

BASIS DATA

DIVISI PENDIDIKAN DAN PELATIHAN

PUB 2022

BASIS DATA | DBMS | TIPE DATA |
ENTITY RELATIONSHIP DIAGRAM |
ATRIBUT | DEPENDENSI |
NORMALISASI | QUERY | DML |
TRIGGER | STORED PROCEDURE |
FUNCTION |

DOWNLOAD E-MODUL
SCAN ME!



Modul ini merupakan modul yang dibuat oleh Divisi Pendidikan dan Pelatihan Program Beasiswa Pemberdayaan Umat Berkelanjutan (PUB) periode yang kemudian ditulis ulang dan diperbaiki oleh Divisi Pendidikan dan Pelatihan periode 2021 – 2022.

TIM PENYUSUN MODUL

Pengarah	: Romi Kusuma Bakti	<i>(PUB Angkatan 18)</i>
Penanggung Jawab	: Yoga Hendrapratama	<i>(PUB Angkatan 18)</i>

ANGGOTA TIM

Hanafi Abdullah	<i>(PUB Angkatan 18)</i>
Tiara Agustin	<i>(PUB Angkatan 18)</i>
Edwar Maulana Sitompul	<i>(PUB Angkatan 19)</i>
Jamil Hamdi Harahap	<i>(PUB Angkatan 19)</i>

Diterbitkan Oleh:

Divisi Pendidikan dan Pelatihan Pemberdayaan Umat Berkelanjutan
Periode 2021 – 2022
Jawa Barat - Bandung, 2022

DAFTAR REVISI

Hlm.	Keterangan
[01]	Mengubah dan memperbaiki materi pengertian basis data; Mengubah dan memperbaiki materi DBMS
[02]	Memperbaiki Sub-Judul Pengkodean data menjadi Pemodelan data
[12]	Menambahkan symbol link pada tabel symbol ERD
[13]	memperbaiki dan menambahkan Kardinalitas notasi chen
[15]	memperbaiki dan menambahkan Kardinalitas notasi crow foot

SILABUS MATERI

NO	MATERI	INDIKATOR	DURASI	ALAT
1	Pengenalan Basis data & pengenalan ERD	Peserta dapat memahami materi: <ul style="list-style-type: none"> - Pengertian basis data - DBMS - Pengkodean data - Tipe data di MySql - Pengenalan ERD - Membuat ERD - Derajat relasi - Rasio Kardinalitas Tugas <ul style="list-style-type: none"> - Membuat ERD (sederhana) 	120 menit	Alat tulis dan kertas
2	Entity Relationship Diagram (ERD)	Peserta dapat memahami dan memperdalam materi: <ul style="list-style-type: none"> - Membuat ERD - Derajat relasi - Rasio kardinalitas - Jenis jenis atribut - Menentukan atribut yang sesuai Tugas <ul style="list-style-type: none"> - Menentukan atribut dari sebuah data 	120 menit	Alat tulis dan kertas
3	Dependensi dan Normalisasi	Peserta dapat memahami dan mengetahui materi tentang: <ul style="list-style-type: none"> - Pengertian Dependensi - Jenis Dependensi - Normalisasi - Menormalisasikan Tabel dari 1NF sampai 3NF Tugas <ul style="list-style-type: none"> - Menormalisasikan tabel 	120 menit	Alat tulis dan kertas
4	Dependensi dan Normalisasi	Peserta dapat memperdalam kembali materi: <ul style="list-style-type: none"> - Menormalisasikan tabel kembali Tugas <ul style="list-style-type: none"> - Menormalisasikan tabel 	120 menit	Alat tulis dan kertas
5	Data Definition Language (DDL)	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Mengetahi apa itu DDL - Mengenal Aplikasi yang digunakan - Membuat Database - Menggunakan database - Membuat tabel - Membuat query alter 	120 menit	Laptop dan Aplikasi SqlYog

		<ul style="list-style-type: none"> - Drop tabel - Drop database Tugas <ul style="list-style-type: none"> - Membuat database dan tabelnya beserta relasinya 		
6	Data Definition Language (DDL)	Peserta dapat mempertadalam kembali materi sebelumnya mengenai: <ul style="list-style-type: none"> - Membuat database dan tabel - Query alter - Query delete Tugas <ul style="list-style-type: none"> - Membuat database dan tabelnya beserta relasinya 	120 menit	Laptop dan Aplikasi SqlYog
7	Persiapan UTS	Peserta dapat mengulang kembali materi yang sudah dipahami dan membuat database dengan query	120 menit	Laptop dan Aplikasi SqlYog
8	Mid Test	Peserta mengerjakan soal UTS tentang materi Bab 1 sampai bab 4	120 menit	Alat tulis
9	Data Manipulation Language (DML)	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Pengertian DDL - Insert - Update - Delete Tugas: <ul style="list-style-type: none"> - DDL 	120 menit	Laptop dan Aplikasi SqlYog
10	Data Manipulation Language (DML)	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Select - Like - Between - And - OR - Order by - Limit Tugas: <ul style="list-style-type: none"> - Select 	120 menit	Laptop dan Aplikasi SqlYog
11	Data Manipulation Language (DML)	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Fungsi Agregat - Sub Select Tugas: <ul style="list-style-type: none"> - Sub select 	120 menit	Laptop dan Aplikasi SqlYog
12	Data Manipulation Language (DML)	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Sub select - Join Tugas: <ul style="list-style-type: none"> - Join 	120 menit	Laptop dan Aplikasi SqlYog

13	Data Manipulation Language (DML)	Memperdalam kembali materi select, join dan sub select	120 menit	Laptop dan Aplikasi SqlYog
14	Trigger, Function dan Store Procedure	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Apa itu trigger - Konsep Triger - Stored Procedure Tugas <ul style="list-style-type: none"> - Trigger 	120 menit	Laptop dan Aplikasi SqlYog
15	Trigger, Function dan Store Procedure	Peserta dapat memahami materi mengenai: <ul style="list-style-type: none"> - Memperdalam Stored Procedure - Function Tugas <ul style="list-style-type: none"> - Stored Procedure - Function 	120 menit	Laptop dan Aplikasi SqlYog
16	Post Test	Peserta dapat mengerjakan UAS dengan materi DDL, DML, sampai Trigger, Function dan Store Procedure	120 menit	Alat tulis

DAFTAR ISI

DAFTAR REVISI	i
SILABUS MATERI	ii
DAFTAR ISI	v
BAB I	1
PENGENALAN BASIS DATA	1
Pengertian Basis Data	1
DBMS (Data Base Management System)	1
Pemodelan Data	2
Tipe Data Dalam Mysql	2
BAB II	10
ENTITY RELATIONSHIP DIAGRAM (ERD)	10
Pengertian ERD	10
Derajat Relasi (Degrees of Relationship)	12
Rasio Kardinalitas (Cardinality Ratio Constraint)	13
[NOTASI CHEN]	13
[NOTASI CROW FOOT]	15
Atribut	20
BAB III	27
DEPENDENSI DAN NORMALISASI	27
DEPENDENSI	27
<i>Dependensi Fungsional</i>	27
NORMALISASI	29
BAB IV	36
DDL (DATA DEFINITION LANGUAGE)	36
Query Create	37
Query Alter	39
Query Drop	40
BAB V	42
DML (DATA MANIPULATION LANGUAGE)	42
Pengertian DML	42

Query INSERT	42
Query UPDATE	43
Query DELETE	43
Query SELECT	44
Fungsi Agregat	47
Sub Select Dan Join	50
BAB VI	61
TRIGGER, FUNCTION DAN STORE PROCEDURE	61
Trigger	61
Stored Procedure	63
Function	65

BAB I

PENGENALAN BASIS DATA

Pengertian Basis Data

Basis Data terdiri dari dua kata, yaitu Basis dan Data. Basis dapat diartikan sebagai markas atau gudang dimana tempat bersarang/berkumpul. Sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

Database atau basis data adalah kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Melalui pengelolaan tersebut pengguna dapat memperoleh kemudahan dalam mencari informasi, menyimpan informasi dan membuang informasi.

DBMS (Data Base Management System)

DBMS adalah suatu sistem atau software yang dirancang khusus untuk mengelola suatu database dan menjalankan operasi terhadap data yang diminta oleh banyak pengguna. DBMS adalah singkatan dari "Database Management System" yaitu sistem penorganisasian dan sistem pengolahan Database pada komputer. DBMS atau database management system ini merupakan perangkat lunak (software) yang dipakai untuk membangun basis data yang berbasis komputerisasi.

DBMS merupakan perantara untuk user dengan basis data, untuk dapat berinteraksi dengan DBMS dapat memakai bahasa basis data yang sudah ditentukan oleh perusahaan DBMS. Bahasa basis data umumnya terdiri dari berbagai macam instruksi yang diformulasikan sehingga instruksi tersebut dapat diproses oleh DBMS. DBMS (Database Management system) ini juga dapat membantu dalam memelihara serta pengolahan data dalam jumlah yang besar,

dengan menggunakan DBMS bertujuan agar tidak dapat menimbulkan kekacauan dan dapat dipakai oleh user sesuai dengan kebutuhan.

Database merupakan salah satu komponen dalam teknologi informasi yang mutlak diperlukan oleh semua organisasi yang ingin mempunyai suatu sistem informasi yang terpadu untuk menunjang kegiatan organisasi demi mencapai tujuannya. Karena pentingnya peran database dalam sistem informasi, tidaklah mengherankan bahwa terdapat banyak pilihan software Database Management System (DBMS) dari berbagai vendor baik yang gratis maupun yang komersial. Beberapa contoh DBMS yang populer adalah MySQL, MS SQL Server, Oracle, Firebird, Database Desktop Paradox dan MS Access.

Pemodelan Data

Komponen yang terdapat dalam basis data antara lain:

1. Entitas

Entitas adalah penyajian data dalam bentuk objek nyata yang keberadaannya dapat kita lihat. Contoh dari entitas adalah mahasiswa, dosen, jurusan dll.

2. Atribut

Atribut adalah keterangan-keterangan yang menjelaskan suatu entitas. Contoh atribut dari entitas mahasiswa adalah NIM, Nama, Alamat, Jurusan dll.

3. Relationship/Hubungan

Relationship atau biasa disebut relasi adalah keterangan yang akan menggambarkan hubungan antara satu entitas dengan entitas lain. Contoh relasi antara entitas mahasiswa dengan entitas mata kuliah adalah mahasiswa mengambil banyak mata kuliah. Kata mengambil tersebut menggambarkan hubungan antara mahasiswa dan mata kuliah.

Tipe Data Dalam Mysql

Tipe data adalah suatu bentuk pemodelan data yang didedikasikan pada saat melakukan pembuatan tabel. Tipe data ini akan mempengaruhi setiap data yang akan dimasukkan ke dalam sebuah tabel. Data yang akan dimasukkan harus sesuai dengan tipe data yang dideklarasikan.

Secara umum tipe data pada MySQL dibagi menjadi 4 yaitu:

1. **Numeric Values** yaitu angka atau bilangan seperti 10; 123; 100.50; -10; 1.2E+17; 2.7e-11; dan sebagainya. Tipe data ini dibagi menjadi dua yaitu Bilangan Bulat (Integer) dan Bilangan Pecahan (Floating-point). Bilangan bulat adalah bilangan tanpa tanda desimal sedangkan bilangan pecahan adalah bilangan dengan tanda desimal. Kedua jenis bilangan ini dapat bernilai positif (+) dan juga negatif (-). Jika bilangan tersebut menggunakan tanda positif (+) atau (-), maka disebut SIGNED. Sebaliknya bila tanpa tanda apapun disebut UNSIGNED. Karena tanda positif (+) dapat diabaikan penulisannya maka pada bilangan yang bernilai positif disebut UNSIGNED.
2. **String/Character Values** adalah semua karakter (atau teks) yang penulisannya selalu diapit oleh tanda kutip baik kutip tunggal (') maupun kutip ganda ("). Hal ini tidak hanya berlaku pada huruf alfabet saja tetapi angka yang ditulis dengan tanda kutip pun akan menjadi karakter atau string.
3. **Date and Time Values** yaitu tanggal dan waktu. Untuk jenis data tanggal dan waktu format standar (default) penulisan tanggalnya adalah "tahun-bulan-tanggal", Misalnya untuk 22 Januari 2001 dituliskan "2002-01-22". Untuk penulisan waktu, formatnya adalah "jam-menit-detik". Contoh, "13:55:07". Data tanggal dan waktu bisa digabung penulisannya menjadi "2002-01-22 13:55:07".
4. **NULL**. NULL sebenarnya bukan data, tapi dia mewakili sesuatu yang "tidak pasti", "tidak diketahui" atau "belum ada nilainya". Sebagai contoh dalam kehidupan sehari-hari Anda melakukan suatu survei berapa jumlah pengguna sistem operasi LINUX dan Windows di Indonesia. Selama survei belum tuntas maka data pastinya belum dapat diketahui. Oleh sebab itu, data tersebut bisa diwakili dengan NULL, alias belum diketahui.

Pada tipe-tipe data MySQL terdapat beberapa atribut yang memiliki arti sebagai berikut:

- M, menunjukkan lebar karakter maksimum. Nilai M maksimum adalah 255.
- D, menunjukkan jumlah angka di belakang koma. Nilai maksimum D adalah 30 tetapi dibatasi oleh nilai M, yaitu tidak boleh lebih besar daripada M-2.
- Atribut yang diberi tanda [dan] berarti pemakaiannya adalah optional.
- Jika atribut ZEROFILL disertakan, MySQL akan otomatis menambahkan atribut UNSIGNED.
- UNSIGNED adalah bilangan tanpa tanda di depannya (misalnya tanda negatif).

