Implementasi Basis Data

E-R & Relational Model

- *Conceptual Data Model* (E-R Model) dibuat untuk memahami kebutuhan data dan aturan-aturan bisnisnya.
- *Logical Data Model* (*Relational Model*) dibuat untuk mengatur struktur data agar data dapat diproses dengan baik.
- Menterjemahkan *Conceptual Design* menjadi *Logical Database Design* untuk kemudian diimplementasikan pada DBMS yang dipilih.

Implementasi Basis Data

- Implementasi basis data merupakan sebuah tahapan yang diupayakan untuk membangun basis data fisik yang ditempatkan dalam media penyimpanan (*disk*) dengan bantuan DBMS.
- Sebuah diagram E-R akan direpresentasikan menjadi sebuah basis data fisik.
- Komponen-komponen diagram E-R yang berupa himpunan entitas dan himpunan relasi akan ditransformasikan menjadi tabel-tabel yang merupakan komponen utama pembentuk basis data.
- Atribut-atribut yang melekat pada masing-masing himpunan entitas dan himpunan relasi akan dinyatakan sebagai *field-field* dari tabel-tabel yang sesuai.

Implementasi Basis Data

Performansi basis data ditentukan oleh :

- Kualitas dan bentuk perancangan basis data
- Kualitas mesin atau komputer
- Platform yang dipilih
- Sistem operasi
- DBMS yang digunakan

Relation

- Tabel data dua dimensi yang memiliki nama
  - Tabel terdiri dari baris dan kolom
  - Tidak semua tabel adalah *relation*
- Syarat *Relation*
  - Setiap *relation* memiliki nama unik
  - Setiap *attribute*nya *atomic* (tidak *multivalued*, tidak *composite*)
  - Setiap baris adalah unik (tidak ada dua baris dengan isi seluruh kolomnya sama)
  - *Attributes* (kolom) dalam tabel memiliki nama unik
  - Urutan baris atau kolom tidak relevan

