Memahami Konsep Database

Yang di bahas pada bab ini :

- **Abstraksi Data**
- **Mode Database Relasi**
- **Model Data Entity Relationship**
- **Diagram Entity Relationship (*Diagram E-R*)**
- **Tahapan Membuat Diagram E-R**

***Database Management System*(DBMS)** merupakan paket program (*Software*) yang dibuat agar memudahkan dan mengefisienkan pemasukan, pengeditan, penghapusan dan pengambilan informasi terhadap database. Software yang tergolong kedalam DBMS antara lain, Microsoft SQL, MySQL, Oracle, MS. Access, dan lain-lain

2.1 Abstraksi Data

Abstraksi data merupakan tingkatan-tingkatan pengguna dalam memandang bagaimana sebenarnya data diolah dalam sebuah sistem database sehingga menyerupai kondisi yang sebenarnya dihadapi oleh pengguna sehari-hari. Sebuah DBMS seringkali menyembunyikan detail tentang bagaimana sebuah data disimpan dan dipelihara (diolah) dalam sebuah sistem database, dengan tujuan untuk memudahkan pengguna dalam menggunakan DBMS tersebut. Karena itu seringkali data yang terlihat oleh pemakai sebelumnya berbeda dengan yang tersimpan secara fisik.

Ada 3 (tiga) tingkatan atau level dalam abstraksi data ini :

1. **Level Fisik (*Physical Level*)**

   Level abstraksi data yang paling rendah, yang menggambarkan bagaimana (***how***) data disimpan dalam kondisi sebenarnya. Level ini sangat kompleks karena struktur data dijelaskan secara rinci.

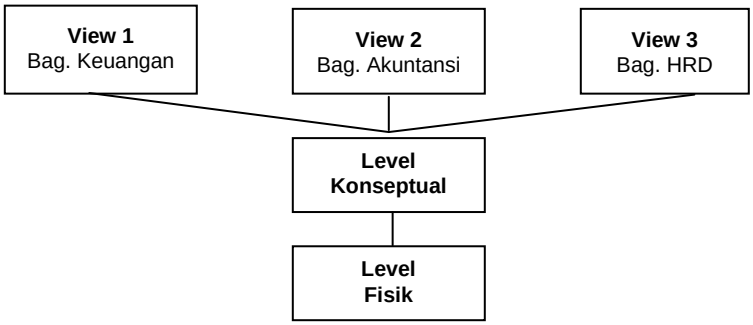
2. **Level Konseptual (*Conceptual Level*)**

   Level ini menggambarkan data apa (***what***) yang disimpan dalam database dan menjelaskan bagaimana hubungan antar datanya secara keseluruhan. Seorang pengguna dalam level ini dapat mengetahui bahwa data mahasiswa disimpan pada tabel mahasiswa, tabel krs, tabel transkrip dan lain sebagainya. Level ini biasa di pakai oleh seorang **Database Administrator** (DBA).

3. Level Pandangan (*View Level*)

Ini merupakan level yang tertinggi, hanya menggambarkan sebagian saja dari keseluruhan database sesuai dengan kebutuhan pengguna. Misalnya : Bagian keuangan hanya membutuhkan data keuangan, jadi yang digambarkan hanya pandangan terhadap data keuangan saja, begitu juga dengan bagian akuntansi, hanya membutuhkan data akuntansi saja. Jadi tidak semua pengguna database membutuhkan seluruh informasi yang terdapat dalam database tersebut.

Hubungan antara ketiga level tersebut, dapat digambarkan sebagai berikut :



Gambar 2.1 : Hubungan level abstraksi data

2.2 Model Database

Model database adalah suatu konsep yang terintegrasi dalam menggambarkan hubungan (*relationships*) antar data dan batasan-batasan (*constraint*) data dalam suatu sistem database. Model data yang paling umum, berdasarkan pada bagaimana hubungan antar record dalam database (***Record Based Data Models***), terdapat tiga jenis, yaitu :

a. Model Database Hirarki (*Hierarchical Database Model*)

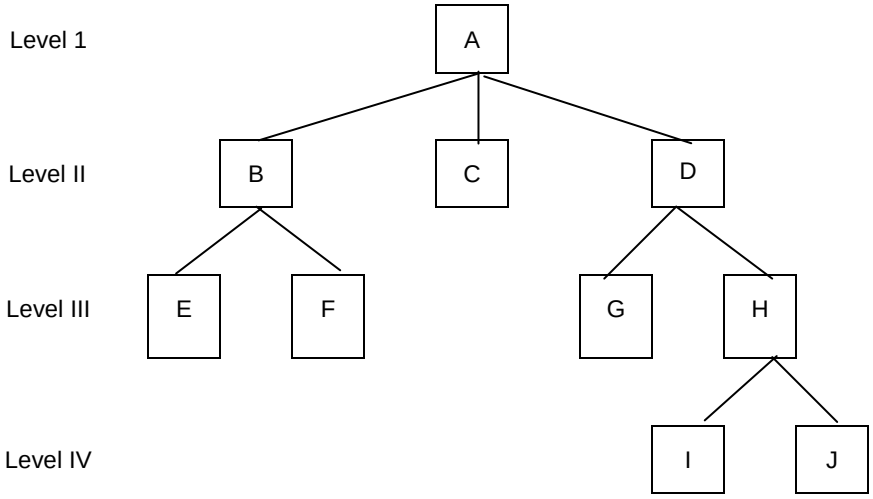
b. Model Database Jaringan (*Network Database Model*)

c. Model Database Relasi (*Relational Database Model*)