Tipe data untuk bilangan

TIPE DATA	BENTUK DASAR PENULISAN	KETERANGAN	UKURAN
TINYINT	TINYINT [(M)] [UNSIGNED] [ZEROFILL]	Digunakan untuk menyimpan data bilangan bulat positif dan negatif. Bilangan bulat terkecil, dengan jangkauan untuk bilangan bertanda -128 sampai dengan 127 dan untuk yang tidak bertanda 0 sampai dengan 255. Bilangan tak bertanda ditandai dengan kata UNSIGNED	1 byte
SMALLINT	SMALLINT [(M)] [UNSIGNED] [ZEROFILL]	Digunakan untuk menyimpan data bilangan bulat positif dan negatif. Jangkauan untuk bilangan bertanda -32768 sampai dengan 32767 dan untuk yang tidak bertanda (jangkauan unsigned) 0 sampai dengan 65535	2 byte
MEDIUMINT	MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]	Digunakan untuk menyimpan data bilangan bulat positif dan negatif. Jangkauan untuk bilangan bertanda -8388608 sampai dengan 8388607 dan untuk yang tidak bertanda	3 byte

		(jangkauan unsigned) 0 sampai dengan 16777215	
INT	INT [(M)] [UNSIGNED] [ZEROFILL]	Digunakan untuk menyimpan data bilangan bulat positif dan negatif . Jangkauan untuk bilangan bertanda -2147483648 sampai dengan 2147483647 dan untuk yang tidak bertanda (jangkauan unsigned) 0 sampai dengan 4294967295	4 byte
	INTEGER [(M)] [UNSIGNED] [ZEROFILL]	Sama dengan INT.	
BIGINT	BIGINT [(M)] [UNSIGNED] [ZEROFILL]	Digunakan untuk menyimpan data bilangan bulat positif dan negatif . Jangkauan untuk bilangan bertanda -9223372036854775808 sampai dengan 9223372036854775807 dan untuk yang tidak bertanda 0 sampai dengan 18446744073709551615	8 byte
FLOAT	FLOAT [(M,D)] [ZEROFILL]	Digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi tunggal.	4 byte

		<p>Jangkauan: -</p> <p>3.402823466E+38 s/d -</p> <p>1.175494351E-38, 0, dan</p> <p>1.175494351E-38 s/d</p> <p>3.402823466E+38</p>	
DOUBLE	DOUBLE [(M,D)] [ZEROFILL]	<p>Digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi ganda.</p> <p>Tidak dapat bersifat unsigned.</p> <p>Nilai yang diijinkan adalah -</p> <p>1.7976931348623157E+308</p> <p>hingga -</p> <p>2.2250738585072014E-308</p> <p>untuk nilai negatif, 0, dan</p> <p>2.2250738585072014E-308</p> <p>hingga</p> <p>1.7976931348623157E+308</p> <p>untuk nilai positif.</p>	8 byte
	DOUBLE PRECISION [(M,D)] [ZEROFILL]	Bilangan pecahan berpresisi ganda.	8 byte
	REAL [(M,D)] [ZEROFILL]	Sinonim dari DOUBLE.	8 byte
DECIMAL	DECIMAL [(M,D)] [ZEROFILL]	<p>Digunakan untuk menyimpan data bilangan pecahan positif dan negatif.</p> <p>Misalnya DECIMAL(5,2) dapat digunakan untuk menyimpan bilangan -99,99 sampai dengan</p>	M byte

		99,99	
	NUMERIC [(M,D)] [ZEROFILL]	Sama dengan Decimal.	M byte.

Tipe data untuk tanggal dan jam

TIPE DATA	BENTUK DASAR PENULISAN	KETERANGAN	UKURAN
DATETIME		Kombinasi tanggal dan waktu dengan jangkauan dari '1000-01-01 00:00:00' sampai dengan '9999-12-31 23:59:59'	8 byte.
DATE		Kombinasi tanggal dan waktu dengan jangkauan dari '1000-01-01' sampai dengan '9999-12-31'	8 byte.
TIMESTAMP	TIMESTAMP [(M)]	Kombinasi tanggal dan waktu dengan jangkauan dari '1970-01-01' sampai dengan '2037'	4 byte.
TIME		Digunakan untuk menyimpan waktu dengan jangkauan dari -838:59:59 sampai dengan 838:59:59	3 byte.
YEAR		Digunakan untuk menyimpan data tahun dari tanggal antara 1901 sampai dengan 2155	1 byte

Tipe data untuk karakter dan lain-lain

TIPE DATA	BENTUK DASAR PENULISAN	KETERANGAN	UKURAN
-----------	---------------------------	------------	--------

CHAR	CHAR(M) [BINARY]	Data string dengan panjang yang tetap. CHAR(1) cukup ditulis dengan CHAR. $1 \leq M \leq 255$, Jika ada sisa, maka sisa tersebut diisi dengan spasi (misalnya nilai M adalah 10, tapi data yang disimpan hanya memiliki 7 karakter, maka 3 karakter sisanya diisi dengan spasi). Spasi ini akan dihilangkan apabila data dipanggil. Nilai dari CHAR akan disortir dan diperbandingkan secara case-insensitive menurut default character set yang tersedia, kecuali bila atribut BINARY disertakan.	M byte.
VARCHAR	VARCHAR (M) [BINARY]	Ukuran L+1 byte dengan $L \leq M$ dan $1 \leq M \leq 255$. Data string dengan panjang bervariasi tergantung datanya. Jika nilai M adalah 10 sedangkan data yang disimpan hanya terdiri dari 5 karakter, maka lebar data tersebut hanya 5 karakter saja, tidak ada tambahan spasi.	L+1 byte
TINYBLOB, TINYTEXT		L+1 byte, dengan $L < 28$. Tipe	L+1 byte

		TEXT atau BLOB dengan panjang maksimum 255 karakter.	
BLOB, TEXT		L+2 byte, dengan L<216. Tipe TEXT atau BLOB dengan panjang maksimum 65535 karakter.	L+2 byte
MEDIUMBLOB, MEDIUMTEXT		L+3 byte, dengan L<224. Tipe TEXT atau BLOB dengan panjang maksimum 1677215 karakter.	L+3 byte
LOB, LONGTEXT		L+4 byte, dengan L<232. Tipe TEXT atau BLOB dengan panjang maksimum 4294967295 karakter.	L+4 byte
ENUM	ENUM('nilai1','nilai2',...)	Ukuran 1 atau 2 byte tergantung nilai enumerasinya maks 65535 nilai	1 atau 2 byte
SET	SET('nilai1','nilai2',...)	Ukuran 1,2,3,4 atau 8 byte tergantung jumlah anggota himpunan maks 64 anggota.	1,2,3,4 atau 8 byte

BAB II



ENTITY RELATIONSHIP DIAGRAM (ERD)

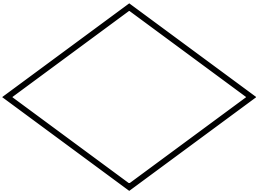
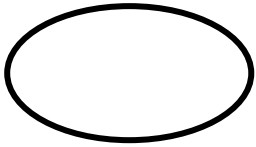
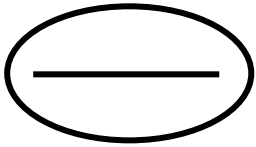
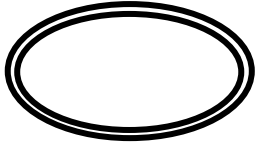
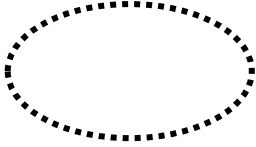
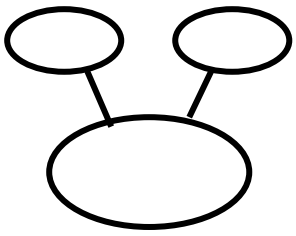
Pengertian ERD


ERD merupakan suatu model untuk menjelaskan hubungan antara data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. ERD untuk memodelkan struktur data dan hubungan antara data, untuk menggambarkannya digunakan beberapa notasi dan simbol.

Menurut salah satu para ahli, Brady dan Loonan (2010), Entity Relationship Diagram (ERD) merupakan Teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh system. Analys dalam tahap analisis persyaratan proyek pengembangan system. Sementara seolah-olah Teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari system informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk database.

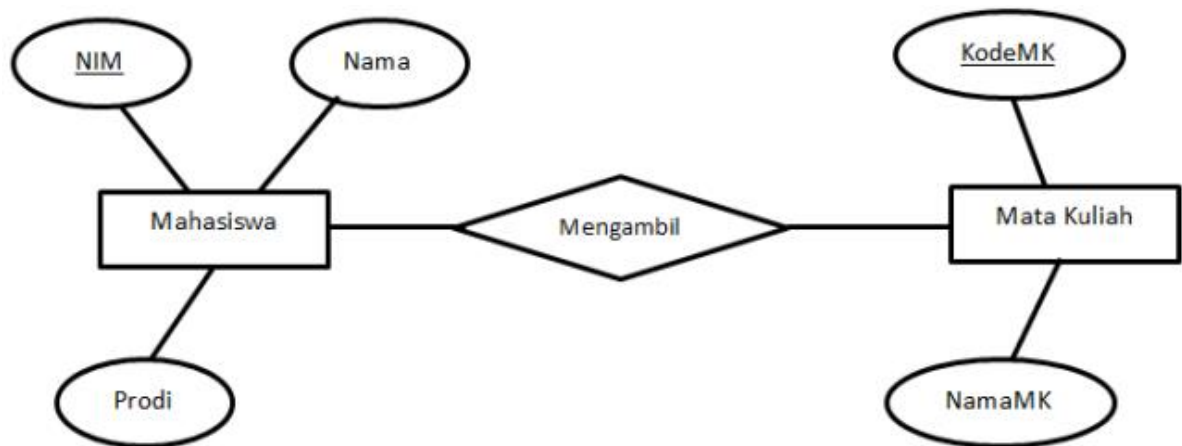
Berikut adalah simbol-simbol yang digunakan ERD:

NO	NAMA	SIMBOL	KETERANGAN
1	Entitas		Entitas, yang diwakili oleh persegi panjang. Entitas adalah objek atau konsep yang ingin Anda simpan informasinya.
2	Weak Entity (Entitas Lemah)		Entitas yang lemah adalah entitas yang harus ditentukan oleh hubungan kunci asing dengan entitas lain karena tidak dapat diidentifikasi secara unik dengan atributnya sendiri.

3	Relationship (Relasi)		Relasi adalah Tindakan, yang diwakili oleh bentuk intan, menunjukkan bagaimana dua entitas berbagi informasi dalam database.
4	Attribute		Atribut, yang diwakili oleh oval. Atribut kunci adalah karakteristik entitas yang unik dan berbeda.
5	Attribute Key		Satu atau beberapa atribut yang mempunyai nilai unik sehingga dapat digunakan untuk membedakan data pada suatu baris/record dengan baris lain pada suatu entitas.
6	Attribute Multivalued		Atribut multivalued dapat memiliki lebih dari satu nilai. Misalnya, entitas karyawan dapat memiliki beberapa nilai keterampilan.
7	Atribut Derivatif (Turunan)		Atribut turunan adalah suatu atribut yang dihasilkan dari atribut lain. Misalnya, gaji bulanan karyawan didasarkan pada gaji tahunan karyawan.
8	Atribut Composite		Atribut Komposita adalah atribut yang terdiri dari beberapa atribut yang lebih mendasar. Contoh: Entity mahasiswa memiliki atribut nama yang terdiri dari nama depan (first name), nama tengah (middle name) dan nama belakang (last name).

9	Link		Digunakan untuk menghubungkan entity dengan relasi dan entity dengan atribut
---	------	---	--

Berikut adalah contoh sederhana dari ERD:



Derajat Relasi (Degrees of Relationship)

Derajat Relasi menunjukkan jumlah entitas yang terlibat dalam suatu relationship.

Derajat relasi secara garis besar ada 3, yaitu:

- Unary Degree (Derajat Satu) : suatu entity yang mempunyai relasi terhadap dirinya sendiri.

Contoh:



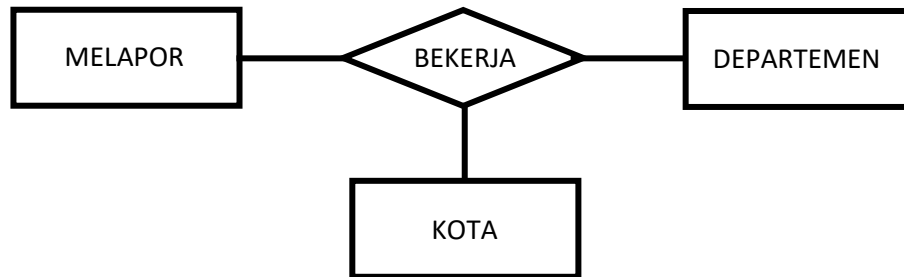
- Binary Degree (Derajat Dua) : Terdapat dua entity yang saling berhubungan.

Contoh:



- Ternary Degree (Derajat Tiga) : Terdapat tiga entity yang saling berhubungan.

Contoh:



Rasio Kardinalitas (Cardinality Ratio Constraint)

Kardinalitas pemetaan atau rasio kardinalitas menunjukkan jumlah entitas yang dapat di hubungkan ke satu entity lain dengan suatu relasi.

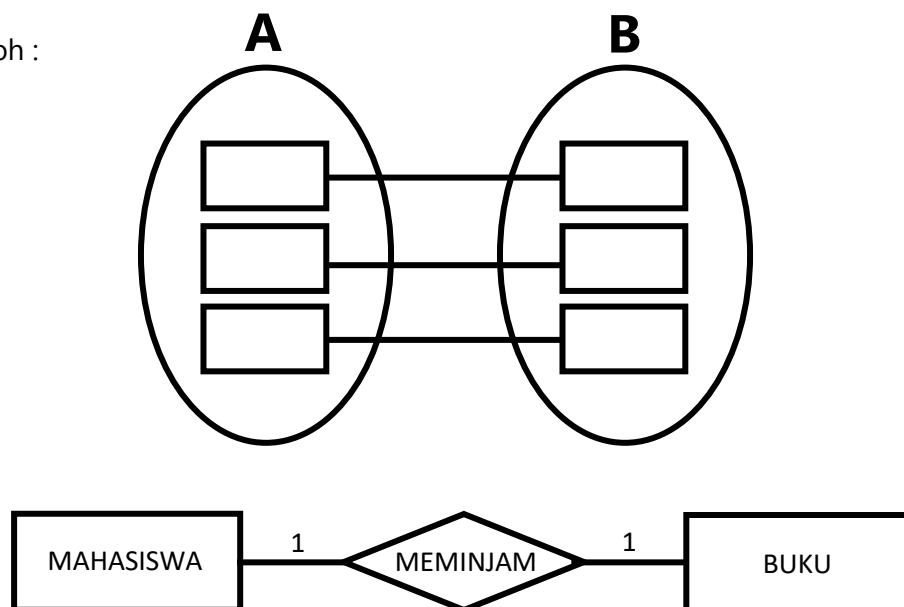
[NOTASI CHEN]

Kardinalitas pemetaan (**notasi chen**) meliputi:

- Satu ke Satu (1:1 / one to one)

1-1 yaitu setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan B, dan begitu juga sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

Contoh :

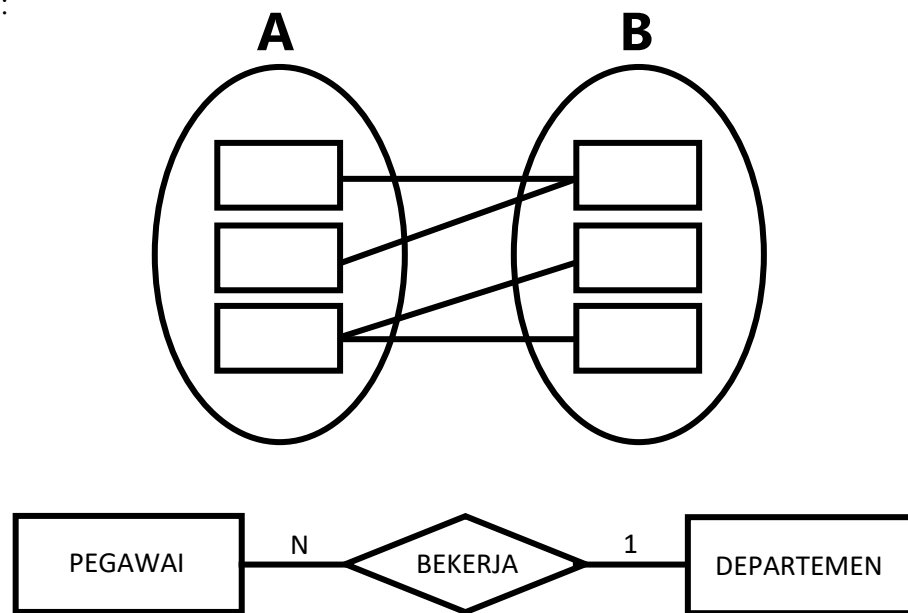


ERD diatas menyatakan bahwa 1 mahasiswa meminjam 1 buku dan 1 buku dipinjam oleh 1 mahasiswa.