Keterkaitan denganE-R Model

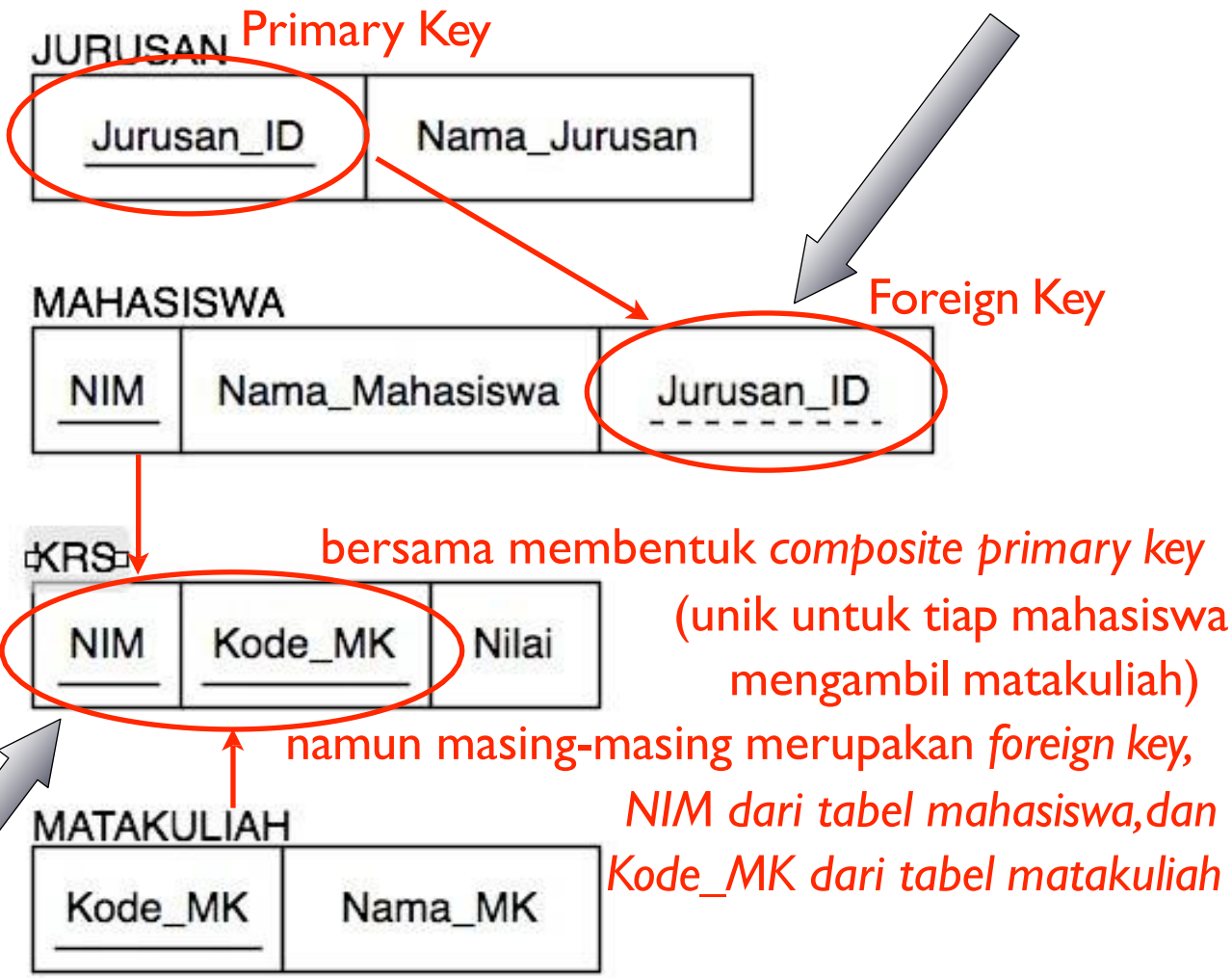
- *Relations* (tabel) terkait dengan *entity types* dan *many-to-many relationship types*.
- Baris-baris terkait dengan *entity instances* dan *many-to-many relationship instances*.
- Kolom-kolom terkait dengan *attributes*.

- Catatan:
  Kata *relation* (dalam *relational database*) **berbeda** dengan *relationship* (dalam E-R model)

Kolom-kolom Kunci

- *Primary key* adalah kolom (atau gabungan beberapa kolom) yang menjadi identitas unik tiap-tiap baris pada sebuah *relation*.
- *Foreign key* adalah kolom (atau gabungan beberapa kolom) yang merupakan *primary key* pada *relation* (tabel) lain.
- Berguna untuk menghubungkan antara *dependent relation* (sisi *many*) dengan *parent relation* (sisi *one*).

implementasi *one-to-many* (1:N) *relationship* antara jurusan dan mahasiswa)