Model database hirarki dan jaringan merupakan model database yang tidak banyak lagi dipakai saat ini, karena adanya berbagai kelemahan dan hanya cocok untuk struktur hirarki dan jaringan saja. Artinya tidak mengakomodir untuk berbagai macam jenis persoalan dalam suatu sistem database. Yang paling banyak dipakai saat ini adalah model database relasi, karena mampu mengakomodir berbagai permasalahan dalam sistem database. Berikut keterangan tentang model database ini.

2.2.1 Model Database Hirarki (*Hierarchical Database Model*)

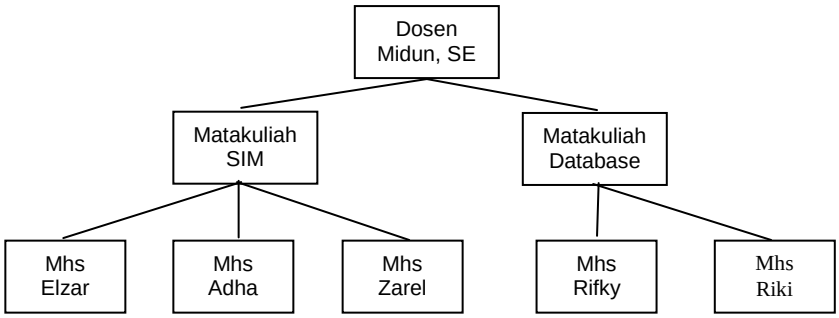
Model database hirarki disebut juga model pohon, karena hubungan antar simpul digambarkan seperti struktur pohon (*tree-structured*) yang dibalik dengan pola hubungan orang tua – anak (*parent – child*). Simpul yang paling atas disebut **akar** (*root*) dan paling bawah disebut **daun**. Setiap simpul digambarkan dengan lingkaran atau kotak. Simpul yang berada di atas simpul lainnya disebut **orang tua**, sedangkan yang berada di bawahnya di sebut **anak,** dimana seorang orang tua bisa mempunyai satu anak (jenis hubungan satu ke satu, ***one to one***) atau mempunya beberapa anak (jenis hubungan satu ke banyak, ***one to many***). Tapi satu anak hanya boleh punya satu orang tua (jenis hubungan satu ke satu, ***one to one***). Untuk jelasnya dapat dilihat pada gambar berikut :



Gambar 2.2 : Contoh model database hirarki

Pada gambar diatas, simpul A disebut **akar** dan juga bertindak sebagai orang tua dengan anak simpul A, B dan C. Simpul E, F, I dan J disebut **daun**, dimana E dan F merupakan anak dari simpul B serta simpul I dan J merupakan anak dari simpul H. Simpul B disebut anak dari simpul A, tapi disisi lain simpul B juga merupakan orang tua dengan anak simpul E dan F.

Dalam aplikasi nyatanya, dapat anda lihat dalam hubungan antara dosen dengan matakuliah yang diasuh serta mahasiswanya. Perhatikan gambar berikut :



Gambar 2.3 : Contoh konkret model database hirarki

Coba anda jelaskan hubungan masing-masing simpul tersebut, mana yang disebut akar, daun, orang tua dan anak ?

Kelemahan utama dari model database hirarki adalah ketidakmampuannya dalam mengelola hubungan banyak ke banyak (***many to many***), sehingga apabila ada jenis hubungan ini pada model database, maka banyaknya redundansi database tidak dapat terelakkan lagi.

Misalnya pada contoh diatas, mahasiswa merupakan anak dari simpul matakuliah, dengan pilihan ini, maka mahasiswa yang sedang cuti (istirahat kuliah) menjadi tidak tertangani, karena yang disimpan hanyalah data mahasiswa (anak) yang mengambil matakuliah (orang tua), akibatnya ada data yang hilang.

Keunggulan model database ini terletak pada keteraturan struktur yang ditunjukkannya dan hanya sangat cocok untuk sistem yang keterkaitan atau hubungan antara recordnya mengikuti struktur hirarki.
Karena keterbatasan pemakaiannya dan adanya kelemahan yang cukup mendasar, penggunaan model database ini dalam pengelolaan sistem database sudah ditinggalkan.

