilaiUTS} \times 0.4 + \text{nilaiUAS} \times 0.6 \text{ jika}) \geq 66$ maka $N = 3$
Jika $(\text{nilaiUTS} \times 0.4 + \text{nilaiUAS} \times 0.6 \text{ jika}) \geq 56$ maka $N = 2$
Jika $(\text{nilaiUTS} \times 0.4 + \text{nilaiUAS} \times 0.6 \text{ jika}) \geq 36$ maka $N = 1$
Lainnya $N = 0$
 $NK = N \times SKS$

$$IP = \frac{\sum (N \times SKS)}{\sum SKS \text{ yang ditempuh}}$$

Hasil dari query ditampilkan seperti contoh sbb:

NIM	NAMA	IP
2000420111	Wong Kang Inan	2

- Buat daftar nilai untuk semua mahasiswa menggunakan cursor dengan hasil yang diharapkan sbb:

NIM : 2000420111

NAMA : Wong Kang Inan

Kode MK	Mata Kuliah	SKS	Nilai

KU1001	Agama	2	C
IF4001	Pemrograman	3	D

Bab IV

Stored Procedure dan Trigger

4.1.Tujuan

- Mahasiswa mampu menyederhanakan pekerjaan dengan menggunakan trigger dan stored procedure
- Mahasiswa mampu mengimplementasikan business rule pada stored procedure maupun trigger.
- Mahasiswa mampu mengurangi network traffic dan melakukan pembagian beban kerja yang optimal antara client dan server.

4.2.Materi

- Stored Procedure
- Trigger

4.3.Alat dan bahan

- a. PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- b. Microsoft SQL Server 2000

4.4.Teori

Stored Procedure (SP) merupakan salah satu object yang ada di SQL Server sama seperti table, view dan object-object lainnya. SP adalah sebuah rutin yang ditulis dengan bahasa T-SQL yang dapat memanipulasi baris-baris database. Semua pernyataan SQL digunakan untuk memanipulasi baris, tetapi SQL tidak memiliki pernyataan alur kendali seperti IF, CASE, serta fungsi-fungsi untuk memanipulasi string dan sebagainya. Dengan SP secara otomatis kita akan menyederhanakan pekerjaan, karena kita tidak perlu menulis berulang-ulang untuk query yang sama dengan parameter yang berbeda sebab kita hanya tinggal memanggil nama SP beserta parameternya. SP dapat mengurangi network traffic dan beban kerja client hal ini dapat terjadi karena SP disimpan, dikompilasi dan dijalankan di server sehingga client tidak perlu memindahkan semua data dari server, client hanya mengirimkan nama SP beserta parameternya server hanya mengirim hasil dari SP tersebut.

Syntax

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]  
    [ { @parameter data_type }  
      [ VARYING ] [ = default ] [ OUTPUT ]  
    ] [ ,...n ]  
  
[ WITH  
  { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]  
  
[ FOR REPLICATION ]  
  
AS  
  sql_statement [ ...n ]
```

procedure_name

Nama SP yang baru. Prosedur Nama harus unik di dalam database.

Prosedur sementara global atau lokal dapat diciptakan dengan menambahkan pada bagian awal *procedure_name* dengan tanda (# *procedure_name*) untuk prosedur temporer lokal dan tanda (## *procedure_name*) untuk prosedur temporer global. Nama, termasuk # atau ##, tidak boleh melebihi 128 karakter.

Number

Bilangan integer optional, digunakan apabila anda ingin menamai procedure dengan nama yang sama dengan satu grup nama tersebut. Sebagai contoh kita mempunyai 2 SP dengan nama KRSProc;1 dan KRSProc;2, dan kita menjalankan perintah DROP PROC KRSProc maka kedua nama SP tersebut akan terhapus.

@*parameter*

Nama dari parameter yang akan digunakan, setiap parameter yang dideklarasikan harus diisi pada saat SP di-execute (kecuali bila nilai default disertakan), jumlah parameter untuk setiap SP maximum 2100 parameter

Nama parameter harus diawali dengan tanda @, dan bersifat sebagai variable local untuk SP tersebut.

data_type

Type data dari parameter. Semua tipe data dapat digunakan untuk SP termasuk text, ntext dan image. Tipe data cursor dapat dipakai sebagai parameter hanya sebagai OUTPUT parameter serta keyword VARYING dan OUTPUT harus disetakan.

VARYING

Digunakan hanya untuk tipe data CURSOR

Default

Default value untuk parameter. Jika default didefinisikan, SP dapat dieksekusi tanpa paramete. Default harus merupakan konstanta atau NULL. Dapat juga berisi wildcard

characters (% , _ , [, and [^]) jika SP mempunyai parameter dengan It can include keyword LIKE.

OUTPUT

Mengindikasikan bahwa parameter merupakan *return parameter*. **Text**, **ntext**, and **image** parameters dapat juga sebagai OUTPUT parameters.

...n

Mengindikasikan sampai dengan n parameter dan maximum 2100 parameter.

{*RECOMPILE* | *ENCRYPTION* | *RECOMPILE, ENCRYPTION*}

RECOMPILE menunjukkan bahwa SP akan di-compile ulang ketika prosedur dijalankan

ENCRYPTION menunjukkan bahwa SQL Server akan meng-enkripsi semua statement yang ada dalam SP, dan tidak di-publish sebagai bagian dari SQL-Server replication

FOR REPLICATION

Menunjukkan bahwa SP hanya dieksekusi untuk saat replikasi saja dan tidak dapat digunakan bersama option WITH RECOMPILE

AS

Merupakan awal dari apa yang akan dikerjakan oleh SP selanjutnya

sql_statement

Isi dari SP yang berupa T-SQL

...n

Menunjukkan bahwa SP dapat berisi lebih dari satu (multiple) statement T-SQL yang besarnya mulai dari CREATE PROC sampai akhir syntax tidak boleh lebih dari 128 MB.

Trigger adalah stored procedure khusus, yang kebanyakan digunakan untuk tugas-tugas administrative. Trigger adalah sebuah stored procedure yang dijalankan oleh SQL-Server secara otomatis bila terjadi kejadian (event) tertentu. Kejadian ini seperti menambah (insert), menghapus (delete) atau merubah(update) record. Dengan kata lain, kita bisa mendefinisikan stored procedure pada sebuah table yang akan dijalankan secara otomatis oleh SQL-Server bila baris (record) pada Tabel tersebut kita tambah, hapus atau kita update. Trigger biasanya juga digunakan untuk mendefinisikan business rules, misalnya kita ingin setiap ada pembelian stock di gudang akan bertambah, bila ada penjualan stock akan berkurang dengan jalan pada table pembelian kita definisikan trigger untuk mengupdate(menambah) table stock dan pada table penjualan kita definisikan trigger untuk mengurangi stock.

Syntax

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
    { { FOR | AFTER | INSTEAD OF } { [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
      [ NOT FOR REPLICATION ]
    AS
      sql_statement [ ...n ]
    }
}
```

trigger_name

Nama dari trigger, harus unik dalam satu database, tidak boleh sama meskipun pada table yang berbeda

Table / view

Nama table atau view dimana trigger akan dijalankan.

WITH ENCRYPTION

ENCRYPTION menunjukkan bahwa SQL Server akan meng-enkripsi semua statement yang ada dalam Trigger, dan tidak di-publish sebagai bagian dari SQL-Server replication

AFTER

Untuk menentukan bahwa trigger hanya dijalankan jika proses insert atau update atau delete sukses dijalankan.

AFTER merupakan default, tapi FOR dijalankan jika dituliskan.

AFTER triggers tidak dapat diletakkan pada trigger views.

INSTEAD OF

INSTEAD OF akan menggantikan proses insert, delete atau update yang sesungguhnya. Misalnya anda melarang untuk menghapus record tertentu anda dapat membuat trigger dengan keyword INSTEAD OF DELETE. Maka proses penghapusan akan diganti dengan menjalankan trigger yang anda buat.

Tetapi INSTEAD OF tidak bisa digunakan jika table berelasi dengan table lain dengan operasi referential relationship ditentukan *cascade action ON DELETE*, demikian juga untuk INSTEAD OF UPDATE tidak bisa digunakan jika referential relationship ditentukan *cascade action ON UPDATE*

NOT FOR REPLICATION

Menunjukkan bahwa TRIGGER tidak akan dieksekusi untuk saat replikasi

AS

Perintah yang akan dijalankan oleh trigger setelah keyword AS ini.

4.5. Latihan

1. Buat Stored Procedure untuk menampilkan daftar nilai dari satu mahasiswa dengan parameter NIM mahasiswa, SP-nya sbb:

```
if exists(Select Name from sysobjects where name ='DaftarNilai')
    DROP Proc DaftarNilai

go

Create Procedure DaftarNilai @Nim varchar(10)
as
Select KRS.KodeMK, NAMAMK,
NilaiUTS*0.4+NilaiUAS*0.6 as NA,
case
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 76 then 'A'
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 66 then 'B'
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 56 then 'C'
    when NilaiUTS*0.4+NilaiUAS*0.6 >= 36 then 'D'
    else 'E'
end as NH
from KRS INNER JOIN MATAKULIAH
    on KRS.KodeMK = MATAKULIAH.KodeMk
WHERE NIM = @NIM
```

2. Buat Trigger untuk mengupdate NIM Mahasiswa jika kodejur pada Mahasiswa dirubah, sehingga selalu sama antara NIM dengan kode jurusannya, trigger-nya sbb:

```
CREATE Trigger UpdNIM on MAHASISWA
for insert, update
as
    if UPDATE(KODEJUR)
    begin
        Update MAHASISWA
        set NIM = left(nim,4) + (select KodeJur from Inserted) +
        right(NIM,4)
        where NIM = (select NIM from Deleted)
    end

    else

    if UPDATE(NIM)
    begin
        Update MAHASISWA
        set KODEJUR = (select substring(NIM,5,2) from inserted)
        where NIM = (select NIM from Inserted)
    end
```

4.6. Tugas

1. Buat Stored Procedure dengan nama *RangkingPerJur* untuk menampilkan ranking berdasarkan IPK per jurusan. Dengan parameter KodeJurusan, misalnya tampilkan daftar IPK mahasiswa jurusan informatika (42).