- b. Satu ke Banyak atau Banyak ke Satu (1:M / one to many atau M:1 / many to one)

1-N / M-1 berarti bahwa setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

Contoh:

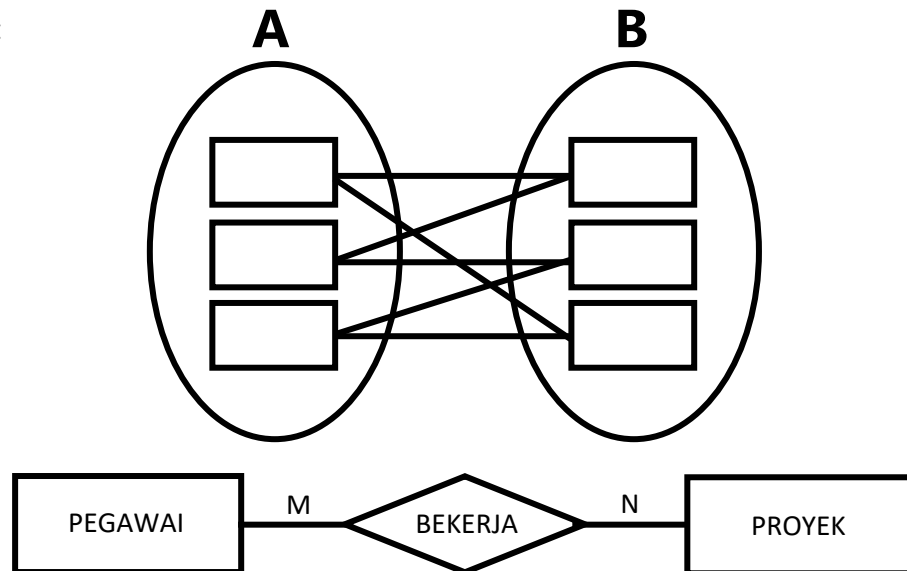


ERD diatas menyatakan bahwa banyak pegawai bekerja di 1 departement, dan 1 departement dikerjakan banyak pegawai.

- c. Banyak ke Banyak (M:N / Many to Many)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B dan demikian juga sebaliknya, dimana setiap entitas pada himpunan entitas B dapat berhubungan banyak entitas pada himpunan entitas A.

Contoh:



ERD diatas menyatakan bahwa banyak pegawai bekerja di banyak proyek, dan proyek dikerjakan oleh banyak pegawai.

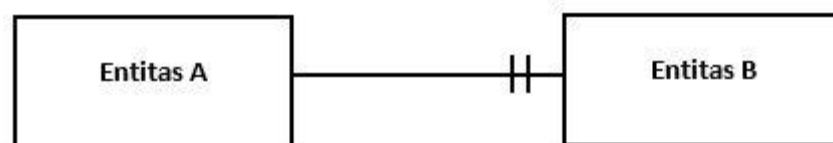
[NOTASI CROW FOOT]

Derajat relasi entitas (kardinalitas) dapat pula digambarkan dengan notasi crow foot. Secara garis besar, terdapat beberapa kemungkinan derajat relasi yang dapat terjadi, yaitu sebagai berikut:

1) one and only one (satu dan hanya satu-satunya);

One and only one adalah derajat relasi yang terjadi antara satu entitas (misalnya entitas A) dan entitas lainnya (misalnya entitas B), dengan ketentuan bahwa anggota himpunan pada entitas A hanya dapat berpasangan dengan satu anggota himpunan pada entitas B.

Contoh:



Penjelasan

Dengan menggunakan notasi crow foot, posisi keterangan derajat relasi untuk entitas A diletakkan dekat dengan entitas B. Bila kita perhatikan terdapat dua garis lurus secara vertikal. Garis tersebut disebut dash.

Posisi garis sebelah kiri menunjukkan jumlah minimum anggota entitas A yang dapat berpasangan dengan anggota entitas B, sedangkan posisi garis sebelah kanan menunjukkan jumlah maksimum anggota entitas A yang dapat berpasangan dengan anggota entitas B.

Mohon diperhatikan, karena derajat relasi dengan notasi crow foot tiap entitas dipisahkan, maka derajat relasi one and only one ini hanya berlaku dari entitas A ke entitas B, tetapi belum tentu derajat relasi dari entitas B ke entitas A sama. Oleh sebab itu, kita juga perlu membuat notasi crow foot derajat relasi dari entitas B ke entitas A.

Contoh lain:



Penjelasan

Derajat relasi antara entitas siswa dengan buku rapor adalah (one and only one). Alasannya, yaitu sebagai berikut:

- satu siswa hanya memiliki satu dan satu-satunya buku rapor.
- satu buku rapor hanya dimiliki oleh satu dan satu-satunya siswa.

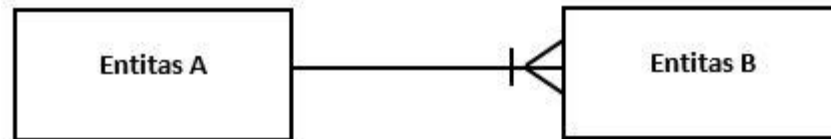
2) **one or many (satu atau banyak);**

One or many adalah derajat relasi yang terjadi antara satu entitas (misalnya entitas A) dan entitas lainnya (misalnya entitas B), dengan ketentuan sebagai berikut:

- anggota himpunan pada entitas A minimal dapat berpasangan dengan satu anggota himpunan pada entitas B;

- Anggota himpunan pada entitas A maksimal dapat berpasangan dengan lebih dari satu anggota himpunan pada entitas B.

Contoh:



Penjelasan

Bila kita perhatikan notasi derajat relasi tersebut, terdapat garis lurus vertikal (disebut dash) dan garis yang membentuk kaki burung gagak (disebut crow foot).

Posisi garis sebelah kiri menunjukkan jumlah minimum anggota entitas A yang dapat berpasangan dengan anggota entitas B (minimal satu), sedangkan posisi kaki burung gagak sebelah kanan, menunjukkan jumlah maksimum anggota entitas A yang dapat berpasangan dengan anggota entitas B (maksimal lebih dari satu).

Contoh lain:



Penjelasan

Derajat relasi antara pembeli dan faktur penjualan adalah one or many. Alasannya, yaitu sebagai berikut:

- satu pembeli minimal memiliki satu faktur penjualan;
- satu pembeli maksimal memiliki banyak faktur penjualan.

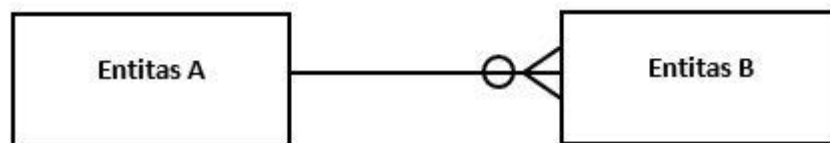
Derajat relasi antara faktur penjualan dan pembeli adalah one and only one. Alasannya, karena tiap satu faktur penjualan hanya dimiliki oleh satu pembeli.

3) **zero or one or many (nol, satu atau banyak);**

Zero or one or many adalah derajat relasi yang terjadi antara satu entitas (misalnya entitas A) dan entitas lainnya (misalnya entitas B), dengan ketentuan sebagai berikut:

- anggota himpunan pada entitas A minimal dapat berpasangan dengan nol anggota himpunan pada entitas B;
- Anggota himpunan pada entitas A maksimal dapat berpasangan dengan banyak anggota himpunan pada entitas B.

Contoh:

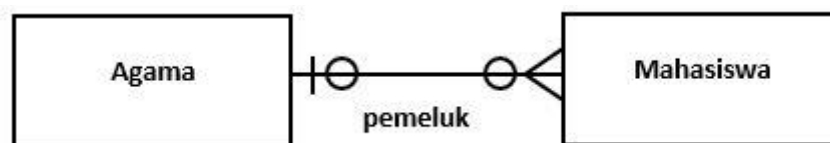


Penjelasan

Bila kita perhatikan notasi derajat relasi tersebut, terdapat lingkaran (disebut ring) dan garis yang membentuk kaki burung gagak (disebut crow foot).

Posisi garis sebelah kiri menunjukkan jumlah minimum anggota entitas A yang dapat berpasangan dengan anggota entitas B (minimal nol), sedangkan posisi kaki burung gagak sebelah kanan, menunjukkan jumlah maksimum anggota entitas A yang dapat berpasangan dengan anggota entitas B (maksimal lebih dari satu).

Contoh lain:



Penjelasan

Derajat relasi entitas agama dengan entitas mahasiswa (di suatu perguruan tinggi) adalah zero or one or many. Alasannya, yaitu sebagai berikut:

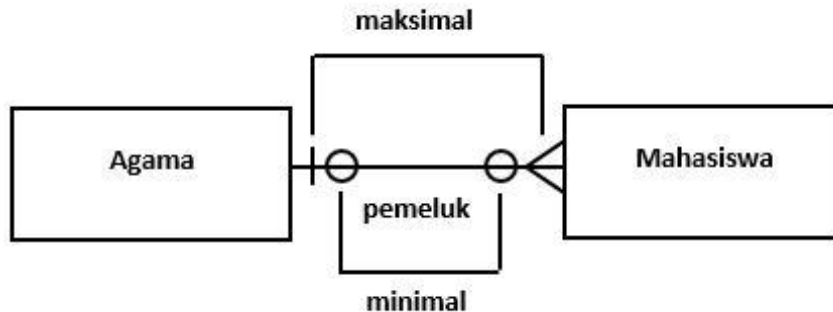
- satu agama dapat memiliki pemeluk minimal nol orang mahasiswa (minimal nol);
- satu agama dapat memiliki pemeluk satu atau banyak orang mahasiswa (maksimal satu atau banyak).

Derajat relasi entitas mahasiswa (di suatu perguruan tinggi) dengan entitas agama adalah zero or one. Alasannya, yaitu sebagai berikut:

- satu mahasiswa bisa jadi bukan pemeluk agama apapun (ateis), sehingga minimal derajat relasi minimal adalah nol;

- Satu mahasiswa maksimal memeluk satu agama, sehingga maksimal derajat relasi adalah satu.

Mohon diperhatikan, posisi penempatan minimal dan maksimal derajat relasi berikut ini!

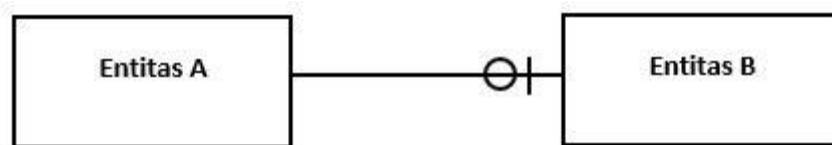


4) zero or one (nol atau satu).

Zero or one adalah derajat relasi yang terjadi antara satu entitas (misalnya entitas A) dan entitas lainnya (misalnya entitas B), dengan ketentuan sebagai berikut:

- anggota himpunan pada entitas A minimal dapat berpasangan dengan nol anggota himpunan pada entitas B;
- Anggota himpunan pada entitas A maksimal hanya dapat berpasangan dengan satu anggota himpunan pada entitas B.

Contoh

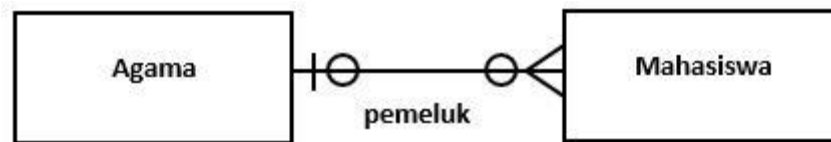


Penjelasan

Bila kita perhatikan notasi derajat relasi tersebut, terdapat lingkaran (disebut ring) dan garis lurus vertikal (disebut dash).

Posisi garis sebelah kiri menunjukkan jumlah minimum anggota entitas A yang dapat berpasangan dengan anggota entitas B (minimal nol), sedangkan posisi kaki burung gagak sebelah kanan, menunjukkan jumlah maksimum anggota entitas A yang dapat berpasangan dengan anggota entitas B (maksimal hanya satu).

Contoh lain



Penjelasan

Derajat relasi antara entitas penduduk dengan kartu tanda penduduk adalah zero or one.

Alasannya, yaitu sebagai berikut:

- satu orang penduduk yang belum berusia 17 tahun atau belum pernah mengurus pembuatan kartu tanda penduduk, maka dia belum memiliki kartu tanda penduduk (minimal nol);
- satu orang penduduk maksimal hanya memiliki satu kartu tanda penduduk (maksimal satu).