implementasi *many-to-many* (N:N) *relationship* antara mahasiswa dan matakuliah

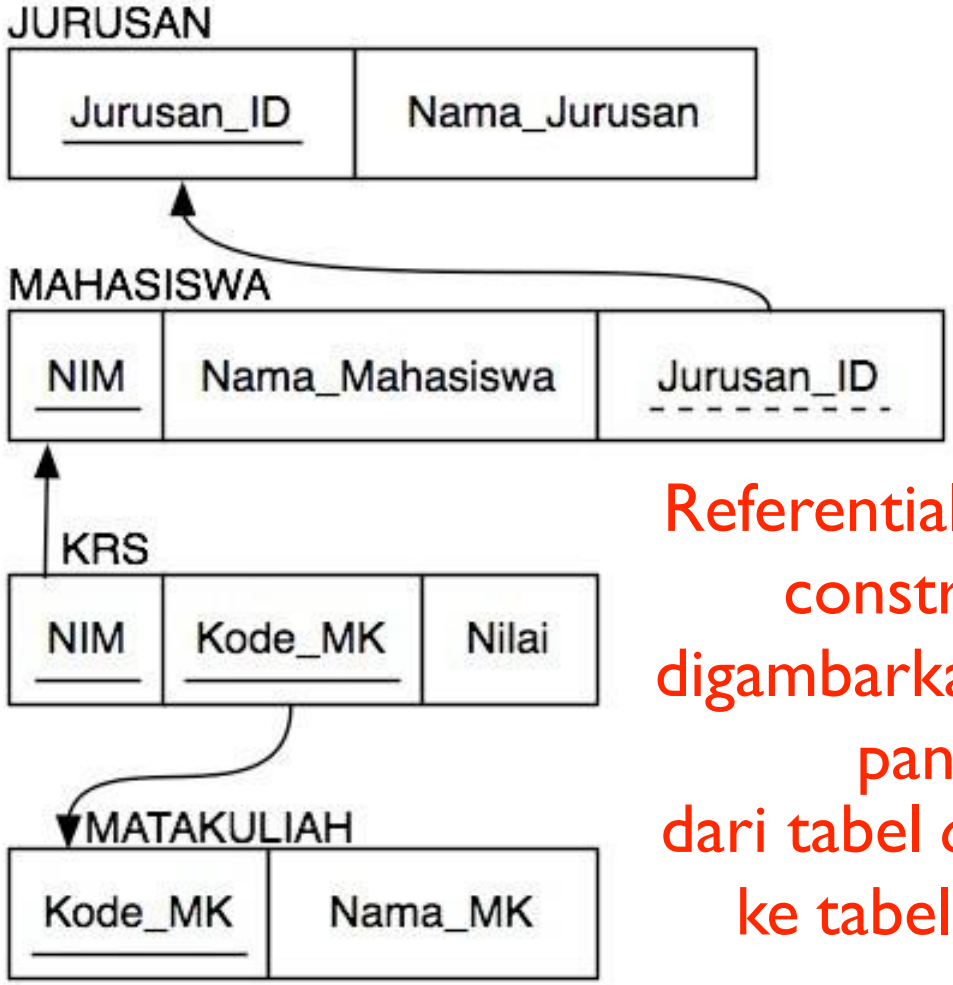
Primary & Foreign Key

Integrity Constraints

- Domain Constraints
  - Batasan mengenai nilai yang boleh muncul dalam sebuah kolom
- Entity Integrity
  - Primary key tidak boleh NULL
  - Primary key harus berisi data

Referential Integrity

- Aturan yang menyatakan bahwa nilai suatu *foreign key* harus cocok dengan sebuah nilai *primary key* dari sisi *one* suatu *relationship*. (atau *foreign key* boleh *null*).
- Contoh : *Delete Rule*
  - *Restrict* - membatasi penghapusan baris dari *parent* apabila ada baris terkait di sisi *dependent*
  - *Cascade* - otomatis menghapus baris-baris terkait di sisi *dependent* apabila *parent* dihapus
  - Set-to-Null - *foreign key* diubah menjadi *null* bila *parent* dihapus (tidak boleh untuk *weak entity*)



Referential integrity constraints digambarkan dengan panah dari tabel *dependent* ke tabel *parent*

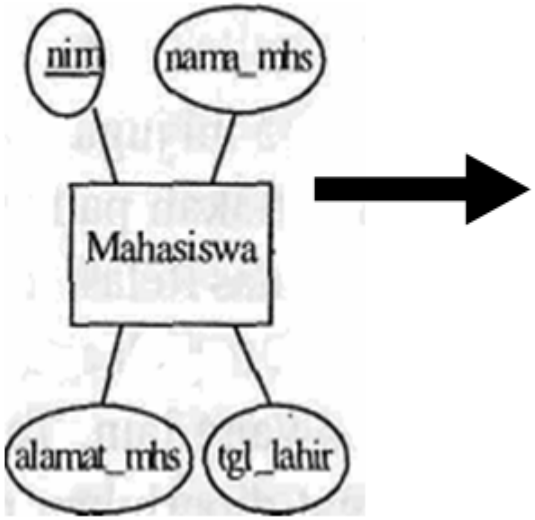
Referential Integrity Constraints

Transformasi Model Data ke Basis Data Fisik

1. Transformasi Umum
2. Implementasi Himpunan Entitas Lemah dan Sub Entitas
3. Implementasi Relasi Tunggal (*Unary Relation*)
4. Implementasi Relasi Multi Entitas (*N-ary Relation*)
5. Implementasi Relasi Ganda (*Redundant Relation*)
6. Implementasi Spesialisasi dan Generalisasi
7. Implementasi Agregasi