Maka eksekusi procedurennya : *exec RangkingPerJur '42'*

Hasil yang diharapkan :

NIM	NAMA	IP
2000420111	Wong Kang Inan	2

2. Buat Trigger untuk mengetahui nilai-nilai pada KRS yang pernah dirubah beserta tanggal kapan nilai tersebut dirubah. Masukkan hasil audit tersebut ke dalam tabel AUDITNILAI dengan struktur sbb:

Nama Kolom	Tipe Data	Panjang	Keterangan
NIM	Varchar	10	NIM dari nilai yang dirubah
KodeMK	Varchar	6	KodeMK dari Nilai yg Dirubah
NilaiUTSAsal	Decimal	5,2	Nilai UTS sebelum Dirubah
NilaiUASAsal	Decimal	5,2	Nilai UAS sebelum Dirubah
NilaiUTSUpd	Decimal	5,2	Nilai UTS Setelah Dirubah
NilaiUASUpd	Decimal	5,2	Nilai UAS Setelah Dirubah
TanggalUpd	Datetime	8	Tanggal Perubahan

Bab V

Pemrograman dengan ADO

5.1.Tujuan

Mahasiswa mampu menghubungkan antara database pada DBMS dengan program aplikasi menggunakan ActiveX Data Object.

5.2.Materi

- Connection
- Command
- Recordset

5.3.Alat dan bahan

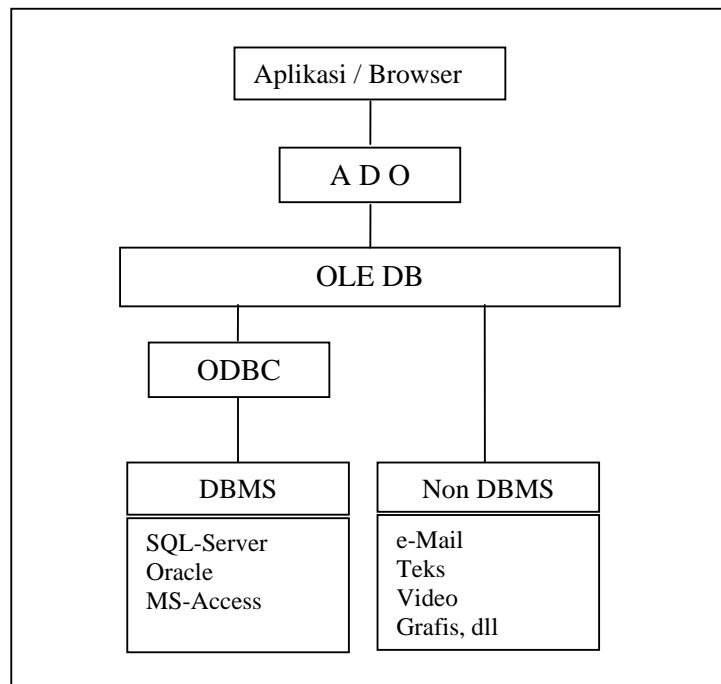
- a. PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- b. Microsoft SQL Server 2000
- c. Visual Basic 6.0

5.4.Teori

Microsoft menyediakan suatu teknologi Universal Data Access (UDA) yang dapat digunakan untuk mengakses ke semua tipe informasi dari berbagai sumber data baik relational maupun non relational seperti database hierarkial, e-mail, teks, dan grafis.

OLE Database (OLE DB) adalah bagian dari UDA yang memungkinkan kita membaca dan memproses data dari manapun tanpa mengkonversi ataupun mengimpor ke dalam bentuk database tradisional. Menggunakan provider OLE DB, kita dapat memproses data dalam pesan e-mail, halaman HTML, spreadsheet, teks, SEL Server, foxpro , oracle dsb.

ActiveX Data Object (ADO) merupakan interface yang menghubungkan aplikasi atau internet browser ke OLE-DB, OLE-DB akan meneruskan hubungan ini dengan sumber data seperti SQL-Server, Access, Oracle dsb. Untuk lebih jelasnya dapat kita lihat ilustrasi berikut:



5.5. Latihan

a. Membangun koneksi database

1. Jalankan Visual Basic 6.0
2. Buat Project Baru
3. Pilih Menu Project > References
4. Aktifkan Microsoft ActiveX Data Objects 2.x Library
5. Click OK
6. Pilih Menu Project > Add module
7. Click Open

Saat ini anda sudah memiliki satu modul dengan nama *Module1* dan satu *form1*, untuk membangun koneksi tambahkan program berikut pada *Module1*

```
Public AkadCon As ADODB.Connection
Dim ConString As String
```

```
Sub Main()
    On Error GoTo ErrCon
    ConString = "Provider=SQLOLEDB.1; " & _
        "Integrated Security=SSPI; " & _
        "Initial Catalog=AKADEMIK;" & _
        "Data Source=(local)"
```

```
Set AkadCon = New ADODB.Connection
AkadCon.CursorLocation = adUseServer
AkadCon.Open ConString
```


MsgBox "Koneksi Sukses"

Exit Sub

ErrCon:

If AkadCon.Errors.Count > 0 Then MsgBox AkadCon.Errors(0).Description

End Sub

Untuk dapat menjalankan program di atas, lakukan langkah berikut:

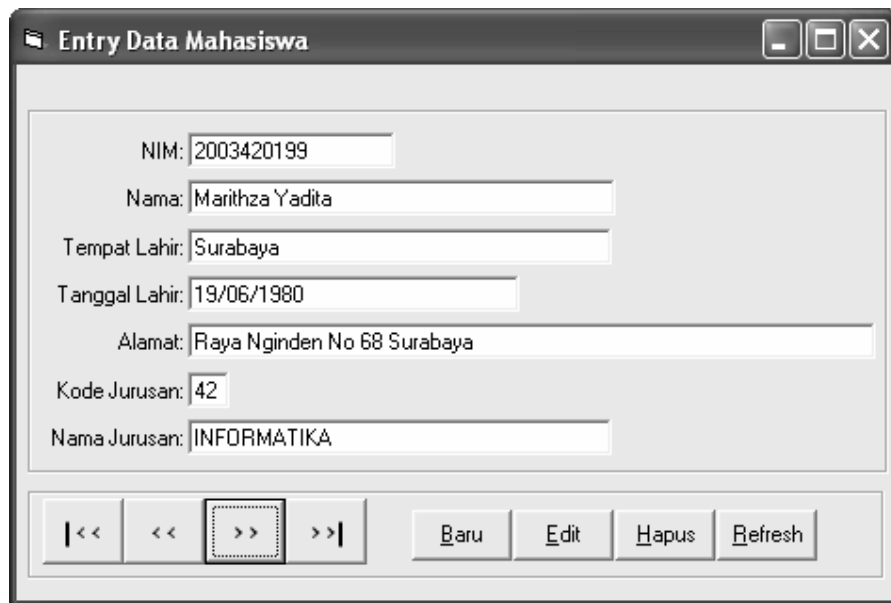
1. Pilih Menu Project > Project Properties
2. Ganti Start Up Object-nya dengan **Sub Main**

Program di atas hanya akan memberikan peringatan jika koneksi yang kita lakukan sukses dengan memberikan peringatan pada message box bahwa koneksi sukses. Apabila terjadi kesalahan akan diberikan peringatan sesuai dengan jenis kesalahannya.

b. Recordset

Pada latihan berikut masih berhubungan dengan latihan-latihan sebelumnya, recordset merupakan object dari connection oleh sebab itu kita akan memanfaatkan module1 di atas sebagai koneksi dari recordset kita.

Latihan berikut adalah program untuk entry mahasiswa, menggunakan ADODB.Recordset, dengan Form dan control sbb:



Modifikasi Form1 yang ada seperti pada gambar di atas dan ganti property-property berikut:

Object	Property	Setting
Form1	Name Caption	FrmEntMhs Entry Data Mahasiswa
Frame	Name Caption	Frame1
Frame	Name Caption	Frame2
TextBox	Name MaxLength	TxtNim 10
TextBox	Name MaxLength	TxtNama 40
TextBox	Name MaxLength	TxtTempatLahir 30
TextBox	Name MaxLength	TxtTanggalLahir 10
TextBox	Name MaxLength	TxtAlamat 40
TextBox	Name MaxLength	TxtKodeJur 2
TextBox	Name MaxLength	TxtNamaJurusan 30
CommandButton	Name	CmdNav
	Index Caption	0 <<
	Index Caption	1 <<
	Index Caption	2 >>
CommandButton	Index Caption	3 >>
	Name	CmdAction
	Index Caption	0 &Baru
	Index Caption	1 &Edit
CommandButton	Index Caption	2 &Hapus
	Index Caption	3 &Refresh

Ketikan listing program berikut pada form FrmEntMhs

Dim rsMHS As ADODB.Recordset 'Deklarasi recordset Mahasiswa
Dim rsJurusan As ADODB.Recordset 'Deklarasi recordset Jurusan
Dim onEdit As Boolean 'Deklarasi Variabel untuk menunjukkan posisi recordset mahasiswa sedang edit atau tidak

Private Sub cmdNav_Click(Index As Integer)

On Error GoTo errNav 'Bila terjadi kesalahan diarahkan ke lable errnav

Select Case Index

Case 0 ' Button |<< ' (untuk Bergerak ke awal baris)

rsMHS.MoveFirst

Case 1 ' Button |<< ' (untuk Bergerak ke baris Sebelumnya)