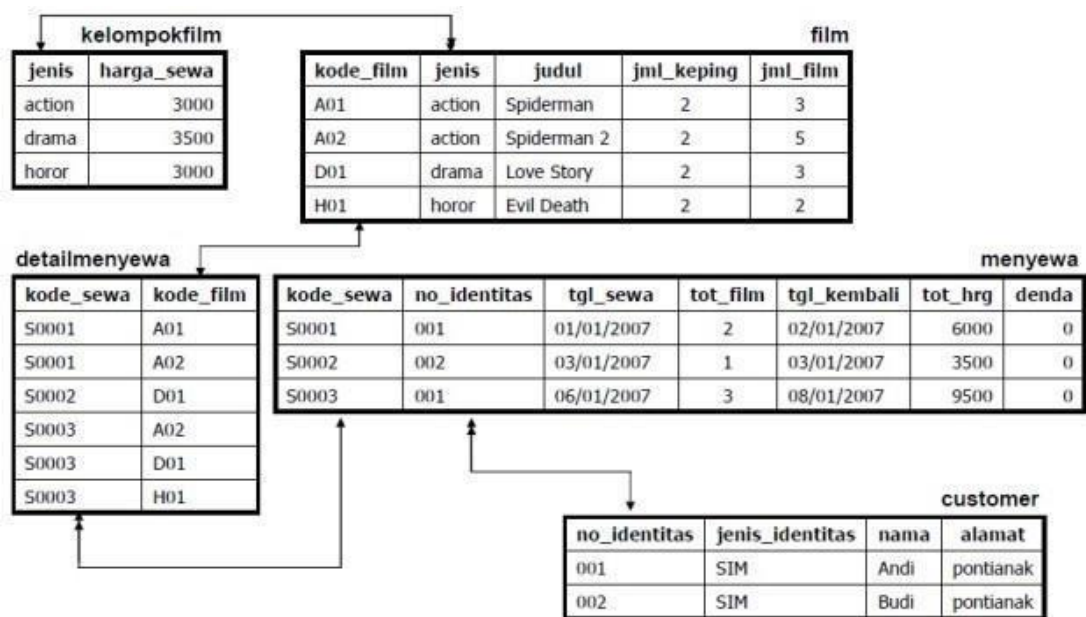
Derajat relasi antara kartu penduduk dengan penduduk adalah one and only one.

Alasannya, yaitu sebagai berikut:

- satu orang penduduk hanya memiliki satu dan satu-satunya kartu tanda penduduk;
- satu kartu tanda penduduk hanya dimiliki oleh satu dan satu-satunya orang penduduk;

Atribut

Atribut identik dengan istilah "Kolom Data" tetapi dapat menunjukkan fungsinya sebagai pembentuk karakteristik (sifat-sifat) yang melekat dalam sebuah tabel. Pada penerapan aturan normalisasi, bisa berdampak pada penghilangan atau penambahan kolom tertentu, atau bahkan dapat membentuk suatu tabel baru.



Contoh:

- Tabel kelompok film memiliki 2 atribut, yaitu : jenis dan harga_sewa
- Tabel film memiliki 5 atribut, yaitu : kode_film, jenis, judul, jml_keping, jml_film

Selain penamaan yang unik berdasarkan fungsinya di tiap tabel, atribut juga dapat dibedakan berdasarkan sejumlah pengelompokkan sbb:

- Atribut Kunci dan Atribut Deskriptif
- Atribut Sederhana (Simple Attribute) dan Atribut Komposit (Composite Attribute)
- Atribut Bernilai Tunggal (Single-Valued Attribute) dan Atribut Bernilai Banyak (Multi-Valued Attribute)
- Atribut Harus Bernilai (Mandatory Attribute) dan Atribut Nilai Null (Null Value Attribute)
- Atribut Turunan (Derived Attribute)

1. **Atribut Kunci**

Atribut kunci adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data dalam tabel secara unik. Dikatakan unik jika pada atribut yang dijadikan kunci tidak boleh ada baris data dengan nilai yang sama.

Dalam metode relasional, ada 6 kunci yaitu:

- Kunci Calon (Candidat Key)*

Kunci Calon adalah salah satu rangkaian yang mempunyai nilai unik untuk membedakan atau mengidentifikasi nilai-nilai kombinasi yang unik diantara semua kejadian yang spesifik dari entitas. Kunci calon ini tidak boleh berisi atribut dari tabel yang lain. Kombinasi dari atribut yang digunakan untuk mengidentifikasi secara unik rekor database tanpa data apapun yang asing. Setiap tabel dapat memiliki satu atau lebih dua kunci calon.

Contoh :

Tabel pegawai berisi atribut

- nip
- no_ktp
- nama
- tempat_lahir
- tanggal_lahir
- alamat
- kota

Kunci calon di sini adalah:

- nip
- no_ktp

b. Kunci Utama (Primary Key)

Kunci utama adalah kunci calon yang telah dipilih untuk mengidentifikasi setiap record secara unik. Kunci utama harus merupakan atribut yang benar-benar unik dan tidak boleh ada nilai NULL. Kunci utama adalah suatu nilai dalam basis data yang digunakan untuk mengidentifikasi suatu baris dalam table. Salah satu dari kunci calon dapat dipilih menjadi kunci utama dengan 3 kriteria sbb:

- Kunci tersebut lebih natural untuk dijadikan acuan
- Kunci tersebut lebih sederhana
- Kunci tersebut cukup unik

Nip dan no_ktp adalah kunci calon (Candidate Key) dan untuk kunci utama (Primary Key) adalah salah satu yang dipilih dari kunci calon. Misalnya nip dijadikan Primary Key, maka Primary Key nya adalah nip.

c. *Kunci Alternatif (Alternate Key)*

Kunci alternatif adalah kunci yang tidak terpilih. Misal: Dalam suatu entitas terdapat dua atribut yang bisa dijadikan sebagai kunci. Sementara yang boleh dijadikan kunci hanya satu, maka anda harus memilih salah satu. Atribut yang dipilih, disebut kunci utama. Sedangkan atribut yang tidak dipilih disebut dengan kunci alternatif.

Contoh :

Tabel pegawai berisi atribut

- nip
- no_ktp
- nama
- tempat_lahir
- tanggal_lahir
- alamat
- kota

Nip dan no_ktp adalah kunci calon (Candidate Key) dan untuk kunci utama (Primary Key) adalah salah satu yang dipilih dari kunci calon. Misalnya nip dijadikan Primary Key, maka kunci alternatif nya adalah no_ktp.

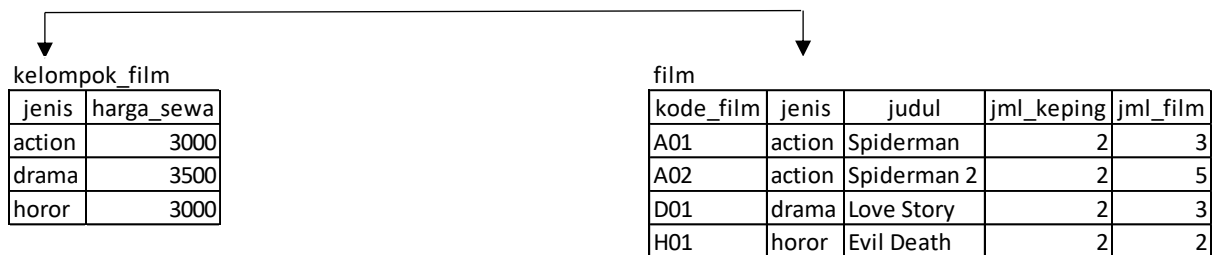
d. *Kunci Tamu (Foreign Key)*

Jika sebuah kunci utama terhubung ke tabel lain, maka keberadaan kunci utama tersebut disebut dengan kunci tamu. Kunci tamu adalah sebuah kumpulan atribut dalam satu relasi yang digunakan me"refer" (menunjuk) ke satu baris (tuple) pada relasi yang lain (harus berkorespondendi dengan kunci utama pada relasi yang kedua), seperti 'logical pointer'.

Sedangkan hubungan antara keduanya (kunci utama dan kunci tamu) dijelaskan sebagai berikut:

Kunci utama adalah atribut kunci dari suatu tabel yang menunjukkan bahwa atribut tersebut tidak bisa diisi dengan data yang sama, atau dengan kata lain kunci utama menjadikan setiap record memiliki identitas sendiri-sendiri yang membedakan satu sama lainnya (unik).

Kunci tamu adalah atribut yang melengkapi satu relationship yang menunjukkan ke induknya, dengan kata lain keduanya saling berkaitan.



Perhatikan gambar di atas, atribut jenis pada tabel kelompok_film adalah kunci utama karena bersifat unik. Pada tabel film juga terdapat atribut jenis, disebut sebagai kunci tamu karena digunakan me"refer" dengan atribut jenis yang terdapat pada tabel kelompok film.

Biasanya tabel yang berisi atribut kunci primer yang di"refer" oleh tabel lain sering disebut tabel induk. Contohnya pada gambar di atas, tabel kelompok_film merupakan tabel induk bagi tabel film. Sedangkan tabel yang mengandung kunci tamu yang me"refer" tabel lain disebut sebagai tabel anak. Dalam contoh di atas maka tabel film merupakan tabel bagi kelompok film.

e. Kunci Komposit (Composite Key)

Dalam desain basis data, kunci komposit adalah kunci yang terdiri dari 2 atau lebih atribut yang secara unik mengidentifikasi suatu kejadian entitas.

detail_menyewa	
kode_sewa	kode_film
S0001	A01
S0001	A02
S0002	D01
S0003	A02
S0003	D01
S0003	H01

Sebagai contoh pada tabel detail_menyewa dapat kita ketahui bahwa atribut kode_sewa maupun kode_film tidaklah bersifat unik karena merupakan kunci tamu dari tabel menyewa dan tabel film. Namun jika digabungkan (dikompositkan), yaitu kode_sewa + kode_film, maka diperoleh atribut komposit yang bersifat unik dimana tidak ada kode sewa yang sekaligus memilih kode_film yang sama.

f. *Kunci Sekunder (Secondary Key)*

Kunci sekunder adalah sebuah atribut atau kombinasi yang digunakan hanya untuk tujuan pengembalian data.

2. *Atribut Deskriptif*

Atribut deskriptif adalah atribut-atribut yang tidak menjadi atau merupakan anggota dari Primary Key. Jadi, dalam tabel mahasiswa yang menjadi atribut deskriptif adalah selain NIM.

3. *Atribut Sederhana (Simple Attribute)*

Atribut sederhana adalah atribut atomic yang tidak dapat dipilah lagi. Contoh atribut sederhana pada tabel customer adalah no_identitas dan jaminan, dimana atribut ini tidak bisa dipecah lagi.

4. *Atribut Komposit (Composite Attribute)*

Atribut komposit adalah atribut yang masih dapat diuraikan lagi menjadi sub-sub atribut yang masing-masing memiliki makna. Contoh pada tabel customer adalah atribut alamat, dimana dapat diuraikan lagi menjadi alamat, kota, dan kode_pos.

5. *Atribut Bernilai Tunggal (Single Valued Attribute)*

Atribut bernilai tunggal adalah atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data. Contoh: Bila seorang mahasiswa memiliki 2 tempat tinggal, maka 1 saja yang boleh diisikan ke atribut alamat_mhs.

6. *Atribut Bernilai Banyak (Multiple Valued Attribute)*

Atribut bernilai banyak adalah atribut-atribut yang dapat diisi dengan lebih dari satu nilai, tetapi jenisnya sama. Contoh: Atribut hobi pada data mahasiswa. Ada

mahasiswa yang punya banyak hobi, ada yang hanya satu hobi dan ada yang tidak sama sekali.

7. *Atribut Harus Bernilai (Mandatory Attribute)*

Atribut harus bernilai adalah atribut yang harus berisi data. Contoh : no_identitas dan nama_customer harus ada nilainya dalam tabel customer.

8. *Atribut Nilai Null (Non Mandatory Attribute)*

Atribut nilai null adalah atribut yang nilainya boleh dikosongkan. Dapat digunakan untuk menyatakan/mengisi atribut-atribut yang nilainya memang belum siap atau tidak ada. Nilai null tidak sama dengan spasi.

9. *Atribut Turunan*

Atribut turunan adalah atribut-atribut yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut tabel lain yang berhubungan. Dapat diiadakan dari sebuah tabel, karena nilainya bergantung pada nilai yang ada di atribut lain.

BAB III

DEPENDENSI DAN NORMALISASI

DEPENDENSI

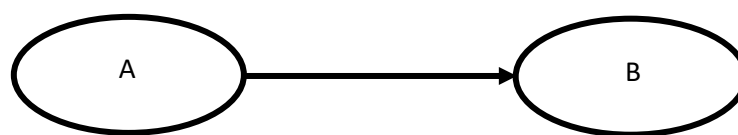
Dalam membuat basis data, kita pasti ingin basis data yang kita punya tidak ruwet dan mudah dalam memodifikasi data. Sehingga kita sebagai database administrator menjadi lebih mudah dalam mengelola basis data tersebut. Untuk mencapai hal tersebut maka kita perlu untuk mempelajari yang namanya Dependensi dan Normalisasi.

Arti umum dari dependensi adalah ketergantungan. Adapun macam-macam dependensi yaitu:

Dependensi Fungsional

Dependensi menggambarkan relasi/hubungan, batasan, dan keterkaitan antara atribut dalam suatu relasi. Suatu atribut dikatakan bergantung pada atribut lain secara fungsional jika kita menggunakan harga atribut lain.

Notasi dependensi fungsional $A \Rightarrow B$ (A secara fungsional menentukan B). artinya bahwa A secara fungsional menentukan B atau B bergantung pada A. Jika ada dua baris data dengan nilai A yang sama, maka nilai B juga sama.



B adalah fungsional dependensi pada A

Contoh:

MataKuliah	NIM	NamaMhs	Grade
Struktur Data	980001	Tiara Agustin	A
Struktur Data	980002	Seli Deslia	A
Basis Data	980001	Tiara Agustin	A
Basis Data	980003	Nushrotummillah Nurul 'Aini	B
Struktur Data	980003	Nushrotummillah Nurul 'Aini	A
Struktur Data	980004	Mita Zuriati	B

Berikut adalah dependensi yang ada di dalam tabel di atas:

- $NIM \Rightarrow NamaMhs$
Karena untuk setiap NIM yang sama, maka NamaMhs juga sama
- $\{MataKuliah, NIM\} \Rightarrow Grade$
Karena atribut Grade tergantung pada MataKuliah dan NIM yang sama, maka Grade juga sama.

Dependensi fungsional terdiri dari 3 jenis:

1) *Dependensi Penuh*

Dependensi penuh menunjukkan jika terdapat atribut A dan B dalam suatu relasi, dimana:

- B memiliki dependensi fungsional secara penuh terhadap A
- B bukan memiliki dependensi terhadap subset A

Contoh:

NIM	Nama	IdRuang
980001	Tiara Agustin	B31
980002	Seli Deslia	B33
980001	Tiara Agustin	B26
980003	Nushrotummillah Nurul 'Aini	B26
980003	Nushrotummillah Nurul 'Aini	B33
980004	Mita Zuriati	B31

- $\{NIM, Nama\} \Rightarrow IdRuang$ bukan dependensi penuh
- $NIM \Rightarrow IdRuang$ adalah dependensi penuh

2) *Dependensi Parsial*

Dependensi parsial atau ketergantungan sebagian merupakan ketergantungan fungsional dimana beberapa atribut dapat dihilangkan dari A dengan ketergantungan tersebut dipertahankan.