Transformasi Umum

1. Setiap himpunan entitas akan diimplementasikan sebagai sebuah tabel (*file* data).

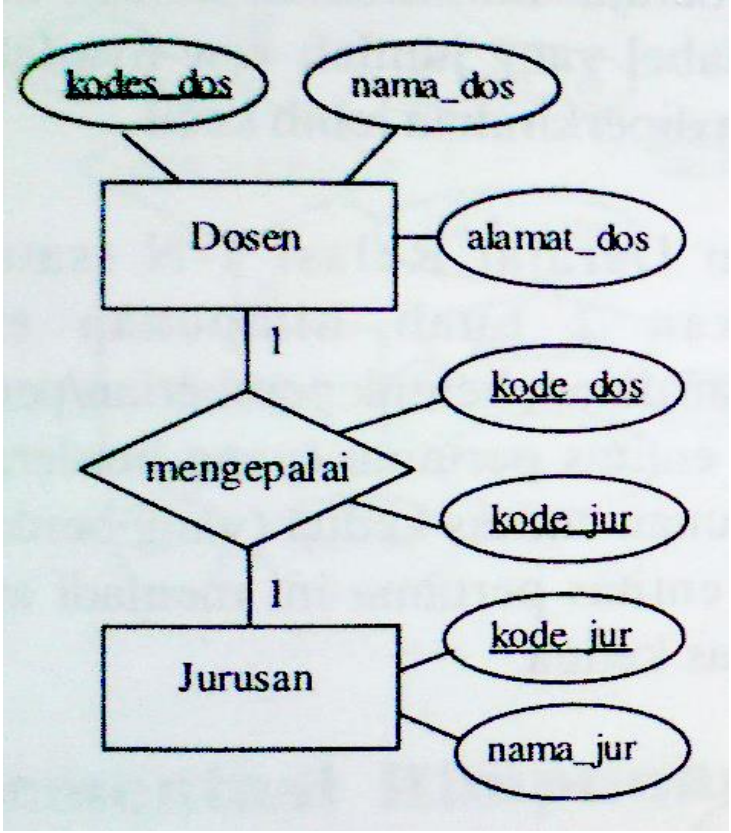


Tabel Mahasiswa

Nim	Nama_mhs	Alamat_mhs	Tgl_lahir

Transformasi Umum

2. Relasi dengan derajat relasi 1 -1 (satu ke satu) yaitu relasi yang menghubungkan dua buah himpunan entitas akan direpresentasikan dalam bentuk penambahan/penyertaan atribut-atribut relasi ke tabel yang mewakili salah satu dari kedua himpunan entitas.



