

# Pertemuan 9

# Bahasa Query Formal

# Bahasa Query Formal

Dalam bahasa Query Formal, ada dua dasar pembentukan bahasa Query, yaitu:

1. Aljabar Relasional
2. Kalkulus Relasional

Dalam pembahasan ini hanya akan membahas tentang Aljabar Relasional karna lebih banyak dijadikan dasar Bahasa Query yang umum digunakan.

# Aljabar Relasional

## ALJABAR RELASIONAL

Adalah kumpulan operasi terhadap relasi, dimana setiap operasi menggunakan satu atau lebih relasi untuk menghasilkan satu relasi yang baru.

Bahasa *Query* yang didasarkan pada operasi-operasi dalam Aljabar Relasional merupakan bahasa *query* yang **Prosedural**.

# Aljabar Relational

## B. OPERATOR RELATIONAL

1. Restrict (  $\sigma$  ) adalah Pemilihan tupel atau record
2. Project (  $\pi$  ) adalah pemilihan attribute atau field
3. Divide (  $\div$  ) adalah membagi
4. Join (  $\theta$  ) adalah menggabungkan

## ALJABAR RELASIONAL

Operator pada aljabar relational dibagi menjadi 2 kelompok:

1. Operator dasar untuk fundamental operational
2. Operator tambahan untuk additional operasional

# Contoh

Tabel dibawah ini adalah contoh untuk mengerjakan perintah – perintah Relation Algebra:

RELASI : MATA KULIAH

KD_MK	NAMA_MK	SKS	NIP
207	LOGIKA & ALGO	4	199910486
310	STRUKTUR DATA	3	200109655
360	SISTEM BASIS DATA	3	200209817
545	IMK	2	200209818
547	APSI	4	200109601
305	PEMR. PASCAL	4	200703073
544	DISAIN GRAFIS	2	200010490

## RELASI : MAHASISWA

NIM	NAMA_MHS	ALAMAT	J_KEL
1105090222	HAFIDZ	DEPOK	LAKI-LAKI
1105091002	RAFFA	DEPOK	LAKI-LAKI
1105095000	NAIA	DEPOK	PEREMPUAN
1104030885	ARIF	P.LABU	LAKI-LAKI
1206090501	LENI	KMP. MELAYU	PEREMPUAN
1206090582	WAHYUNI	TANGERANG	PEREMPUAN
1205097589	ARIS	DEPOK	LAKI-LAKI
1106094586	YANI	CILEDUG	PEREMPUAN
110709	BAMBANG	SALEMBA	LAKI-LAKI

## RELASI : REGISTRASI

KD_MK	NIM
360	1105090222
545	1206090501
547	1105095000

## RELASI : DOSEN

NIP	NAMA_DOS	GAJI
199910486	BILLY	3500000
200109655	MARDIANA	4000000
200209817	INDRIYANI	4500000
200209818	SURYANI	4250000
200109601	DWINITA	3500000
200703073	MALAU	2750000
200010490	IRFIANI	3500000

# Operator Dasar

## a. Selection ( $\sigma$ ) Lower Case Omega

Operasi selection menyeleksi tupel-tupel pada sebuah relation yang memenuhi predicate/syarat yang sudah ditentukan

Contoh :

1. Mencari tuple-tuple dari MAHASISWA yang memiliki jenis kelamin laki-laki, Ekspresi aljabar relational :

$\sigma$  J\_KEL="LAKI-LAKI" (MAHASISWA)

2. Tampilkan data mata kuliah yang memiliki kode 360 atau yang memiliki sks 4

$\sigma$  KD\_MK="306"  $\vee$  SKS=4 (MATAKULIAH)



# Operator Dasar lanjutan

## b. Projection ( $\pi$ )

Operator projection beroperasi pada sebuah relation, yaitu membentuk relation baru dengan mengcopy attribute-attribute dan domain-domain dari relation tersebut berdasarkan argumen-argumen pada operator tersebut.

Contoh :

Tampilkan nama beserta gaji dari dosen

$\pi_{\text{nama\_dos}, \text{gaji}}$  (DOSEN)

# Operator Dasar lanjutan

## c. Cartesian product ( X )

Operator dengan dua relasi untuk menghasilkan tabel hasil perkalian kartesian.

Contoh :

Tampilkan nid,nama\_d (dari relasi Dosen), nama\_mk (dari relasi Matakuliah), thn\_akademik,smt,hari,jam\_ke,waktu,kelas (dari relasi Mengajar) dimana semester mengajar adalah pada semester '1'.

**$\pi$  nid, nama\_d, nama\_mk, thn\_akademik,smt, hari,jam\_ke, waktu, kelas (  $\sigma$  smt=1  $\wedge$  Dosen.nid = Mengajar.nid  $\wedge$  mengajar.kdmk = Matakuliah.kdmk (DosenxMatakuliahxMengajar))**

# Operator Dasar lanjutan

## d. Union ( $\cup$ )

Operasi untuk menghasilkan gabungan tabel dengan syarat kedua tabel memiliki atribut yang sama yaitu domain atribut ke-i masing-masing tabel harus sama

$R \cup S = \{ X \mid X \in R \text{ atau } X \in S \}$

Contoh :

Penggabungan berdasarkan kolom kota dari tabel mahasiswa dengan tabel dosen

$\pi \text{ kota (mahasiswa)} \cup \pi \text{ kota (Dosen)}$

# Operator Dasar lanjutan

## e. Set difference ( - )

Operasi untuk mendapatkan tabel dis uatu relasi tapi tidak ada di relasi lainnya.

$$R - S = \{ X \mid X \in R \text{ dan } X \notin S \}$$

Contoh : Tampilkan nama dari mahasiswa yang tinggal di depok tetapi bukan berjenis kelamin perempuan

Query I : tampilkan nama yang tinggal di depok  
 $\pi_{\text{nama\_mhs}}(\sigma_{\text{alamat}=\text{"DEPOK"}}(\text{MAHASISWA}))$

Query II : tampilkan nama yang berjenis kelamin perempuan  
 $\pi_{\text{nama\_mhs}}(\sigma_{\text{j\_kel}=\text{"PEREMPUAN"}}(\text{MAHASISWA}))$

Tampilkan query I minus query II :

$\pi_{\text{nama\_mhs}}(\sigma_{\text{alamat}=\text{"DEPOK"}}(\text{MAHASISWA})) -$   
 $\pi_{\text{nama\_mhs}}(\sigma_{\text{j\_kel}=\text{"PEREMPUAN"}}(\text{MAHASISWA}))$

# Operator Tambahan

## 1. SET INTERSECTION ( $\cap$ )

Operasi untuk menghasilkan irisan dua tabel dengan syarat kedua tabel memiliki atribut yang sama, domain atribut ke-i kedua tabel tersebut sama.

## 2. THETA JOIN

Operasi yang menggabungkan operasi cartesian product dengan operasi selection dengan suatu kriteria.

## 3. NATURAL JOIN

Operasi menggabungkan operasi selection dan cartesian product dengan suatu kriteria pada kolom yang sama

# Operator Tambahan lanjutan

## 4. DIVISION

Merupakan operasi pembagian atas tuple-tuple dari 2 relation

Contoh:

Sno	Pno
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2

A

B

Pno
P2
A/B
Sno
S1
S2

# Pertemuan 10

## Bahasa Query Terapan

# *Structured Query Language* (SQL)

**SQL** merupakan bahasa query terapan yang banyak digunakan oleh berbagai DBMS, diterapkan dalam berbagai *development tools* dan program aplikasi untuk berinteraksi dengan basis data.

## **Subdivisi SQL:**

1. *Data Definition Language (DDL)*

Query-query ini digunakan untuk mendefinisikan struktur atau skema basis data.

2. *Data Manipulation Language (DML)*

Query-query ini digunakan untuk manajemen data dalam basis data.



# SQL lanjutan

## PENGELOMPOKAN STATEMEN SQL

### 1. *Data Definition Language* (DDL)

CREATE DATABASE	DROP DATABASE
CREATE TABEL	DROP TABEL
CREATE INDEX	DROP INDEX
CREATE VIEW	DROP VIEW
ALTER TABLE	

### 2. *Data Manipulation Language* (DML)

INSERT, SELECT, UPDATE, DELETE

# SQL lanjutan

- 3. Data Access  
GRANT , REVOKE
- 4. Data Integrity  
RECOVER TABLE
- 5. Auxiliary  
SELECT INTO OUTFILE,  
LOAD, RENAME TABLE

# Data Definition Language (DDL)

## A. CREATE

### 1. Pembuatan Database

Nama Database adalah yang dapat mewakili suatu kejadian dapat berupa nama organisasi atau perusahaan.

Sintaks : `CREATE DATABASE nama_database`

Contoh : Buat database dengan nama KAMPUS  
`CREATE DATABASE KAMPUS`

### 2. Pembuatan Tabel

Sintaks : `CREATE TABLE nama_table  
( nama_kolom1 tipe_data_kolom1,  
nama_kolom2,tipe_data_kolom2,...)`

Contoh :

Buat struktur tabel dengan nama tabel Mahasiswa dengan data NIM char(8), NAMA char(25), ALAMAT char(30)

`CREATE TABLE Mahasiswa (NIM char(8) not null,  
NAMA char(25) notnull, ALAMAT char(30) notnull)`

# DDL lanjutan

## 3. Pembuatan Index

Sintaks : `CREATE [UNIQUE] INDEX nama_index  
ON nama_table (nama_kolom) ;`

Contoh :

Buat index data Mahasiswa berdasarkan NIM dengan nama MHSIDX  
Dimana NIM tidak boleh sama

```
CREATE UNIQUE INDEX MHSIDX ON Mahasiswa(NIM)
```

## 4. Pembuatan View

Sintaks :

```
CREATE VIEW nama_view [ (nama_kolom1,...) ]  
AS SELECT statement  
[WITH CHECK OPTION] ;
```

Contoh :

Buat view dengan nama MHSVIEW yang berisi semua data mahasiswa

```
CREATE VIEW MHSVIEW  
AS SELECT * FROM Mahasiswa
```

# DDL lanjutan

## B. DROP (MENGHAPUS)

1. Menghapus Database

Sintaks : DROP DATABASE nama\_db ;

2. Menghapus Tabel

Sintaks : DROP TABLE nama\_table ;

3. Menghapus Index

Sintaks : DROP INDEX nama\_index ;

4. Menghapus View

Sintaks : DROP VIEW nama\_view ;

Contoh :

DROP DATABASE **KAMPUS**;

DROP TABLE MHS;

DROP INDEX MHSIDX;

DROP VIEW MHSVIEW;

# DDL lanjutan

## C. ALTER TABLE (MERUBAH STRUKTUR TABEL)

Sintaks: ALTER TABLE nama\_tabel

ADD nama\_kolom jenis\_kolom

[FIRST | AFTER nama\_kolom]

CHANGE [COLUMN] oldnama newnama

MODIFY nama\_kolom jenis kolom, ...

DROP nama\_kolom

RENAME newnama\_tabel

Contoh :

1. Tambahkan kolom JKEL dengan panjang 1 char pada tabel Mahasiswa

```
ALTER TABLE Mahasiswa ADD JKEL char(1);
```

2. Ubah panjang kolom JKEL menjadi 15 char

```
ALTER TABLE Mahasiswa MODIFY COLUMN JKEL char(15);
```

3. Hapus kolom JKEL dari data table MHS

```
ALTER TABLE Mahasiswa DROP JKEL;
```

# Data Manipulation Language (DML)

## A. INSERT

Sintaks SQL yang digunakan untuk penambahan record baru kedalam sebuah tabel.

Sintaks: `INSERT INTO Nama_tabel [(nama_kolom1,...)]  
values (nilai atribut1, ...)`

Contoh:Masukan data Mahasiswa dengan Nim 10296832, Nama Nurhayati beralamat di Jakarta

```
INSERT INTO Mahasiswa (Nim, Nama, Alamat) values  
("10296832","Nurhayati","Jakarta");
```

# DML lanjutan

## B. UPDATE

Sintaks SQL yang digunakan untuk mengubah nilai atribut pada suatu record dari sebuah tabel.

Sintaks : UPDATE nama\_tabel  
          SET nama\_kolom = value\_1  
          WHERE kondisi ;

Contoh:

Ubah alamat menjadi “Depok” untuk mahasiswa yang memiliki NIM “10296832”

```
UPDATE Mahasiswa  
SET ALAMAT="Depok"  
WHERE NIM=" 10296832";
```



# DML lanjutan

## C. DELETE

Sintaks SQL yang digunakan untuk menghapus record dari sebuah tabel.

Sintaks: DELETE FROM nama\_table  
WHERE kondisi

Contoh:

Hapus data Mahasiswa yang mempunyai NIM  
"21198002"

```
DELETE FROM Mahasiswa  
WHERE NIM=" 21198002"
```

# DML lanjutan

Tabel dibawah ini untuk mengerjakan perintah **SELECT**

Tabel Nilai

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0

Tabel Mahasiswa

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananigrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor
10296001	Fintri	Depok
21198002	Julizar	Jakarta

Tabel MataKuliah

KD_MK	NAMA_MK	SKS
KK021	Sistem Basis Data	2
KD132	Sistem Informasi Manajemen	3
KU122	Pancasila	2

# DML lanjutan

## D. SELECT

Sintaks : `SELECT [DISTINCT | ALL] nama_kolom  
FROM nama_tabel  
[ WHERE condition ]  
[ GROUP BY column_list ]  
[HAVING condition ]  
[ ORDER BY column_list [ASC | DESC]]`

# DML lanjutan

Contoh :

- a. Tampilkan semua data Mahasiswa

```
SELECT NIM,NAMA,ALAMAT FROM Mahasiswa;
```

Atau

```
SELECT * FROM Mahasiswa;
```

**Maka hasilnya adalah :**

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor

# DML lanjutan

b. Tampilkan Mata Kuliah yang SKS nya 2

```
SELECT NAMA_MK FROM MataKuliah WHERE SKS=2
```

Maka Hasilnya:

NAMA_MK
Sistem Basis Data Pancasila

# DML lanjutan

c. Tampilkan semua data nilai dimana nilai MID lebih besar sama dengan 60 atau nilai finalnya lebih besar 75.

maka penulisannya :

```
SELECT * FROM Nilai WHERE MID >= 60 OR FINAL > 75
```

Hasilnya:

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
41296525	KU122	90	80
21196353	KU122	75	75

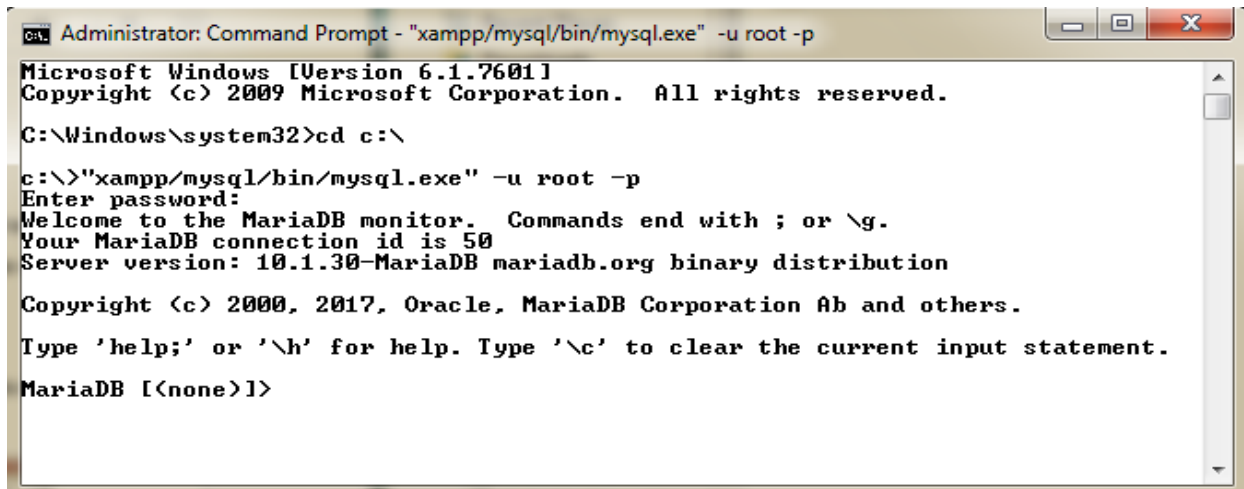
**Aplikasi yang digunakan sebagai contoh adalah Mysql**

Dari Address ketik : <http://localhost/phpmyadmin>

Tampilan user ketik **root** dan **password** dikosongkan

# Petunjuk Slide ke -10

1. Pilih menu *Start - All Programs - Accessories* Kemudian Pilih *Command Prompt*.
2. Klik Kanan pada command prompt atau cmd, lalu pilih *Run as administrator*.
3. Ketikan perintah `cd c:\` (lalu kemudian tekan Enter)
4. Lalu ketikan perintah `"xampp/mysql/bin/mysql.exe" -u root -p` (lalu kemudian tekan Enter)



```
Administrator: Command Prompt - "xampp/mysql/bin/mysql.exe" -u root -p
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\

c:\>"xampp/mysql/bin/mysql.exe" -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 50
Server version: 10.1.30-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

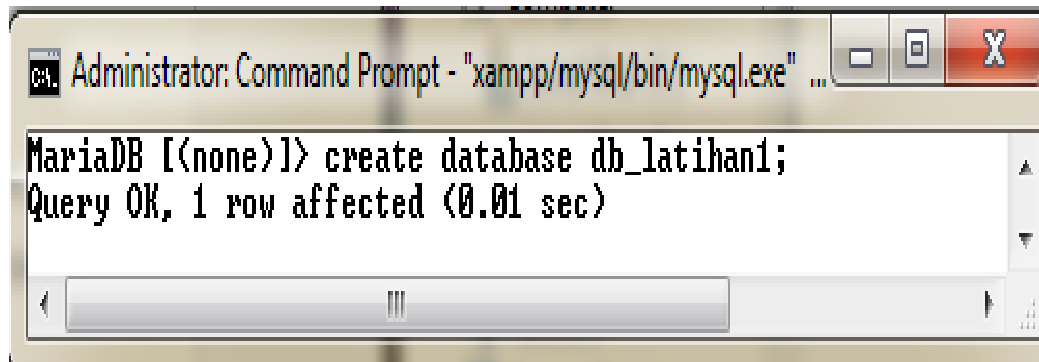
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```



## Membuat Database dan Tabel Sederhana

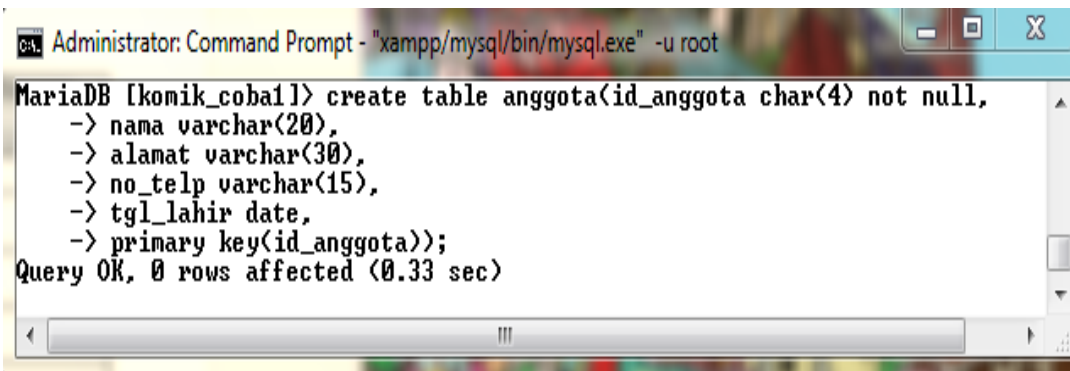
- Setelah masuk MariaDB, ketikkan perintah berikut: `create database db_latihan1;`
- Lihat hasilnya dengan perintah ***show databases;***
- Lalu, untuk membuat tabel terlebih dahulu kita aktifkan database `db_latihan1` dengan perintah ***use.***