- B memiliki dependensi terhadap subset A

Contoh:

NIM	Nama	IdRuang
980001	Tiara Agustin	B31
980002	Seli Deslia	B33
980001	Tiara Agustin	B26
980003	Nushrotummillah Nurul 'Aini	B26
980003	Nushrotummillah Nurul 'Aini	B33
980004	Mita Zuriati	B31

- $\{NIM, Nama\} \Rightarrow IdRuang$ merupakan dependensi parsial, dimana jika dihilangkan maka ketergantungan tetap ada

3) *Dependensi Transitif*

Dependensi transitif adalah kondisi dimana A, B, C merupakan atribut sebuah relasi, dimana $A \Rightarrow B$ dan $B \Rightarrow C$

- C dikatakan dependensi transitif terhadap A melalui B

Contoh:

NIP	Nama	Kdcabang	Alamat
980001	Tiara Agustin	A1	Purwakarta
980002	Seli Deslia	A2	KBB
980003	Nushrotummillah Nurul 'Aini	A3	Boyolali
980004	Mita Zuriati	A4	Pekan Baru

- $NIP \Rightarrow \{Nama, KdCabang, Alamat\}$
- $KdCabang \Rightarrow Alamat$
- Maka Alamat bergantung transitif terhadap NIP melalui KdCabang

NORMALISASI

Normalisasi adalah suatu teknik untuk mengorganisasi data ke dalam tabel-tabel untuk memenuhi kebutuhan pemakai di dalam suatu organisasi.

Tujuan dari normalisasi:

- Untuk menghilangkan kerangkapan data
- Untuk mengurangi kompleksitas
- Untuk mempermudah pemodifikasian data

Proses normalisasi:

- Data diuraikan ke Dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke eberapa tingkat.
- Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel tersebut perlu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.

Tahapan normalisasi:

Bentuk Tidak Normal



Menghilangkan perulangan grup

Bentuk Normal Pertama (1NF)



Menghilangkan ketergantungan sebagian

Bentuk Normal Kedua (2NF)



Menghilangkan ketergantungan transitif

Bentuk Normal Ketiga (3NF)



Menghilangkan anomali-anomali hasil dari ketergantungan fungsional

Bentuk Normal Boyce-Codd (BCNF)



Menghilangkan ketergantungan multivalue

Bentuk Normal Keempat (4NF)



Menghilangkan anomaly-anomali yang tersisa

Bentuk Normal Kelima (5NF)

Bentuk Normal Kesatu (1NF)

Suatu relasi dikaakan sudah memenuhi bentuk normal kesatu bila setiap data bersifat atomik, yaitu setiap irisan baris dan kolom hanya mempunyai satu nilai data.

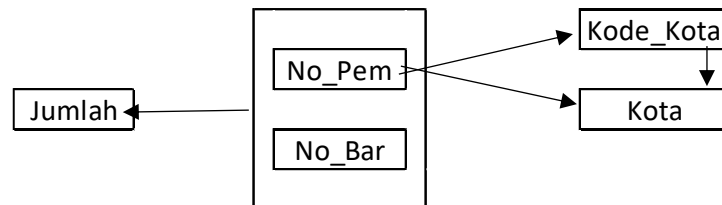
Tabel KIRIM-1 (Unnormal)

No_Pem	Kode_KotaKota	Kota	No_Bar	Jumlah
P01	1	Jakarta	B01	1000
			B02	1500
			B03	2000
P02	3	Bandung	B03	1000
P03	2	Surabaya	B02	2000

Tabel KIRIM-2 (1NF)

No_Pem	Kode_KotaKota	Kota	No_Bar	Jumlah
P01	1	Jakarta	B01	1000
P02	2	Jakarta	B02	1500
P03	3	Jakarta	B03	2000
P02	3	Bandung	B03	1000
P03	2	Surabaya	B02	2000

Digram Ketergantungan Fungsional



Bentuk Normal Kedua (2NF)

Suatu relasi dikatakan sudah memenuhi bentuk normal kedua bila relasi tersebut sudah memenuhi bentuk normal kesatu, dan atribut yang bukan key sudah tergantung penuh terhadap keynya.

Tabel PEMASOK-1 (2NF)

No_Pem	Kode_Kota	Kota
P01	1	Jakarta
P02	2	Bandung
P03	3	Surabaya

Bentuk Normal Ketiga (3NF)

Suatu relasi dikatakan sudah memenuhi bentuk normal ketiga bila relasi tersebut sudah memenuhi bentuk normal kedua dan atribut yang bukan key tidak tergantung transitif terhadap keynya.

No_Pem	Kode_KotaKota	Kota	No_Bar	Jumlah
P01	1	Jakarta	B01	1000
P02	2	Jakarta	B02	1500
P03	3	Jakarta	B03	2000
P02	3	Bandung	B03	1000
P03	2	Surabaya	B02	2000

Ketergantungan transitif:

- No_Pem -> Kode_Kota
- Kode_Kota -> Kota, maka
- No_Pem -> Kota

Tabel Kirim – 3 (2NF)

No_Pem	No_Bar	Jumlah
P01	B01	1000
P01	B02	1500
P01	B03	2000
P02	B03	1000
P03	B02	2000

Tabel Pemasok – 2 (2NF)

No_Pem	Kode_kota
P01	1
P02	3
P03	2

Tabel Pemasok – 3 (2NF)

Kode_Kota	Kota
1	Jakarta
2	Surabaya
3	Bandung

Contoh Lain :

Normalisasi pada database perkuliahan

Asumsi :

- Seorang mahasiswa dapat mengambil beberapa mata kuliah
- Satu mata kuliah dapat diambil oleh lebih dari satu mahasiswa
- Satu mata kuliah hanya diajarkan oleh satu dosen
- Satu dosen dapat mengajar beberapa mata kuliah
- Seorang mahasiswa pada mata kuliah tertentu hanya mempunyai satu nilai

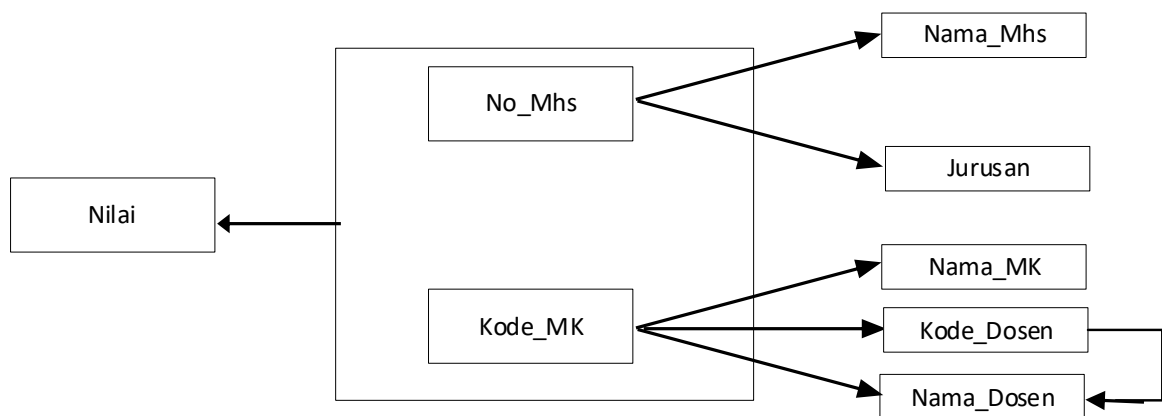
Tabel Mahasiswa – 1 (Unnormal)

No_Mhs	Nama_Mhs	Jurusan	Kode_MK	Nama_MK	Kode_Dosen	Nama_Dosen	Nilai
2683	Rere	MI	MI350	Menkeu	B104	Didi	A
			MI465	RPL	B317	Andra	B
5432	Setia	TI	AKN201	Akuntansi Keuangan	D310	Zafram	B
			KG210	Komputer Grafis	B343	Cahyana	C
			MI350	Menkeu	B104	Didi	A

Tabel Mahasiswa – 2 (1NF)

No_Mhs	Nama_Mhs	Jurusan	Kode_MK	Nama_MK	Kode_Dosen	Nama_Dosen	Nilai
2683	Rere	MI	MI350	Menkeu	B104	Didi	A
2683	Rere		MI465	RPL	B317	Andra	B
5432	Setia	TI	AKN201	Akuntansi Keuangan	D310	Zafram	B
5432	Setia	TI	KG210	Komputer Grafis	B343	Cahyana	C
5432	Setia	TI	MI350	Menkeu	B104	Didi	A

Diagram Ketergantungan Fungsional



Tabel Kuliah (2NF)

Kode_MK	Nama_MK	Kode_Dosen	Nama_Dosen
MI350	Mankeu	B104	Didi
MI465	RPL	B317	Andra
AKN201	Akuntansi Keuangan	D310	Zafram
KG210	Komputer Grafis	B343	Cahyana

Tabel Mahasiswa – 3 (2NF)

No_Mhs	Nama_Mhs	Jurusan
2683	Rere	MI
5432	Setia	TI

Tabel Nilai (2NF)

No_Mhs	Kode_MK	Nilai
2683	MI350	A
2683	MI465	B
5432	AKN201	B
5432	KG210	C
5432	MI350	A

Tabel Matakuliah (2NF)

Kode_MK	Nama_MK	Kode_Dosen
MI350	Mankeu	B104
MI465	RPL	B317
AKN201	Akuntansi Keuangan	D310
KG210	Komputer Grafis	B343

Tabel Dosen (2NF)

Kode_Dosen	Nama_Dosen
B104	Didi
B317	Zahra
D310	Zafram

BAB IV

DDL (DATA DEFINITION LANGUAGE)

Setelah kita mengenal DBMS untuk membuat database, kita tentu harus mengenal Bahasa yang akan kita gunakan untuk berkomunikasi dengan computer untuk membuat database tersebut. Dalam modul ini kita akan menggunakan sebuah Bahasa yang akan kita gunakan untuk berkomunikasi dengan computer melalui DBMS SQLyog, yang kita sebut dengan SQL (Structured Query Language) yang nanti akan kita panggil akrab dengan sebutan QUERY. Bahasa SQL/QUERY ini cukup manusiawi, di mana pemilihan kata-kata untuk sebuah perintah sangat mudah di ingat karena benar - benar menggunakan keyword - keyword dalam Bahasa inggris.

Nah setelah kita tahu Bahasa yang akan kita gunakan untuk berkomunikasi dengan computer via SQLyog, kita akan mengenal yang Namanya DDL (Data Definition Language). DDL adalah bagian dari Bahasa SQL/QUERY yang digunakan untuk mendefinisikan sebuah permodelan data, baik itu berupa Entitas, Atribut, ataupun sebuah Relasi.

Sekilas tentang Entitas, Atribut dan Relasi, dalam DLL ini agar untuk lebih cepat paham kita kiaskan Entitas itu sebagai nama Tabel dan Atribut sebagai Field.

Contoh:

Tabel_Mahasiswa -> Entitas (Tabel_Mahasiswa)

nama	nim
(NULL)	(NULL)

Atribut (Nama,Nim)/Field.

DDL (Data Definition Language) dibagi menjadi 3 perintah utama, yaitu:

- CREATE (Membuat),
- ALTER (Mengubah)

- DROP (Hapus).

Agar lebih dimengerti kita belajar DDL menggunakan studi kasus sebagai berikut.

Sebuah universitas yang bernama PASIM ingin membuat database untuk merekam data mahasiswa, matakuliah, dan data dosen yang mengajar matakuliah yang bersangkutan seperti umumnya sebuah universitas mempunyai fakultas dan jurusan masing-masing sekarang kita coba membuat database yang bernama db_PASIM yang di dalamnya ada entitas dengan nama mahasiswa, dosen, matakuliah, fakultas, jurusan.

Atribut untuk mahasiswa adalah id_mahasiswa (primary key), nim, nama, id_jurusan (foreign key). Atribut untuk matakuliah adalah id_matakuliah (primary key), nama_MK, Sks. Atribut untuk dosen adalah id_dosen (primary key), NID, nama_dosen. Atribut untuk fakultas adalah id_fakultas (primary key), nama_fakultas. Atribut untuk jurusan adalah id_jurusan (primary key), nama_jurusan, id_fakultas (foreign key). Dari semua atribut yang ada silahkan sesuaikan dengan type data masing-masing.

Query Create

Dari studi kasus diatas adalah beberapa pernyataan create yang akan kita buat Query nya. Langsung saja ke Langkah awal.

CREATE DATABASE nama_database

Contoh:

CREATE DATABASE db_pasim

Setelah kita membuat query diatas. Berarti kita telah membuat database yang bernama db_pub di server SQL.

Selanjutnya kita membuat tabel-tabel. Sebelum itu ada 1 query yang harus kita execute dulu. Agar tabel tabel yang akan kita buat nantinya akan disimpan di db_pub. Adapun query nya adalah

USE nama_database: berarti -> **USE** db_pasim

Pernyataan USE wajib kita execute Ketika kita akan menggunakan suatu database baik itu pada awal setelah kita membuat database. Maupun terhadap database yang kita buat sebelumnya. Karena dalam 1 server SQL terdiri dari lebih dari 1 database. Jadi kita harus memiliki dulu database mana yang akan kita gunakan.

Langkah selanjutnya kita membuat tabel di db_pasim query nya sebagai berikut.

```
CREATE TABLE nama_tabel (  
  
    Field1 Type_data (Panjang data),  
  
    Field2 Type_data (Panjang data),  
  
    Field3 Type_data (Panjang data)  
  
);
```

Contoh kita akan membuat tabel fakultas dan jurusan sesuai dengan ketentuan di atas.

```
CREATE TABLE Mst_fakultas (  
  
    Id_fakultas CHAR (3) NOT NULL,  
  
    Nama_fakultas VARCHAR (40) NOT NULL,  
  
    CONSTRAINT pk_fak PRIMARY KEY (id_fakultas)  
  
);
```

```
CREATE TABLE Mts_jurusan (  
  
    Id_jurusan CHAR (3) NOT NULL,  
  
    Nama_jurusan VARCHAR (50) NOT NULL,  
  
    Id_fakultas CHAR (3) NOT NULL,  
  
    CONSTRAINT pk_jur PRIMARY KEY (id_jurusan),  
  
    CONSTRAINT fk_jur FOREIGN KEY (id_fakultas) REFERENCES  
    Mts_fakultas (id_fakultas)  
  
);
```

Keterangan:

- **CREATE TABLE** -> Perintah /Query dasar untuk membuat tabel.
- **I NOT NULL** Artinya menyatakan bahwa field tersebut tidak boleh kosong atau tidak menerima nilai null/kosong.
- **CONSTRAINT** /batasan → suatu cara untuk memastikan integrasi dari database melalui perryaringan informasi yang dimasukan kedalam suatu baris atau kalom.