If Not rsMHS.BOF Then

rsMHS.MovePrevious

End If

If rsMHS.BOF Then rsMHS.MoveFirst

Case 2 ' Button |<< (untuk Bergerak ke baris Berikutnya)

If Not rsMHS.EOF Then

rsMHS.MoveNext

End If

If rsMHS.EOF Then rsMHS.MoveLast

Case 3 ' Button |<< (untuk Bergerak ke baris Akhir)

rsMHS.MoveLast

End Select

Exit Sub

errNav:

MsgBox Err.Description, vbOKOnly, "Latihan ADO"

End Sub

Private Sub cmdAction_Click(Index As Integer)

On Error GoTo errAction

Select Case Index

Case 0

If cmdAction(0).Caption = "&Baru" Then 'Button untuk menambahkan baris baru

rsMHS.AddNew

onEdit = True

Else 'Button untuk Menyimpan

rsMHS.Update

onEdit = False

End If

Case 1

If cmdAction(1).Caption = "&Edit" Then 'Button untuk Edit baris

onEdit = True

Else 'Button untuk Membatalkan (cancel)

rsMHS.CancelUpdate

rsMHS.Move 0

onEdit = False

End If

Case 2 'Button untuk Menghapus baris

If MsgBox("Anda Yakin Menghapus Mahasiswa dg NIM:" + txtNIM + " ?",

vbQuestion + vbYesNo, "Latihan ADO") = vbYes Then

rsMHS.Delete

If Not rsMHS.EOF Then

rsMHS.MoveNext

End If

If rsMHS.EOF Then rsMHS.MoveLast

End If

Case 3

rsMHS.Requery

End Select

Setbutton onEdit

Exit Sub

errAction:

MsgBox Err.Description, vbOKOnly, "Latihan ADO"

If rsMHS.EditMode <> adEditNone Then

rsMHS.CancelUpdate 'Cek apakah kesalahan terjadi pada saat proses penambahan atau edit jika ya batalkan

rsMHS.Move 0 'Untuk mererefresh baris yang aktif saat ini

End If

Resume Next 'bila terjadi kesalahan akan dilanjutkan ke baris berikutnya setelah kesalahan

End Sub

Private Sub Form_Load() ' Procedure saat form dipanggil

Dim SqlString As String

Set rsJurusan = New ADODB.Recordset 'Inisialisasi object recordset jurusan

rsJurusan.CursorLocation = adUseClient 'Gunakan cursor client

*rsJurusan.Open "Select KodeJurusan, NamaJurusan from Jurusan
order By KodeJurusan", AkadCon, adOpenDynamic, dLockReadOnly*

*SqlString = "SELECT MAHASISWA.NIM, MAHASISWA.Nama, " & _
"MAHASISWA.TempatLahir, MAHASISWA.TanggalLahir, " & _
"MAHASISWA.Alat, MAHASISWA.KodeJur " & _
"FROM MAHASISWA " & _
"ORDER BY MAHASISWA.NIM"*

Set rsMHS = New ADODB.Recordset

rsMHS.Open SqlString, AkadCon, adOpenDynamic, adLockOptimistic

'Binding recordset ke masing-masing textbox

Set txtNIM.DataSource = rsMHS

txtNIM.DataField = "NIM"

Set txtNama.DataSource = rsMHS

txtNama.DataField = "NAMA"

Set txtTempatLahir.DataSource = rsMHS

txtTempatLahir.DataField = "TempatLahir"

Set txtTanggalLahir.DataSource = rsMHS

txtTanggalLahir.DataField = "TanggalLahir"

Set txtAlamat.DataSource = rsMHS

txtAlamat.DataField = "Alamat"

Set txtKodeJur.DataSource = rsMHS

txtKodeJur.DataField = "KodeJur"

Frame1.Enabled = False

End Sub

'Bila text kode jurusan berubah maka textbox nama jurusan akan berubah

Private Sub txtKodeJur_Change()

rsJurusan.Find "KodeJurusan =" + txtKodeJur + "" , , adSearchForward, 1

If Not rsJurusan.EOF Then

txtNamaJurusan = rsJurusan!NamaJurusan

Else

txtNamaJurusan = ""

End If

End Sub

Private Sub txtTanggalLahir_Validate(Cancel As Boolean)

If Not IsDate(txtTanggalLahir) Then

*If MsgBox("Tanggal yang anda isikan salah, click Yes untuk mengosongkan " & _
dan No untuk meneruskan", vbQuestion + vbYesNo, "Latihan ADO")
= vbNo Then*

Cancel = True

Else

txtTanggalLahir = ""

End If

End If

End Sub

'Untuk validasi kode jurusan yang dimasukkan, jika tidak ada akan diulangi sampai ketemu

Private Sub txtKodeJur_Validate(Cancel As Boolean)

rsJurusan.Find "KodeJurusan =" + txtKodeJur + "" , adSearchForward, 1

If rsJurusan.EOF Then

*If MsgBox("Kode Jurusan Tidak Ada, click Yes untuk mengosongkan dan No
untuk meneruskan", vbQuestion + vbYesNo, "Latihan ADO") = vbNo Then*

Cancel = True

Else

txtKodeJur = ""

txtNamaJurusan = ""

End If

Else

txtNamaJurusan = rsJurusan!NamaJurusan

End If

End Sub

'Procedure untuk mengaktifkan atau tidak mengaktifkan frame dan button

Private Sub Setbutton(ByVal bEdit)

If bEdit Then

cmdAction(0).Caption = "&Simpan"

cmdAction(1).Caption = "&Batal"

Else

cmdAction(0).Caption = "&Baru"

cmdAction(1).Caption = "&Edit"

End If

For i = 0 To CmdNav.Count - 1

CmdNav(i).Enabled = Not bEdit

Next

For i = 2 To cmdAction.Count - 1

cmdAction(i).Enabled = Not bEdit

Next

Frame1.Enabled = bEdit

End Sub

Agar Program tersebut dapat berjalan ganti statement *MsgBox "Koneksi Sukses"*
Pada Module1 di atas dengan *FrmEntMhs.Show*

c. Command

Pada latihan berikut masih berhubungan dengan latihan-latihan sebelumnya, commabd merupakan object dari connection oleh sebab itu kita akan memanfaatkan module1 di atas sebagai koneksi dari command kita.

Latihan berikut adalah program untuk menampilkan KHS mahasiswa dari stored procedure *DaftarNilai*, menggunakan ADODB.Recordset, dengan Form dan control sbb:

	KodeMK	NAMAMK	NA	NH
1	KU1001	AGAMA	84	A
2	KU1002	Matematika	78	A
3	TI4101	Client Server	62	C

Modifikasi Form yang ada seperti pada gambar di atas dan ganti property-property berikut:

Object	Property	Setting
Form1	Name Caption	FrmCommand Daftar Nilai
TextBox	Name MaxLength	TxtNim 10
Label	Name	LblNama
CommandButton	Name Caption	CmdExecute &Execute
MSFlexGrid	Name AllowUserResizing	Flex 1-FlexResizeColoumn

Ketikan listing program berikut pada form FrmCommand

```
Dim CmdNilai As ADODB.Command
Dim RsNilai As ADODB.Recordset
Dim PrmNilai As ADODB.Parameter
Dim rsMhs As ADODB.Recordset

Private Sub cmdExecute_Click()
    Set CmdNilai = New ADODB.Command
    CmdNilai.ActiveConnection = AkadCon
    CmdNilai.CommandText = "DaftarNilai"
    CmdNilai.CommandType = adCmdStoredProc

    Set PrmNilai = New ADODB.Parameter
    PrmNilai.Name = "Nim"
    PrmNilai.Type = adVarChar
    PrmNilai.Size = 10
    PrmNilai.Direction = adParamInput
    PrmNilai.Value = txtNIM.Text
    CmdNilai.Parameters.Append PrmNilai

    Set RsNilai = New ADODB.Recordset
    Set RsNilai = CmdNilai.Execute

    Flex.Cols = RsNilai.Fields.Count + 1
    Flex.Rows = 2
    Flex.Clear
    For i = 0 To RsNilai.Fields.Count - 1
        Flex.ColWidth(i + 1) = If(RsNilai.Fields(i).DefinedSize >=
Len(RsNilai.Fields(i).Name), RsNilai.Fields(i).DefinedSize,
Len(RsNilai.Fields(i).Name)) * TextWidth("W"))
        Flex.TextMatrix(0, i + 1) = RsNilai.Fields(i).Name
    Next

    If RsNilai.EOF Then
        MsgBox "Tidak ada daftar nilai untuk mahasiswa tersebut"
    Else
        iRow = 1
        With RsNilai
            While Not .EOF
                If iRow > 1 Then
                    Flex.AddItem iRow
                Else
                    Flex.TextMatrix(iRow, 0) = iRow
                End If

                For icol = 0 To RsNilai.Fields.Count - 1
                    Flex.TextMatrix(iRow, icol + 1) = .Fields(icol).Value
                Next
                .MoveNext
                iRow = iRow + 1
            Wend
        End With
    End If
End Sub

Private Sub Form_Load()
    Set rsMhs = New ADODB.Recordset
    rsMhs.CursorLocation = adUseClient
    rsMhs.Open "Mahasiswa", AkadCon, adOpenDynamic, adLockReadOnly
End Sub
```

```

Private Sub txtNIM_Validate(Cancel As Boolean)
    rsMhs.Find "NIM=" + txtNIM.Text + "", , adSearchForward, 1
    If Not rsMhs.EOF Then
        lblNama = rsMhs!Nama
    Else
        MsgBox "NIM Mahasiswa Tidak Ditemukan"
        Cancel = True
    End If
End Sub

```

Agar Program tersebut dapat berjalan ganti statement *FrmEntMhs.Show* Pada Module1 di atas dengan *FrmCommand.Show*

5.6. Tugas

1. Buat program untuk entry KRS, dengan model master detail dan bentuk form sebagai berikut:

The screenshot shows a Windows application window titled "Entry KRS". The window has a standard Windows XP-style title bar with minimize, maximize, and close buttons. The main content area is a form with the following elements:

- Three text input fields at the top: "NIM:", "Nama: Label Nama Mahasiswa", and "Nama Jurusan: Nama Jurusan".
- A table with three columns: "Kode Mata Kuliah", "Mata Kuliah", and "S K S". The table has one empty row below the headers.
- Buttons to the right of the table: "Edit", "Delete", and "Baru".
- A large, empty rectangular area below the table, likely for displaying a list of courses or a detailed view.
- A row of buttons at the bottom: "<<", "<<", ">>", ">>|", "Baru", "Edit", and "Hapus".