Tabel Dosen

Kode_dos	Nama_dos	Alamat_dos

Tabel Jurusan

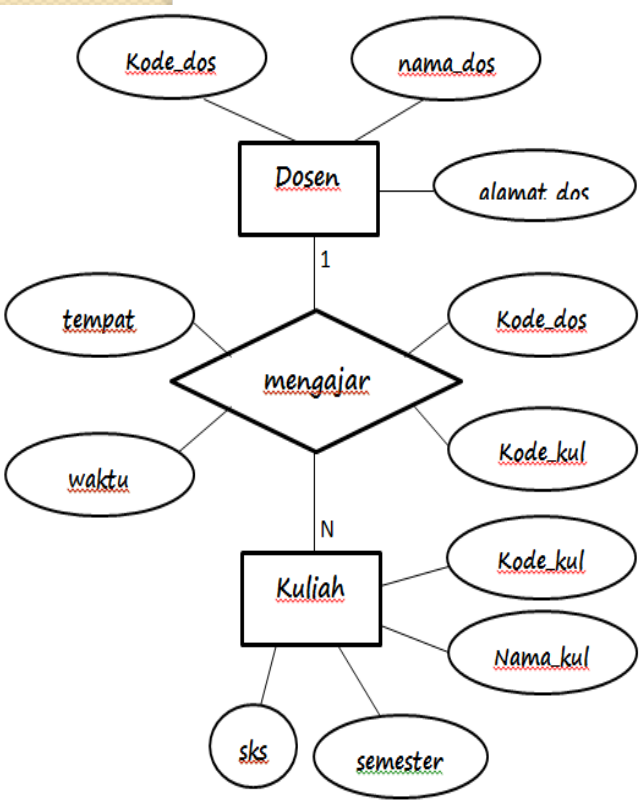
Kode_jur	Nama_jur	Kode_dos

(kode_dos) pada tabel jurusan : Field yang ditambah dari relasi mengepalai

Transformasi Umum

3. Relasi 1-N (Satu ke Banyak)

   Relasi dengan derajat relasi satu-ke-banyak, yaitu relasi yang menghubungkan dua buah himpunan entitas, juga akan direpresentasikan dalam bentuk pemberian/pencantuman atribut kunci *(key)* dari himpunan entitas pertama (yang berderajat 1) yang mewakili himpunan entitas kedua (yang berderajat N)



Tabel Dosen

Kode_dos	Nama_dos	Alamat_dos

Tabel Kuliah

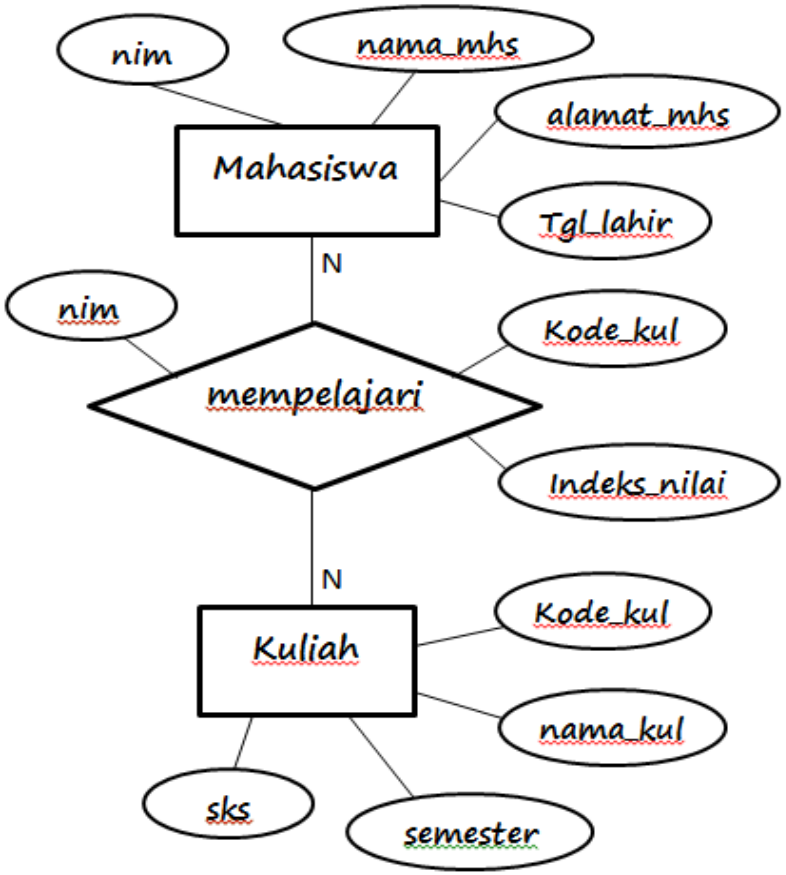
Kode_kul	Nama_kul	sks	semester	Kode_dos	tempat	waktu

Kode_dos, tempat, waktu pada tabel kuliah :
Field yang ditambah dari relasi Mengajar

Transformasi Umum

4. Relasi N-N (Banyak ke Banyak)

   Relasi dengan derajat relasi banyak-ke-banyak, yang menghubungkan dua buah himpunan entitas akan diwujudkan dalam bentuk tabel khusus, yang memiliki field *(atau foreign key)* yang berasal dari kunci-kunci dari himpunan entitas yang dihubungkannya.