Query Alter

Query ALTER digunakan untuk mengubah struktur tabel baik itu nama tabel, nama field-field nya type datanya ataupun pajangnya. Selain itu juga bisa digunakan untuk menambahkan field baru dalam suatu tabel dan masih banyak lagi. Bentuk umum dari alter adalah sebagai berikut:

ALTER TABLE nama_tabel alter_option.

ALTER_OPTION adalah pilihan alter yang akan menentukan Tindakan, perubahan yang akan kita lakukan terhadap tabel yang bersangkutan.

- 1.1. **ADD** → Digunakan untuk menambah field baru pada suatu tabel.
Contoh : kita ingin menambahkan field tgl_lahir di Tb_mahasiswa.

ALTER TABLE Tb_Mahasiswa **ADD** Tgl_Lahir **DATE NOT NULL**:

- 1.2. **ADD PRIMARY KEY** → Digunakan untuk menjadikan suatu di tabel menjadi primary key. Contoh Tb Mahasiswa kita lupa untuk menjadikan NIM sebagai Primary key, maka kita gunakan ALTER dengan alter option ADD PRIMARY KEY untuk menjadikan NIM sebagai Primary key. Cara penulisanya adalah sebagai berikut:

ALTER TABLE Tb Mahasiswa **ADD CONSTRAINT** pk_mhs **PRIMARY KEY (NIM);**

- 1.3. **CHANGE** → Digunakan untuk mengubah nama dan type data, serta panjang suatu field. Contoh kita ingin mengubah field nama di Tb Mahasiswa menjadi Nama_Mhs dengan type varchar dan panjangnya jadi 100. Maka Query nya sebagai berikut:

```
ALTER TABLE Tb_Mahasiswa CHANGE nama NamaMhs VARCHAR  
(100) NOT NULL;
```

- 1.4. **MODIFY** → Digunakan untuk mengubah Type data ataupun panjang dari suatu field. Contoh kita ingin mengubah panjang data Nama_Mhs menjadi 50. Maka Querynya adalah sebagai berikut:

```
ALTER TABLE Tb_mahasiswa MODIFY Nama_Mhs VARCHAR (50) NOT  
NULL;
```

- 1.5. **DROP** Digunakan untuk menghapus suatu field dalam suatu tabel. Contoh kita ingin menghapus fiels Tgl_Lahir di tabel mahasiswa maka Querynya adalah sebagai berikut.

```
ALTER TABEL Tb_mahasiswa DROP Tgl_Lahir;
```

- 1.6. **RENAME TO** -> Digunakan untuk mengubah nama tabel. Contoh kita ingin mengubah nama Tb Mahasiswa menjadi Tb_Mhs maka Query-nya adalah sebagai berikut.

```
RENAME TABLE Tb_Mahasiswa TO Tb_Mhs;
```

Query Drop

Query DROP digunakan untuk menghapus suatu Database dan Tabel. untuk Menghapus suatu Data Base Query nya adalah sebagai berikut.

```
DROP DATABASE nama_database:
```

Contah kita ingin menghapus db_PASIM maka Query-nya adalah sebagai berikut.

```
DROP DATABASE db_PASIM:
```


Selain untuk menghapus Database DROP juga bisa digunakan untuk menghapus suatu tabel dalam database. Query-nya adalah sebagai berikut.

DROP TABLE nama_tabel;

Contoh kita ingin menghapus Tb_Mhs maka Query-nya adalah sebagai berikut.

DROP TABLE Tb_Mhs.

BAB V

DML (DATA MANIPULATION LANGUAGE)

Pengertian DML

DML merupakan kumpulan perintah query yang digunakan untuk memanipulasi data pada database. Misalnya digunakan untuk menambah data, merubah data, maupun menghapus data pada database.

Bisa diibaratkan DDL adalah yang membuat kerangkanya, sedangkan DML adalah yang mengubah isi dari kerangka tersebut.

Query INSERT

Query Insert Berfungsi untuk menyisipkan, memasukkan, atau menyimpan data kedalam database.

Ada 3 cara untuk mengisikan data ke dalam tabel, yaitu:

- **INSERT INTO** nama_tabel **VALUES** (value1, value2);

```
INSERT INTO mhs VALUES (1, 'Romi kusuma bakti', '02042011020');
```

- **INSERT INTO** nama_tabel (field1, field2) **VALUES** (value1, value2);

```
INSERT INTO tb_mahasiswa (id,nama,nim,) VALUES (2, 'Seli', '02042011025');
```

- **INSERT INTO** nama_tabel **SET** field1='value1', field2='value2';

```
INSERT INTO mhs SET id=3,nama='jamil',nim='02042011016';
```

Hasil dari Query insert seperti dibawah ini.

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	1	Romi Kusuma Bakti	02042011020
<input type="checkbox"/>	2	Seli	02042011025
<input type="checkbox"/>	3	jamil	02042011016

Query UPDATE

Query Update berfungsi untuk mengubah satu atau data field yang terdapat pada satu atau lebih record update digunakan jika ada data yang salah Ketika di insert.

UPDATE nama_tabel **SET** field='new value' **WHERE** kondisi

Contoh:

Merubah nama mahasiswa berdasarkan nimnya.

```
UPDATE tb_mahasiswa SET nama='Seli Deslia' WHERE nim='02042011025';
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	1	Romi Kusuma Bakti	02042011020
<input type="checkbox"/>	2	Seli Deslia	02042011025
<input type="checkbox"/>	3	jamil	02042011016

Maksudnya adalah merubah nama pada tabel tb_mahasiswa yang mempunyai nim='02042011025' yang Namanya 'Seli' menjadi 'Seli Deslia'.

Query DELETE

Query delete digunakan untuk menghapus data dalam tabel.

DELETE FROM nama_tabel **WHERE** kondisi;

Contoh:

1. Menghapus semua data dalam tabel.

```
DELETE FROM tb_mahasiswa;
```

2. Menghapus data berdasarkan nimnya.

```
DELETE FROM tb_mahasiswa WHERE nim='02042011016';
```

Namun, ada juga fungsi **TRUNCATE** yang digunakan untuk mengkosongkan isi tabel, sama saja dengan **DELETE FROM** nama_tabel.

Contoh:

```
TRUNCATE nama_tabel;
```

```
TRUNCATE tb_mahasiswa;
```

Hasil:

<input type="checkbox"/>	id	nama	nim
*	(NULL)	(NULL)	(NULL)

Query SELECT

Query select digunakan Ketika kita ingin menampilkan data dalam tabel.

1. Menampilkan semua isi tabel.

```
SELECT * FROM tb_mahasiswa;
```

2. Menampilkan nama saja

```
SELECT nama FROM tb_mahasiswa;
```

Hasil:

<input type="checkbox"/>	nama
<input type="checkbox"/>	Romi kusuma bakti
<input type="checkbox"/>	Seli Deslia
<input type="checkbox"/>	jamil

QUERY khusus yang biasanya digunakan dalam select.

a. **LIKE** (Seperti)

Perintah Like digunakan untuk menampilkan data tertentu yang hanya berkaitan dengan kata-kata yang digunakan.

```
SELECT * FROM nama_tabel WHERE nama_field LIKE kondisi;
```

- **LIKE %Kata%**, Menampilkan semua record yang mengandung 'Kata' pada satu field. Bisa diawal, ditengah ataupun dibelakang.
- **LIKE %Kata**, Menampilkan semua record yang mengandung 'kata' pada suatu field yang berada di akhir.
- **LIKE Kata%**, Menampilkan semua record yang mengandung 'kata' pada suatu field yang berada di awal.

Contoh:

```
SELECT * FROM tb_mahasiswa WHERE nama LIKE '%mi%';
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	1	Romi kusuma bakti	02042011020
<input type="checkbox"/>	3	jamil	02042011016

Query diatas maksudnya menampilkan semua data tabel Tb_mahasiswa berdasarkan nama yang mengandung kata 'mi'.

b. BETWEEN (Diantara)

Between berfungsi menampilkan data di antara dua kondisi dihubungkan dengan tanda penghubung AND.

```
SELECT * FROM tb_mahasiswa WHERE nim BETWEEN '02042011002' AND '02042011007';
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	2	Aris Purnama	02042011002
<input type="checkbox"/>	3	Edwar maulana	02042011003
<input type="checkbox"/>	4	Dani Hidayat	02042011004
<input type="checkbox"/>	5	Habib Jannata	02042011005
<input type="checkbox"/>	6	Jamil Hamdi	02042011006
<input type="checkbox"/>	7	Romi Kusuma Bakti	02042011007

c. AND atau &&

Menampilkan data jika kedua kondisi terpenuhi. Penulisannya bisa menggunakan AND atau &&.

```
SELECT * FROM tb_mahasiswa WHERE id='5' && nim='02042011005';
```

Atau

```
SELECT * FROM tb_mahasiswa WHERE id='5' AND nim='02042011005';
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	5	Habib Jannata	02042011005

d. OR atau ||

Menampilkan data dengan salah satu kondisi terpenuhi penulisannya bisa menggunakan OR atau ||

```
SELECT * FROM mhs WHERE id='5' OR nim='02042011008';
```

Atau

```
SELECT * FROM mhs WHERE id='5' || nim='02042011008';
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	5	Habib Jannata	02042011005
<input type="checkbox"/>	8	Yoga Hendra P	02042011008

e. ORDER BY

Order by digunakan untuk mengurutkan data berdasarkan field.

Bisa dengan ascending(asc), mengurutkan dari kecil ke besar, Descending (desc) mengurutkan dari besar ke kecil.

```
SELECT * FROM tb_mahasiswa ORDER BY nama ASC;
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	1	Ari sandi	02042011001
<input type="checkbox"/>	2	Aris Purnama	02042011002
<input type="checkbox"/>	4	Dani Hidayat	02042011004
<input type="checkbox"/>	3	Edwar maulana	02042011003
<input type="checkbox"/>	5	Habib Jannata	02042011005
<input type="checkbox"/>	6	Jamil Hamdi	02042011006
<input type="checkbox"/>	7	Romi Kusuma Bakti	02042011007
<input type="checkbox"/>	8	Yoga Hendra P	02042011008

f. LIMIT(BATAS)

Limit awal, jumlah_record;

Contoh:

Limit 2.5

2 artinya dimulai dari baris index ke 2, dan 5 artinya jumlah baris yang akan ditampilkan adalah 5 baris

```
SELECT * FROM tb_mahasiswa LIMIT 0,3;
```

Hasil:

<input type="checkbox"/>	id	nama	nim
<input type="checkbox"/>	1	Ari sandi	02042011001
<input type="checkbox"/>	2	Aris Purnama	02042011002
<input type="checkbox"/>	3	Edwar maulana	02042011003

Fungsi Agregat

a. GREATEST

Fungsi greatest digunakan untuk mencari nilai terbesar dari beberapa data.

```
SELECT GREATEST(1,2,4,10,3,15) AS terbesar;
```

Hasil:

<input type="checkbox"/>	terbesar
<input type="checkbox"/>	15

b. COUNT(RANGE)

Fungsi Count digunakan untuk menghitung jumlah record(baris) dalam tabel.

```
SELECT COUNT(*) AS jumlah FROM tb_mahasiswa;
```

Hasil:

<input type="checkbox"/>	jumlah
<input type="checkbox"/>	8

c. MAX (maksimal)

Fungsi MAX digunakan untuk mencari nilai terbesar dari suatu field dalam tabel.

```
SELECT MAX(nim) AS terbesar FROM tb_mahasiswa;
```

Hasil:

<input type="checkbox"/>	terbesar
<input type="checkbox"/>	02042011008

d. MIN (minimal)

Fungsi MIN digunakan untuk mencari nilai terkecil dari suatu field dalam tabel.

```
SELECT MIN(nim) AS terkecil FROM tb_mahasiswa;
```

Hasil:

<input type="checkbox"/>	terkecil
<input type="checkbox"/>	02042011001

e. SUM (RANGE)

Fungsi SUM digunakan untuk menjumlahkan nilai yang ada dalam field

Contoh:

stok_barang
0
27
162

```
SELECT SUM(stok_barang) jumlah FROM barang;
```

Hasil:

<input type="checkbox"/>	jumlah
<input type="checkbox"/>	189

f. AVG (Rata-Rata)

Fungsi AVG digunakan untuk mencari nilai rata-rata nilai dari suatu field.

Contoh:

harga_barang
300000
65000000
10000

```
SELECT AVG(harga_barang) rata_rata FROM barang;
```

Hasil:

rata_rata
21770000.0000

g. GROUP BY (mengelompokkan)

Fungsi GROUP BY digunakan untuk mengelompokkan data berdasarkan nama_fieldnya.

```
SELECT nama,COUNT(*) AS jumlah FROM mhs GROUP BY nim;
```

Hasil:

nama	jumlah
Ari sandi	1
Aris Purnama	1
Edwar maulana	1
Dani Hidayat	1
Habib Jannata	1
Jamil Hamdi	1
Romi Kusuma Bakti	1
Yoga Hendra P	1

h. HAVING (yang mempunyai)

Having itu layaknya **WHERE**, Namun perbedaannya adalah having digunakan saat akan menggunakan fungsi agregat.

```
SELECT nama,COUNT(*) AS jumlah FROM tb_mahasiswa GROUP BY nim HAVING SUM(id) > 3
```

Jika kita baca Query diatas kita menampilkan nama. Hasil jumlah dari seluruh baris dalam tabel tb_mahasiswa yang digrupkan berdasarkan nim yang jumlah id nya lebih besar dari 3.

Hasil:

<input type="checkbox"/>	nama	jumlah
<input type="checkbox"/>	Dani Hidayat	1
<input type="checkbox"/>	Habib Jannata	1
<input type="checkbox"/>	Jamil Hamdi	1
<input type="checkbox"/>	Romi Kusuma Bakti	1
<input type="checkbox"/>	Yoga Hendra P	1

Sub Select Dan Join

a. SUB SELECT

SUB SELECT atau biasa disebut select dalam select digunakan Ketika kita membutuhkan lebih dari satu tabel untuk menghasilkan suatu tampilan laporan.