2. Buat program untuk mengeksekusi stored procedure *RangkingPerJur*, gunakan object command seperti pada latihan 2 bentuk form sebagai berikut:

The screenshot shows a Java Swing window titled "Rangking IP Mahasiswa". The window has a standard title bar with minimize, maximize, and close buttons. Inside the window, there are two text input fields. The first field is labeled "Kode Jurusan" and is empty. The second field is labeled "Jurusan" and contains the text "Nama Jurusan". To the right of these fields is a button labeled "Execute". Below the input fields is a large rectangular area, likely a table or list, which is currently empty.

Bab VI

Class Data Aware

6.1. Tujuan

- Mahasiswa mampu mengembangkan aplikasi dengan arsitektur *three tier*

6.2. Materi

- Class
- Implementasi Class

6.3. Alat dan bahan

- a. PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- b. Microsoft SQL Server 2000
- c. Visual Basic 6.0

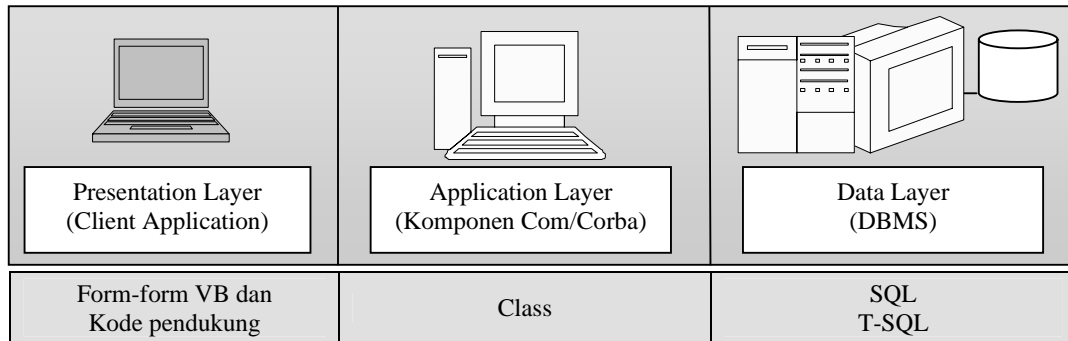
6.4. Teori

Sebelum kita mempelajari class yang berhubungan dengan data (*class data aware*) serta bagaimana class tersebut dapat menyederhanakan aplikasi, mari kita melihat ke dalam model *three tier*. Perlu diketahui bahwa aplikasi dengan arsitektur ini bukanlah jawaban bagi semua akses data. Model ini tidak harus diterapkan, karena memang tidak semua aplikasi membutuhkannya. Model dua tingkat (*client server*) dengan komponen ADO sudah cukup untuk membangun aplikasi yang berhubungan dengan database. Tetapi jika kita menginginkan aplikasi yang dapat dikembangkan lagi atau membuat aplikasi untuk internet, kita harus mengerti keuntungan arsitektur ini. Arsitektur ini juga sangat bermanfaat bila kita mengembangkan aplikasi dalam team. Tanpa class berarti setiap programmer harus memahami struktur database dan menulis perintah SQL pada saat memanipulasi database.

Gambar dibawah adalah diagram dari arsitektur tiga tingkat. Aplikasi client adalah user interface yang dilihat oleh user (*User Service*). Aplikasi ini bertanggung jawab untuk mepresentasikan data kepada user (*Presentation Layer*). Pada arsitektur *client server* aplikasi client berhubungan langsung dengan server database. Seperti pada bab sebelumnya (ADO), aplikasi client mengirimkan perintah SQL atau prosedur tersimpan. Pada arsitektur *three tier* ini aplikasi client

memanggil *methode (function)* dari class dan mengijinkan class mengakses tabel-tabel pada server. Application layer atau logika presentasi adalah kode program yang memproses data dan mengimplementasikan aturan bisnis.

Data layer mewakili satu atau lebih sumber data dan menyediakan data bagi lapisan aplikasi.



6.5.Latihan

Latihan berikut, kita akan membuat class yang memiliki methode-methode yang dibutuhkan untuk mengakses tabel mahasiswa. Methode-methode ini adalah addMahasiswa, updateMahasiswa, deleteMahasiswa, dan getMahasiswa yang diimplementasikan dalam class clsMahasiswa. Berikut langkah – langkah untuk membentuk class tersebut:

1. Pilih menu File, New Project dan ActiveX DLL.
2. Ganti nama module1 menjadi clsMahasiswa
3. Ganti nama Project1 menjadi classAkademik
4. Pilih menu project, reference
5. Aktifkan Microsoft ActiveX Data Objects 2.x library
6. Ketikkan program berikut pada jendela kode class

```
Dim AkadCN As ADODB.Connection
```

```
Private Sub Class_Initialize()
```

```
Set AkadCN = New ADODB.Connection
```

```
AkadCN.ConnectionString = "Provider=SQLOLEDB.1; Integrated  
Security=SSPI; Data Source=(local); Initial Catalog =AKADEMIK"
```

```
AkadCN.CursorLocation = adUseServer  
AkadCN.Open
```

```
End Sub
```

```
Private Sub Class_Terminate()
```

```
AkadCN.Close
```

```
Set AkadCN = Nothing
```

```
End Sub
```

Public Function addMahasiswa(sNim, sNama, sTempatLahir, dTanggalLahir, sAlamat, sKodeJur) As Boolean

Dim SqlAdd As String

On Error GoTo erradd

If sNim = "" Or sNim = "<NULL>" Or sNim = "NULL" Then

sNim = "null"

Else

sNim = "" + sNim + ""

End If

If sNama = "" Or sNama = "<NULL>" Or sNama = "NULL" Then

sNama = "null"

Else

sNama = "" + sNama + ""

End If

If sTempatLahir = "" Or sTempatLahir = "<NULL>" Or sTempatLahir = "NULL" Then

sTempatLahir = "null"

Else

sTem = "" + sTempatLahir + ""

End If

If dTanggalLahir = "" Or dTanggalLahir = "<NULL>" Or dTanggalLahir = "NULL" Then

dTanggalLahir = "null"

Else

dTanggalLahir = "" + Format(dTanggalLahir, "YYYY-MM-DD") + ""

End If

If sAlamat = "" Or sAlamat = "<NULL>" Or sAlamat = "NULL" Then

sAlamat = "null"

Else

sAlamat = "" + sAlamat + ""

End If

If sKodeJur = "" Or sKodeJur = "<NULL>" Or sKodeJur = "NULL" Then

sKodeJur = "null"

Else

sKodeJur = "" + sKodeJur + ""

End If

SqlAdd = "Insert into Mahasiswa " & _

"(Nim, Nama, TempatLahir, TanggalLahir, Alamat, KodeJur) " & _

"VALUES " & _

"(" & sNim & "," + sNama + "," + sTempatLahir + "," + dTanggalLahir

+ "," + sAlamat + "," + sKodeJur + ")"

AkadCN.Execute SqlAdd

addMahasiswa = True

Exit Function

erradd:

addMahasiswa = False

End Function

Public Function updateMahasiswa(sNim, sNama, sTempatLahir, dTanggalLahir, sAlamat, sKodeJur) As Boolean

Dim SqlUpd As String

On Error GoTo errUpd

If sNim = "" Or sNim = "<NULL>" Or sNim = "NULL" Then

```

        sNim = "null"
    Else
        sNim = "" + sNim + ""
    End If

    If sNama = "" Or sNama = "<NULL>" Or sNama = "NULL" Then
        sNama = "null"
    Else
        sNama = "" + sNama + ""
    End If

    If sTempatLahir = "" Or sTempatLahir = "<NULL>" Or sTempatLahir = "NULL" Then
        sTempatLahir = "null"
    Else
        sTempatLahir = "" + sTempatLahir + ""
    End If

    If dTanggalLahir = "" Or dTanggalLahir = "<NULL>" Or dTanggalLahir = "NULL" Then
        dTanggalLahir = "null"
    Else
        dTanggalLahir = "" + Format(dTanggalLahir, "YYYY-MM-DD") + ""
    End If

    If sAlamat = "" Or sAlamat = "<NULL>" Or sAlamat = "NULL" Then
        sAlamat = "null"
    Else
        sAlamat = "" + sAlamat + ""
    End If

    If sKodeJur = "" Or sKodeJur = "<NULL>" Or sKodeJur = "NULL" Then
        sKodeJur = "null"
    Else
        sKodeJur = "" + sKodeJur + ""
    End If

    SqlUpd = "Update Mahasiswa " & _
        "Set Nama      =" & sNama & "," & _
        "TempatLahir =" & sTempatLahir & "," & _
        "TanggalLahir =" & dTanggalLahir & "," & _
        "Alamat      =" & sAlamat & "," & _
        "KodeJur     =" & sKodeJur + " " & _
        "Where NIM =" & sNim
    Debug.Print SqlUpd
    AkadCN.Execute SqlUpd
    updateMahasiswa = True
    Exit Function
errUpd:
    MsgBox Err.Description
    updateMahasiswa = False
End Function

Public Function deleteMahasiswa(sNim) As Boolean
    Dim SqlDel As String

    On Error GoTo errDel

    SqlDel = "Delete From Mahasiswa " & _
        "Where NIM =" & sNim & ""
    AkadCN.Execute SqlDel
    deleteMahasiswa = True
    Exit Function
errDel:

```

```

errDel:
    deleteMahasiswa = False
End Function

Public Function getMahasiswa(Optional sNim As String) As ADODB.Recordset
Dim SqlGet As String
On Error GoTo errGet

    If sNim = "" Or IsNull(sNim) Then
        SqlGet = "Select Mahasiswa.*,Jurusan>NamaJurusan From Mahasiswa left join
Jurusan on Mahasiswa.KodeJur=Jurusan.KodeJurusan"
    Else
        SqlGet = "Select Mahasiswa.*,Jurusan>NamaJurusan From Mahasiswa left join
Jurusan on Mahasiswa.KodeJur=Jurusan.KodeJurusan Where NIM ="" & sNim & ""
    End If

    Set getMahasiswa = AkadCN.Execute(SqlGet)
Exit Function
errGet:
    Set getMahasiswa = Nothing
    MsgBox "Ada Kesalahan pada class getmahasiswa"
End Function

```

7. Tambahkan satu class baru dengan nama clsJurusan
8. Klik kanan classAkademik.vbp, pilih add, class module
9. Ganti nama class1 menjadi clsJurusan
10. Pada jendela kode clsJurusan ketikkan program berikut:

```

Dim AkadCN As New ADODB.Connection

Private Sub Class_Initialize()
    Set AkadCN = New ADODB.Connection
    AkadCN.ConnectionString = "Provider=SQLOLEDB.1; Integrated Security=SSPI;
Data Source=(local); Initial Catalog =AKADEMIK"
    AkadCN.CursorLocation = adUseServer
    AkadCN.Open
End Sub

Private Sub Class_Terminate()
    AkadCN.Close
    Set AkadCN = Nothing
End Sub

Public Function getJurusan(Optional sKdJur As String) As ADODB.Recordset
Dim SqlGet As String
On Error GoTo errGet

    If sKdJur = "" Or IsNull(sKdJur) Then
        SqlGet = "Select * From Jurusan order by kodejurusan"
    Else
        SqlGet = "Select * From Jurusan where KodeJurusan="" + sKdJur + ""
    End If

    Set getJurusan = AkadCN.Execute(SqlGet)
Exit Function
errGet:
    Set getJurusan = Nothing
    MsgBox "Ada Kesalahan pada class getJurusan"
End Function

```

Saat ini kita mempunyai satu project dengan dua class, untuk menguji atau mencoba class tersebut buatlah form untuk entry data mahasiswa, agar project antara class dan form terpisah tambahkan project baru.

1. Pilih menu File, Add Project, Standard EXE
2. Pilih menu project, reference
3. Aktifkan library classAkademik, pilih menu Project, Project References, aktifkan classAkademik.
4. Aktifkan Project1 sebagai startup project, pada jendela project group klik kanan Project1, pilih set as start up.
5. Modifikasi Form1 seperti gambar dan daftar property berikut:

	NIM	Nama
2	2003110128	Yadi
3	2003320121	Heni Zaki Ananda
4	2003320127	Firman Yadita
5	2003320129	Gandi Sugandi
6	2003320130	Shanty Nirmala
7	2003420126	Kabulo Kajate
8	2003420199	Marithza Yadita

NIM:
 Nama:
 Tempat Lahir:
 Tanggal Lahir:
 Alamat:
 Kode Jurusan:
 Nama Jurusan:

Object	Property	Setting
Form1	Name Caption	FrmTestClass Test Class
MSFlexGrid	Name HighLight FocusRect SelectionMode AllowUserResizing	Flex 2 - flexHighLightWithFocus 0 - flexFocusNone 1 - flexSelectionByRow 0 - flexResizeColumns
Frame	Name	Frame1

	Caption	
Frame	Name Caption	Frame2
TextBox	Name MaxLength	TxtNim 10
TextBox	Name MaxLength	TxtNama 40
TextBox	Name MaxLength	TxtTempatLahir 30
TextBox	Name MaxLength	TxtTanggalLahir 10
TextBox	Name MaxLength	TxtAlamat 40
TextBox	Name MaxLength	TxtKodeJur 2
TextBox	Name MaxLength	TxtNamaJurusan 30
CommandButton	Name Index Caption Index Caption Index Caption Index Caption	CmdNav 0 << 1 << 2 >> 3 >>
CommandButton	Name Index Caption Index Caption Index Caption Index Caption Index Caption Index Caption	CmdAction 0 &Baru 1 &Edit 2 &Hapus 3 &Refresh 4 Simpan 5 Batal

Pada jendela kode FrmTestClass ketikkan listing program berikut:

```
Dim cMhs As New ClassAkademik.clsMahasiswa
Dim cJur As New ClassAkademik.clsJurusan
Dim onEdit As Boolean

Private Sub cmdAction_Click(Index As Integer)
    Dim bSuccess
    On Error GoTo errAction
    Select Case Index
        Case 0 'Button untuk Menambah Baris Baru
            txtNIM.Locked = False
            onEdit = True
            ClearTxt
        Case 1
            txtNIM.Locked = True
            onEdit = True
        Case 2 'Button untuk Menghapus baris
            If MsgBox("Anda Yakin Menghapus Mahasiswa dg NIM:" + txtNIM + " ?",
vbQuestion + vbYesNo, "Latihan Class") = vbYes Then
                If Not cMhs.deleteMahasiswa(txtNIM) Then
                    MsgBox "Mahasiswa dengan NIM ." + txtNIM + " Tidak dapat dihapus!"
                Else
                    Flex.RemoveItem Flex.Row
                    ShowTxt
                End If
            End If
        Case 3
            ShowMhs
        Case 4

            If Trim(txtNIM) = "" Then
                MsgBox "NIM harus diisi"
                txtNIM.SetFocus
            Exit Sub
            End If

            If Trim(txtNama) = "" Then
                MsgBox "Nama harus diisi"
                txtNama.SetFocus
            Exit Sub
            End If

            If Not txtNIM.Locked Then
                If Not cMhs.addMahasiswa(txtNIM.Text, txtNama.Text, txtTempatLahir.Text,
txtTanggalLahir.Text, txtAlamat.Text, txtKodeJur.Text) Then
                    MsgBox "Terjadi kesalahan pada saat penambahan baris baru"
                Else
                    Flex.AddItem Str(Flex.Rows) + vbTab + txtNIM.Text + _
vbTab + txtNama.Text + vbTab + txtTempatLahir.Text + _
vbTab + txtTanggalLahir.Text + vbTab + txtAlamat.Text + _
vbTab + txtKodeJur.Text + vbTab + txtNamaJurusan.Text
                    Flex.Row = Flex.Rows - 1
                    Flex.RowSel = Flex.Rows - 1
                End If
            Else
                If Not cMhs.updateMahasiswa(txtNIM.Text, txtNama.Text,
txtTempatLahir.Text, txtTanggalLahir.Text, txtAlamat.Text, txtKodeJur.Text) Then
                    MsgBox "Terjadi kesalahan pada saat update"
                Else
                    Flex.TextMatrix(Flex.Row, 0) = Flex.Row
                End If
            End If
        End Sub
```

```

        Flex.TextMatrix(Flex.Row, 1) = txtNIM.Text
        Flex.TextMatrix(Flex.Row, 2) = txtNama.Text
        Flex.TextMatrix(Flex.Row, 3) = txtTempatLahir.Text
        Flex.TextMatrix(Flex.Row, 4) = txtTanggalLahir.Text
        Flex.TextMatrix(Flex.Row, 5) = txtAlamat.Text
        Flex.TextMatrix(Flex.Row, 6) = txtKodeJur.Text
        Flex.TextMatrix(Flex.Row, 7) = txtNamaJurusan.Text
    End If
End If
onEdit = False
txtNIM.Locked = True
Case 5
    onEdit = False
    Flex.Row = Flex.Row
    ShowTxt
End Select
txtNIM.Locked = True
Setbutton onEdit

Exit Sub
errAction:
    MsgBox Err.Description, vbOKOnly, "Latihan Class"
    Resume Next 'bila terjadi kesalahan akan dilanjutkan ke baris berikutnya setelah
kesalahan
End Sub

```

Private Sub CmdNav_Click(Index As Integer)

```

    Select Case Index
    Case 0
        Flex.Row = 1
    Case 1
        If Flex.Row > 1 Then
            Flex.Row = Flex.Row - 1
        End If
    Case 2
        If Flex.Row < Flex.Rows - 1 Then
            Flex.Row = Flex.Row + 1
        End If
    Case 3
        Flex.Row = Flex.Rows - 1
    End Select
    Flex.SetFocus
End Sub

```

Private Sub Flex_RowColChange()

```

    txtNIM = Flex.TextMatrix(Flex.Row, 1)
    txtNama = Flex.TextMatrix(Flex.Row, 2)
    txtTempatLahir = Flex.TextMatrix(Flex.Row, 3)
    txtTanggalLahir = Flex.TextMatrix(Flex.Row, 4)
    txtAlamat = Flex.TextMatrix(Flex.Row, 5)
    txtKodeJur = Flex.TextMatrix(Flex.Row, 6)
    txtNamaJurusan = Flex.TextMatrix(Flex.Row, 7)
End Sub

```

Private Sub Form_Load()

```

    onEdit = False
    Setbutton onEdit
    ShowMhs
    Flex.Row = 1
    ShowTxt
End Sub