Tabel Mahasiswa

nim	Nama_mhs	Alamat_mhs	Tgl_lahir

Tabel Mempelajari/Tabel Nilai

nim	Kode_kul	Indeks_nilai

Tabel Kuliah

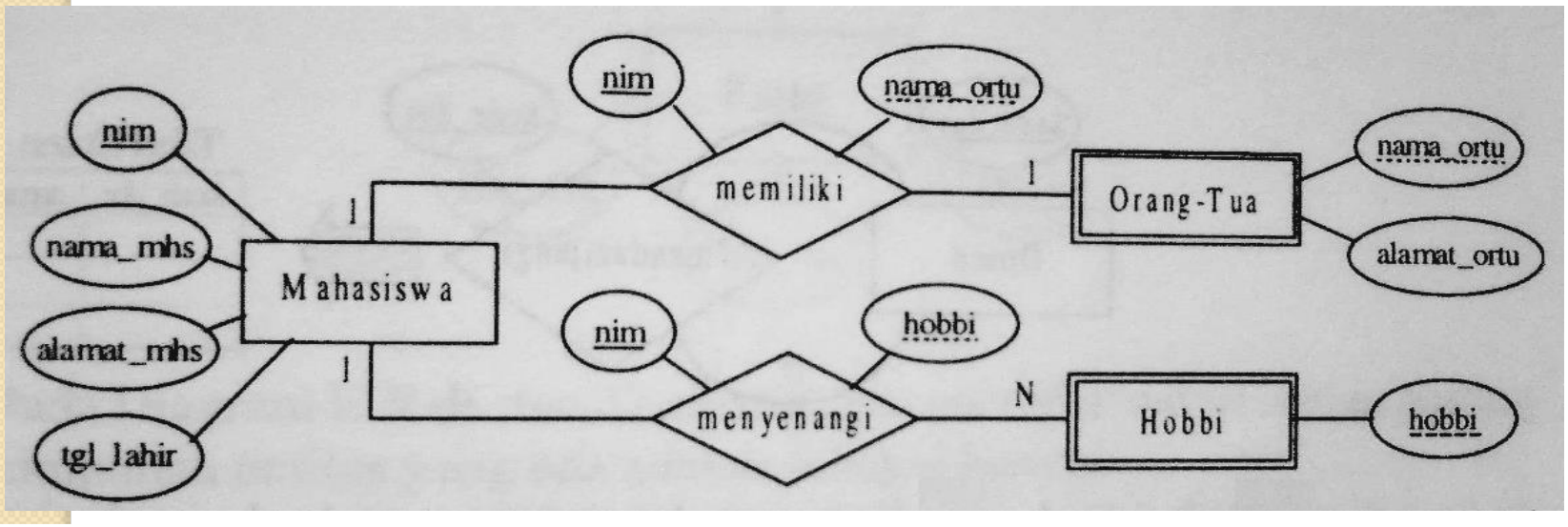
Kode_kul	Nama_kul	sks	semester

Implementasi Himpunan Entitas Lemah dan Sub Entitas

Perbedaan Himpunan Entitas Lemah dan Entitas Kuat :

Himpunan Entitas Kuat sudah dapat langsung menjadi sebuah tabel utuh/sempurna, walaupun tanpa melihat relasinya dengan himpunan entitas yang lain.

Himpunan entitas lemah dan sub entitas hanya dapat ditransformasikan menjadi sebuah tabel dengan menyertakan atribut key yang ada di himpunan entitas kuat yang berelasi dengannya.