```
Administrator: Command Prompt - "xampp/mysql/bin/mysql.exe" ...  
MariaDB [(none)]> create database db_latihan1;  
Query OK, 1 row affected (0.01 sec)
```

## Membuat Database dan Tabel Sederhana

- Perintah untuk membuat tabel, yaitu dengan perintah *create table nama\_tabel (spesifikasi tabel);*



```
Administrator: Command Prompt - "xampp/mysql/bin/mysql.exe" -u root
MariaDB [komik_coba11]> create table anggota(id_anggota char(4) not null,
-> nama varchar(20),
-> alamat varchar(30),
-> no_telp varchar(15),
-> tgl_lahir date,
-> primary key(id_anggota));
Query OK, 0 rows affected (0.33 sec)
```

- Untuk melihat daftar tabel pada suatu database, gunakan perintah *show tables;*
- Lanjutkan untuk membuat seluruh table yang ada di slide ke-10

# Tugas

- Gunakan perintah-perintah Bahasa Query terapan yang dipelajari pada pertemuan ke-10 pada project yang dipilih pada pertemuan sebelumnya

# Pertemuan 11

## **Bahasa Query Terapan Lanjutan**

# Query Terapan lanjutan

Tabel dibawah ini untuk mengerjakan perintah **JOIN**

Tabel Nilai

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0

Tabel Mahasiswa

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananigrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor
10296001	Fintri	Depok
21198002	Julizar	Jakarta

Tabel MataKuliah

KD_MK	NAMA_MK	SKS
KK021	Sistem Basis Data	2
KD132	Sistem Informasi Manajemen	3
KU122	Pancasila	2

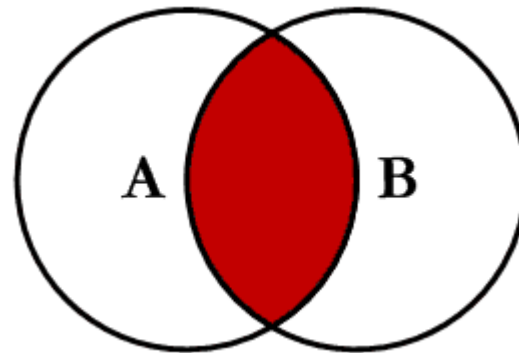
# JOIN

**JOIN** digunakan untuk memilih data dari dua tabel atau lebih.

## 1. INNER JOIN

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian.

```
SELECT select_list  
FROM Table_A A  
JOIN Table_B B  
ON A.Key = B.Key
```



Source : [https://pojokprogrammer.net/id/content/representasi-sql-join-secara-visual?language\\_content\\_entity=id](https://pojokprogrammer.net/id/content/representasi-sql-join-secara-visual?language_content_entity=id)

# Contoh INNER JOIN

```
SELECT Nilai.NIM, Mahasiswa.NAMA, Nilai.KD_MK, Nilai.MID  
FROM Nilai INNER JOIN Mahasiswa  
ON Nilai.NIM = Mahasiswa.NIM
```

Hasil :

NIM	NAMA	KD_MK	MID
10296832	Nurhayati	KK021	60
10296126	Astuti	KD132	70
31296500	Budi	KK021	55
41296525	Prananigrum	KU122	90
21196353	Quraish	KU122	75
50095487	Pipit	KD132	80

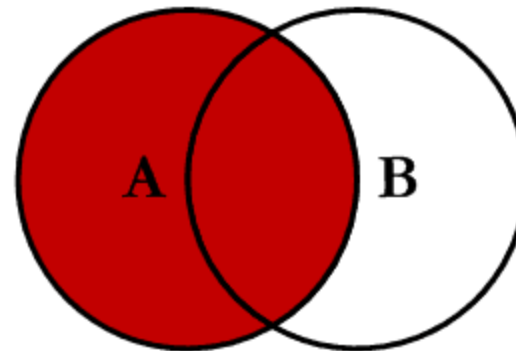
# JOIN

**JOIN** digunakan untuk memilih data dari dua tabel atau lebih.

## 2. LEFT JOIN atau LEFT OUTER JOIN

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian dan juga semua record pada tabel sebelah kiri.

```
SELECT select_list  
FROM Table_A A  
LEFT JOIN Table_B B  
ON A.Key = B.Key
```



Source : [https://pojokprogrammer.net/id/content/representasi-sql-join-secara-visual?language\\_content\\_entity=id](https://pojokprogrammer.net/id/content/representasi-sql-join-secara-visual?language_content_entity=id)



# Contoh LEFT JOIN

```
SELECT Mahasiswa.NIM, Mahasiswa.NAMA, Nilai.KD_MK,  
Nilai.MID  
FROM Mahasiswa LEFT OUTER JOIN Nilai  
ON Nilai.NIM = Mahasiswa.NIM
```

Hasil:

NIM	NAMA	KD_MK	MID
10296832	Nurhayati	KK021	60
10296126	Astuti	KD132	70
31296500	Budi	KK021	55
41296525	Prananigrum	KU122	90
21196353	Quraish	KU122	75
50095487	Pipit	KD132	80
10296001	Fintri	-	-
21198002	Julizar	-	-

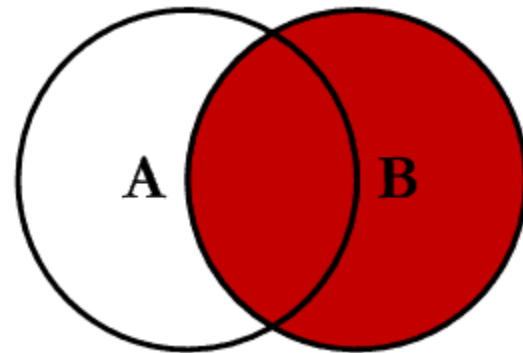
# JOIN

**JOIN** digunakan untuk memilih data dari dua tabel atau lebih.

## 3. RIGHT JOIN atau RIGHT OUTER JOIN

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian dan juga semua record pada tabel sebelah kanan.

```
SELECT select_list  
FROM Table_A A  
RIGHT JOIN Table_B B  
ON A.Key = B.Key
```



Source : [https://pojokprogrammer.net/id/content/representasi-sql-join-secara-visual?language\\_content\\_entity=id](https://pojokprogrammer.net/id/content/representasi-sql-join-secara-visual?language_content_entity=id)

# Query Terapan lanjutan

Tabel dibawah ini untuk mengerjakan perintah **JOIN**

Tabel Nilai

NIM	KD_MK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0

Tabel Mahasiswa

NIM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananigrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor
10296001	Fintri	Depok
21198002	Julizar	Jakarta

Tabel MataKuliah

KD_MK	NAMA_MK	SKS
KK021	Sistem Basis Data	2
KD132	Sistem Informasi Manajemen	3
KU122	Pancasila	2

# Contoh RIGHT JOIN

SELECT Mahasiswa.NIM, Mahasiswa.NAMA, Nilai.KD\_MK,  
Nilai.MID

FROM Nilai RIGHT OUTER JOIN Mahasiswa  
ON Nilai.NIM = Mahasiswa.NIM

Hasil :

NIM	NAMA	KD_MK	MID
10296832	Nurhayati	KK021	60
10296126	Astuti	KD132	70
31296500	Budi	KK021	55
41296525	Prananigrum	KU122	90
21196353	Quraish	KU122	75
50095487	Pipit	KD132	80
10296001	Fintri	-	-
21198002	Julizar	-	-

# Contoh Lain Join

*tb\_kota*

id_kota	nama_kota	id_provinsi
1	Jakarta	1
2	Semarang	2
3	Pati	2
4	Bandung	4
5	Surabaya	5
6	Medan	6

*tb\_provinsi*

id_provinsi	nama_provinsi
1	DKI Jakarta
2	Jawa Tengah
3	Papua Barat
4	Jawa Barat
5	Jawa Timur

Tampilkan data-data yang mempunyai nilai sama antara id\_provinsi pada table tb\_kota dan id\_provinsi pada table tb\_provinsi

SELECT \* FROM tb\_kota INNER JOIN tb\_provinsi ON tb\_kota.id\_provinsi = tb\_provinsi.id\_provinsi;

id_kota	nama_kota	id_provinsi	id_provinsi	nama_provinsi
1	Jakarta	1	1	DKI Jakarta
2	Semarang	2	2	Jawa Tengah
3	Pati	2	2	Jawa Tengah
4	Bandung	4	4	Jawa Barat
5	Surabaya	5	5	Jawa Timur

Tampilkan hanya nama\_kota dan nama\_provinsi saja

SELECT nama\_kota, nama\_provinsi FROM tb\_kota INNER JOIN tb\_provinsi ON tb\_kota.id\_provinsi = tb\_provinsi.id\_provinsi;

nama_kota	nama_provinsi
Jakarta	DKI Jakarta
Semarang	Jawa Tengah
Pati	Jawa Tengah
Bandung	Jawa Barat
Surabaya	Jawa Timur

<https://yukcoding.id/belajar-inner-left-right-join-pada-sql/>

# Contoh Lain Join

*tb\_kota*

id_kota	nama_kota	id_provinsi
1	Jakarta	1
2	Semarang	2
3	Pati	2
4	Bandung	4
5	Surabaya	5
6	Medan	6

*tb\_provinsi*

id_provinsi	nama_provinsi
1	DKI Jakarta
2	Jawa Tengah
3	Papua Barat
4	Jawa Barat
5	Jawa Timur

**Tampilkan data-data Pada table *tb\_provinsi* (kanan).**

**SELECT \* FROM *tb\_kota* LEFT JOIN *tb\_provinsi* ON *tb\_kota*.id\_provinsi = *tb\_provinsi*.id\_provinsi;**

id_kota	nama_kota	id_provinsi	id_provinsi	nama_provinsi
1	Jakarta	1	1	DKI Jakarta
2	Semarang	2	2	Jawa Tengah
3	Pati	2	2	Jawa Tengah
4	Bandung	4	4	Jawa Barat
5	Surabaya	5	5	Jawa Timur
6	Medan	6	NULL	NULL

<https://yukcoding.id/belajar-inner-left-right-join-pada-sql/>

# Contoh Lain Join

*tb\_kota*

id_kota	nama_kota	id_provinsi
1	Jakarta	1
2	Semarang	2
3	Pati	2
4	Bandung	4
5	Surabaya	5
6	Medan	6

*tb\_provinsi*

id_provinsi	nama_provinsi
1	DKI Jakarta
2	Jawa Tengah
3	Papua Barat
4	Jawa Barat
5	Jawa Timur

```
SELECT * FROM tb_kota RIGHT JOIN tb_provinsi ON  
tb_kota.id_provinsi = tb_provinsi.id_provinsi;
```

id_kota	nama_kota	id_provinsi	id_provinsi	nama_provinsi
1	Jakarta	1	1	DKI Jakarta
2	Semarang	2	2	Jawa Tengah
3	Pati	2	2	Jawa Tengah
4	Bandung	4	4	Jawa Barat
5	Surabaya	5	5	Jawa Timur
NULL	NULL	NULL	3	Papua Barat

<https://yukcoding.id/belajar-inner-left-right-join-pada-sql/>



# Data Access

**1. GRANT** → Memberikan hak akses / hak istimewa pengguna

Sintaks : GRANT hak\_akses ON nama\_db  
TO nama\_pemakai  
[IDENTIFIED BY] [PASSWORD] 'Password'  
[WITH GRANT OPTION] ;

GRANT hak\_akses ON [nama\_db]nama\_tabel  
TO nama\_pemakai  
[IDENTIFIED BY] [PASSWORD] 'Password'  
[WITH GRANT OPTION];

Contoh :

Berikan hak akses kepada Adi untuk menampilkan nilai final test pada tabel Nilai.

GRANT SELECT (FINAL) ON NILAI TO ADI



# Data Access lanjutan

2. **REVOKE** → Menarik hak akses pengguna yang diberikan lewat perintah GRANT

Sintaks : REVOKE hak\_akses ON nama\_db  
FROM nama\_pemakai ;

REVOKE hak\_akses ON nama\_tabel  
FROM nama\_pemakai ;

Contoh :

Tarik kembali dari Adi hak akses untuk menampilkan nilai final test

REVOKE SELECT (FINAL) ON NILAI FROM ADI

# Data Integrity

## RECOVER TABLE

Sintaks : `RECOVER TABLE nama_tabel`

Contoh :

Kembalikan keadaan data mahasiswa seperti pada saat sebelum terjadi kerusakan

`RECOVER TABLE MHS ;`

# Auxiliary

## 1. **SELECT ... INTO OUTFILE** 'filename'

Sintaks ini digunakan untuk mengekspor data dari tabel ke file lain.

Sintaks :                **SELECT ... INTO**  
                             **OUTFILE** 'Nama File'  
                             **[FIELDS | COLUMNS]**  
                             **[TERMINATED BY 'string']**  
                             **[[OPTIONALLY] ENCLOSED BY 'char']**  
                             **[ESCAPED BY 'char'] ]**

Contoh :

Ubah semua data mahasiswa ke bentuk ASCII dan disimpan ke file teks di directory/home/adi dengan pemisah antar kolom '|'

```
SELECT * FROM MHS
INTO OUTFILE "/home/adi/teks"
FIELDS TERMINATED BY "|";
```

# Auxiliary lanjutan

## 2. LOAD

Sintaks query ini digunakan untuk mengimpor data dari file lain ke tabel.

Sintaks :     LOAD DATA INFILE “ nama\_path”  
                  INTO TABLE nama\_tabel [ nama\_kolom] ;  
                  [FIELDS | COLUMNS]  
                  [TERMINATED BY '*string*']  
                  [[OPTIONALLY] ENCLOSED BY '*char*']  
                  [ESCAPED BY '*char*'] ]

Contoh :

Memasukkan data-data dari file teks yang berada pada direktori “/home/adi” ke dalam tabel MHS\_2. Dimana pemisah antara kolom dalam file teks adalah tab (\t) :

```
LOAD FROM “/home/adi/teks”  
INTO MHS_2  
FIELDS TERMINATED BY ‘\t’;
```

# Auxiliary lanjutan

## 3. RENAME TABLE

Sintaks :

```
RENAME TABLE OldnamaTabel  
TO NewNamaTabel
```

Contoh :

```
RENAME TABLE MHS  
TO MAHASISWA
```

# Fungsi Aggregate

## MENGGUNAKAN FUNGSI AGGREGATE :

1. COUNT digunakan untuk menghitung jumlah.  
Menghitung jumlah record mahasiswa dari tabel MAHASISWA  
`SELECT COUNT(*) FROM MAHASISWA`
2. SUM digunakan untuk menghitung total dari kolom yang mempunyai tipe data numerik.  
`SELECT SUM(SKS) AS 'TOTAL SKS' FROM MATAKULIAH`

# Fungsi Aggregate lanjutan

3. AVG digunakan untuk menghitung rata-rata dari data-data dalam sebuah kolom.

```
SELECT AVG(FINAL) AS 'FINAL' FROM Nilai
```

4. MIN digunakan untuk menghitung nilai minimal dalam sebuah kolom.

```
SELECT MIN(FINAL) FROM Nilai
```

5. MAX digunakan untuk menghitung nilai maksimum dalam sebuah kolom

```
SELECT MAX(MID) FROM Nilai
```

# SUBQUERY

## SUBQUERY

Adalah subselect yang dapat digunakan di klausa WHERE dan HAVING dipernyataan select luar untuk menghasilkan tabel akhir.

Aturan-aturan untuk membuat subquery, yaitu :

1. Klausa Order By tidak boleh digunakan di subquery, Order By hanya dapat digunakan di pernyataan Select luar.
2. Klausa subquery Select harus berisi satu nama kolom tunggal atau ekspresi kecuali untuk subquery-subquery menggunakan kata kunci EXIST
3. Secara default nama kolom di subquery mengacu ke nama tabel di klausa FROM dari subquery tersebut.
4. Saat subquery adalah salah satu dua operan dilibatkan di perbandingan, subquery harus muncul disisi kanan perbandingan



# Subquery lanjutan

## Penggunaanan ANY dan ALL

Jika subquery diawali kata kunci ALL, syarat hanya akan bernilai TRUE jika dipenuhi semua nilai yang dihasilkan subquery itu.

Jika subquery diawali kata kunci ANY, syaratnya akan bernilai TRUE jika dipenuhi sedikitnya satu nilai yang dihasilkan subquery tersebut.

# Subquery lanjutan

## Penggunaan **EXIST** DAN **NOT EXIST**

**EXIST** akan mengirim nilai **TRUE** jika dan hanya jika terdapat sedikitnya satu baris di tabel hasil yang dikirim oleh subquery dan **EXIST** mengirim nilai **FALSE** jika subquery mengirim tabel kosong.

Untuk **NOT EXIST** kebalikan dari **EXIST**.

(Masing-masing dosen membuat contoh untuk subquery)

# Subquery lanjutan

## CONTOH SUBQUERY :

1. Ambil nilai mid dan final dari mahasiswa yang bernama Astuti.

```
SELECT MID, FINAL FROM NILAI WHERE NIM=(  
SELECT NIM FROM MAHASISWA WHERE  
NAMA='Astuti')
```

2. Ambil nilai kode matakuliah, mid dan final dari mahasiswa yang tinggal di jakarta.

```
SELECT KD_MK, MID, FINAL FROM NILAI WHERE NIM  
IN(SELECT NIM FROM MAHASISWA WHERE ALAMAT =  
'Jakarta')
```

# Subquery lanjutan

3. Ambil nama-nama mahasiswa yang mengikuti ujian.

```
SELECT NAMA FROM MAHASISWA WHERE EXISTS  
(SELECT NIM FROM NILAI WHERE NILAI.NIM=  
MAHASISWA.NIM)
```

4. Ambil nama-nama mahasiswa yang tidak mengikuti ujian.