<input type="checkbox"/>	id_jurusan	nama
<input type="checkbox"/>	1	D3 MI
<input type="checkbox"/>	2	S1 TI
<input type="checkbox"/>	3	S1 Psikologi
<input type="checkbox"/>	4	S1 Akuntansi
<input type="checkbox"/>	*	(NULL) (NULL)

<input type="checkbox"/>	id	nama	nim	id_jurusan
<input type="checkbox"/>	1	Ari sandi	02042011001	1
<input type="checkbox"/>	2	Aris Purnama	02042011002	1
<input type="checkbox"/>	3	Edwar maulana	02042011003	1
<input type="checkbox"/>	4	Dani Hidayat	02042011004	1
<input type="checkbox"/>	5	Habib Jannata	02042011005	2
<input type="checkbox"/>	6	Jamil Hamdi	02042011006	1
<input type="checkbox"/>	7	Romi Kusuma Bakti	02042011007	2
<input type="checkbox"/>	8	Yoga Hendra P	02042011008	3

Nah kita bisa menggabungkan kedua tabel dengan sub select atau join dengan memilih field yang sama. Sudah jelas field yang sama dari kedua kedua tabel tersebut yaitu id_jurusan. Field yang sama ini nantinya digunakan sebagai acuan relasinya.

```
SELECT nama,nim FROM mhs WHERE id_jurusan IN
(SELECT id_jurusan FROM jurusan WHERE nama='S1 Akuntansi');
```

Artinya kita pilih semua nama,nim di tb_mahasiswa yang id_jurusannya di (pilih id_jurusan yang nama jurusan S1 Akuntansi).

Perhatikan tabel mst_mahasiswa dan mst_jurusan berikut:

Tabel mst_mahasiswa

<input type="checkbox"/>	nim	nama	kode_jurusan
<input type="checkbox"/>	0204161012	Erik Sutiawan	05
<input type="checkbox"/>	0204161015	Fauzi Alamsyah	04
<input type="checkbox"/>	0204161036	Muhammad Zada Widiyanto	03
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz	01
<input type="checkbox"/>	0204161041	Resi Meliyanti	04
<input type="checkbox"/>	0204161042	Roihatul Jannah	01
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki	02
<input type="checkbox"/>	0204161046	Siti Kulsum Saadah	05
*	(NULL)	(NULL)	(NULL)

Tabel mst_jurusan

<input type="checkbox"/>	kode_jurusan	nama_jurusan	kode_fakultas
<input type="checkbox"/>	01	S1 Teknik Informatika	001
<input type="checkbox"/>	02	D3 Manajemen Informatika	001
<input type="checkbox"/>	03	S1 Sastra Jepang	002
<input type="checkbox"/>	04	D3 Bahasa Inggris	002
<input type="checkbox"/>	05	S1 Manajemen	003
<input type="checkbox"/>	06	S1 Akuntansi	003
<input type="checkbox"/>	07	S1 Psikologi	004
*	(NULL)	(NULL)	(NULL)

Bisa kita lihat pada tabel mst_mahasiswa terdapat field nim, nama dan kode_jurusan, dan pada tabel mst_jurusan terdapat field kode_jurusan, nama_jurusan, kode_fakultas. Nah untuk menggabungkan kedua tabel dengan sub select atau join adalah dengan memilih field yang sama. Sudah jelaskan field yang sama dari kedua tabel tersebut ialah kode_jurusan. Field yang sama ini nantinya digunakan untuk acuan relasinya.

Contoh 1:

**SELECT nim,nama FROM mst_mahasiswa WHERE kode_jurusan IN(SELECT kode_jurusan
FROM mst_jurusan WHERE nama_jurusan = 'S1 Teknik Informatika');**

Artinya pilih semua nim dan nama di tabel mst_mahasiswa yang kode_jurusannya di (pilih kode_jurusan yang nama jurusan S1 Teknik Informatika).

Query IN ini berarti di, dia mempunyai kebalikan yaitu NOT IN yang berarti tidak di

Hasil:

<input type="checkbox"/>	nim	nama
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz
<input type="checkbox"/>	0204161042	Roihatul Jannah

Contoh 2:

Misalnya kita ingin menampilkan nama mahasiswa yang berada di fakultas ilmu komputer. Maka kita harus menyambung ke tabel mst_fakultas. Bagaimana cara menyambungkannya ? Sedangkan tabel mst_mahasiswa dan mst_jurusan tidak memiliki filed yang sama.

Tabel mst_mahasiswa

<input type="checkbox"/>	nim	nama	kode_jurusan
<input type="checkbox"/>	0204161012	Erik Sutiawan	05
<input type="checkbox"/>	0204161015	Fauzi Alamsyah	04
<input type="checkbox"/>	0204161036	Muhammad Zada Widiyanto	03
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz	01
<input type="checkbox"/>	0204161041	Resi Meliyanti	04
<input type="checkbox"/>	0204161042	Roihatul Jannah	01
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki	02
<input type="checkbox"/>	0204161046	Siti Kulsum Saadah	05
*	(NULL)	(NULL)	(NULL)

Tabel mst_jurusan

<input type="checkbox"/>	kode_jurusan	nama_jurusan	kode_fakultas
<input type="checkbox"/>	01	S1 Teknik Informatika	001
<input type="checkbox"/>	02	D3 Manajemen Informatika	001
<input type="checkbox"/>	03	S1 Sastra Jepang	002
<input type="checkbox"/>	04	D3 Bahasa Inggris	002
<input type="checkbox"/>	05	S1 Manajemen	003
<input type="checkbox"/>	06	S1 Akuntansi	003
<input type="checkbox"/>	07	S1 Psikologi	004
<input type="checkbox"/>	* (NULL)	(NULL)	(NULL)

Tabel mst_fakultas

<input type="checkbox"/>	kode_fakultas	nama_fakultas
<input type="checkbox"/>	001	Ilmu Komputer
<input type="checkbox"/>	002	Sastra
<input type="checkbox"/>	003	Ekonomi
<input type="checkbox"/>	004	Psikologi
<input type="checkbox"/>	* (NULL)	(NULL)

Dari ketiga field tersebut, coba analisis nama field yang dapat dijadikan sebagai relasi.

Untuk menyambungkan antara tabel mst_mahasiswa dengan tabel mst_fakultas adalah dengan bantuan tabel mst_jurusan. Diatas sudah disebutkan field yang sama antara mst_mahasiswa dengan mst_jurusan adalah kode_jurusan. Nah field apakah yang sama antara mst_jurusan dengan mst_fakultas. Yapss bener banget fieldnya yaitu kode_fakultas.

Contoh:

```
SELECT nim,nama FROM mst_mahasiswa WHERE kode_jurusan IN (SELECT kode_jurusan  
FROM mst_jurusan WHERE kode_fakultas IN (SELECT kode_fakultas FROM mst_fakultas  
WHERE nama_fakultas = 'Ilmu Komputer'));
```

Artinya : pilih semua nim, nama yang ada di tabel mst_mahasiswa yang kode_jurusannya di (pilih semua kode_jurusan yang ada di tabel mst_jurusan yang kode_fakultasnya di (pilih semua kode_fakultas di tabel mst_fakultas yang nama fakultasnya = 'Ilmu Komputer')).

Hasil:

<input type="checkbox"/>	nim	nama
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz
<input type="checkbox"/>	0204161042	Roihatul Jannah
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki

IN juga dapat diganti dengan =

SELECT nama_jurusan FROM mst_jurusan WHERE kode_fakultas = (SELECT kode_fakultas FROM mst_fakultas WHERE nama_fakultas = 'Sastra');

Hasil:

<input type="checkbox"/>	nama_jurusan
<input type="checkbox"/>	S1 Sastra Jepang
<input type="checkbox"/>	D3 Bahasa Inggris

b. JOIN

Untuk menggabungkan 2 (dua) atau lebih tabel, kita dapat menggabungkan bentuk perintah JOIN.

1. Inner Join

Dengan inner join, tabel akan digabungkan dua arah, sehingga tidak ada data yang NULL di satu sisi.

➤ Penggabungan dengan WHERE

Penggabungan dengan klausa WHERE memiliki bentuk umum sebagai berikut :

SELECT tabel1.*, tabel2.*, FROM tabel1, tabel2 WHERE tabel1.PK = tabel2. FK;

Contoh :

SELECT mst_mahasiswa.nama FROM mst_mahasiswa, mst_jurusan WHERE mst_mahasiswa.kode_jurusan = mst_jurusan.kode_jurusan AND mst_jurusan.nama_jurusan = 'S1 Manajemen';

Hasil:

<input type="checkbox"/>	nama
<input type="checkbox"/>	Erik Sutiawan
<input type="checkbox"/>	Siti Kulsum Saadah

Bisa dilihat data tersebut di akses oleh dua tabel yaitu tabel mst_mahasiswa dan mst_jurusan. Karena dua tabel maka penulisan nama pun harus disertai dengan asal tabel. Mengapa? Karena jika tidak, SQL akan bingung memilih nama di tabel yang mana. Kondisi WHERE diambil dari data yang sama, dalam hal ini data yang sama dari kedua tabel adalah kode jurusan. Dan AND digunakan untuk kondisi yang mencari nama jurusannya 'S1 Manajemen'.

Selain itu penulisannya juga dapat di persingkat dengan mengaliaskan nama tabelnya seperti ini :

```
SELECT m.nama FROM mst_mahasiswa m, mst_jurusan j WHERE  
m.`kode_jurusan` = j.`kode_jurusan` AND j.`nama_jurusan` = 'S1 Manajemen';
```

ATAU

```
SELECT m.nama FROM mst_mahasiswa as m, mst_jurusan as j WHERE  
m.`kode_jurusan` = j.`kode_jurusan` AND j.`nama_jurusan` = 'S1 Manajemen';
```

Hasil:

<input type="checkbox"/>	nama
<input type="checkbox"/>	Erik Sutiawan
<input type="checkbox"/>	Siti Kulsum Saadah

➤ **Penggabungan dengan INNER JOIN**

Menggabungkan dua tabel dimana diantara dua tabel datanya bersesuaian. Berikut ini bentuk Umumnya :

```
SELECT tabel1.* , tabel2.* FROM tabel1 INNER JOIN tabel2 ON tabel1.PK =  
tabel2.FK;
```

Contoh :

Misalnya kita ingin menampilkan nim, nama mahasiswa beserta nama jurusan.

```
SELECT m.nim, m.nama, j.nama_jurusan FROM mst_mahasiswa m INNER JOIN  
mst_jurusan j ON m.`kode_jurusan` = j.`kode_jurusan`;
```

Hasil :

<input type="checkbox"/>	nim	nama	nama_jurusan
<input type="checkbox"/>	0204161012	Erik Sutiawan	S1 Manajemen
<input type="checkbox"/>	0204161015	Fauzi Alamsyah	D3 Bahasa Inggris
<input type="checkbox"/>	0204161036	Muhammad Zada Widiyanto	S1 Sastra Jepang
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz	S1 Teknik Informatika
<input type="checkbox"/>	0204161041	Resi Meliyanti	D3 Bahasa Inggris
<input type="checkbox"/>	0204161042	Roihatul Jannah	S1 Teknik Informatika
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki	D3 Manajemen Informatika
<input type="checkbox"/>	0204161046	Siti Kulsum Saadah	S1 Manajemen

Atau misal kita akan menampilkan nim, nama mahasiswa dan nama jurusan yang beradadi jurusan D3 Bahasa Inggris

```
SELECT m.nim, m.nama, j.nama_jurusan FROM mst_mahasiswa m INNER JOIN  
mst_jurusan j ON m.`kode_jurusan`=j.`kode_jurusan`  
WHERE j.`nama_jurusan` = 'D3 Bahasa Inggris';
```

Hasil :

<input type="checkbox"/>	nim	nama	nama_jurusan
<input type="checkbox"/>	0204161015	Fauzi Alamsyah	D3 Bahasa Inggris
<input type="checkbox"/>	0204161041	Resi Meliyanti	D3 Bahasa Inggris

Perbedaan penggabungan menggunakan WHERE dan INNER JOIN

Jika kita menggunakan INNER JOIN,
pembandingan kode jurusan menggunakan ON,
dan kondisi nama jurusan menggunakan WHERE
seperti biasa

Menggabungkan dari 3 tabel:

Misalnya kita ingin menampilkan nim, nama mahasiswa, nama jurusan dan nama fakultas yang berada di fakultas ilmu komputer.

```
SELECT m.nim, m.nama, j.nama_jurusan, f.nama_fakultas FROM mst_mahasiswa  
m INNER JOIN mst_jurusan j INNER JOIN
```


mst_fakultas f ON m.`kode_jurusan` = j.`kode_jurusan` AND j.`kode_fakultas` = f.`kode_fakultas` WHERE f.`nama_fakultas` = 'Ilmu Komputer';

Hasil :

<input type="checkbox"/>	nim	nama	nama_jurusan	nama_fakultas
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz	S1 Teknik Informatika	Ilmu Komputer
<input type="checkbox"/>	0204161042	Roihatul Jannah	S1 Teknik Informatika	Ilmu Komputer
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki	D3 Manajemen Informatika	Ilmu Komputer

Jika kita ingin menjoin 3 tabel maka penulisannya bisa seperti query diatas. Jangan lupa ON nya pun harus ada yang menyambungkan tabel mahasiswa dengan tabel Jurusan, tabel Jurusan dengan tabel Fakultas.

Perbedaan dari menggunakan WHERE atau menggunakan INNER JOIN adalah pada WHERE tidak menggunakan ON, sedangkan INNER JOIN menggunakan ON

➤ **Penggabungan dengan OUTER JOIN**

Dengan outer join, tabel akan digabungkan satu arah, sehingga memungkinkan ada data yang **NULL** (kosong) disatu sisi. Outer Join terbagi menjadi 2 (dua) yaitu LEFT JOIN dan RIGHT JOIN.

a. LEFT JOIN atau LEFT OUTER JOIN

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

Bentuk umum:

SELECT tabel1.* , tabel2.* FROM tabel1 LEFT JOIN tabel2 ON tabel1.PK = tabel2.FK;

Contoh :

SELECT m.nim, m.nama, j.nama_jurusan FROM mst_mahasiswa m LEFT JOIN mst_jurusan j ON m.`kode_jurusan` = j.`kode_jurusan`;

Diquery tersebut yang disebelah kiri tulisan LEFT JOIN adalah tabel mst_mahasiswa. Jadi hasilnya akan menampilkan semua yang ada di mst_mahasiswa.