```

Private Sub ShowMhs()

Dim rsMhs As ADODB.Recordset

Set rsMhs = cMhs.getMahasiswa

If rsMhs Is Nothing Then

MsgBox "Data Mahasiswa Tidak ditemukan"

Flex.Rows = 1

Exit Sub

Else

Flex.Rows = 1

Flex.Cols = rsMhs.Fields.Count + 1

Flex.ColWidth(0) = 500

For i = 0 To rsMhs.Fields.Count - 1

Flex.TextMatrix(0, i + 1) = rsMhs.Fields(i).Name

*Flex.ColWidth(i + 1) = rsMhs.Fields(i).DefinedSize * TextWidth("W")*

Next

While Not rsMhs.EOF

Flex.Rows = Flex.Rows + 1

Flex.TextMatrix(Flex.Rows - 1, 0) = Flex.Rows - 1

For icol = 0 To rsMhs.Fields.Count - 1

If IsNull(rsMhs.Fields(icol)) Then

Flex.TextMatrix(Flex.Rows - 1, icol + 1) = ""

Else

Flex.TextMatrix(Flex.Rows - 1, icol + 1) = rsMhs.Fields(icol)

End If

Next

rsMhs.MoveNext

Wend

End If

End Sub

Private Sub Setbutton(ByVal bEdit)

For i = 0 To cmdAction.Count - 1

cmdAction(i).Visible = Not bEdit

Next

cmdAction(0).Visible = Not bEdit

cmdAction(1).Visible = Not bEdit

cmdAction(4).Visible = bEdit

cmdAction(5).Visible = bEdit

For i = 0 To CmdNav.Count - 1

CmdNav(i).Enabled = Not bEdit

Next

Frame1.Enabled = bEdit

End Sub

Private Sub ClearTxt()

txtNIM = ""

txtNama = ""

txtTempatLahir = ""

txtTanggalLahir = ""

txtAlamat = ""

txtKodeJur = ""

txtNamaJurusan = ""

End Sub

Private Sub ShowTxt()

```
txtNIM = Flex.TextMatrix(Flex.Row, 1)
txtNama = Flex.TextMatrix(Flex.Row, 2)
txtTempatLahir = Flex.TextMatrix(Flex.Row, 3)
txtTanggalLahir = Flex.TextMatrix(Flex.Row, 4)
txtAlamat = Flex.TextMatrix(Flex.Row, 5)
txtKodeJur = Flex.TextMatrix(Flex.Row, 6)
txtNamaJurusan = Flex.TextMatrix(Flex.Row, 7)
```

End Sub***Private Sub txtKodeJur_Validate(Cancel As Boolean)***

```
Dim rsJur As ADODB.Recordset
If txtKodeJur <> "" Then
    Set rsJur = cJur.getJurusan(txtKodeJur)
    If Not rsJur.EOF Then
        txtNamaJurusan = rsJur!NamaJurusan
    Else
        MsgBox "Kode Jurusan Tidak Ditemukan"
        Cancel = True
    End If
Else
    txtNamaJurusan = ""
End If
End Sub
```

6.7.Tugas

- Buatlah class-class untuk mengakses table-table lain (jurusan, Jurusan, mata kuliah dan KRS) seperti latihan diatas. Buat class-class tersebut menjadi satu library classAkademik.dll.
- Gunakan class-class tersebut untuk membuat form-form entryan untuk masing-masing tabel.
- Gabungkan ke dalam satu menu.

Bab VII

Crystal Report

7.1.Tujuan

Mahasiswa mampu membuat dan menghubungkan antara Crystal Report dengan program aplikasi menggunakan ActiveX Data Object.

7.2.Materi

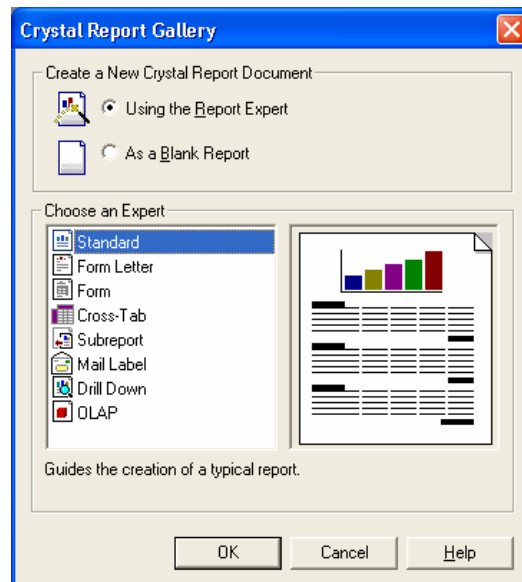
- Crystal Report
- Implementasi Crystal Report

7.3.Alat dan bahan

- PC yang telah terinstall system operasi Windows 2000/ Windows NT/ Windows 9x.
- Microsoft SQL Server 2000
- Visual Basic 6.0
- Crystal Report

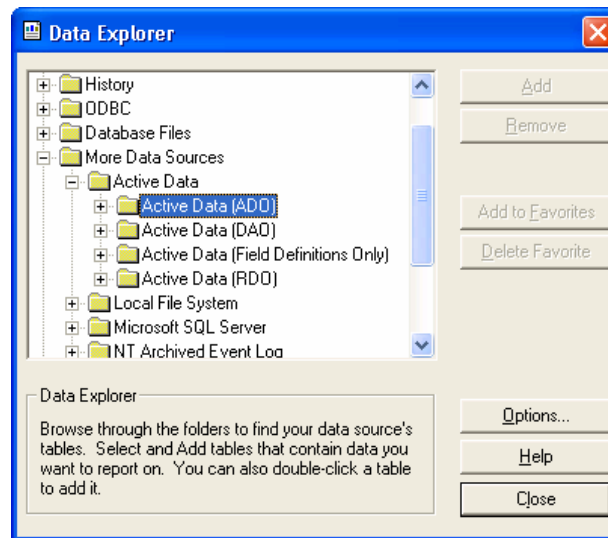
7.4.Langkah-langkah Pembuatan Report dengan Crystal Report

- Dari menu Start >Program>Crystal Report Tools>Crystal Reports
- Buat Report Baru dari menu File>New, maka akan muncul form *Crystal Report Gallery* seperti dibawah ini :

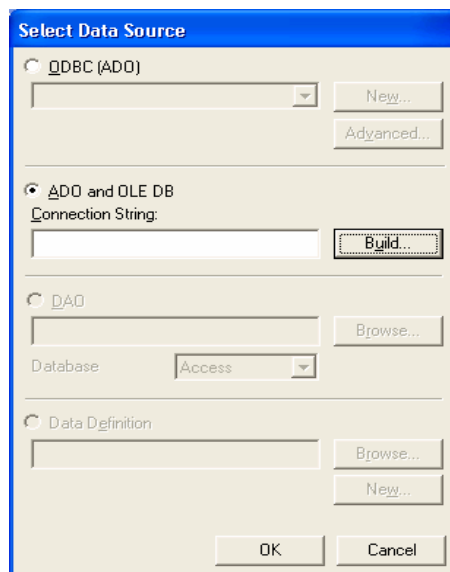


- Klik OK
- Pilih Database

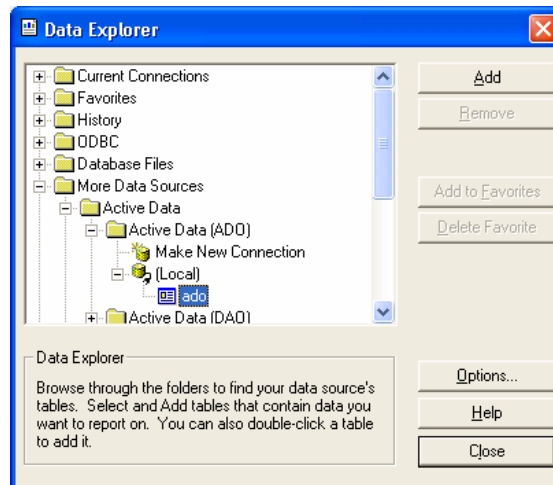
- e. Lalu pilih More Data Sources> Active Data> Active Data(ADO)>



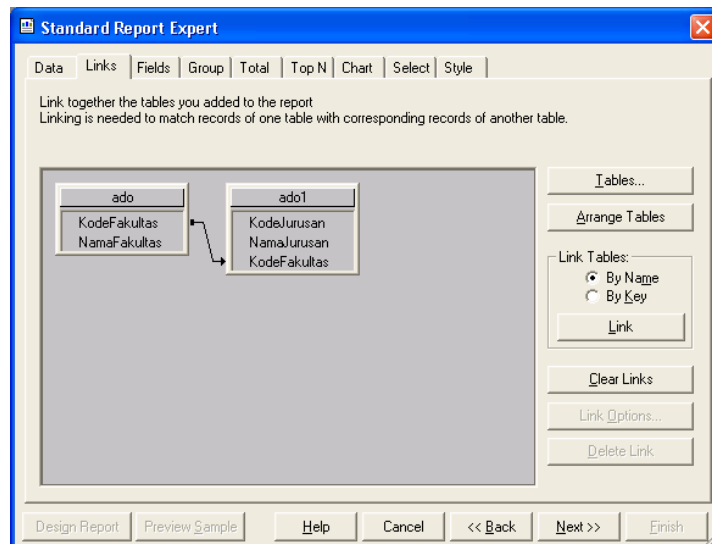
- f. Pilih ADO and OLE DB lalu tekan Build, ikuti petunjuk yang ada.