```
SELECT NAMA FROM MAHASISWA WHERE NOT  
EXISTS (SELECT NIM FROM NILAI WHERE NILAI.NIM=  
MAHASISWA.NIM)
```

## Pertemuan 12

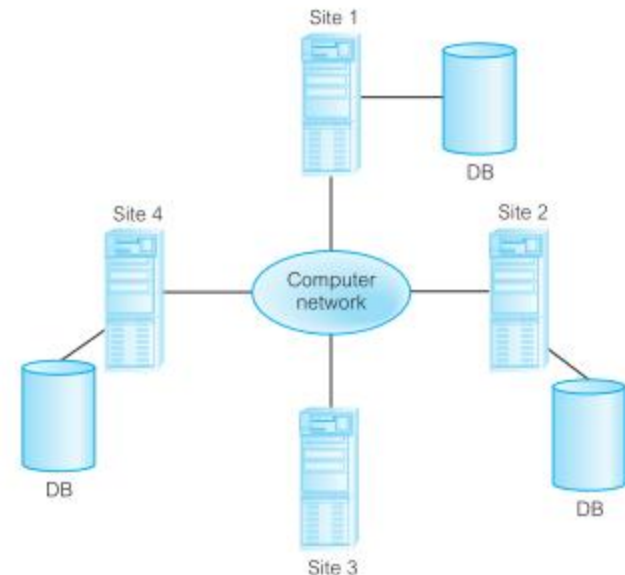
# Basis Data Terdistribusi

# Basis Data Terdistribusi

## Basis Data Terdistribusi

Yaitu kumpulan data yang digunakan bersama yang saling terhubung secara logik tetapi tersebar secara fisik pada suatu jaringan komputer.

Dalam sebuah database terdistribusi, database disimpan pada beberapa komputer. Komputer-komputer tersebut berhubungan satu sama lain melalui jaringan komunikasi.



Gambar : Connolly, T., & Begg, C. (2015). **Database Systems: A Practical Approach To Design, Implementation, and Management**. 6 th Edition. Pearson Education. England. ISBN: 978-1-292-06118-4 /

# Karakteristik Database Terdistribusi

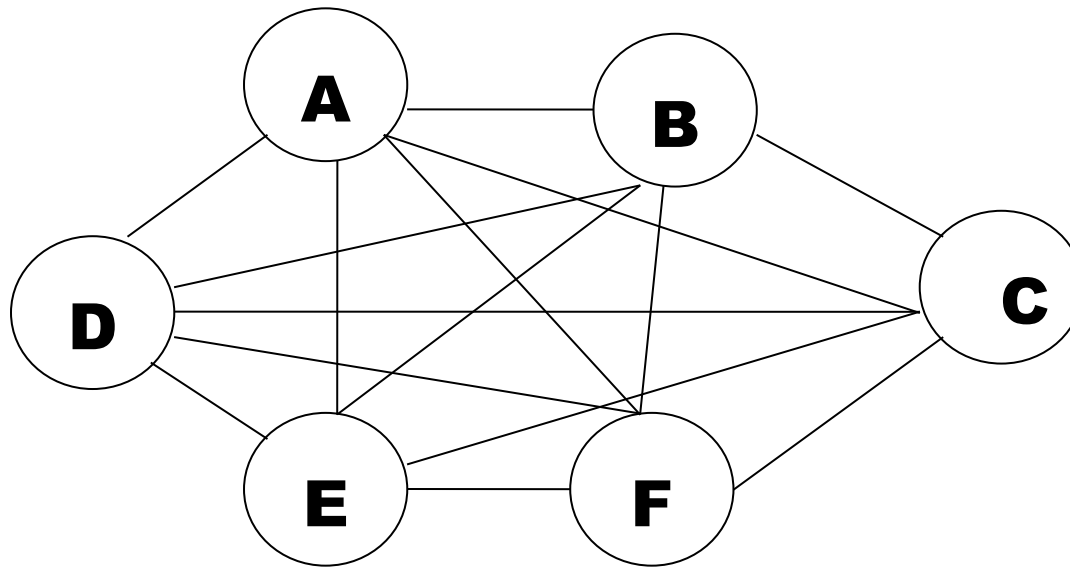
Karakteristik Database terdistribusi, yaitu :

1. Kumpulan data yang digunakan bersama secara logik tersebar pada sejumlah komputer yang berbeda
2. Komputer yang dihubungkan menggunakan jaringan komunikasi
3. Data pada masing-masing situs dapat menangani aplikasi-aplikasi lokal secara otonom
4. Data pada masing situs dibawah kendali satu DBMS
5. Masing-masing DBMS berpartisipasi dalam sedikitnya satu aplikasi global

# Topologi Distribusi Data

## BENTUK-BENTUK TOPOLOGI DISTRIBUSI DATA :

### a. Fully Connected network



Keuntungan : kalau salah satu node rusak, yang lainnya masih dapat berjalan (tetapi biaya mahal).

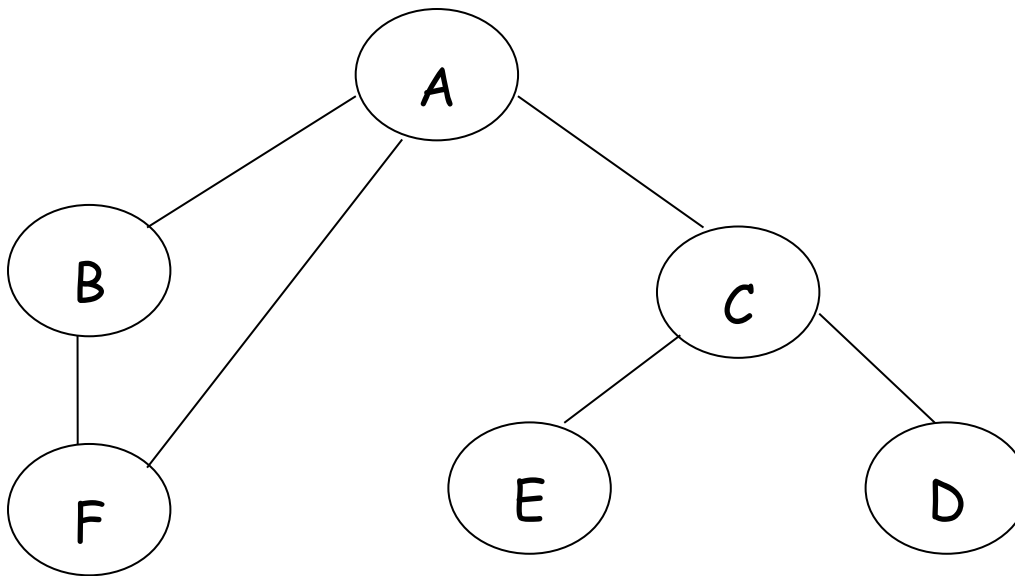
Kerugian : control management tidak terjamin

[https://repository.dinus.ac.id/docs/ajar/c-12\\_Reff\\_DB\\_Terdistribusi.pdf](https://repository.dinus.ac.id/docs/ajar/c-12_Reff_DB_Terdistribusi.pdf)



# Topologi lanjutan

## b. Partialy conneted network



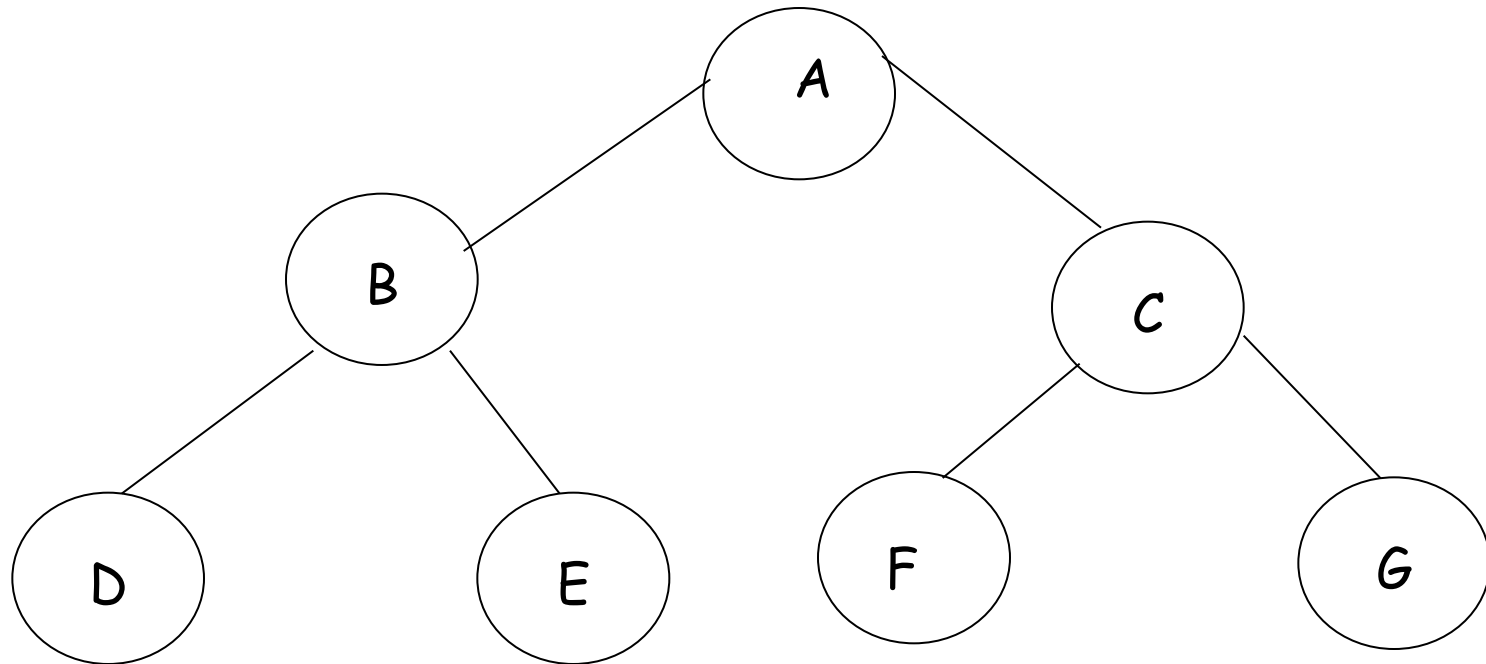
Keuntungan : reliability rendah, biaya dapat ditekan

Kerugian : control management tidak terjamin

[https://repository.dinus.ac.id/docs/ajar/c-12\\_Reff\\_DB\\_Terdistribusi.pdf](https://repository.dinus.ac.id/docs/ajar/c-12_Reff_DB_Terdistribusi.pdf)

# Topologi lanjutan

## C. Tree Strutured Network



Keuntungan : bersifat sentral, control management lebih terjamin

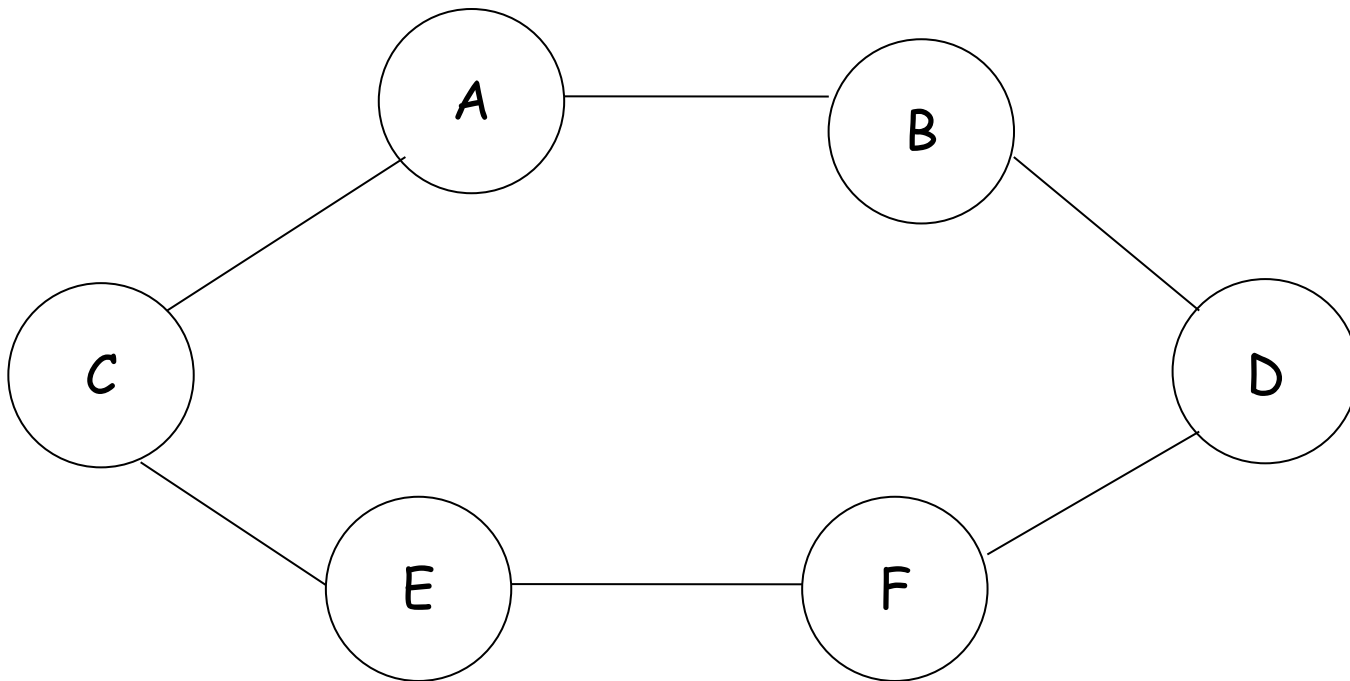
Kerugian : kalau node pusat (A) rusak, semua akan rusak.

Cat : setiap proses dimulai dari bawah

[https://repository.dinus.ac.id/docs/ajar/c-12\\_Reff\\_DB\\_Terdistribusi.pdf](https://repository.dinus.ac.id/docs/ajar/c-12_Reff_DB_Terdistribusi.pdf)

# Topologi lanjutan

## d. Ring network



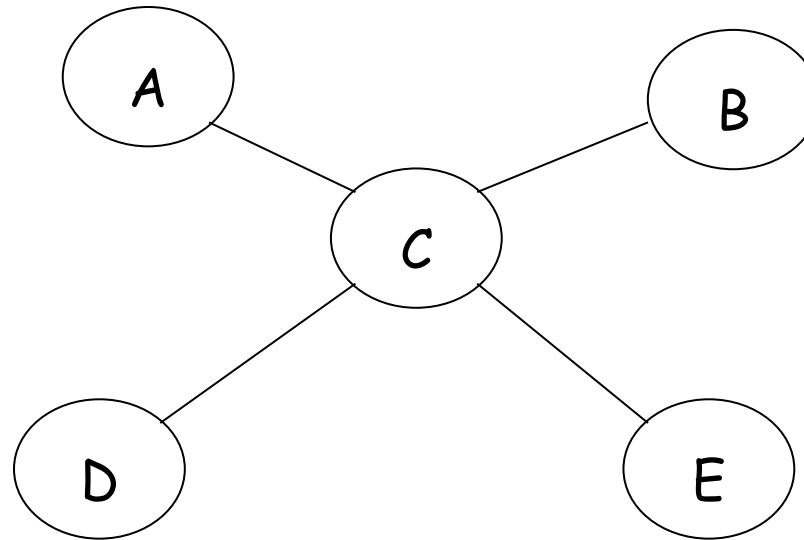
Keuntungan : rusak satu, yang lain masih berjalan

Kerugian : Control management kurang terjamin karena bersifat desentralisasi

[https://repository.dinus.ac.id/docs/ajar/c-12\\_Reff\\_DB\\_Terdistribusi.pdf](https://repository.dinus.ac.id/docs/ajar/c-12_Reff_DB_Terdistribusi.pdf)

# Topologi lanjutan

## e. Star network



Keuntungan : - control management lebih terjamin, karena bersifat sentral -  
reliability rendah

Kerugian : kalau pusat rusak, yang lainnya rusak

[https://repository.dinus.ac.id/docs/ajar/c-12\\_Reff\\_DB\\_Terdistribusi.pdf](https://repository.dinus.ac.id/docs/ajar/c-12_Reff_DB_Terdistribusi.pdf)

# Keuntungan Basis Data Terdistribusi

## **KEUNTUNGAN :**

1. Secara alami mengikuti struktur organisasi
2. Adanya otonomi lokal
3. Sifatnya dapat dipakai secara bersama
4. Peningkatan ketersediaan
5. Peningkatan kehandalan
6. Peningkatan kinerja
7. Ekonomis
8. Pertumbuhan yang modular

# Kerugian Basis Data Terdistribusi

## **KERUGIAN :**

1. Harga software mahal (Biaya)
2. Kompleksitas
3. Kelemahan dalam keamanan
4. Sulitnya menjaga keutuhan data
5. Kurangnya standar
6. Kurangnya pengalaman
7. Perancangan basisdata lebih kompleks

# Fragmentasi Data

**FRAGMENTASI** Merupakan sebuah proses pembagian atau pemetaan database dimana database dipecah-pecah berdasarkan kolom dan baris yang kemudian disimpan didalam site atau unit komputer yang berbeda dalam suatu jaringan data, sehingga memungkinkan untuk pengambilan keputusan terhadap data yang telah terbagi.

**Fragmentasi data** merupakan langkah yang diambil untuk menyebarkan data dalam basis data terdistribusi.

# Alasan Diperlukannya Fragmentasi

## Alasan-alasan diperlukannya fragmentasi, yaitu :

1. **Penggunaan** → Secara umum, aplikasi bekerja dengan tampilan daripada seluruh relasi. Oleh karena itu, untuk distribusi data, tampaknya tepat untuk bekerja dengan himpunan bagian dari relasi sebagai unit distribusi.
2. **Efisiensi** → Data disimpan dekat dengan tempat yang paling sering digunakan. Tambahan, data yang tidak diperlukan oleh aplikasi lokal tidak disimpan.
3. **Parallesisme** → Dengan fragmen sebagai unit distribusi, suatu transaksi dapat dibagi menjadi beberapa subquery yang beroperasi pada fragmen. Ini bisa meningkatkan tingkat konkurensi, atau paralelisme, dalam sistem, sehingga memungkinkan transaksi dapat dilakukan dengan aman untuk dieksekusi secara paralel.