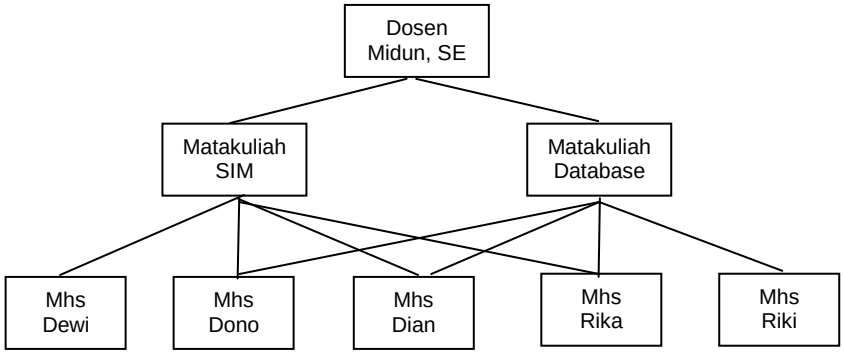
2.2.2 Model Database Jaringan (*Network Database Model*)

Model database jaringan merupakan pengembangan dari model database hirarki, dimana kelemahan yang ada pada model database hirarki yaitu ketidakmampuannya dalam mengelola hubungan banyak ke banyak (***Many to Many***) telah dapat diatasi dengan model database jaringan ini.

Dalam model ini, data di representasikan sebagai koleksi record dan hubungan antar record direpresentasikan sebagai pointer.

Oleh karena itu, model database jaringan mampu menyatakan hubungan :

- **Satu ke Satu (*One To One,* 1 : 1)**, satu orang tua punya satu anak.
- **Satu ke Banyak (*One To Many*, 1 : M)** Satu orang tua punya beberapa anak,
- **Banyak ke Banyak (*Many To Many*, N : M)**, beberapa anak punya beberapa orang tua.



Gambar 2.4 : Contoh konkret model database jaringan

Kelemahan dalam model database ini adalah lebih kompleks dan sulitnya dalam proses *query***,** begitu juga halnya dalam manipulasi data yang harus dilaksanakan dengan menelusuri data pointer pada setiap recordnya.

Kelebihan model database ini adalah dari segi efisiensi penyimpanan data, karena tidak adanya data yang duplikat (*redundansi*) dan akses yang cepat karena langsung memanfaatkan pointer ke alamat fisik data.

Karena kompleksitas yang tinggi, apalagi diterapkan pada sistem database yang begitu kompleks, maka model database ini tidak tepat lagi untuk digunakan. Saat ini, model database jaringan sudah jarang sekali dipakai, kecuali untuk keperluan penelitian (*research*) saja.

2.2.3 Model Database Relasi (*Relational Database Model*)

Model database relasi merupakan model database yang paling banyak digunakan saat ini, karena paling sederhana dan mudah digunakan serta yang paling penting adalah kemampuannya dalam mengakomodasi berbagai kebutuhan pengelolaan database. Sebuah database dalam model ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (***record***) dan kolom (***field***), pertemuan antara baris dengan kolom disebut **item data (*data value*)**, tabel-tabel yang ada dihubungkan (***relationship***) sedemikian rupa menggunakan **field-field kunci (*Key* field)** sehingga dapat meminimalkan duplikasi data.

Model database relasi ini dikemukakan pertama kali oleh **E.F. Codd**, salah seorang pakar dalam bidang database. Sering juga model ini disebut Database relasi.

2.2.3.1 Tingkatan Data Dalam Database Relasi

Dalam suatu sistem database relasi, data yang tersimpan dalam DBMS mempunyai tingkatan-tingkatan, sebagai berikut :

1. **Karakter (*Characters*)**

   Merupakan bagian terkecil dalam database, dapat berupa karakter numerik (angka 0 s.d 9), huruf (A - Z, a - z) ataupun karakter-karakter khusus, seperti *, &. %, # dan lain-lain.

2. **Field** atau **Attribute**

   Merupakan bagian dari record yang menunjukkan suatu item data yang sejenis, Misalnya : field nama, file NIM dan lain sebagainya. Setiap field harus mempunyai nama dan tipe data tertentu. Isi dari field di sebut **Data Value**. Dalam tabel database, field ini disebut juga **kolom**.