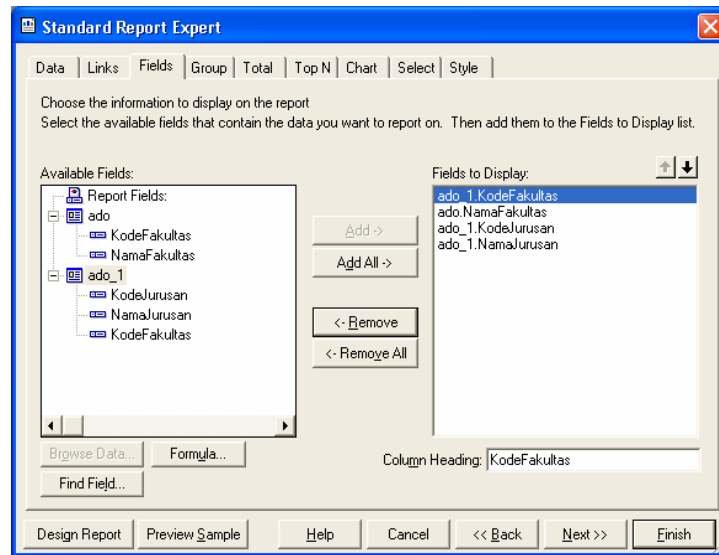
- g. Klik OK. Sehingga form menjadi seperti berikut :



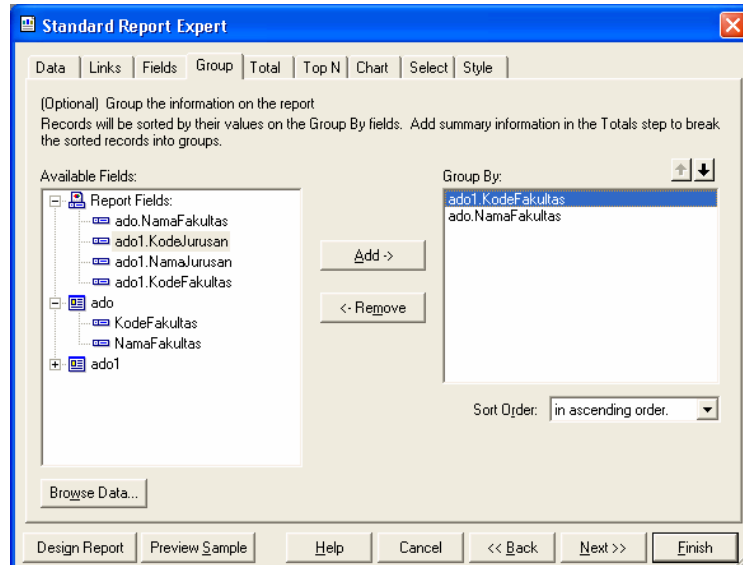
- h. Klik ganda **ado**, pada combo OBJECT pilih nama table yang ingin ditampilkan lalu tekan OK. Jika table yang ditampilkan lebih dari satu maka ulangi lagi langkah tsb.
- i. Klik Close. Maka muncul form *Standard Report Expert*. Tekan **Next >>**.



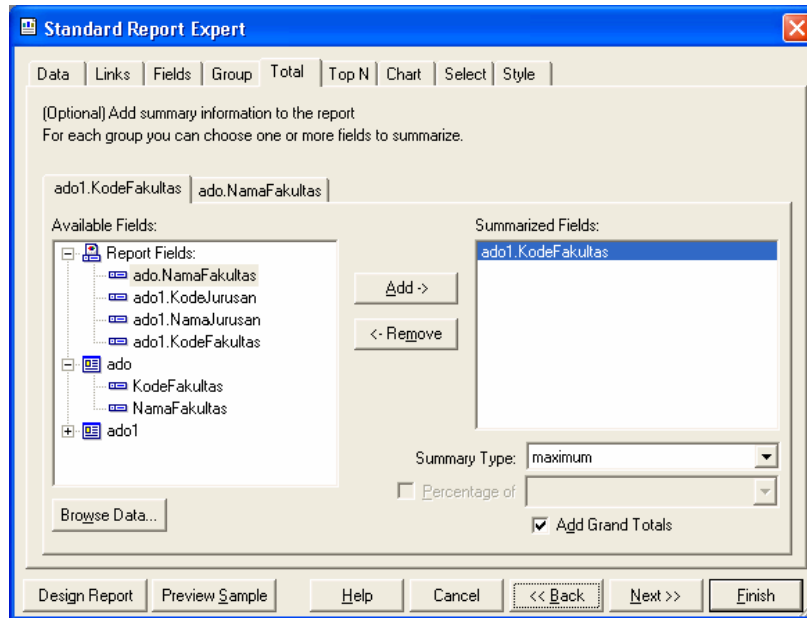
- j. Pilih / select field-field yang ingin ditampilkan, tekan NEXT



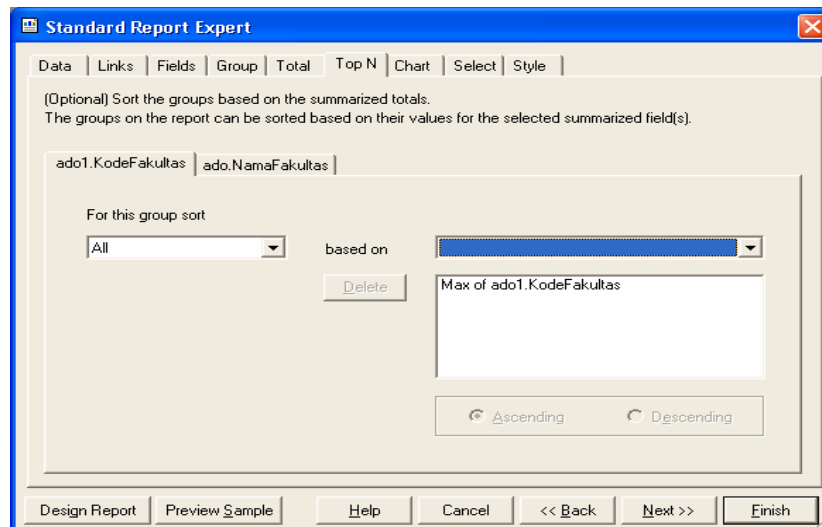
k. Pilih Field yang ingin di GROUP BY, tekan NEXT



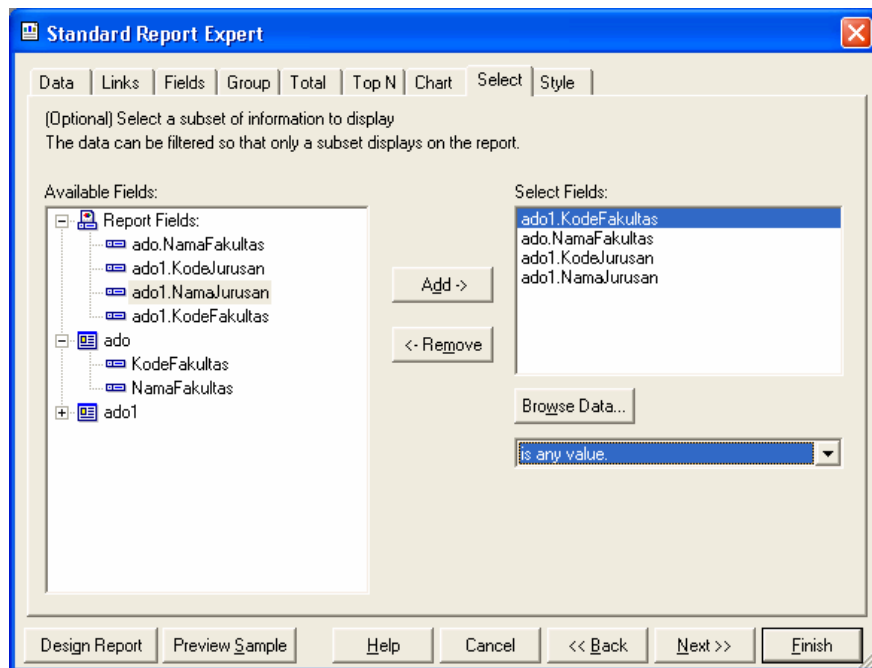
1. Pilih field yang ingin diberi fungsi *Agregate Function*



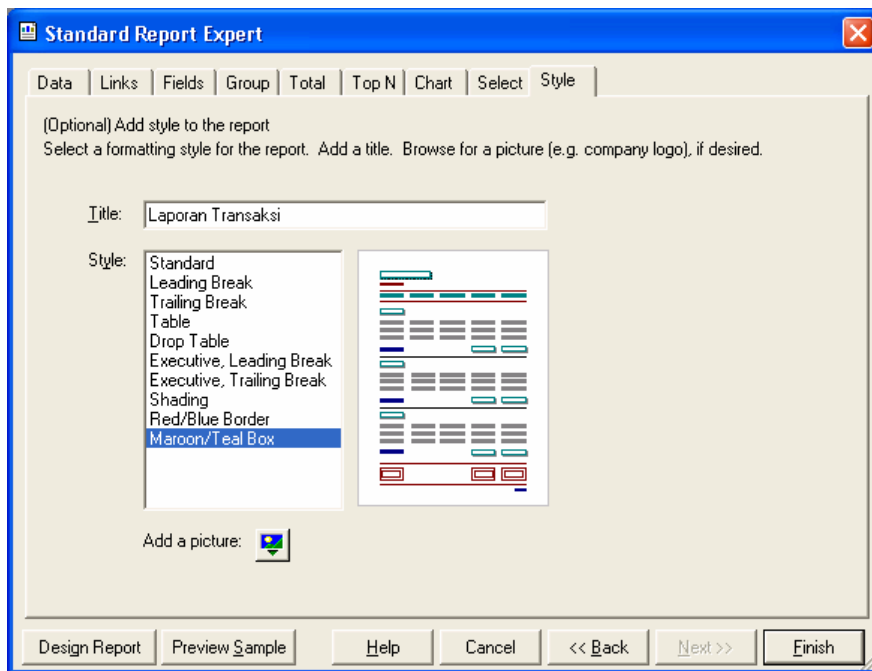
- m. Pilih field untuk mengurutkan data. Tekan NEXT.