4. **Keamanan** → Data yang tidak diperlukan oleh aplikasi lokal tidak disimpan dan akibatnya data tidak tersedia untuk pengguna yang tidak sah.



# Fragmentasi lanjutan

## **Kerugian fragmentasi yaitu :**

### **1. Performance**

Kinerja yang dapat turun karena data tersebar dan butuh proses untuk penggabungan kembali

### **2. Integrity**

Integritas yang dapat terganggu dikarenakan kegagalan pada salah satu site database server

# Fragmentasi lanjutan

## **BEBERAPA PERATURAN YANG HARUS DIDEFINISIKAN KETIKA MENDEFINISIKAN FRAGMENT :**

1. Kondisi lengkap (*Completeness*)  
sebuah unit data yang masih dalam bagian dari relasi utama, maka data harus berada dalam satu fragmen. Ketika ada relasi, pembagian datanya harus menjadi satu kesatuan dengan relasinya.

Contoh :

Jika relasi  $R$  didekomposisi menjadi fragmen  $R_1, R_2, \dots$ , setiap item data yang dapat ditemukan di  $R$  harus muncul setidaknya dalam satu fragmen.

Aturan ini diperlukan untuk memastikan bahwa tidak ada kehilangan data selama fragmentasi.

# Fragmentasi lanjutan

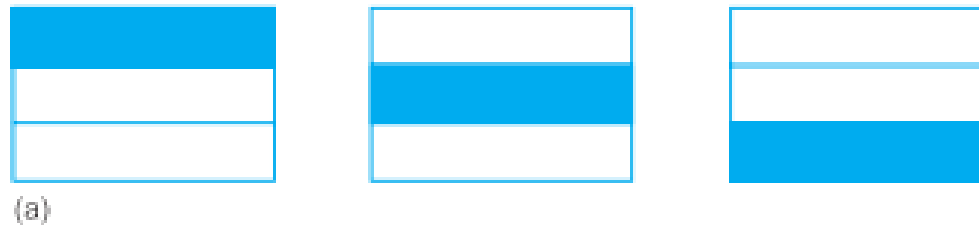
2. Rekontruksi (*Reconstruction*)  
sebuah relasi asli dapat dibuat kembali atau digabungkan kembali dari sebuah fragmen. Ketika telah dipecah-pecah, data masih memungkinkan untuk digabungkan kembali dengan tidak mengubah struktur data.  
Aturan ini memastikan bahwa ketergantungan fungsional dipertahankan.
3. Disjointness  
data didalam fragmen tidak boleh diikuti dalam fragmen lain agar tidak terjadi redundancy data, kecuali untuk atribut primary key dalam fragmentasi vertikal  
Aturan ini memastikan redundansi data minimal.

# Jenis Fragmentasi

## TIGA JENIS FRAGMENTASI :

### 1. Fragmentasi horizontal

terdiri dari tuple dari fragment global yang kemudian dipecah-pecah atau disekat menjadi beberapa sub-sets



$$\sigma_p(R)$$

R = Relation

P = predikat yang didasarkan pada satu atau lebih atribut relasi

Gambar : Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach To Design, Implementation, and Management*. 6 th Edition. Pearson Education. England. ISBN: 978-1-292-06118-4 /

# Fragmentasi Horizontal

$P_1: \sigma_{type = 'House'}(PropertyForRent)$

$P_2: \sigma_{type = 'Flat'}(PropertyForRent)$

Fragment  $P_1$

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003

Fragment  $P_2$

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	SG14	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Completeness → Setiap tuple dalam relasi muncul di salah satu fragmen  $P_1$  atau  $P_2$

Reconstruction → Relasi PropertyForRent dapat direkonstruksi dari fragmen dengan menggunakan operasi Union:  $P_1 \cup P_2 = PropertyForRent$

Disjointness → tidak ada type property yang keduanya “house” dan “flat”

Untuk Melihat Tabel Lengkap di : Connolly, T., & Begg, C. (2015). **Database Systems: A Practical Approach To Design, Implementation, and Management**. 6 th Edition. Pearson Education. England. ISBN: 978-1-292-06118-4  
Halaman 162

# Jenis Fragmentasi

## 2. Fragmentasi vertikal

Membagi atribut-atribut dari fragment global yang tersedia menjadi beberapa grup.

R = Relation

$a_1, \dots, a_n$  = atribut dari R

$$\Pi_{a_1, \dots, a_n}(R)$$



# Fragmentasi Vertikal

Aplikasi penggajian DreamHome membutuhkan nomor staf no staff dan posisi, jenis kelamin, DOB, dan atribut gaji setiap anggota staf; departemen SDM membutuhkan atribut staffNo, fName, lName, dan branchNo.

$S_1: \Pi_{\text{staffNo, position, sex, DOB, salary}}(\text{Staff})$

$S_2: \Pi_{\text{staffNo, fName, lName, branchNo}}(\text{Staff})$

Completeness → Setiap atribut dalam relasi Staf muncul di salah satu fragmen  $S_1$  atau  $S_2$

Reconstruction → Relasi Staf dapat direkonstruksi dari fragmen menggunakan Operasi Natural Join :  $S_1 \bowtie S_2 = \text{Staff}$

Disjointness → seluruh fragment disjoint kecuali primary key, yang dibutuhkan untuk rekonstruksi

Untuk Melihat Tabel Lengkap di : Connolly, T., & Begg, C. (2015).

**Database Systems: A Practical Approach To Design, Implementation, and Management.** 6 th Edition. Pearson

Education. England. ISBN: 978-1-292-06118-4

Halaman 162

Fragment  $S_1$

staffNo	position	sex	DOB	salary
SL21	Manager	M	1-Oct-45	30000
SG37	Assistant	F	10-Nov-60	12000
SG14	Supervisor	M	24-Mar-58	18000
SA9	Assistant	F	19-Feb-70	9000
SG5	Manager	F	3-Jun-40	24000
SL41	Assistant	F	13-Jun-65	9000

Fragment  $S_2$

staffNo	fName	lName	branchNo
SL21	John	White	B005
SG37	Ann	Beech	B003
SG14	David	Ford	B003
SA9	Mary	Howe	B007
SG5	Susan	Brand	B003
SL41	Julie	Lee	B005

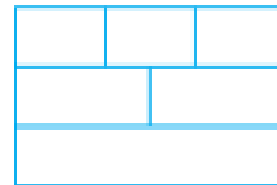
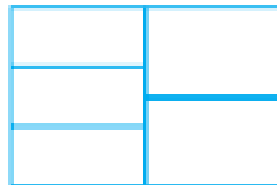


# Jenis Fragmentasi

## 3. Fragmentasi campuran

Cara yang sederhana untuk membangun fragmentasi campuran sbb :

- Menggunakan fragmentasi horizontal pada fragmentasi vertikal
- Menggunakan fragmentasi vertikal pada fragmentasi horizontal