3. **Record** atau **Tupple**

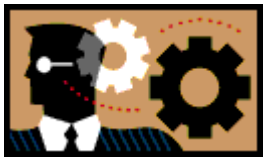
   Tuple/Record adalah kumpulan data value dari attribute yang berkaitan sehingga dapat menjelaskan sebuah entity secara lengkap. Misal : Record entity mahasiswa adalah kumpulan data value dari field nobp, nama, jurusan dan alamat per-barisnya. Dalam tabel database, Record disebut juga **baris.**

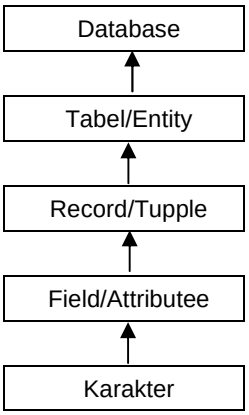
4. **Table/Entity**

   Entity merupakan sesuatu yang dapat diidentifikasi dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian atau konsep yang informasinya akan disimpan dalam database. Misal. Pada sistem database akademik, yang menjadi entity adalah, mahasiswa, dosen, matakuliah dan lain-lain. Dalam aplikasi, penggunaan istilah Entity sering di samakan dengan istilah Tabel. (Entity = table). Disebut tabel, karena dalam merepresentasikan datanya di atur dalam bentuk baris dan kolom. Baris mewakili 1 record dan kolom mewakili 1 field. Dalam sistem database tradisional, entity/table ini disebut juga dengan file.

5. **Database**

   Kumpulan dari tabel-tabel yang saling berelasi, disusun secara logis, sehingga menghasilkan informasi yang bernilai guna dalam proses pengambilan keputusan.





Gambar 2.5 : Tingkatan data dalam database

Perhatikan tabel berikut : Contoh Entity/Tabel Mahasiswa :

NOBP	**NamaMahasiswa**	**Alamat**	
03156001	Helga Elzar Adha	Curup	> baris **record** ke-1
03156002	Rifky Zarel Putra	Bengkulu	> baris **record** ke-2
03156003	Fikri Putra Zarel	Padang	> baris **record** ke-3
03156004	Reinhard Steven	Padang Panjang	> baris **record** ke-4
03156005	Edi Brocoli	Lubuk Alung	> baris **record** ke-5
Field Nobp	Kolom **Field** Nama Mahasiswa	Kolom **field** Alamat	

Data Value

Tabel 2.1 : Contoh sebuah tabel

Ada beberapa sifat yang melekat pada suatu tabel :

- Tidak boleh ada record yang sama (kembar)
- Urutan record tidak terlalu penting, karena data dalam record dapat diurut sesuai dengan kebutuhan.
- Setiap field harus mepunyai nama yang unik (tidak boleh ada yang sama).
- Setiap field mesti mempunyai tipe data dan karakteristik tertentu.

2.2.3.2 Jenis Hubungan Antar Tabel

Jenis hubungan antar tabel dalam model database relasi, juga didefinisikan dengan hubungan :

- **Satu ke satu (*One to One*)**
- **Satu ke Banyak (*One to Many*)**
- **Banyak ke satu (*Many to One*)**
- **Banyak ke Banyak (*Many to Many*)**

Untuk lebih jelasnya penggunaan hubungan ini, sering digunakan **Diagram Entity Relationship (*Diagram E-R*)** yang merupakan bagian dari **Model Data Entity Relationship.**

2.3 Model Data Entity Relationship.

Seperti telah dijelaskan di atas, bahwa hubungan antar data dan batasan-batasannya dalam suatu sistem database, dapat diolah secara hirarki, jaringan dan relasional. Ketiga tipe model data ini mengacu kepada hubungan antar record (***Record Based Data Models***) dalam masing-masing entity/tabel. Tapi disisi lain, hubungan dan batasan data ini, dapat juga berupa **Object Based Data Model** (Model Data Berbasis Object).

Konsep utama dalam model data berbasis object ini adalah penggunaan entity, atribut dan hubungan antar entitynya (***Entity Relationship***)..

Yang tergolong kedalam **Object Based Data Model** ini adalah **Model Data Entity Relationship**. Model data entity relationship sering dijadikan acuan dalam merancang suatu sistem database.

Pada **Model Data Entity Relationship** ini, data yang ada ditransformasikan dengan memanfaatkan sejumlah perangkat konseptual menjadi diagram data, yang sering disebut **Diagram Entity Relationship (*Diagram E-R*)**. Ada dua komponen utama pembentuk model data ini, yaitu :

- **Entity** beserta attributenya.
- **Relasi** dan jenis hubungannya

Contoh :

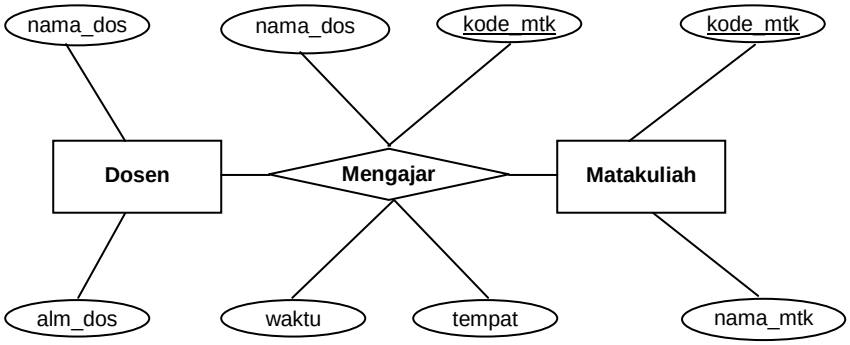
Dalam dunia akademis terdapat entity mahasiswa, dosen, dan matakuliah. Setiap entity mempunyai atribut atau field. Attribute adalah ciri khas yang melekat pada suatu entity. Misalnya entity mahasiswa, ciri khas dari mahasiswa adalah mempunyai nama, tempat tanggal lahir, alamat dan lain sebagainya, begitu juga dengan dosen, mempunyai nama, alamat, dan lain sebagainya. Lalu keterkaitan antara entity mahasiswa dengan entity dosen, mahasiswa dengan matakuliah atau dosen dengan matakuliah, digambarkan dengan simbol-simbol sehingga lebih mudah dipahami. Penggambaran hubungan inilah yang disebut dengan **diagram E-R.**