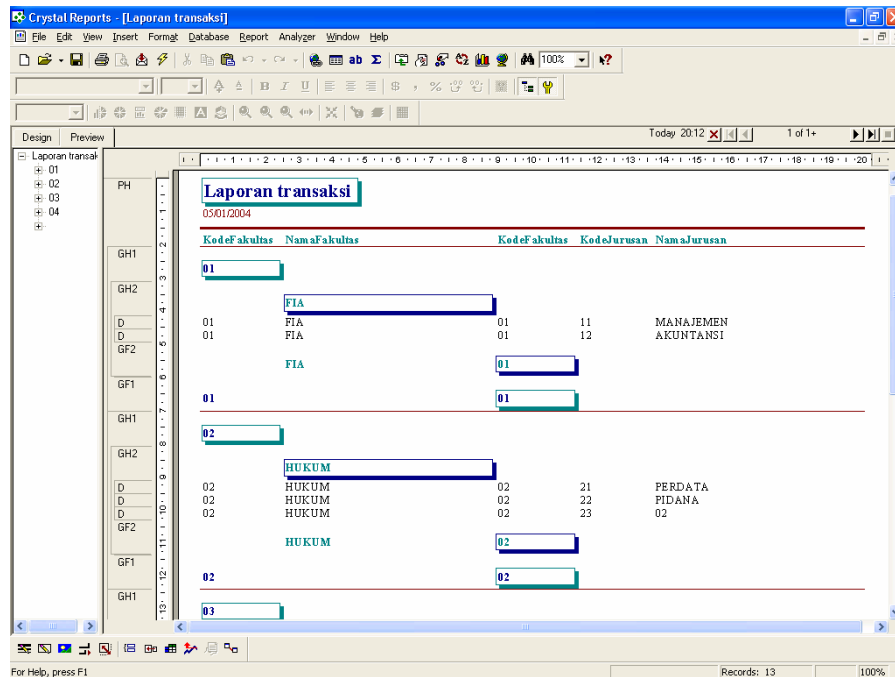
- n. Pilih Field-field yang inginditampilkan.



- o. Tulis judul Laporan, pilih Style: *Maroon/Teal Box*, lalu klik FINISH.



- p. Maka hasil reportnya seperti dibawah ini :



- q. Simpan.
- r. Form Master berisi Tombol Report

KodeFakultas	NamaFakultas
01	FIA
02	HUKUM
03	SASTRA
04	TEKNIK
05	FIKOM
06	EKONOMI

- s. Tulis program dibawah ini pada Command Report.

Private Sub cmdReport_Click()

CrystalReport1.ReportFileName = ".....\ Laporan Data Master Fakultas.rpt"

CrystalReport1.WindowState = crptMaximized

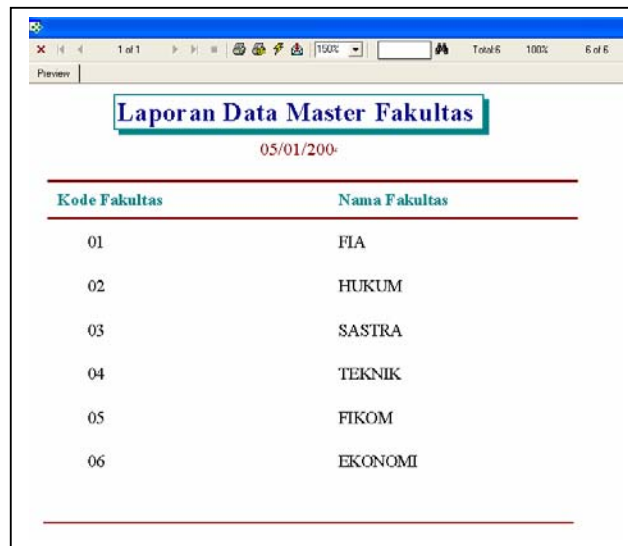
CrystalReport1.Destination = crptToWindow

CrystalReport1.RetrieveDataFiles

CrystalReport1.Action = 1

End Sub

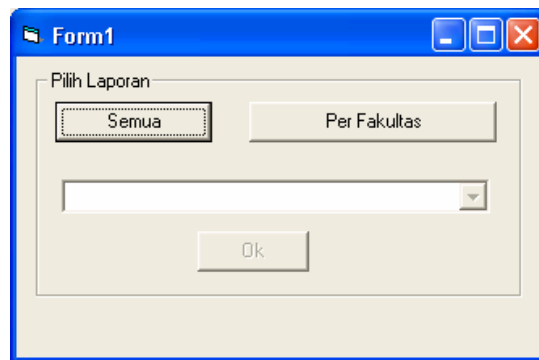
- t. Tampilan Report Data Master.



The screenshot shows a Crystal Report window with a title bar. The report title is 'Laporan Data Master Fakultas' in a blue box. Below the title is the date '05/01/200'. The report contains a table with two columns: 'Kode Fakultas' and 'Nama Fakultas'. The table lists six faculties: 01 FIA, 02 HUKUM, 03 SASTRA, 04 TEKNIK, 05 FIKOM, and 06 EKONOMI. The report is displayed in a preview window with a toolbar at the top.

Kode Fakultas	Nama Fakultas
01	FIA
02	HUKUM
03	SASTRA
04	TEKNIK
05	FIKOM
06	EKONOMI

- u. Form untuk menentukan Laporan Transaksi yang ingin ditampilkan.



The screenshot shows a form window titled 'Form1'. It has a section titled 'Pilih Laporan' with two buttons: 'Semua' and 'Per Fakultas'. Below these buttons is a text box with a dropdown arrow. At the bottom of the form is an 'Ok' button.

- v. Tulis program dibawah ini:

```
Private Sub cmdSemua_Click()  
CrystalReport1.ReportFileName = ".....\ TRANS.rpt"  
CrystalReport1.WindowState = crptMaximized  
CrystalReport1.Destination = crptToWindow  
CrystalReport1.RetrieveDataFiles  
CrystalReport1.Action = 1  
End Sub
```

- w. Tampilan Report Transaksi untuk semua Fakultas.

1 of 1 100% Total:13 100% 13 of 13

Preview

Laporan Transaksi
05/01/2004

Kode fakultas	Nama fakultas	Kode jurusan	Nama jurusan
01	FIA	11 12	MANAJEMEN AKUNTANSI
02	HUKUM	21 22 23	PERDATA PIDANA 02
03	SASTRA	31 32 33	INGGRIS JEPANG 03
04	TEKNIK	41 42 43	SIPIL INFORMATIKA 04
05			

- x. Untuk menentukan Report perFakultas.

Form1

Pilih Laporan

Semua Per Fakultas

TEKNIK

FIA
HUKUM
SASTRA
TEKNIK
FIKOM
EKONOMI

- y. Tulis program dibawah ini:

Private Sub cmdPerFakultas_Click()

Combo1.Enabled = True

cmdOk.Enabled = True

Set rsRFak = New ADODB.Recordset

rsRFak.CursorLocation = adUseClient

*rsRFak.Open "Select * From Fakultas", db, adOpenDynamic,
adLockOptimistic*

With rsRFak

.MoveFirst

While Not .BOF And Not .EOF

```

Combo1.AddItem rsRFak!NamaFakultas
.MoveNext
Wend
End With
End Sub

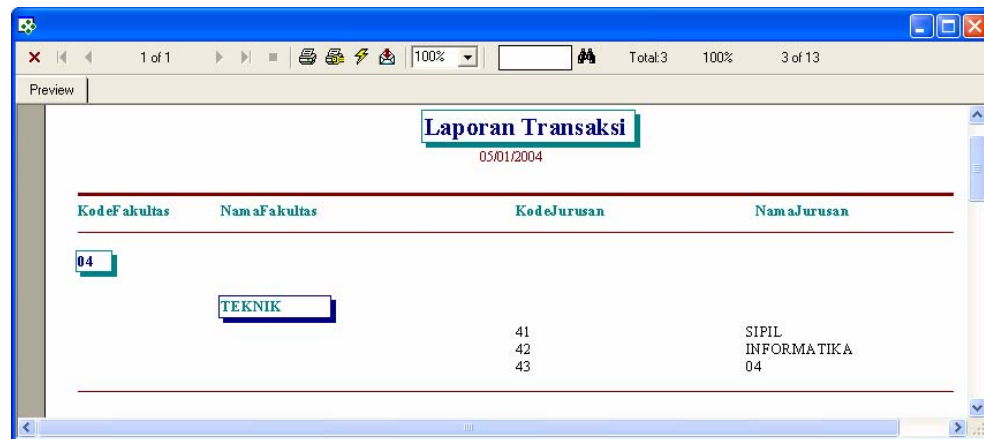
```

```

Private Sub cmdOk_Click()
CrystalReport1.ReportFileName = "D:\Document
Assisten\dnx\ReportPrak\trans.rpt"
CrystalReport1.WindowState = crptMaximized
CrystalReport1.SelectionFormula = "{ado.NamaFakultas} = '" &
Combo1.Text & "'"
CrystalReport1.Destination = crptToWindow
CrystalReport1.RetrieveDataFiles
CrystalReport1.Action = 1
End Sub

```

- z. Tampilan Report Transaksi PerFakultas.



KodeFakultas	NamaFakultas	KodeJurusan	NamaJurusan
04	TEKNIK	41	SIPIL INFORMATIKA 04
		42	
		43	

7.5.Tugas :

Buat Laporan Master Mahasiswa dan Mata Kuliah serta Laporan Transaksi KRS.