Fragmen campuran didefinisikan menggunakan operasi Selection dan Projection dari aljabar relasional.

$$\sigma_p(\Pi_{a_1, \dots, a_n}(R))$$

or

$$\Pi_{a_1, \dots, a_n}(\sigma_p(R))$$

di mana  $p$  adalah predikat berdasarkan satu atau lebih atribut dari  $R$ , dan  $a_1, \dots, a_n$  adalah atribut dari  $R$



# Fragmentasi Campuran

Dibagi secara vertikal Staf untuk departemen penggajian dan SDM menjadi:

$S_1: \Pi_{\text{staffNo, position, sex, DOB, salary}}(\text{Staff})$

$S_2: \Pi_{\text{staffNo, fName, lName, branchNo}}(\text{Staff})$

Kita sekarang dapat secara horizontal memecah  $S_2$  sesuai dengan nomor cabang (untuk memudahkannya, maka diasumsikan bahwa hanya ada tiga cabang):

$S_{21}: \sigma_{\text{branchNo} = 'B003'}(S_2)$

$S_{23}: \sigma_{\text{branchNo} = 'B005'}(S_2)$

$S_{23}: \sigma_{\text{branchNo} = 'B007'}(S_2)$

**Completeness** → Setiap atribut dalam relasi Staf muncul di salah satu fragmen  $S_1$  atau  $S_2$  ; setiap (bagian) tupel muncul di fragmen  $S_1$  dan salah satu fragmen  $S_{21}$  ,  $S_{22}$  , atau  $S_{23}$

**Reconstruction** → Relasi Staf dapat direkonstruksi dari fragmen menggunakan Operasi Natural Join dan Union :  $S_1 \bowtie (S_{21} \cup S_{22} \cup S_{23}) = \text{Staff}$

**Disjointness** → seluruh fragment disjoint, tidak boleh ada anggota staf yang bekerja di lebih dari satu cabang, dan  $S_1$  dan  $S_2$  disjoint kecuali untuk duplikasi yang diperlukan oleh kunci utama.

Fragment  $S_1$

staffNo	position	sex	DOB	salary
SL21	Manager	M	1-Oct-45	30000
SG37	Assistant	F	10-Nov-60	12000
SG14	Supervisor	M	24-Mar-58	18000
SA9	Assistant	F	19-Feb-70	9000
SG5	Manager	F	3-Jun-40	24000
SL41	Assistant	F	13-Jun-65	9000

Fragment  $S_{21}$

staffNo	fName	lName	branchNo
SG37	Ann	Beech	B003
SG14	David	Ford	B003
SG5	Susan	Brand	B003

Fragment  $S_{22}$

staffNo	fName	lName	branchNo
SL21	John	White	B005
SL41	Julie	Lee	B005

Fragment  $S_{23}$

staffNo	fName	lName	branchNo
SA9	Mary	Howe	B007

# Contoh Fragmentasi

## CONTOH KASUS JENIS-JENIS FRAGMENTASI

Ujian (NIM,Nama\_Mhs,Kode\_MK,Mt\_Kuliah,Nil\_Akhir,Grade)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
123	Fathi	101	Sistem Basis Data	78	B
124	Farah	102	Peranc. Sistem	60	C
125	Sarah	101	Sistem Basis Data	40	D
126	Salsabila	101	Sistem Basis Data	90	A
127	Azizah	103	Visual Basic	70	B
128	Farhan	103	Visual Basic	40	D
129	Faiz	102	Peranc. Sistem	80	A

# Contoh Fragmentasi Horizontal

Fragmentasi **Horizontal** terbagi menjadi 3 fragment yang berbeda berdasarkan Mt\_Kuliah

1. Relasi Mt\_Kuliah="Sistem Basis Data"  
 $\sigma$  Mt\_Kuliah="Sistem Basis Data" (Ujian)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
123	Fathi	101	Sistem Basis Data	78	B
125	Sarah	101	Sistem Basis Data	40	D
126	Salsabila	101	Sistem Basis Data	90	A

# Fragmentasi Horizontal lanjutan

2. Relasi Mt\_Kuliah="Peranc. Sistem"

$\sigma$  Mt\_Kuliah="Peranc. Sistem" (Ujian)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
124	Farah	102	Peranc. Sistem	60	C
129	Faiz	102	Peranc. Sistem	80	A

3. Relasi Mt\_Kuliah="Visual Basic"

$\sigma$  Mt\_Kuliah="Visual Basic" (Ujian)

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Akhir	Grade
127	Azizah	103	Visual Basic	70	B
128	Farhan	103	Visual Basic	40	D

# Contoh Fragmentasi Vertical

Fragment di atas memenuhi kondisi jika Nama\_Mhs dan Mt\_Kuliah adalah hal-hal yang memenuhi syarat **Fragmentasi vertical**: berdasarkan dekomposisi-nya dengan menambahkan tuple\_id

NIM	Nama_Mhs	Kode_MK	Mt_Kuliah	Nil_Ak hir	Grade	Tuple_ID
123	Fathi	101	Sistem Basis Data	78	B	1
124	Farah	102	Peranc. Sistem	60	C	2
125	Sarah	101	Sistem Basis Data	40	D	3
126	Salsabila	101	Sistem Basis Data	90	A	4
127	Azizah	103	Visual Basic	70	B	5
128	Farhan	103	Visual Basic	40	D	6
129	Faiz	102	Peranc. Sistem	80	A	7

# Contoh Fragmentasi Vertikal

Relasi 1 = NIM, Nama\_Mhs, Mt,Kuliah, Nil\_Akhir, Grade, Tuple\_ID

$\pi$  NIM,Nama\_Mhs,Mt,Kuliah,Nil\_Akhir,Grade,Tuple\_ID (UJian)

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
123	Fathi	Sistem Basis Data	78	B	1
124	Farah	Peranc. Sistem	60	C	2
125	Sarah	Sistem Basis Data	40	D	3
126	Salsabila	Sistem Basis Data	90	A	4
127	Azizah	Visual Basic	70	B	5
128	Farhan	Visual Basic	40	D	6
129	Faiz	Peranc. Sistem	80	A	7

# Contoh Fragmentasi Vertical

Relasi 2 = NIM,Kode\_MK,Nil\_Akhir,Grade,Tuple\_ID

$\pi$  NIM,Kode\_MK,Nil\_Akhir,Grade,Tuple\_ID (Ujian)

NIM	Kode_MK	Nil_Akhir	Grade	Tuple_ID
123	101	78	B	1
124	102	60	C	2
125	101	40	D	3
126	101	90	A	4
127	103	70	B	5
128	103	40	D	6
129	102	80	A	7



# Contoh Fragmentasi Campuran

Terdapat relasi berdasarkan Mata Kuliah yang sama

Relasi 1a.

$\pi$  NIM, Nama\_Mhs, Mt\_Kuliah, Nil\_Akhir, Grade, Tuple\_ID ( $\sigma$  Mt\_Kuliah="Sistem Basis Data" (Ujian))

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
123	Fathi	Sistem Basis Data	78	B	1
125	Sarah	Sistem Basis Data	40	D	3
126	Salsabila	Sistem Basis Data	90	A	4

Relasi 1b.