2.4 Diagram Entity Relationship (*Diagram E-R*)

Diagram E-R digunakan untuk menggambarkan secara sistematis hubungan antar entity-entity yang ada dalam suatu sistem database menggunakan simbol-simbol sehingga lebih mudah dipahami. Simbol-simbol yang boleh digunakan adalah :

- **Persegi Panjang**, berfungsi untuk menyatakan suatu entity.
- **Elips**, berfungsi untuk menyatakan attribute, jika diberi garis bawah menandakan bahwa attribute tersebut merupakan attribute/field kunci.
- **Belah Ketupat**, menyatakan jenis relasi.
- **Garis**, penghubungan antara relasi dengan entity dan antara entity dengan attribute.

Misal : Hubungan antara entity Dosen dengan entity matakuliah.



Gambar 2.6 : Contoh sebuah diagram E-R

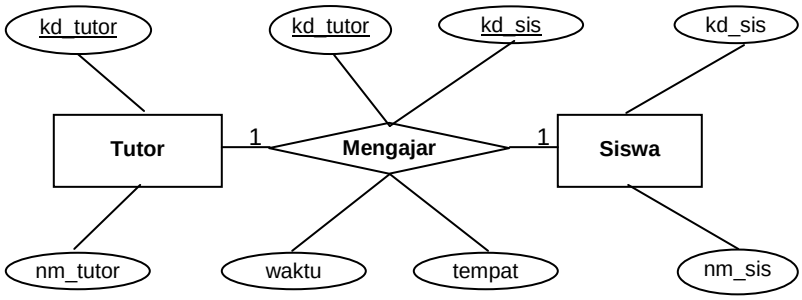
Dalam hubungan antar entity, juga harus ditentukan derajat relasi antar entity. Derajat relasi menunjukkan jumlah maksimum record suatu entity ber-relasi dengan record pada entity yang lainnya. Misalnya pada contoh sebelumnya, entity mahasiswa dapat berelasi dengan **lebih dari satu** record yang ada pada entity matakuliah sebaliknya **satu** record pada entity matakuliah hanya boleh ber-relasi dengan **satu** mahasiswa yang sama pada entity mahasiswa, begitu juga satu record pada entity matakuliah berhubungan paling banyak satu record juga pada entity dosen, dan seterusnya.

Berikut dijelaskan secara lengkap jenis relasi antar entity (misalnya entity A dan B) beserta derajat relasinya disertai contohnya menggunakan diagram E-R.

2.4.1 Relasi Satu ke Satu (*One to One*)

Artinya **satu** record pada entity A ber-relasi paling banyak **satu** record juga pada entity B, begitu juga sebaliknya, satu record pada entity B, ber-relasi paling banyak satu record juga dengan entity A. Dalam diagram E-R, relasi ini disimbolkan dengan angka 1.

Contoh : Dalam proses belajar mengajar secara privat misalnya, seorang (**satu**) tutor hanya mengajar **satu** siswa, begitu juga sebaliknya, satu siswa hanya diajar oleh satu tutor. Hubungan antar entity tutor dengan siswa ini dapat digambar dengan diagram E-R sebagai berikut :



Gambar 2.7 : Contoh relasi satu ke satu.

Keterangan :

- ☑ Entity tutor mempunya dua attribute, yaitu kode tutor (**kd_tutor**) yang berfungsi sebagai field kunci, dan nama tutor (**nm_tutor**).
- ☑ Entity siswa juga mempunyai dua attribute, yaitu kode siswa (**kode_sis**) sebagai field kunci dan nama siswa (**nm_sis**).
- ☑ Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode tutor (**kd_tutor**) dan kode siswa (**kd_sis**) yang berfungsi sebagai kunci tamu (*foreign key*) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan tempat mengajar (**tempat**).
- ☑ Derajat relasi dinyatakan dengan **1 : 1**, yang menandakan bahwa hubungan antar entity adalah satu ke satu, seperti terlihat pada gambar.