Hasil :

<input type="checkbox"/> nim	nama	nama_jurusan
<input type="checkbox"/> 0204161012	Erik Sutiawan	S1 Manajemen
<input type="checkbox"/> 0204161015	Fauzi Alamsyah	D3 Bahasa Inggris
<input type="checkbox"/> 0204161036	Muhammad Zada Widiyanto	S1 Sastra Jepang
<input type="checkbox"/> 0204161037	Nispi Abdul Aziz	S1 Teknik Informatika
<input type="checkbox"/> 0204161041	Resi Meliyanti	D3 Bahasa Inggris
<input type="checkbox"/> 0204161042	Roihatul Jannah	S1 Teknik Informatika
<input type="checkbox"/> 0204161044	Sahdan Hidayatul Muzaki	D3 Manajemen Informatika
<input type="checkbox"/> 0204161046	Siti Kulsum Saadah	S1 Manajemen

Atau sebaliknya yang disebelah kiri tulisan LEFT JOIN adalah tabel mst_jurusan. Maka hasilnya :

<input type="checkbox"/> nim	nama	nama_jurusan
<input type="checkbox"/> 0204161037	Nispi Abdul Aziz	S1 Teknik Informatika
<input type="checkbox"/> 0204161042	Roihatul Jannah	S1 Teknik Informatika
<input type="checkbox"/> 0204161044	Sahdan Hidayatul Muzaki	D3 Manajemen Informatika
<input type="checkbox"/> 0204161036	Muhammad Zada Widiyanto	S1 Sastra Jepang
<input type="checkbox"/> 0204161015	Fauzi Alamsyah	D3 Bahasa Inggris
<input type="checkbox"/> 0204161041	Resi Meliyanti	D3 Bahasa Inggris
<input type="checkbox"/> 0204161012	Erik Sutiawan	S1 Manajemen
<input type="checkbox"/> 0204161046	Siti Kulsum Saadah	S1 Manajemen
<input type="checkbox"/> (NULL)	(NULL)	S1 Akuntansi
<input type="checkbox"/> (NULL)	(NULL)	S1 Psikologi

Query tersebut akan menampilkan semua data yang berada di sebelah kiri tulisan LEFT JOIN tidak peduli ada atau tidaknya relasi dengan tabel yang direlasikan. Jika di tabel yang direlasikan tidak ada datanya maka akan di tampilkan NULL.

INGAT

LEFT JOIN artinya YANG DISEBELAH KIRI

**Tak peduli siapapun dia asal dia
DISEBELAH KIRI, itu yang di tampilkan**

b. **RIGHT JOIN atau RIGHT OUTER JOIN**

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

Bentuk Umum :

**SELECT tabel1.* , tabel2.* FROM tabel1 RIGHT JOIN tabel2 ON tabel1.PK =
tabel2.FK;**

Contoh :

```
SELECT m.nim, m.nama, j.nama_jurusan FROM mst_mahasiswa m RIGHT  
JOIN mst_jurusan j ON m.`kode_jurusan` = j.`kode_jurusan`;
```

Hasil :

<input type="checkbox"/>	nim	nama	nama_jurusan
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz	S1 Teknik Informatika
<input type="checkbox"/>	0204161042	Roihatul Jannah	S1 Teknik Informatika
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki	D3 Manajemen Informatika
<input type="checkbox"/>	0204161036	Muhammad Zada Widiyanto	S1 Sastra Jepang
<input type="checkbox"/>	0204161015	Fauzi Alamsyah	D3 Bahasa Inggris
<input type="checkbox"/>	0204161041	Resi Meliyanti	D3 Bahasa Inggris
<input type="checkbox"/>	0204161012	Erik Sutiawan	S1 Manajemen
<input type="checkbox"/>	0204161046	Siti Kulsum Saadah	S1 Manajemen
<input type="checkbox"/>	(NULL)	(NULL)	S1 Akuntansi
<input type="checkbox"/>	(NULL)	(NULL)	S1 Psikologi

Bisa kita lihat bahwa yang tampil adalah semua data yang ada di tabel sebelah kanan. Maksudnya adalah kanannya RIGHT JOIN.

```
SELECT m.nim, m.nama, j.nama_jurusan FROM mst_mahasiswa m RIGHT  
JOIN mst_jurusan j ON m.`kode_jurusan` = j.`kode_jurusan`;
```

Disini tabel yang di sebelah kanannya RIGHT JOIN adalah mst_jurusan. Jadi query akan menampilkan semua data yang ada di tabel mst_jurusan meskipun di tabel mst_mahasiswa tidak ada datanya. Maka akan menampilkan NULL, seperti gambar pada tabel diatas.

Jika penempatan penulisan tabel nya berbeda maka hasilnya juga akan berbeda.

Contoh :

```
SELECT m.nim, m.nama, j.nama_jurusan FROM mst_jurusan j RIGHT JOIN  
mst_mahasiswa m ON m.`kode_jurusan` = j.`kode_jurusan`;
```

Hasil :

<input type="checkbox"/>	nim	nama	nama_jurusan
<input type="checkbox"/>	0204161012	Erik Sutiawan	S1 Manajemen
<input type="checkbox"/>	0204161015	Fauzi Alamsyah	D3 Bahasa Inggris
<input type="checkbox"/>	0204161036	Muhammad Zada Widiyanto	S1 Sastra Jepang
<input type="checkbox"/>	0204161037	Nispi Abdul Aziz	S1 Teknik Informatika
<input type="checkbox"/>	0204161041	Resi Meliyanti	D3 Bahasa Inggris
<input type="checkbox"/>	0204161042	Roihatul Jannah	S1 Teknik Informatika
<input type="checkbox"/>	0204161044	Sahdan Hidayatul Muzaki	D3 Manajemen Informatika
<input type="checkbox"/>	0204161046	Siti Kulsum Saadah	S1 Manajemen

Tabel akan menampilkan semua data di sebelah kanan.

**RIGHT JOIN artinya YANG SEBELAH
KANAN**

**Tak peduli siapapun dia asal dia
DISEBELAH KANAN, itu yang ditampilkan**

INGAT

BAB VI

TRIGGER, FUNCTION DAN STORE PROCEDURE

Trigger

Trigger adalah sebuah script MySQL Command yang memicu suatu kejadian dalam database MySQL berupa aksi insert, update dan delete setelah syarat tertentu. Lokasi penulisan Trigger adalah di dalam database yang bersangkutan, dan Trigger tidak di tempatkan di php script. Hasil penulisan dari sebuah Trigger akan menimbulkan efek manipulasi database secara otomatis sesuai dengan yang kita inginkan. Misalnya, setelah insert pada tabel_a dan kolom a1 maka akan otomatis insert pada tabel_b pada kolom b1. Begitu pula untuk aksi update dan delete.

Contoh kasus dalam trigger, misalkan kita mempunyai database pengolahan data barang yang mempunyai 3 tabel, yaitu : tabel pembelian, penjualan dan tabel stock. Ketika membeli barang, kita akan memasukkan data ke tabel pembelian dan seharusnya tabel stock pun terupdate menjadi bertambah. Begitu juga dengan kita menjual barang maka otomatis kita akan menginputkan data barang di tabel penjualan dan secara otomatis di tabel stock akan berkurang nah didalam kasus ini kita membutuhkan yang namanya trigger.

Cara pembuatan Trigger sebagai berikut :

```
DELIMITER $$ atau DELIMITER //  
CREATE TRIGGER nama_trigger  
AFTER/BEFORE INSERT/DELETE/UPDATE  
ON nama_tabel (yang dikenai trigger)  
FOR EACH ROW BEGIN  
    Query yang diperlukan untuk menindak lanjuti  
trigger  
END;
```

\$\$

DELIMITER;

Contoh kita akan menambah/mengupdate tabel stock Ketika terjadi pembelian atau penambahan data di tabel pembelian. Catatan di tabel pembelian ada ID_barang, jumlah_beli. Di tabel stock ada ID_barang, jumlah_persediaan.

Maka Triggernya sebagai berikut :

DELIMITER \$\$

CREATE TRIGGER tambah_stock

AFTER INSERT

ON pembelian

FOR EACH ROW BEGIN

UPDATE stock SET jumlah_persediaan =

Jumlah_persediaan + new.jumlah_beli WHERE

ID_barang = new.ID_barang;

END;

\$\$

DELIMITER;

Keterangan :

- DELIMITER (adalah untuk memberi tahu kepada mysql soal delimiter yang digunakan, secara default menggunakan (;) jadi bila ada tanda (;) mysql akan mengartikan akhir dari statement, pada contoh diatas delimiter yang digunakan \$\$ jadi akhir statementnya adalah \$\$)
- CREATE TRIGGER (menandakan pembuatan trigger)
- AFTER/BEFORE INSERT/DELETE/UPDATE (kondisi yang dibutuhkan dalam trigger)
- FOR EACH ROW BEGIN (untuk menghitung kolom dalam tabel)
- END (tanda untuk mengakhiri pembuatan trigger)

Stored Procedure

Stored Procedure adalah salah satu objek routine yang tersimpan pada database MySQL dan dapat digunakan untuk menggantikan berbagai kumpulan perintah yang sering kita gunakan, seperti misalkan sejumlah row ke table lain dengan filter tertentu.

Stored Procedure sangat berguna Ketika kita tidak ingin user mengakses table secara langsung, atau dengan kata lain membatasi hak akses user dan mencatat operasi yang dilakukan. Dengan demikian resiko kebocoran dan kerusakan data dapat lebih diminimalisir.

Untuk melakukan retrieving suatu data dalam table, kita bisa menggunakan stored procedure. Stored Procedure ini sebaiknya digunakan apabila database server terpisah secara fisik dengan aplikasi atau disebut aplikasi Multi Tier. Dengan menggunakan stored procedure SQL tidak akan melakukan loading semua table yang ter-relasi, tetapi langsung melakukan filtering berdasarkan query yang kita maksud. Stored Procedure menyimpan statement-statement SQL dalam sebuah berkas yang disimpan di database server, sehingga dari sisi performa eksekusi, utilitas jaringan, dan keamanan, stored procedure banyak dipakai sebagai solusi akses data.

Setiap kali Query Processing menjalankan query, gambaran prosesnya sbb :

- a. Pengecekan Syntax
- b. Pemilihan execution plan yang optimal
- c. Eksekusi query

Query yang ada di SP sudah di-compile terlebih dahulu, jadi ada 1 step yang di-skip pada SP. Compile maksudnya adalah pemilihan mana execution plan yang paling optimal.

Keuntungan menggunakan Stored Procedure :

- Stored Procedure lebih fleksibel karena ada parameter didalamnya.
- Proses dilakukan di Database Server sehingga lebih cepat, aplikasi cukup memanggil stored procedure dan mengirim parameter yang diperlukan.

- Untuk membagi beban resource yang terpakai saat aplikasi dijalankan. Jika semua query dijalankan pada aplikasi/client (Front-End) maka resource yang terpakai pada client tersebut akan besar, oleh karena itu perintah query tersebut dibuat pada stored procedure (eksekusi pada server).
- Untuk mempermudah maintainance aplikasi. Apabila ada proses query yang sama dan berulang, dengan stored procedure akan lebih simple dalam proses pembuatan aplikasi.
- Mendukung ANSI model terhadap database. ANSI model adalah sebuah model database yang memodelkan penglihatan user terhadap database menjadi 3 komponen yaitu user view, logical view dan fisik view.
- Stored Procedure mencegah terjadinya SQL injection.

Kekurangan :

Apabila ingin mengganti Database Server, misalnya dari Oracle ke SQL Server, Porting Stored Procedurenya menyulitkan, antara PL SQL ke T-SQL. Berbeda jika embedded di Aplikasi, kita cukup mengganti Koneksinya karena Logiknya dilakukan di Aplikasi dan menggunakan SQL Standard.

Cara pembuatan Stored Procedure :

```
DELIMITER // atau DELIMITER $$
CREATE PROCEDURE nama_procedure()
BEGIN
    Query yang dibutuhkan;
END //
DELIMITER;
```

Contoh pemanggilan Stored Procedure :

```
Call nama_stored_procedure();
```


Contoh :

Call nama_mhs();

Keterangan :

- Jika diawali dengan DELIMITER // di akhirnya juga harus memakai // DELIMITER atau jika menggunakan DELIMITED \$\$ harus diakhirinya juga dengan \$\$ DELIMITER
- Body sql dimulai dengan BEGIN dan diakhiri dengan END

Variabel di Stored Procedure

Variable digunakan untuk menyimpan procedure ke penyimpanan hasil dengan segera.

Contoh :

DECLARE nama_variabel tipe data (ukuran) DEFAULT

nilai_default;

Contoh :

DECLARE total_sales INT(11) DEFAULT

Function

Function (fungsi) hampir sama dengan stored procedure. Hanya saja fungsi mempunyai sejumlah parameter input dan hanya mengembalikan 1 output.

Contoh penggunaan :

Struktur tabel siswa

```

CREATE TABLE data_siswa . tbl_siswa (
    nis char(10) NOT NULL,
    nama varchar(255) NOT NULL,
    kelas INT(11) NOT NULL)
ENGINE = MyISAM DEFAULT CHARSET = latin1;

```

Function untuk mengembalikan jumlah data dari setiap kelas

```

DELIMITER $$
CREATE FUNCTION sf_tampil_siswa_kelas(p_kelas int)
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE jml INT;
    SELECT COUNT(*) AS jml_kelas INTO jml FROM
    tb_siswa WHERE kelas = p_kelas;
    RETURN jml;
END $$
DELIMITER;

```

Penjelasannya sebagai berikut :

- DELIMITER = adalah untuk memberitahu kepada mysql soal delimiter yang digunakan, secara default menggunakan (;) jadi bila ada tanda (;) mysql akan mengartikan akhir dari statement, pada contoh di atas delimiter yang digunakan \$\$ jadi akhir dari statementnya adalah \$\$
- CREATE FUNCTION = adalah header untuk membuat function
- RETURNS = adalah untuk menentukan tipe data yang direturn-kan oleh function

- DETERMINISTIC / NOT DETERMINISTIC = adalah untuk menentukan yang bisa menggunakan function ini adalah user pembuatnya saja (deterministic) atau user siapa saja (not deterministic)
- BEGIN END = adalah body dari function jadi semua SQL nya ditulis disini

Contoh Pemanggilannya seperti dibawah ini :

SELECT sf_tampil_siswa_kelas("2");

Sebuah function hanya bisa memberikan return berupa nilai saja dan tidak bisa berupa resultset

Untuk penulisan DETERMINISTIC bisa ditulis secara implisit dengan memberikan setting global pada mysql dan secara default bernilai NOT DETERMINISTIC, cara nya dibawah ini :

SET GLOBAL log_bin_trust_function_creators = 1;