$\pi$  NIM, Nama\_Mhs, Mt\_Kuliah, Nil\_Akhir, Grade, Tuple\_ID ( $\sigma$  Mt\_Kuliah="Peranc. Sistem" (Ujian))

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
124	Farah	Peranc. Sistem	60	C	2
129	Faiz	Peranc. Sistem	80	A	7



# Fragmentasi Campuran lanjutan

## Relasi 1c

$\pi$  NIM, Nama\_Mhs, Mt\_Kuliah, Nil\_Akhir, Grade, Tuple\_ID ( $\sigma$  Mt\_Kuliah="Visual Basic" (Ujian))

NIM	Nama_Mhs	Mt_Kuliah	Nil_Akhir	Grade	Tuple_ID
127	Azizah	Visual Basic	70	B	5
128	Farhan	Visual Basic	40	D	6

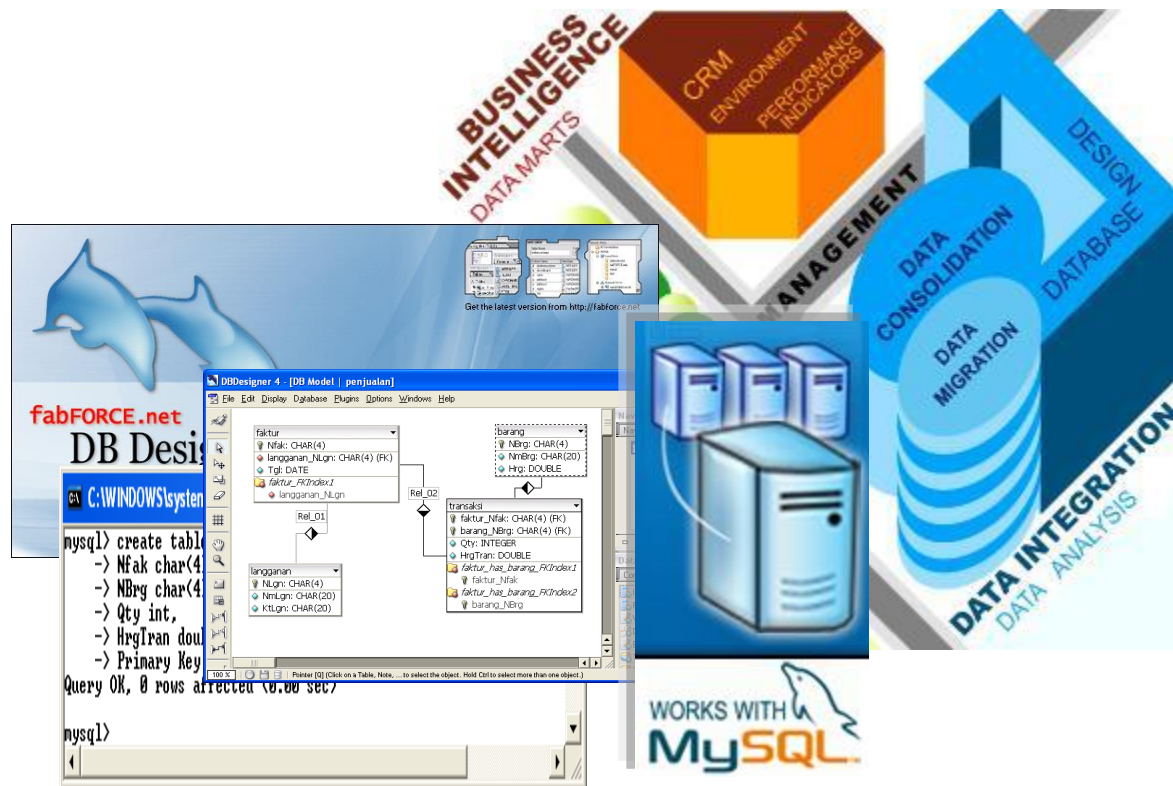
# Referensi Buku

1. Ceri, Stefano & Pelagatti G, Distributed Databases : Principles & Systems, McGraw-Hill, Singapore, 1984
2. Korth, H.F & Siferschatz, Database System Concepts, McGraw-Hill, USA, 1986
3. Özsu, M.T & Valduriez, Principles of Distributed Database Systems, PrenticeHall, New Jersey, 1991
4. Connolly, T., & Begg, C. (2015). ***Database Systems: A Practical Approach To Design, Implementation, and Management***. 6 th Edition. Pearson Education. England. ISBN: 978-1-292-06118-4

# Pertemuan 13

## **Perancangan Dan Implementasi Basis Data Menggunakan DB Designer**

# PERANCANGAN DAN IMPLEMENTASI BASIS DATA MENGGUNAKAN MYSQL

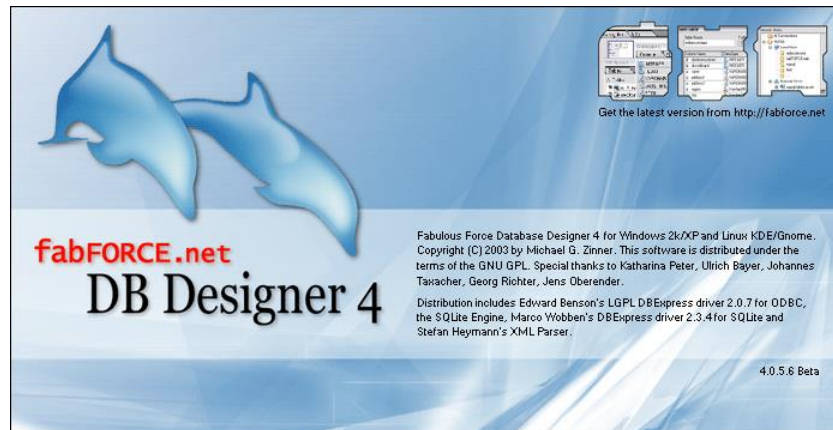


# Lanjutan

## Perangkat Lunak Bantu untuk Perancangan Basis Data

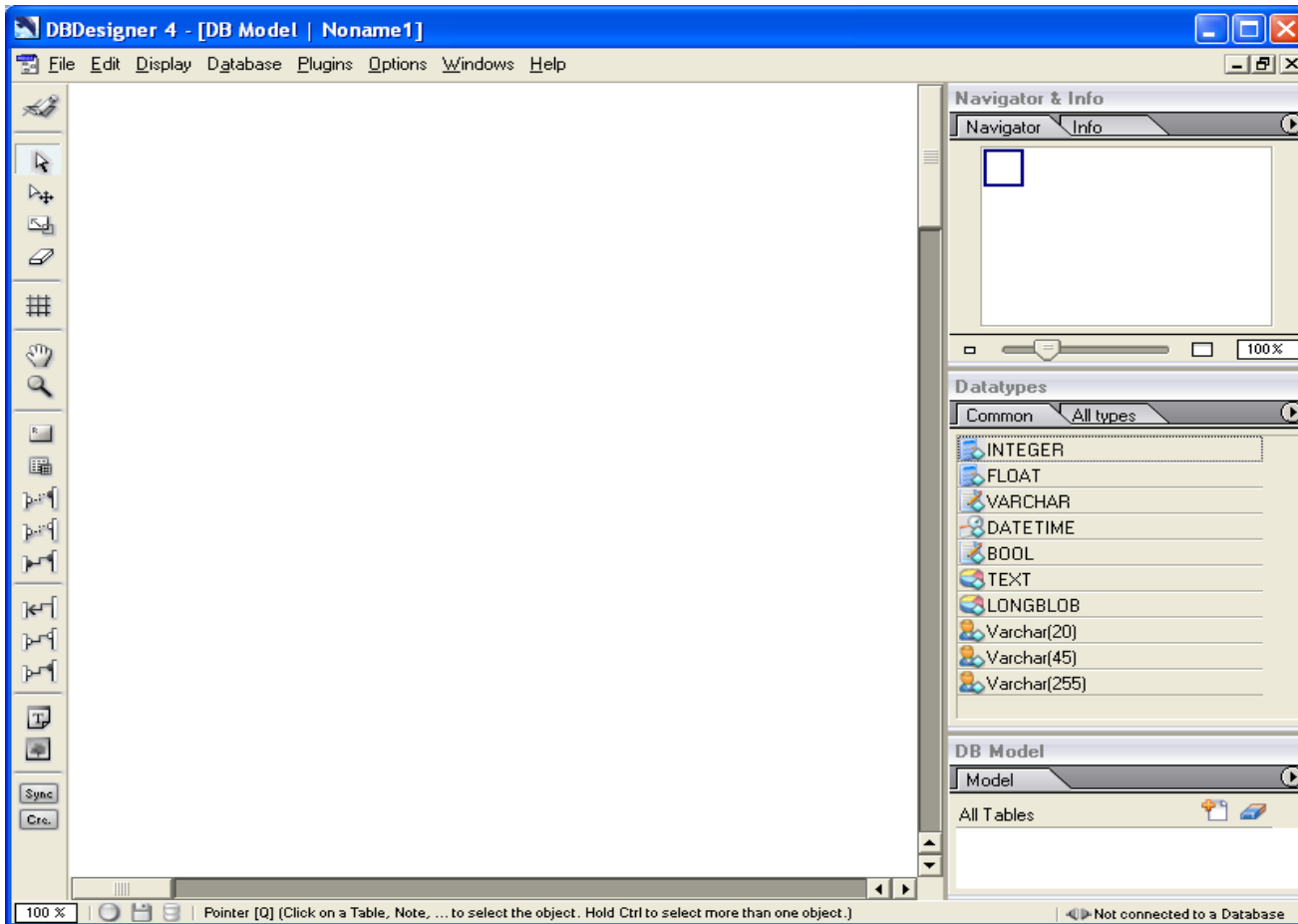
Pada perangkat lunak bantu telah tersedia komponen-komponen (notasi-notasi) perancangan basis data.

Salah satu perangkat lunak bantu untuk keperluan semacam itu adalah DBDesigner yang dioptimalkan untuk MySQL Database.



# Lanjutan

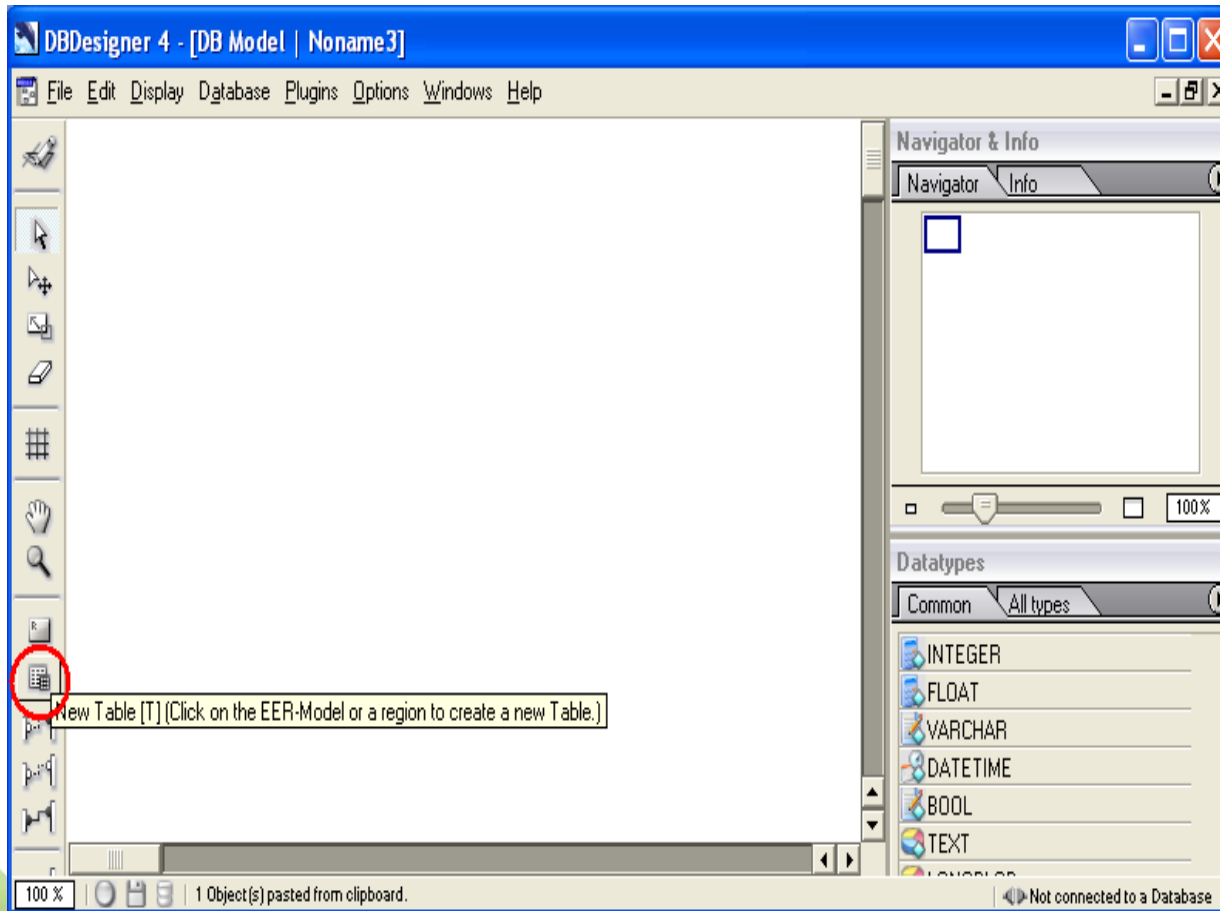
- Tampilan jendela DBDesigner.





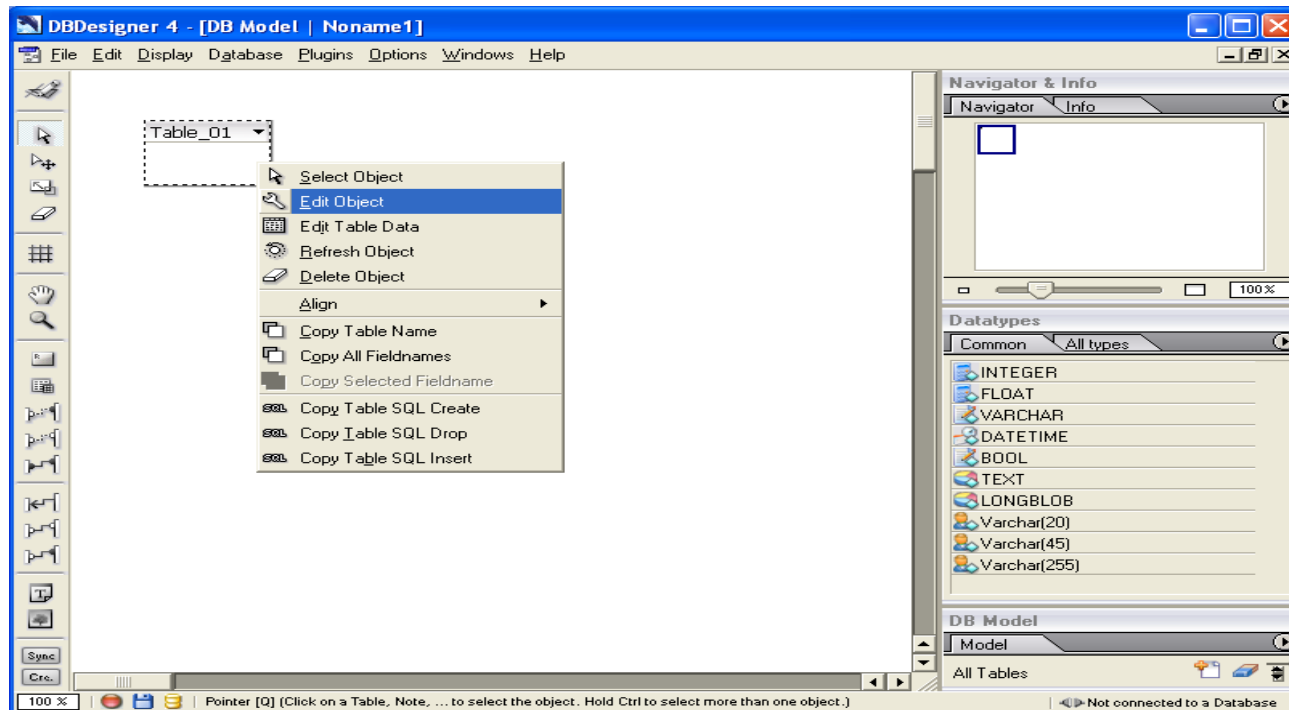
# Contoh penggunaan DBDesigner

**Menggunakan Komponen TABEL dan RELASI** Klik komponen **Tabel** pada toolbar seperti di gambar berikut.



Letakan komponen tsb. pada page area sehingga muncul komponen **Tabel** (Table\_01) pada page area, kemudian klik kanan komponen tsb sehingga muncul menu dan pilihlah **Edit Object** seperti berikut.

# Lanjutan



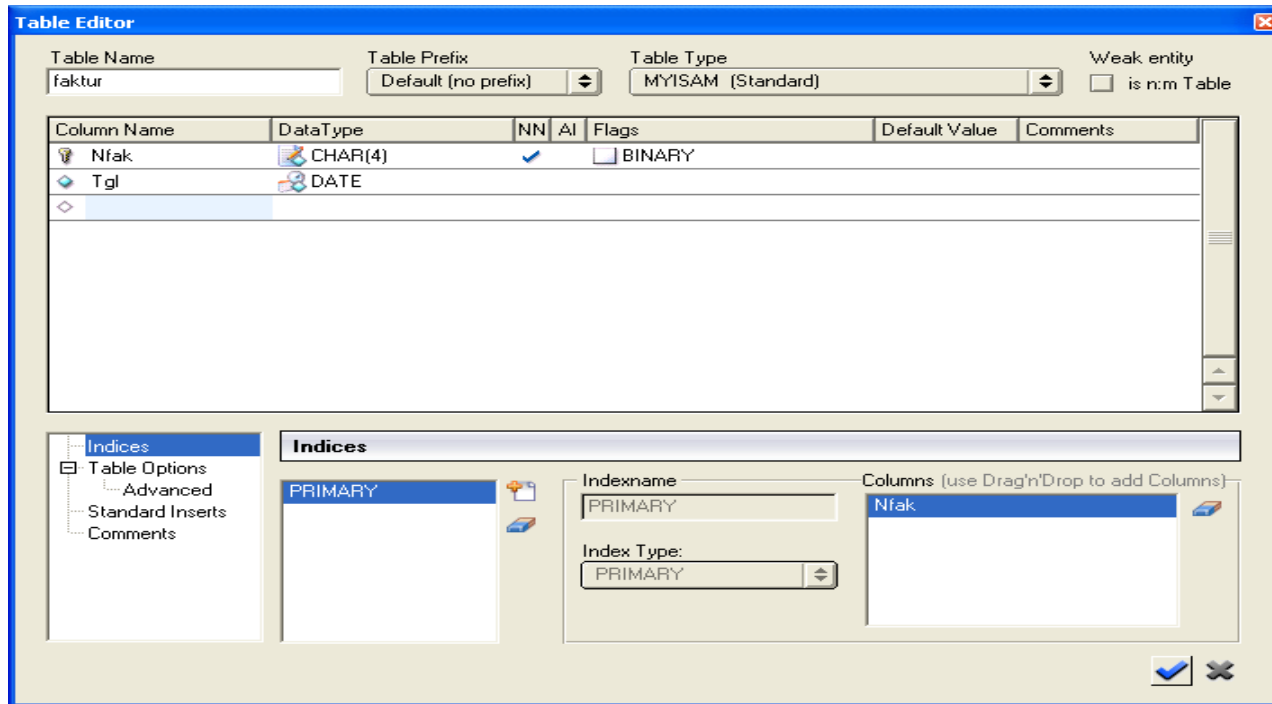
Menu Edit Object akan menampilkan jendela **Table Editor**.

Pada **Table Editor** kita bisa menentukan properties dari tabel seperti nama tabel, tipe data, primary key dsb.

Ubah dan simpanlah properties tabel (Table \_01) menjadi tabel **faktur** (struktur tabel seperti pada pembahasan LRS tanpa ada FK) seperti berikut.



# Lanjutan



**Table Editor**

Table Name: faktur    Table Prefix: Default (no prefix)    Table Type: MYISAM (Standard)    Weak entity: ☐ is n:m Table

Column Name	DataType	NN	AI	Flags	Default Value	Comments
Nfak	CHAR(4)	<input checked="" type="checkbox"/>		<input type="checkbox"/> BINARY		
Tgl	DATE					

**Indices**

Table Options: ☒ Table Options, ☐ Advanced, ☐ Standard Inserts, ☐ Comments

**Indices**

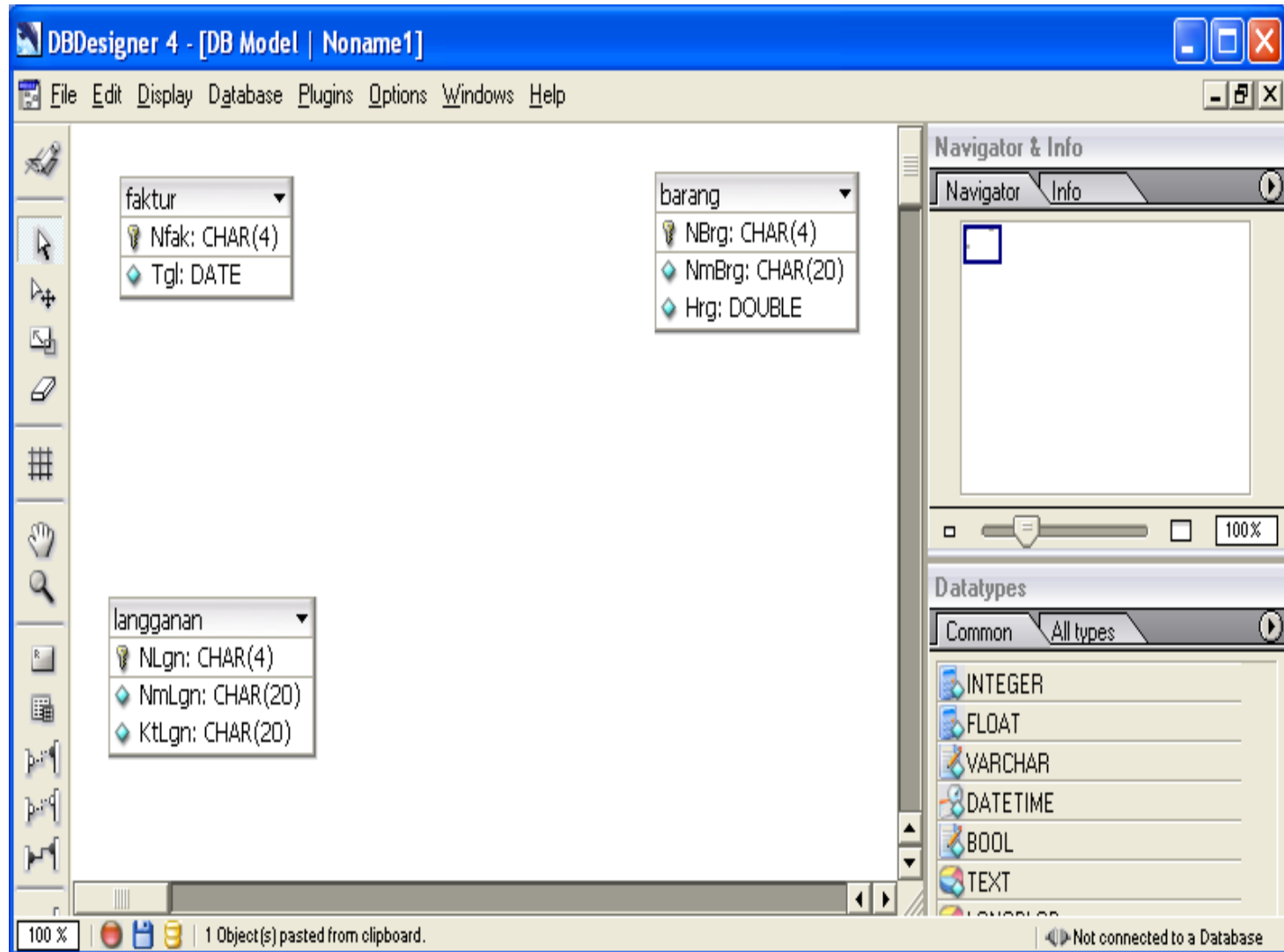
PRIMARY

Indexname: PRIMARY    Index Type: PRIMARY

Columns (use Drag'n'Drop to add Columns): Nfak

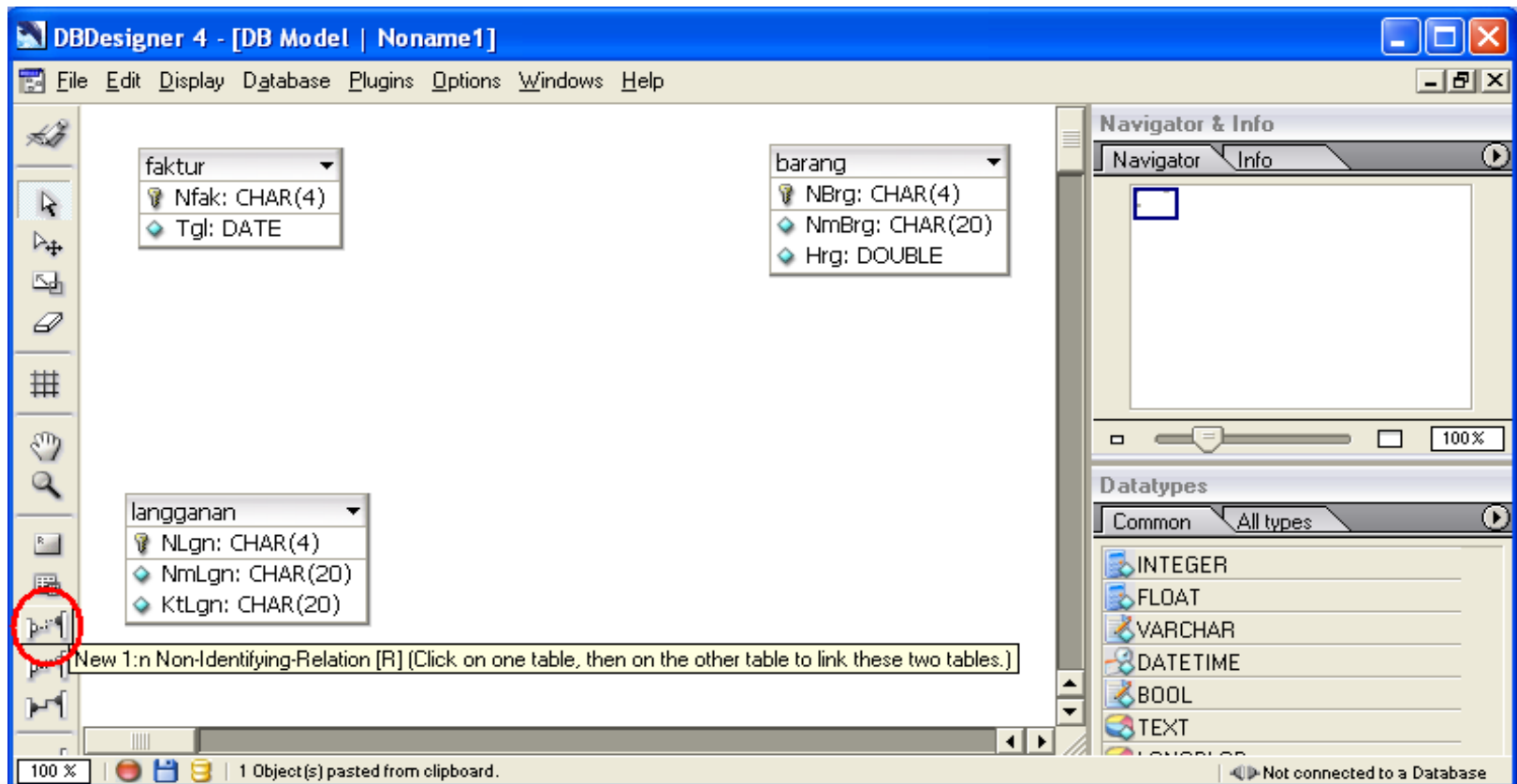
Ulangi langkah-langkah menggunakan komponen **Table** di atas (**tabel faktur**) untuk tabel **barang** dan **langganan** (struktur tabel seperti pada pembahasan LRS tanpa ada FK). Sehingga ada 3 komponen Table seperti gambar berikut

# Lanjutan



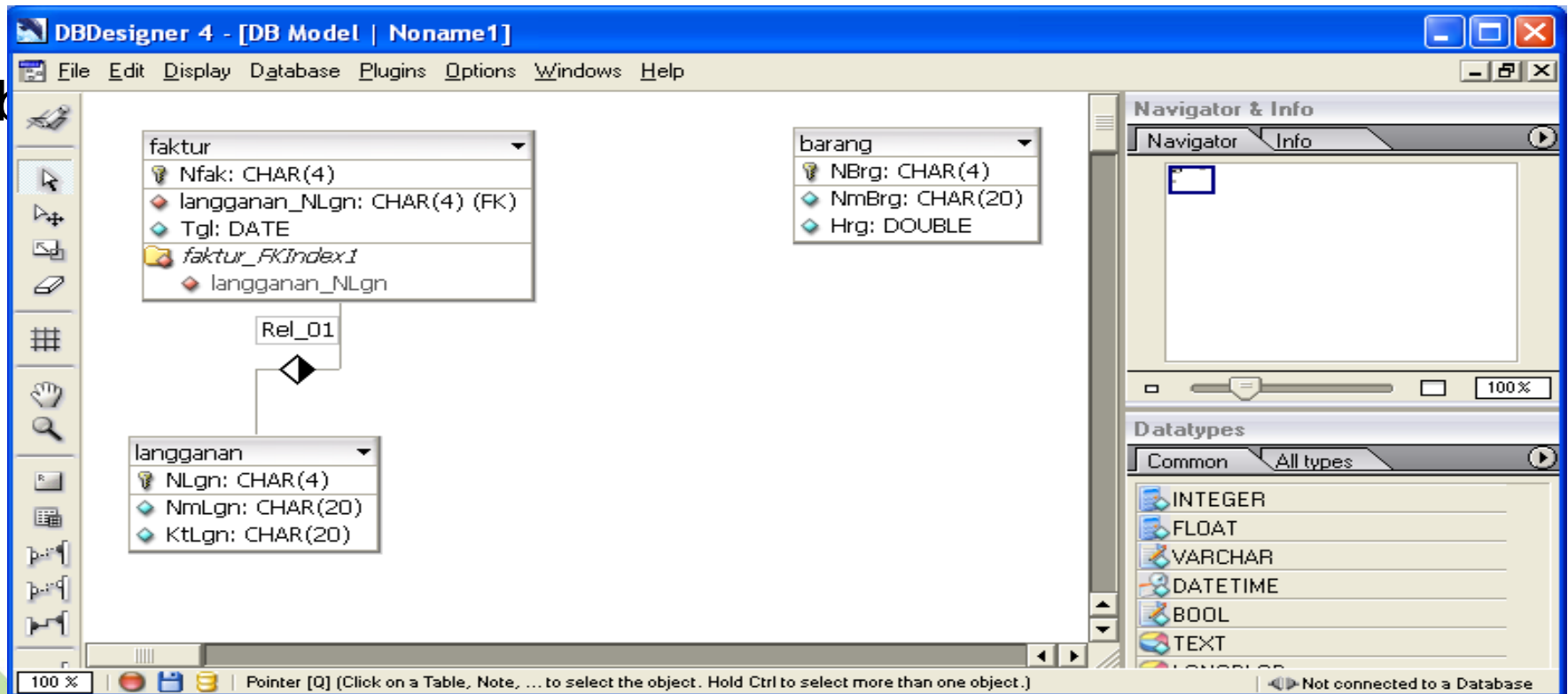
# Lanjutan

Langkah berikutnya membuat relasi **1-M** antara **langganan** dengan **faktur** dengan cara klik komponen **1-n Relation** pada toolbar seperti di gambar berikut.



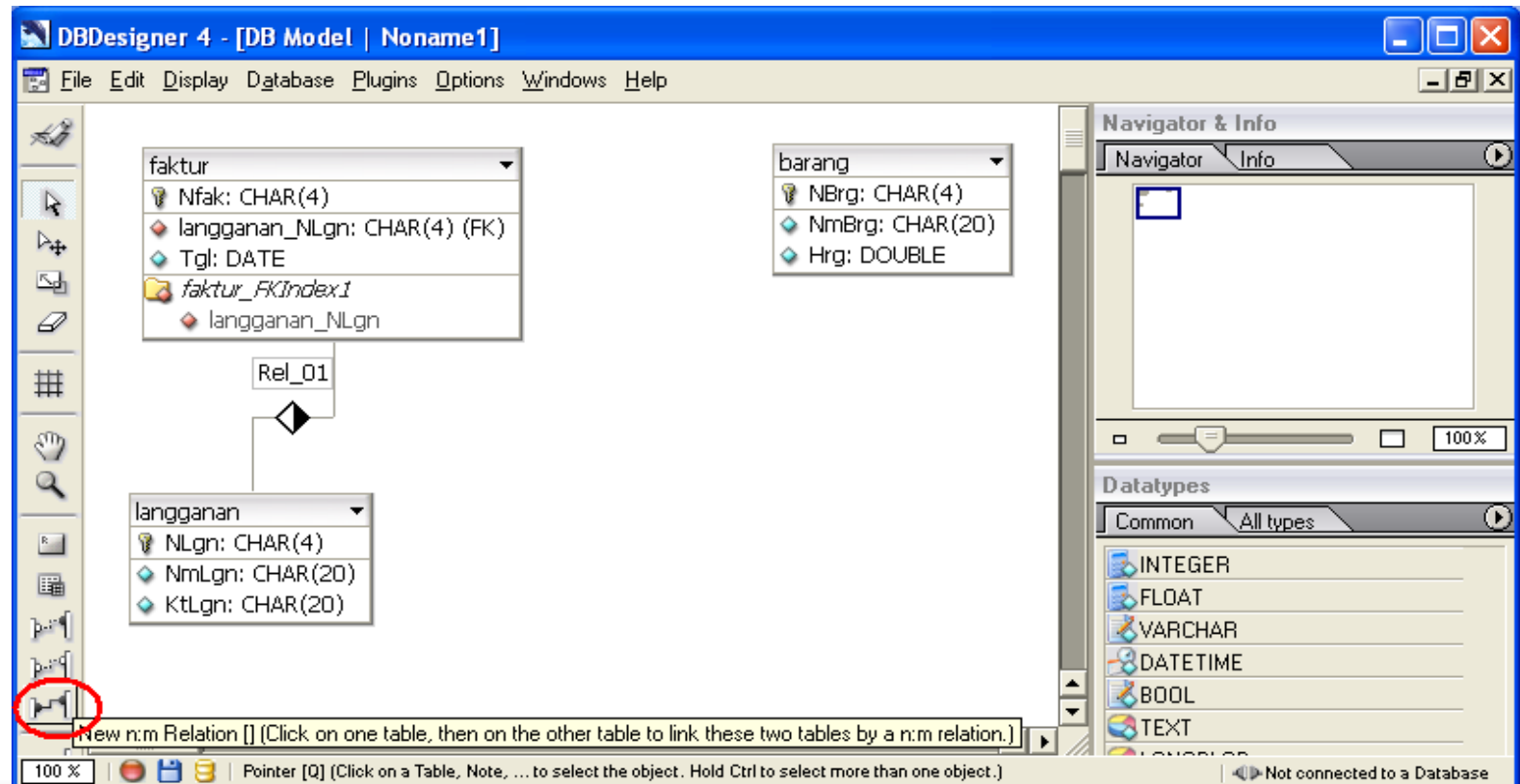
# Lanjutan

Klik di tabel **langganan** kemudian klik di tabel **faktur**, sehingga muncul komponen relasi yang menghubungkan kedua tabel tsb.



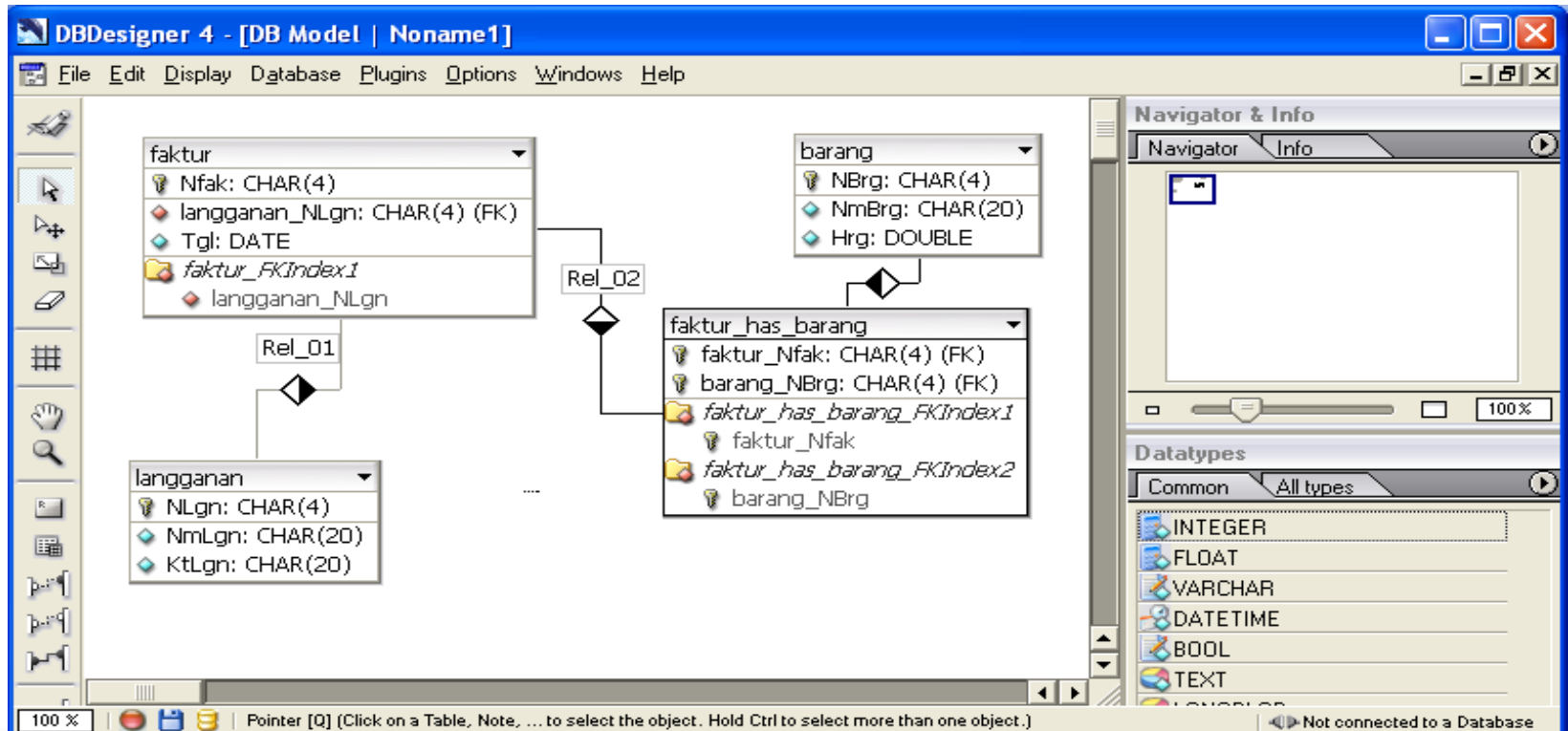
# Lanjutan

Langkah berikutnya membuat relasi **M-M** antara **faktur** dengan **barang** dengan cara klik komponen **n-m Relation** pada toolbar seperti di gambar berikut



# Lanjutan

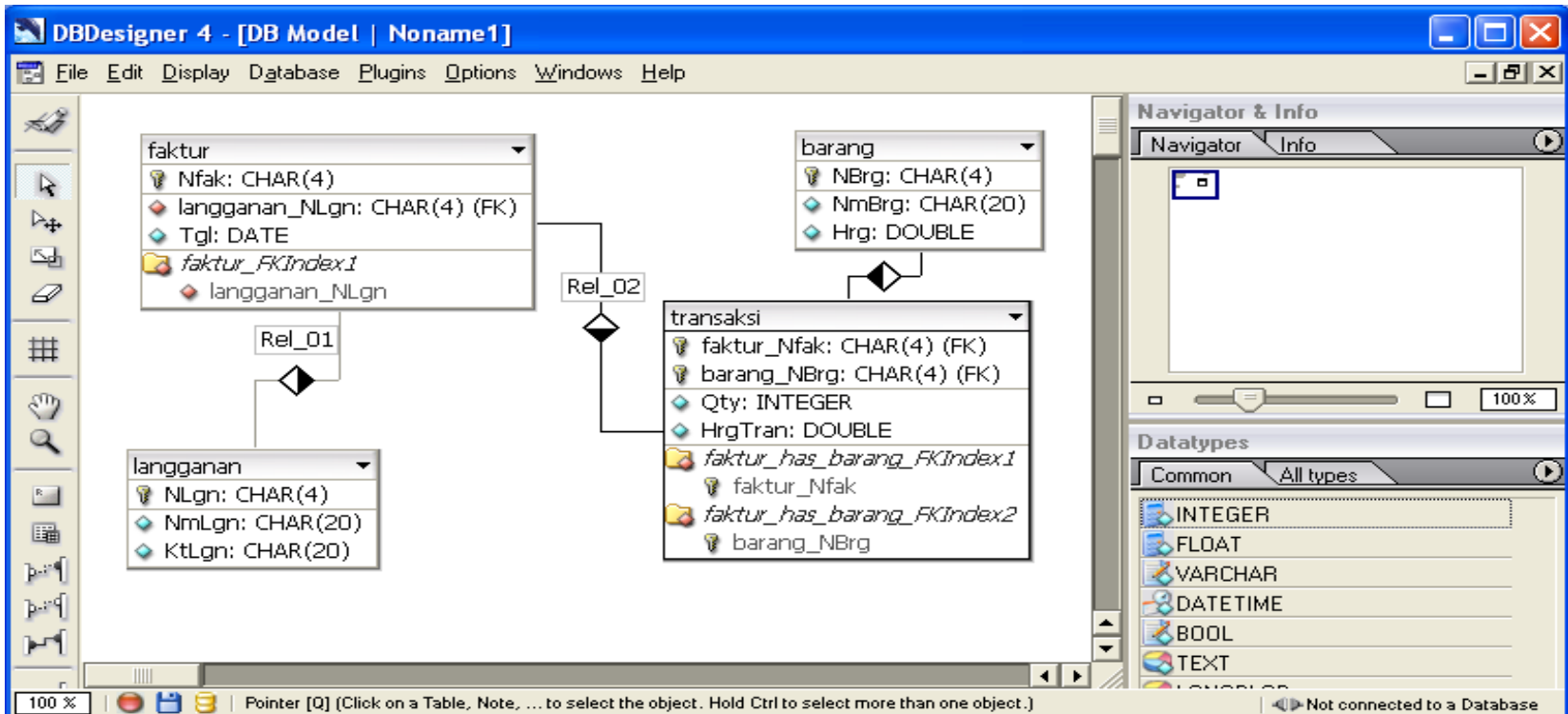
Klik di tabel **faktur** kemudian klik di tabel **barang**, sehingga muncul komponen relasi yang disertai munculnya tabel baru (**faktur\_has\_barang**) dan FK (Nfak & NBrng) berada pada tabel tsb, seperti gambar berikut.





# Lanjutan

Edit properties tabel `faktur_has_barang` yaitu dengan mengganti nama menjadi tabel transaksi dan menambahkan field Qty dan HrgTran. Sehingga menjadi seperti gambar berikut.



# Lanjutan

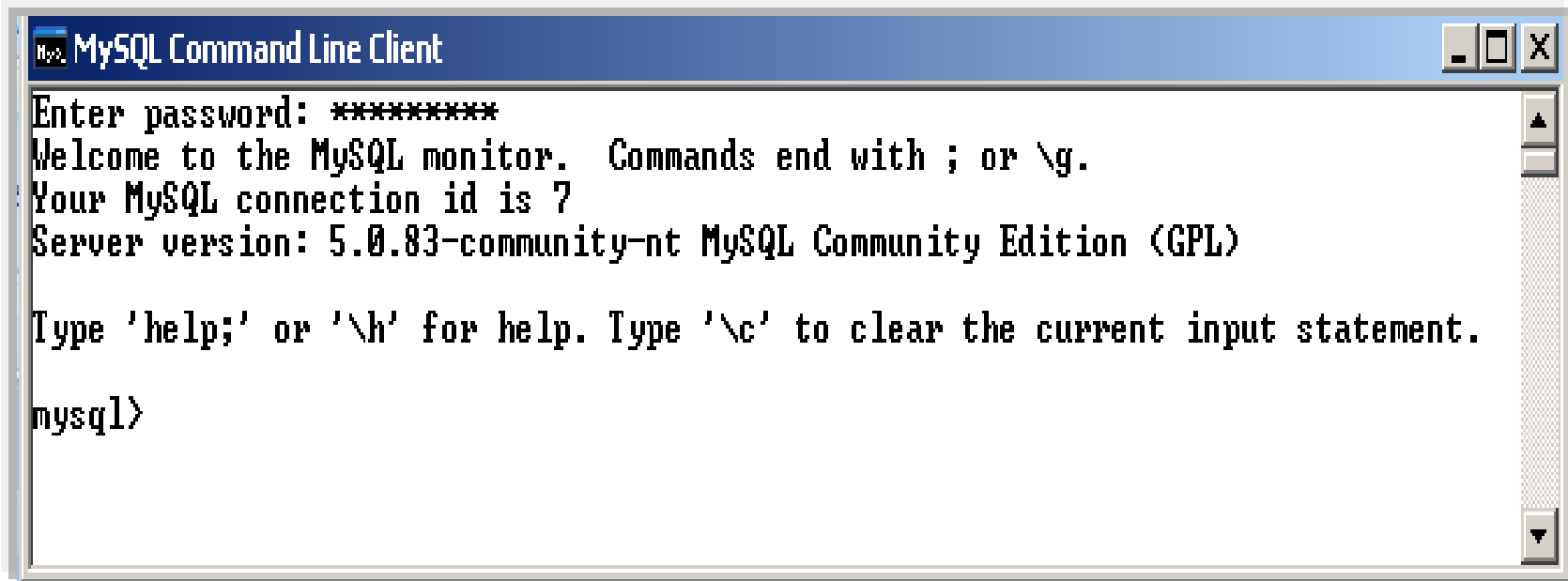
Untuk mengekspor hasil rancangan database ke dalam database digunakan **Database Synchronization**. Database yang digunakan pada contoh ini adalah MySQL.

Sebelum melakukan sinkronisasi, kita perlu membuat koneksi ke database MySQL terlebih dahulu. Jika remote connection dengan root diperbolehkan maka gunakan user root. Jika tidak maka kita butuh membuat user baru terlebih dahulu. Berikut ini adalah cara bagaimana membuat user baru yaitu db\_owner.



# Lanjutan

Lakukan login terlebih dahulu ke MySQL dengan memasukkan password root.