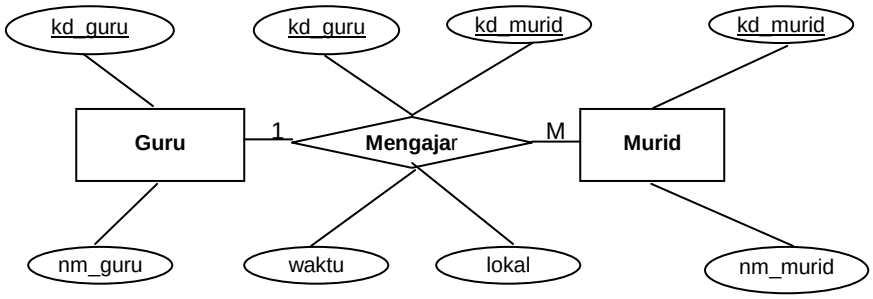
Dalam diagram E-R, attribute yang berfungsi sebagai field kunci harus di beri garis bawah, sebagai pembeda dengan attribute yang lain.

2.4.2 Relasi Satu ke Banyak (*One to Many*)

Artinya **satu** record pada entity A ber-relasi dengan beberapa record pada entity B, tapi tidak sebaliknya, setiap record pada entity B ber-relasi paling banyak satu record dengan entity A. Dalam diagram E-R, relasi ini disimbolkan dengan angka **1** untuk menyatakan satu dan huruf **M** atau **N** untuk menyatakan banyak.

Contoh : Dalam proses belajar mengajar di sekolah dasar misalnya, **satu** orang guru mengajar beberapa (**banyak**) murid, tetapi satu kelas (beberapa murid) hanya di ajar

oleh satu guru. Hubungan antara entity guru dan murid seperti ini dapat dibuat diagram E-R, sebagai berikut :



Gambar 2.8 : Contoh relasi satu ke banyak.

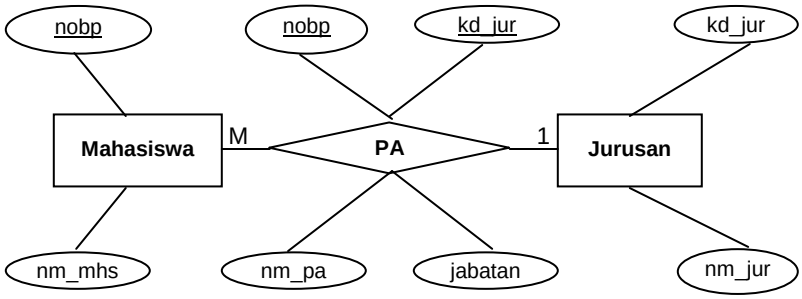
Keterangan :

- © Entity guru mempunya dua attribute, yaitu kode guru (**kd_guru**) yang berfungsi sebagai field kunci, dan nama guru (**nm_guru**).
- © Entity murid juga mempunyai dua attribute, yaitu kode murid (**kd_murid**) sebagai field kunci dan nama murid (**nm_murid**).
- © Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode guru (**kd_guru**) dan kode murid (**kd_murid**) yang berfungsi sebagai kunci tamu (*foreign key*) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan lokal mengajar (**lokal**).
- © Derajat relasi dinyatakan dengan **1 : M**, yang menandakan bahwa hubungan antar entity adalah satu ke banyak, seperti terlihat pada gambar diatas.

2.4.3 Relasi Banyak ke Satu (*Many to One*)

Ini adalah kebalikan dari relasi satu ke banyak, dimana setiap record pada entity A hanya dapat ber-relasi paling banyak 1 record pada entity B, tapi tidak sebaliknya, satu record pada entity B dapat ber-relasi dengan beberapa record pada entity A.. Dalam diagram E-R, relasi ini disimbolkan dengan angka **1** untuk menyatakan satu dan huruf **M** atau **N** untuk menyatakan banyak..

Contoh : Dalam dunia akademik misalnya, beberapa (**banyak**) mahasiswa hanya mempunyai **satu** pilihan jurusan, sebaliknya **satu** jurusan dapat dipilih oleh beberapa (**banyak**) mahasiswa. Dalam diagram E-R, hal ini dapat digambar sebagai berikut :



Gambar 2.9 : Contoh relasi banyak ke satu.

Keterangan :

® Entity mahasiswa mempunya dua attribute, yaitu nomor buku pokok (**nobp**) yang berfungsi sebagai field kunci, dan nama mahasiswa (**nm_mhs**).

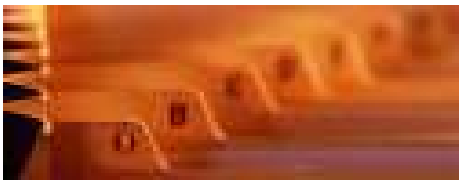
® Entity jurusan juga mempunyai dua attribute, yaitu kode jurusan (**kode_jur**) sebagai field kunci dan nama jurusan (**nm_jur**).