```
MySQL Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.0.83-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\\h' for help. Type '\\c' to clear the current input statement.

mysql>
```

# Lanjutan

Buat user baru bernama dbo dengan password "owner".  
Ketikkan 3 perintah dibawah ini.

```
mysql>  
mysql>  
mysql> CREATE USER dbo IDENTIFIED BY 'owner';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> GRANT ALL ON *.* TO dbo;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> SET PASSWORD FOR dbo = OLD_PASSWORD('owner');  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

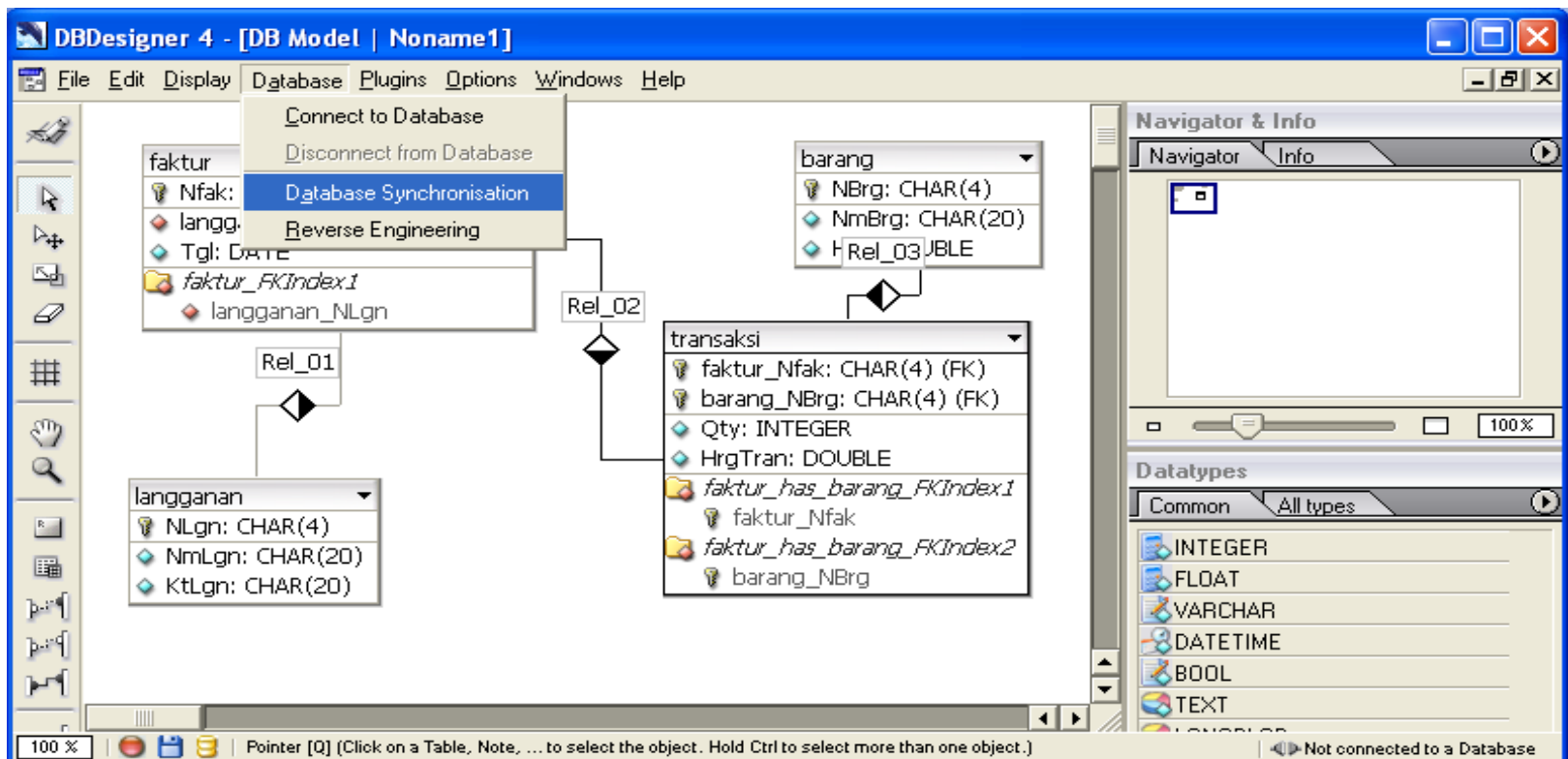
Buat Database baru yaitu dbpenjualan

```
mysql> create database dbpenjualan;  
Query OK, 1 row affected (0.03 sec)
```

# Lanjutan

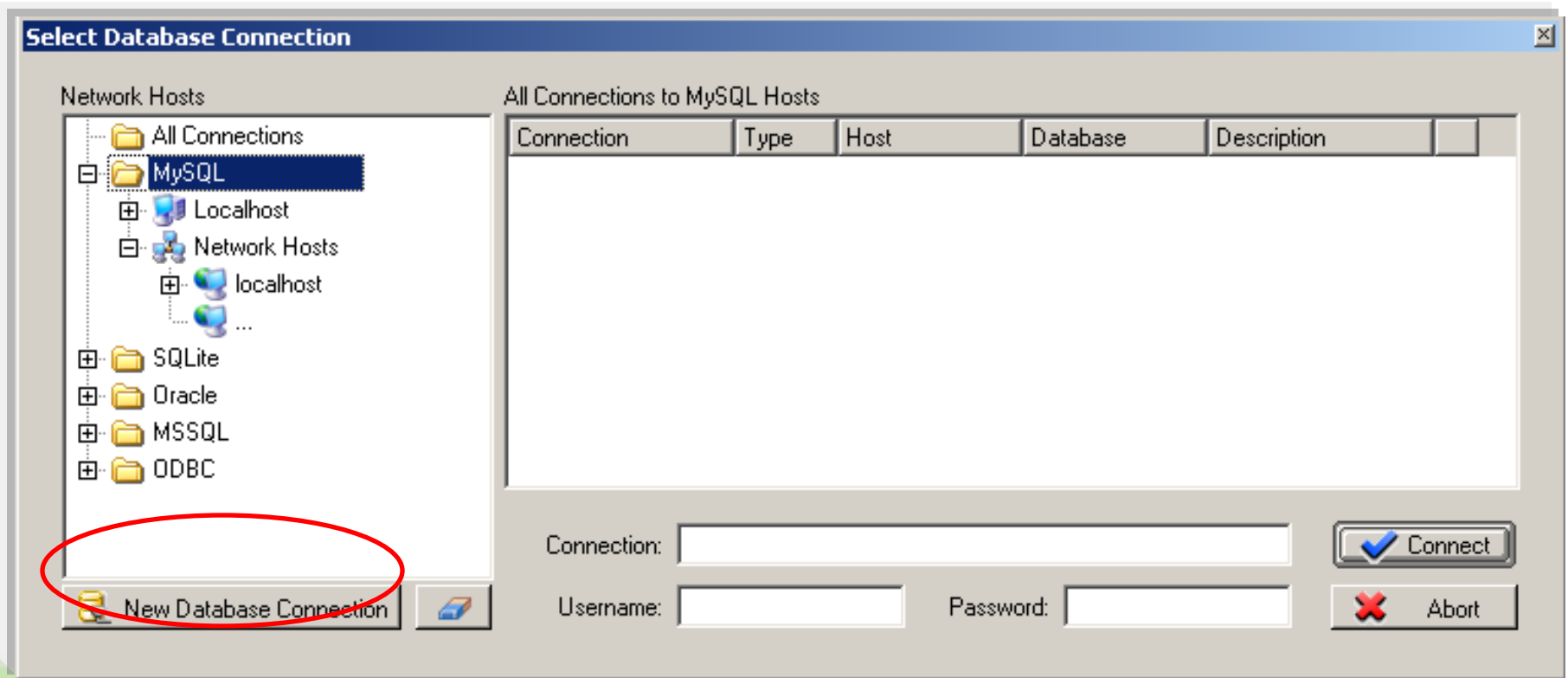
## Mengekspor Tabel Hasil Rancangan Ke Server Database

Mengekspor tabel ke server database bisa dilakukan dari menu **Database** → **Database Synchronisation** seperti gambar berikut.

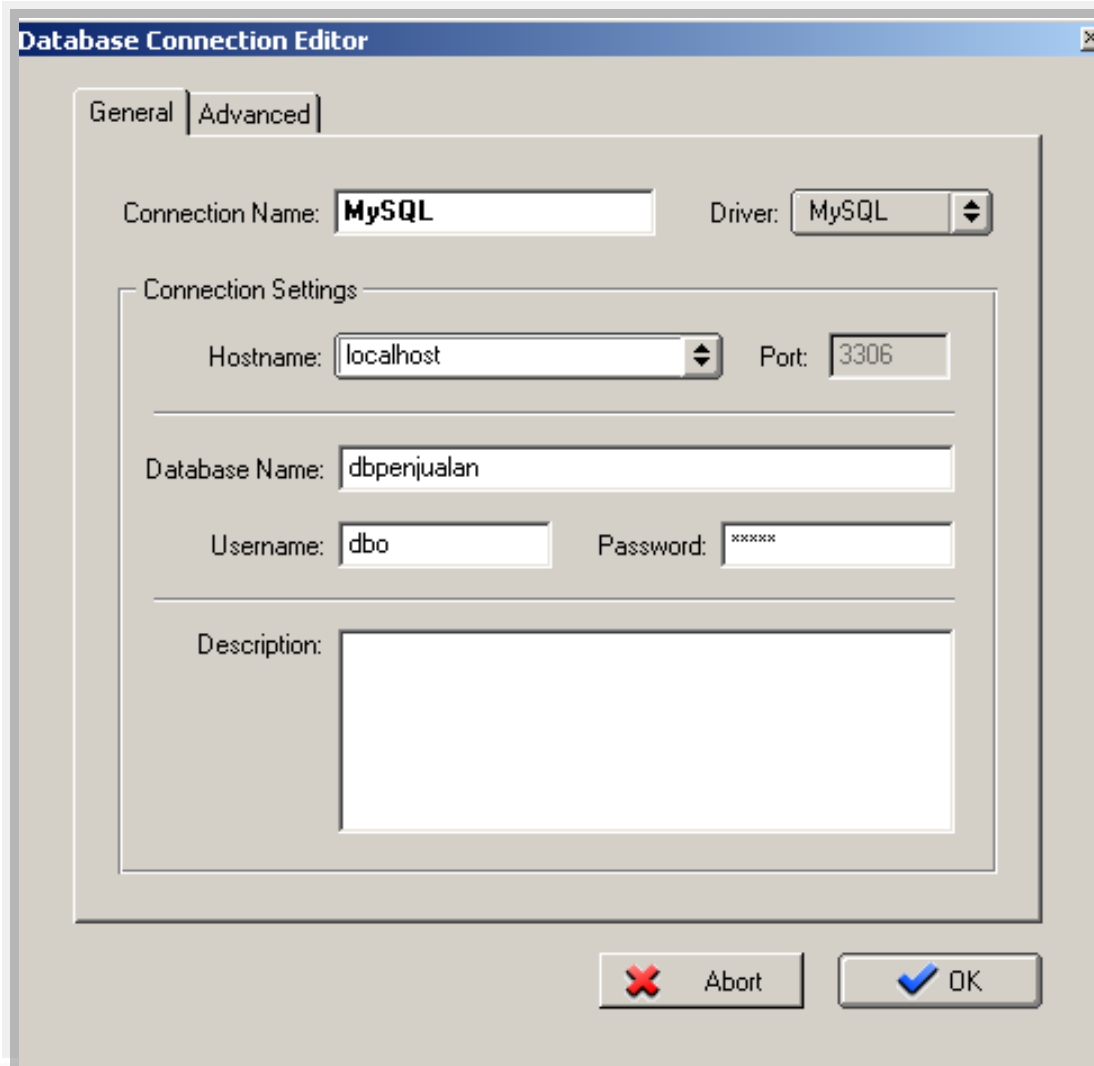


# Lanjutan

Lalu pilih MySQL sebagai database dan kemudian klik **New Database Connection**



# Lanjutan



Database Connection Editor

General | Advanced

Connection Name:  Driver:

Connection Settings

Hostname:  Port:

Database Name:

Username:  Password:

Description:

Masukkan Nilai berikut:

**Connection Name :**  
*MySQL*

**Hostname :** *localhost*

**Database Name :**  
*dbpenjualan*

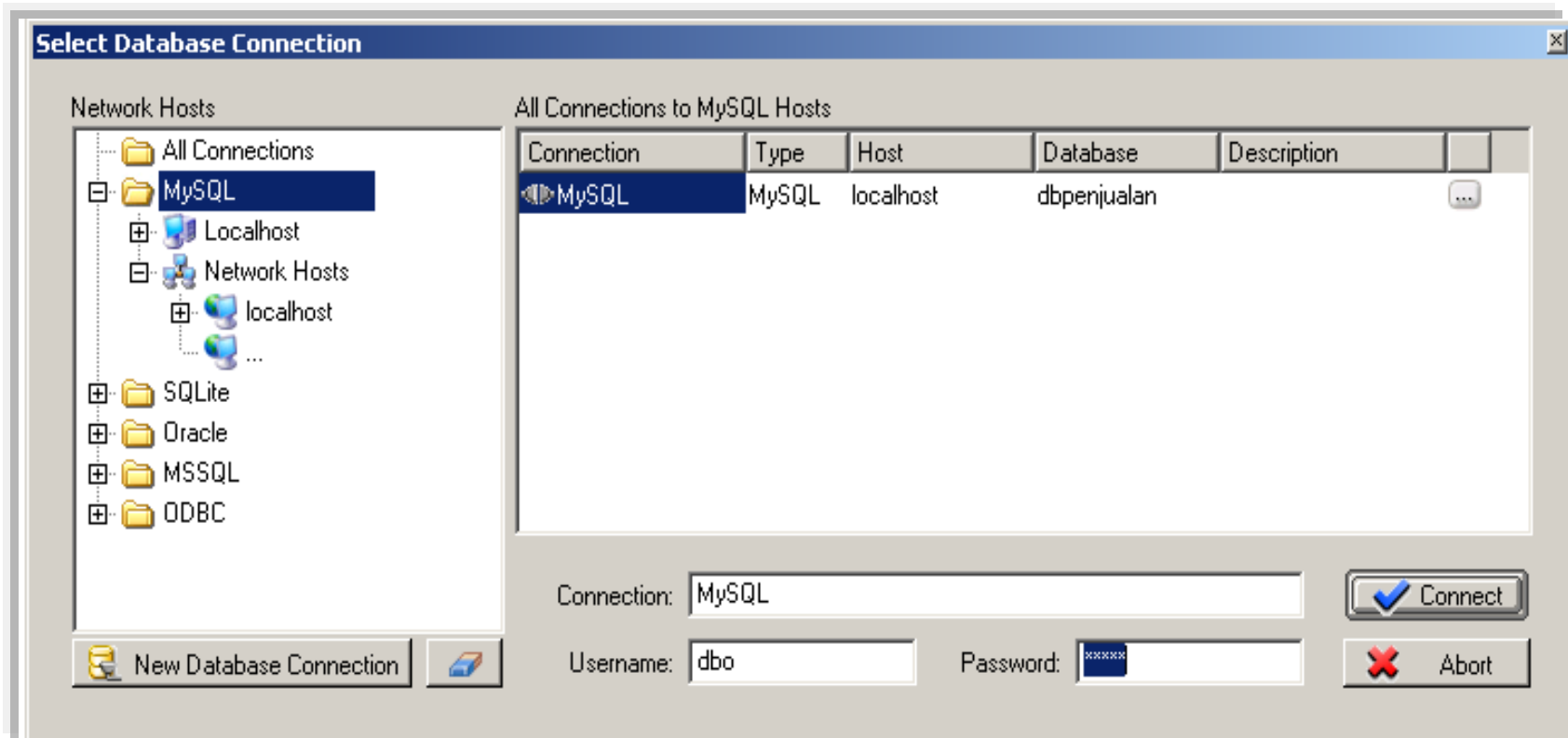
**UserName :** *dbo*

**Password :** *owner*

Lalu klik OK

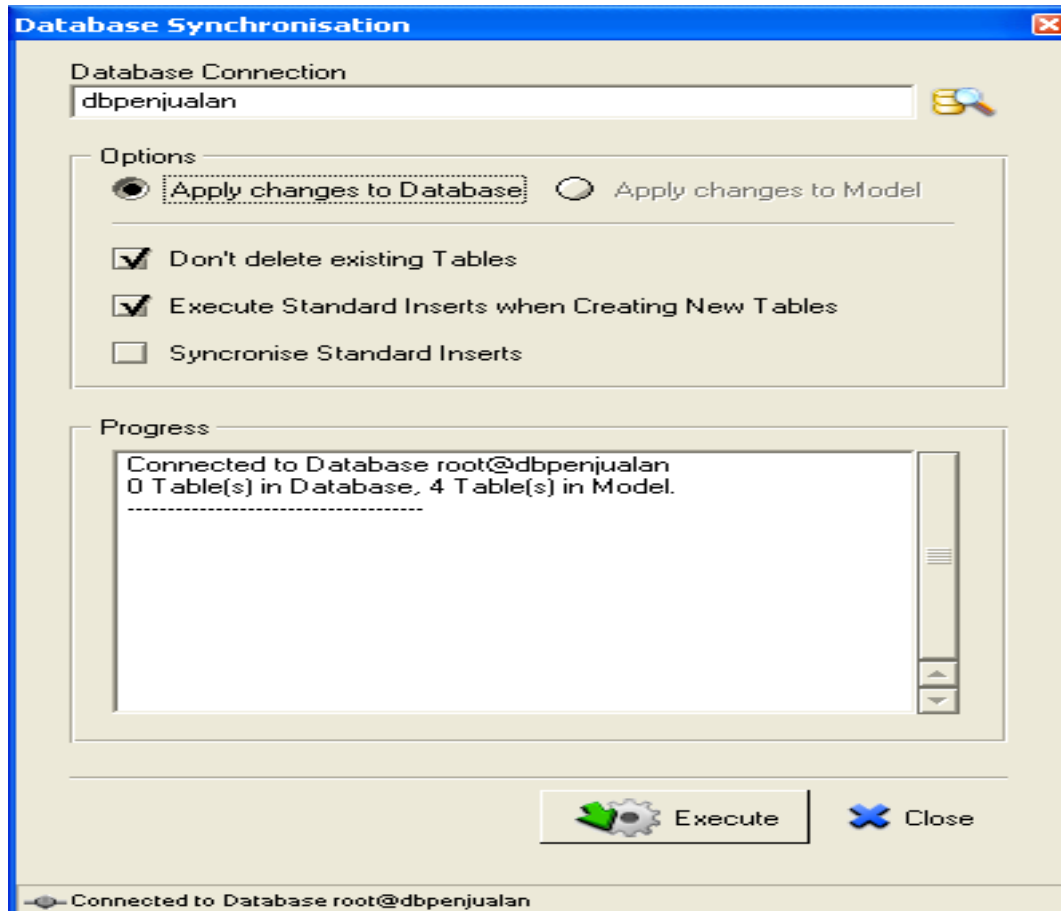
# Lanjutan

Klik **Connect** untuk terkoneksi ke MySQL



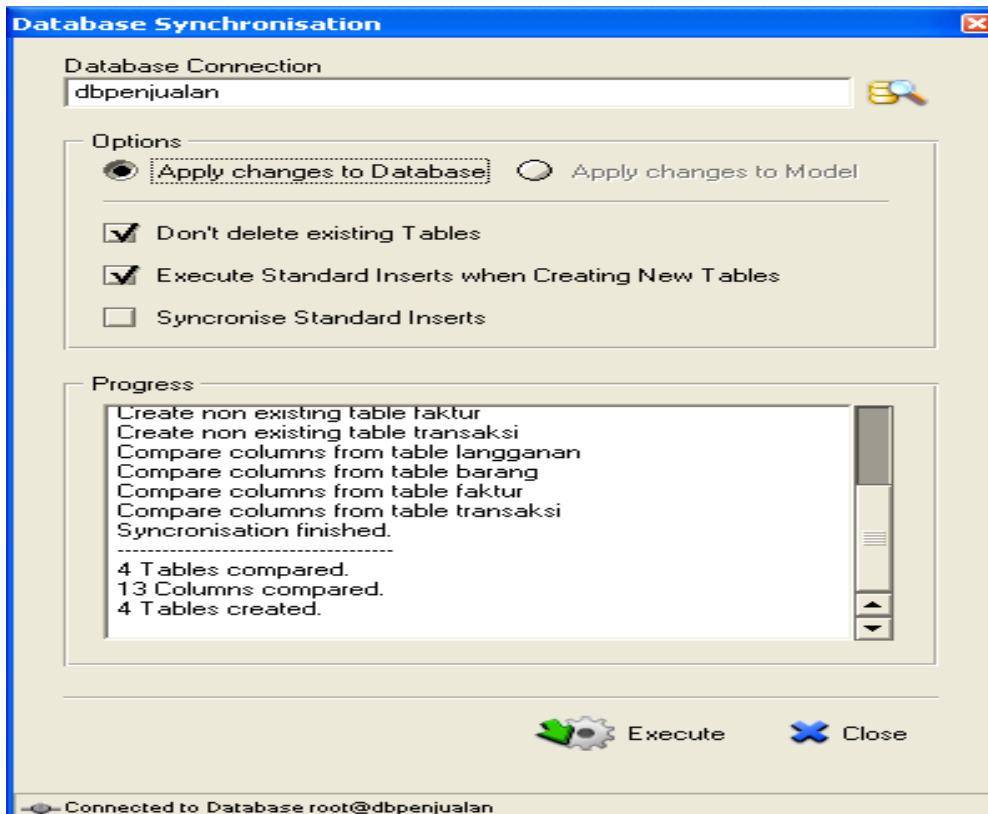
# Lanjutan

Klik **Execute** untuk mengeksekusi sinkronisasi



# Lanjutan

Setelah tampil jendela seperti di atas, selanjutnya klik tombol **EXECUTE** untuk mengekspor tabel ke server database MySQL dan akan tampil progress report seperti berikut





# Latihan 1

1. Sebuah perusahaan yang melayani pemesanan barang/produk umum memerlukan sebuah program aplikasi yang berfungsi untuk menyimpan data produk beserta suppliernya dan juga berfungsi untuk mencatat transaksi pemesanan produk dari customer. Setiap produk yang dipesan akan dikirim ke customer yang mememesannya. Rancanglah database untuk program aplikasi tersebut dengan menggunakan DBdesigner dan ekspor hasilnya ke server MySQL, untuk memenuhi keinginan perusahaan tersebut.

# Latihan 2

2. Seorang kolektor mobil ingin mendata seluruh mobil miliknya dan memerlukan program aplikasi yang bisa berfungsi untuk menyimpan data koleksi mobilnya. Rancanglah database untuk program aplikasi tersebut dengan menggunakan DBdesigner dan ekspor hasilnya ke server MySQL, sehingga program yang dikembangkan bisa memenuhi keinginan kolektor tersebut.

# Pertemuan 14

## Lingkungan Basis Data

# KONKURENSI

## CONCURRENCY (KONKURENSI)

Ada 3 masalah yang disebabkan oleh Concurrency :

1. Masalah kehilangan modifikasi (*Lost Update Problem*)

Masalah ini timbul jika dua transaksi mengakses item database yang sama yang mengakibatkan nilai dari database tersebut menjadi tidak benar.

# lanjutan

Transaksi A	Waktu	Transaksi B
=	↓	=
Baca R	t1	=
=	↓	=
=	t2	Baca R
=	↓	=
Modifikasi R	t3	=
=	↓	=
=	t4	Modifikasi R
=	↓	=

## Contoh Lost Update problem

Data transaksi pada rekening bersama (Ika dan Susi)

Waktu	Transaksi Ika	Transaksi Susi	Saldo
T1	Read Saldo	.....	1.000.000
T2	.....	Read Saldo	1.000.000
T3	Saldo:=Saldo-50.000	.....	1.000.000
T4	Write Saldo	.....	950.000
T5	.....	Saldo:= saldo+100.000	1.000.000
T6	.....	Write Saldo	1.100.000

Nilai saldo menjadi tidak benar disebabkan transaksi Susi membaca nilai saldo sebelum transaksi Ika mengubah nilai tersebut dalam database, sehingga nilai yang sudah di update yang dihasilkan dari transaksi Ika menjadi hilang.

# lanjutan

## 2. Masalah Modifikasi Sementara (*uncommitted Update Problem*)

Masalah ini timbul jika transaksi membaca suatu record yang sudah dimodifikasi oleh transaksi lain tetapi belum terselesaikan (*uncommitted*), terdapat kemungkinan kalau transaksi tersebut dibatalkan (*rollback*).

# lanjutan

Transaksi A	Waktu	Transaksi B
-	↓	-
Baca R	t1	Modifikasi R
-	↓	-
-	t2	-
-	↓	-
Modifikasi R	t3	Rollback
-	↓	-



## Contoh *uncommitted Update Problem*

Waktu	Transaksi Simpanan	Transaksi Bunga	Saldo
T1	Read Saldo	.....	1.000.000
T2	Saldo:=saldo+1.000.0000	.....	1.000.000
T3	Write Saldo	.....	2.000.000
T4	.....	Read Saldo	2.000.000
T5	.....	Saldo:= saldo*0.15	2.000.000
T6	.....	Write Saldo	2.300.000
T7	.....	RollBack	2.300.000

Nilai saldo menjadi tidak benar disebabkan terjadi RollBack pada T7 yang membatalkan transaksi sebelumnya (T6), sehingga saldo seharusnya tetap 2.000.000

# lanjutan

## 3. Masalah Analisa yang tidak konsisten (*Problem of inconsistency Analysis*)

Masalah ini timbul jika sebuah transaksi membaca suatu nilai tetapi transaksi yang kedua mengupdate beberapa nilai tersebut selama eksekusi transaksi pertama

# Contoh Problem of Inconsistency Analysis

⊕ Nilai 1 = 40	Nilai 2 = 50	Nilai 3 = 30
Transaksi A	Waktu	Transaksi B
-	↓	-
Baca nilai 1(40)	t1	-
Jum=40	↓	-
-	↓	-
Baca nilai 2(50)	t2	-
Juml=90	↓	-
-	↓	-
-	t3	baca nilai 3(30)
-	↓	-
-	t4	modifikasi nilai 3
-	↓	30 → 20
-	↓	-
-	t5	baca nilai 1(40)
-	↓	-
-	t6	modifikasi nilai 1
-	↓	40 → 50
-	↓	-
-	t7	-
-	↓	commit
Baca nilai 3(20)	t8	-
Juml=110(bukan 120)	↓	-
-		-

Transaksi A menjumlahkan nilai 1, nilai 2 dan nilai 3  
 Transaksi B → nilai 1 + 10, nilai 3 -10

# Locking

**LOCKING** adalah salah satu mekanisme pengontrol concurrency

## KONSEP DASAR :

Ketika sebuah transaksi memerlukan jaminan kalau record yang diinginkan tidak akan berubah secara mendadak, maka diperlukan kunci untuk record tersebut

## FUNGSI

Locking berfungsi untuk menjaga record tersebut agar tidak dimodifikasi oleh transaksi lain.

# lanjutan

Jenis- Jenis Lock :

## 1. Share (S)

Kunci ini memungkinkan pengguna dan para pengguna konkuren yang lain dapat membaca record tetapi tidak mengubahnya.

## 2. Exclusive (X)

Kunci ini memungkinkan pengguna untuk membaca dan mengubah record. Sedangkan pengguna konkuren lain tidak diperbolehkan membaca ataupun mengubah record tersebut.

	X	S	-
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

X = kunci X

S = kunci S

N = No

Y = Yes

# Cara Kerja Locking

Masalah kehilangan modifikasi (Lost Update Problem)

Transaksi A	Waktu	Transaksi B
-		-
baca R11 (kunci S)		-
-	t2	baca R(kunci S)
-	↓	-
modifikasi R (kunci X)	t3	-
tunggu	↓	-
⋮	t4	modifikasi R (kunci X)
⋮	↓	tunggu
⋮	↓	⋮
⋮	↓	⋮
tunggu	↓	tunggu

# lanjutan

Masalah Modifikasi Sementara (uncommitted Update Problem)

Transaksi A	Waktu	Transaksi B
-	↓	-
-	t1	modifikasi R (kunci X)
-	↓	-
baca R	t2	-
kunci (S)	↓	-
tunggu	↓	-
⋮	t3	synchpoint (kunci X dilepas)
⋮	↓	-
tunggu	t4	-
baca R kembali (Kunci S)	↓	-



# lanjutan

Transaksi A	Waktu	Transaksi B
-	t1	modifikasi R
-	↓	(kunci X)
Modifikasi R	t2	-
Kunci (X)	↓	-
tunggu	↓	-
:	t3	synchpoint
:	↓	(kunci X dilepas)
tunggu	↓	-
modifikasi R	t4	-
(Kunci X)	↓	-

# lanjutan

Masalah Analisa yang tidak konsisten (Problem of inconsistency Analisa)

⊕ Nilai 1 = 40	Nilai 2 = 50	Nilai 3 = 30
Transaksi A	Waktu	Transaksi B
-	↓	-
Baca nilai 1(40)	t1	-
(kunci S)	↓	-
Juml=40	↓	-
-	↓	-
Baca nilai 2(50)	t2	-
(kunci S)	↓	-
Juml=90	↓	-
-	t3	baca nilai 3(30)
-	↓	(kunci S)
-	t4	-
-	↓	modifikasi nilai 3
-	↓	(kunci X)
-	↓	30 → 20
-	t5	-
-	↓	baca nilai 1(40)
-	↓	(kunci S)
-	t6	-
-	↓	modifikasi nilai 1
-	↓	(kunci X)
modifikasi nilai 3	t7	tunggu
(kunci S)	↓	⋮
tunggu	↓	tunggu

# Timestamping

## **TIMESTAMPING**

Adalah salah satu alternatif mekanisme kontrol konkurensi yang dapat menghilangkan masalah dead lock

Dua masalah yang timbul pada Timestamping :

1. Suatu transaksi memerintahkan untuk membaca sebuah item yang sudah di update oleh transaksi yang belakangan.
2. Suatu transaksi memerintahkan untuk menulis sebuah item yang nilainya sudah dibaca atau ditulis oleh transaksi yang belakangan

# Crash dan Recovery

## **PENGERTIAN :**

Crash adalah suatu failure atau kegagalan dari suatu sistem

## **PENYEBAB DARI KEGAGALAN ADALAH :**

1. Disk Crash yaitu informasi yang ada di disk akan hilang
2. Power failure yaitu informasi yang disimpan pada memori utama dan register akan hilang
3. Software Error yaitu output yang dihasilkan tidak betul dan sistem databasenya sendiri akan memasuki suatu kondisi tidak konsisten

# Klasifikasi Failure

Berdasarkan Jenis storage

1. Volatile storage, biasanya informasi yang terdapat pada volatile akan hilang, jika terjadi kerusakan sistem (system crash) contoh: RAM
2. Non Volatile Storage, biasanya informasi yang terdapat pada non volatile storage tidak akan hilang jika terjadi kerusakan sistem contoh: ROM
3. Stable Storage, informasi yang terdapat dalam stable storage tidak pernah hilang. contoh: Harddisk RAID

# Jenis-Jenis Kegagalan

1. Logical Error, program tidak dapat lagi dilaksanakan disebabkan oleh kesalahan input, data tidak ditemukan, over flow
2. System Error, sistem berada pada keadaan yang tidak diinginkan, seperti terjadi deadlock, sebagai akibat program tidak dapat dilanjutkan namun setelah beberapa selang waktu program dapat dijalankan kembali.
3. System Crash, kegagalan fungsi perangkat keras, menyebabkan hilangnya data pada volatile storage, tetapi data pada non volatile storage masih tetap ada.
4. Disk Failure, hilangnya data dari sebuah blok disk disebabkan oleh kerusakan head atau kesalahan pada waktu pengoperasian transfer data

# Security

**SECURITY** adalah suatu proteksi data terhadap perusakan data dan pemakaian oleh pemakai yang tidak mempunyai ijin.

## **BEBERAPA MASALAH SECURITY SECARA UMUM :**

1. Di dalam suatu perusahaan siapa yang diijinkan untuk mengakses suatu sistem
2. Bila sistem tersebut menggunakan password, bagaimana kerahasiaan dari password tersebut dan berapa lama password tersebut harus diganti
3. Di dalam pengontrolan hardware, apakah ada proteksi untuk penyimpanan data (data storage)



# lanjutan

## **DUA KATAGORI PENYALAHGUNAAN DATABASE :**

1. Katagori yang tidak disengaja  
Contoh: Anomali yang disebabkan oleh pendistribusian data pada beberapa komputer
2. Katagori yang disengaja  
Contoh: Insert, Delete & Update oleh pihak yang tidak berwenang

## **BEBERAPA TINGKATAN MASALAH SECURITY :**

1. Phisical, berkaitan dengan pengamanan lokasi fisik database
2. Man, berkaitan dengan wewenang user
3. Sistem operasi, berkaitan dengan keamanan sistem operasi yang digunakan dalam jaringan
4. Sistem database, sistem dapat mengatur hak akses user

# Pemberian Wewenang dan View

**KONSEP VIEW** adalah cara yang diberikan pada seorang pemakai untuk mendapatkan model database yang sesuai dengan kebutuhan perorangan

Database relational membuat pengamanan pada level :

- Relasi, seorang pemakai diperbolehkan atau tidak mengakses langsung suatu relasi
- View, seorang pemakai diperbolehkan atau tidak mengakses data yang terdapat pada view
- Read Authorization, data dapat dibaca tapi tidak boleh dimodifikasi
- Insert Authoroization, pemakai boleh menambah data baru, tetapi tidak dapat memodifikasi data yang sudah ada

# lanjutan

- Update Authorization, pemakai boleh memodifikasi tetapi tidak dapat menghapus data
- Delete Authorization, pemakai boleh menghapus data
- Index Authorization, pemakai boleh membuat atau menghapus index
- Resource Authorization, mengizinkan pembuatan relasi – relasi baru
- Alternation Authorization, mengizinkan penambahan atau penghapusan atribut dalam satu relasi
- Drop Authorization, pemakai boleh menghapus relasi yang ada

# Integrity

## Integrity

Berarti memeriksa keakuratan dan validasi data

### **BEBERAPA JENIS INTEGRITY :**

1. **Integrity Konstains**, memberikan suatu sarana yang memungkinkan pengubahan database oleh pemakai berwenang sehingga tidak akan menyebabkan data inkonsistensi
2. **Integrity Rule** (pada basisdata relational), terbagi menjadi:
  - *Integrity Entity*, contoh: tidak ada satu komponen kunci primer yang bernilai kosong (null)
  - *Integrity Referensi*, suatu domain dapat dipakai sebagai kunci primer bila merupakan atribut tunggal pada domain yang bersangkutan