® Hubungan antara kedua entity tersebut dinyatakan dalam entity pembimbing akademik (**PA**), yang mempunyai 4 attribute, yaitu nomor buku pokok (**nobp**) mahasiswa dan kode jurusan (**kd_jur**) yang berfungsi sebagai kunci tamu (*foreign key*) pada entity **PA** serta attribute nama PA (**nm_pa**) dan jabatan PA(**jabatan**).

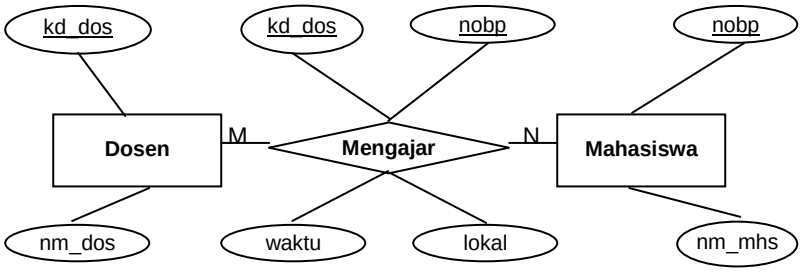
® Derajat relasi dinyatakan dengan **M : 1**, yang menandakan bahwa hubungan antar entity adalah banyak ke satu, seperti terlihat pada gambar diatas.

2.4.4 Relasi Banyak ke Banyak (*Many to Many*)

Artinya beberapa record pada entity A dapat ber-relasi dengan beberapa record juga pada entity B, begitu juga sebaliknya, beberapa record pada entity B dapat ber-relasi dengan beberapa record juga pada entity A.. Dalam diagram E-R, relasi ini disimbolkan dengan huruf **M** atau **N** untuk menyatakan banyak..



Contoh : Dalam hubungan antara mahasiswa dengan dosen pada perguruan tinggi, yaitu seorang seorang dosen mengajar banyak mahasiswa, sebaliknya seorang mahasiswa dapat diajar oleh beberapa dosen, sehingga terjadihubungan **banyak ke banyak**. Perhatikan diagram E-R berikut :



Gambar 2.10 : Contoh relasi banyak ke banyak.

Keterangan :

- Entity dosen mempunya dua attribute, yaitu kode dosen (**kd_dos**) yang berfungsi sebagai field kunci, dan nama dosen (**nm_dos**).
- Entity mahasiswa juga mempunyai dua attribute, yaitu No. Buku Pokok (**NoBP**) sebagai field kunci dan nama mahasiswa (**nm_mhs**).
- Hubungan antara kedua entity tersebut dinyatakan dalam entity mengajar, yang mempunyai 4 attribute, yaitu kode dosen (**kd_dos**) dan No.Buku Pokok mahasiswa (**NoBP**) yang berfungsi sebagai kunci tamu (*foreign key*) pada entity mengajar serta attribute waktu mengajar (**waktu**) dan tempat mengajar (**lokal**).
- Derajat relasi dinyatakan dengan **M : N**, yang menandakan bahwa hubungan antar entity adalah banyak ke banyak, seperti terlihat pada gambar diatas.

2.5 Tahapan Membuat Diagram E-R.

Berikut panduan bagaimana tahapan dalam membuat Diagram E-R;

- **Mengidentifikasi dan menetapkan seluruh entity yang terlibat dalam sistem database tersebut.**

  Sebagaimana telah dijelaskan diawal, bahwa entity merupakan sesuatu yang dapat diidentifikasi dengan mudah dari suatu sistem database, bisa berupa objek, orang, tempat, kejadian atau konsep yang informasinya akan disimpan. Dari sekian banyak kemungkinan entity yang ada, anda harus memilah-milah entity mana saja yang sesuai dan mampu mengakomodasi kebutuhan sistem yang akan dirancang. Misalnya dalam ruang lingkup perkuliahan, ada banyak entity yang mungkin, misalnya entity mahasiswa, matakuliah, dosen, asistensi, jadwal, jurusan dan lain sebagainya. Namun dalam ruang lingkup perkuliahan sederhana, anda dapat menggunakan tiga entity saja, yaitu entity mahasiswa, dosen dan matakuliah.

  Proses menentukan entity ini memang agak sulit tapi akan menjadi mudah apabila diserta latihan dengan berbagai macam kasus database.

- **Menentukan attribute-attribute atau field dari masing-masing entity beserta kunci (*key*)-nya.**

  Menentukan attribute dari suatu entity sangat menentukan baik atau tidaknya sistem database yang dirancang, karena attribute ini sangat menentukan nantinya dalam proses relasi. Attribute merupakan ciri khas yang melekat pada suatu entity, misalnya attribute pada mahasiswa dapat berupa nobp, nama, tempat lahir, tanggal lahir, alamat, nama orang tua, pekerjaan orang tua dan lain-lain. Dari sekian banyak kemungkinan attribute yang ada pada entity mahasiswa, anda dapat menggunakan hanya yang perlu saja. Setelah menentukan attributenya selanjutnya adalah menentukan field kunci. Field kunci adalah penanda entity tersebut sehingga bisa digunakan untuk relasi nantinya dan field kunci ini harus bersifat unik. Misalnya pada entity mahasiswa, attribute nobp bisa dijadikan field kunci, karena bersifat unik dan tidak ada mahasiswa yang mempunyai nobp sama.

- **Mengidentifikasi dan menetapkan seluruh himpunan relasi diantara himpunan-himpunan entity yang ada beserta kunci tamu (*foreign key*)-nya.**

  Setelah menentukan entity dan attribute beserta field kuncinya, maka selanjutnya adalah menentukan entity yang terbentuk akibat adanya relasi antar entity. Misalnya antara entity mahasiswa dengan entity dosen, terjadi suatu hubungan proses mengajar, maka proses mengajar ini merupakan entity baru. Entity mengajar ini harus anda tentukan juga attribute yang melekat padanya beserta kunci tamu (*foreign key*). Kunci tamu adalah field kunci utama pada tabel lain, dan field tersebut digunakan juga pada tabel yang satu lagi. Misalnya nobp adalah field kunci dari entity mahasiswa, pada entity mengajar terdapat juga attribute NoBP, maka keberadaan attribute nobp pada entity mengajar disebut sebagai kunci tamu. Proses menentukan hubungan antar entity juga sangat menentukan kualitas sistem database yang dirancang.

- **Menentukan derajat relasi untuk setiap himpunan relasi.**

  Setelah semua entity dan attribute yang dibutuhkan terbentuk, maka selanjutnya adalah menentukan derajat relasi antar entity tersebut, apakah satu kesatu, satu ke banyak atau sebaliknya, atau banyak ke banyak. Berhati-hatilah dalam menentukan derajat relasi ini, karena nantinya akan berhubungan dengan proses query terhadap data.

2.6 Jenis-Jenis Kunci (*Key*)

Key atau kunci adalah suatu field yang dapat mewakili dari suatu record. Misal : nobp merupakan field kunci dari entity mahasiswa, sehingga setiap melakukan pencarian atas entity mahasiswa cukup menyebutkan nobp saja, maka field nama, jurusan dan alamat dapat diketahui. Syarat utama pemilihan suatu field kunci dari entity adalah field tersebut harus unik dan tidak boleh bernilai NULL.

Ada 4 jenis dari key ini :

a. **Candidate Key** (Kunci Calon)

Sebuah attribute atau lebih yang secara unit mengidentifikasi sebuat record, disebut candidate key. Attribute ini mempunyai nilai yang unik pada hampir setiap recordnya. Fungsi dari candidate key ini adalah sebagai calon primary key.

Misal : Entity dosen dalam database akademik :

KodeDosen	NIP	Nama_Dosen	Alamat	Pendidikan
01	131656765	Johan, SE	Padang	S.1
02	130876543	Rafdinal, Drs, M.Si	Padang	S.2
03	132098675	Ratna, SE	Padang	S.1
04	130987567	Effendi Bakar, SE	Padang	S.1

Tabel 2.2 : Tabel entity dosen

Dalam kasus ini, field KodeDosen dan NIP merupakan Candidate Key karena masing-masing bersifat unik dan tidak boleh ada yang sama atau bernilai Null.

b. **Primary Key** (Kunci Utama)

Merupakan candidate key yang telah dipilih untuk mengidentifikasi setiap record secara unik. Primary key harus merupakan field yang benar-benar unik dan tidak boleh ada nilai NULL. **Misal :** Perhatikan kembali entity dosen di atas, yaitu mempunyai dua Candidate Key, maka anda dapat mengambil field KodeDosen sebagai Primary Key, Karena lebih sederhana dalam jumlah digit dan unik. Boleh juga menggunakan NIP sebagai Primary Key-nya, hal ini tergantung kepada si perancang database itu sendiri.

c. **Alternate Key** (Kunci Alternatif)

Adalah candidate key yang tidak terpilih. Misal : dalam suatu entity terdapat dua field yang bisa dijadikan sebagai kunci. Sementara yang boleh dijadikan kunci hanya satu, maka anda harus memilih salah satu. Field yang anda pilih, disebut primary key, sedangkan field yang tidak dipilih disebut dengan alternate key.
Misal : Kembali ke kasus entity dosen diatas, jika anda pilih field KodeDosen sebagai Primary Key, maka otomatis field NIP menjadi Alternate Key-nya, begitu juga sebaliknya.

d. **Foreign Key** (kunci Tamu)

Jika sebuah primary key terhubungan ke table/entity lain, maka keberadaan primary key pada entity tersebut di sebut sebagai foreign key. Misal : Primary Key KodeDosen dari entity Dosen digunakan juga pada field entity KRS, maka keberadaan field KodeDosen pada entity KRS disebut sebagai foreign key.