Tabel Mahasiswa

nim	Nama_mhs	Alamat_mhs	Tgl_lahir

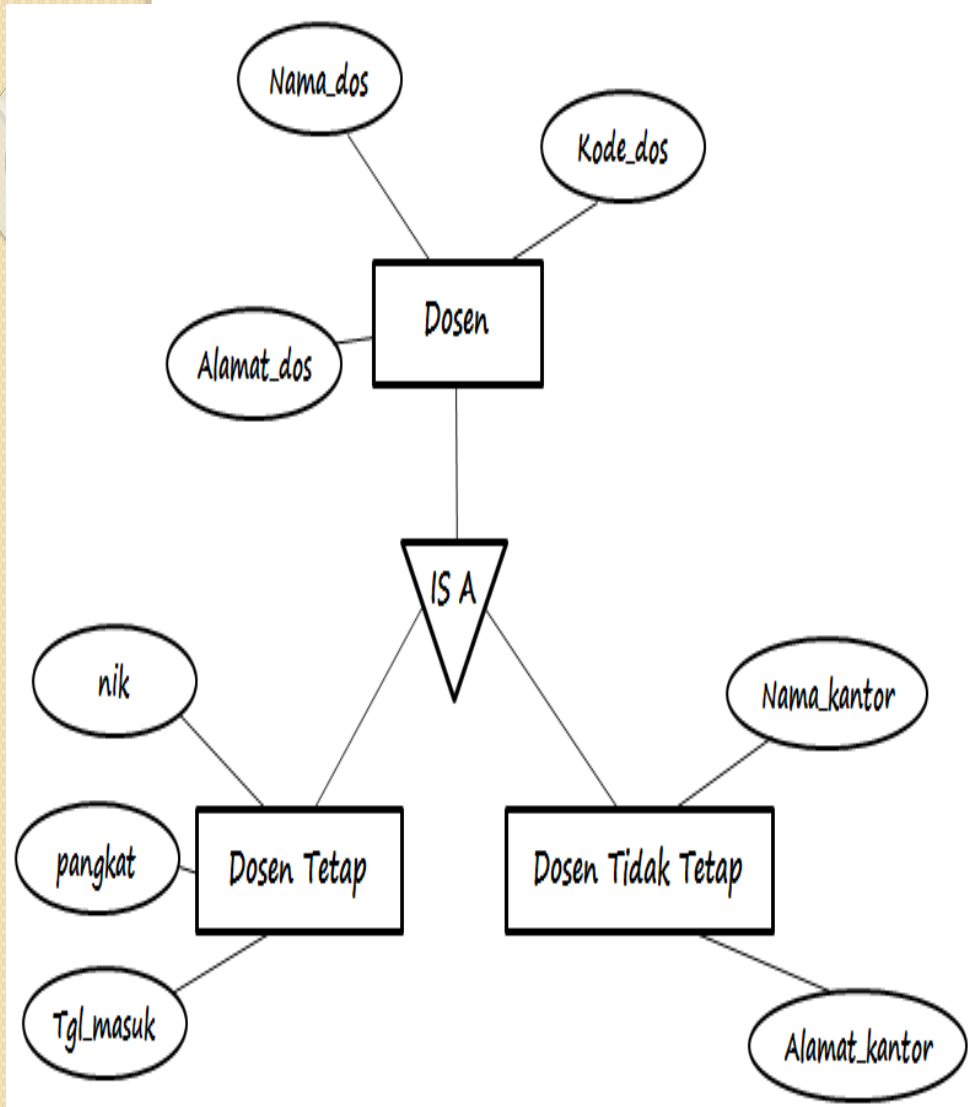
Tabel orang tua

nim	Nama_ortu	Alamat_ortu

Tabel Hobbi

nim	hobbi

Implementasi Sub Entitas Hasil Spesialisasi



Tabel Dosen

Kode_dos	Nama_dos	Alamat_dos

Tabel Dosen Tetap

Kode_dos	nik	Pangkat	Tgl_masuk

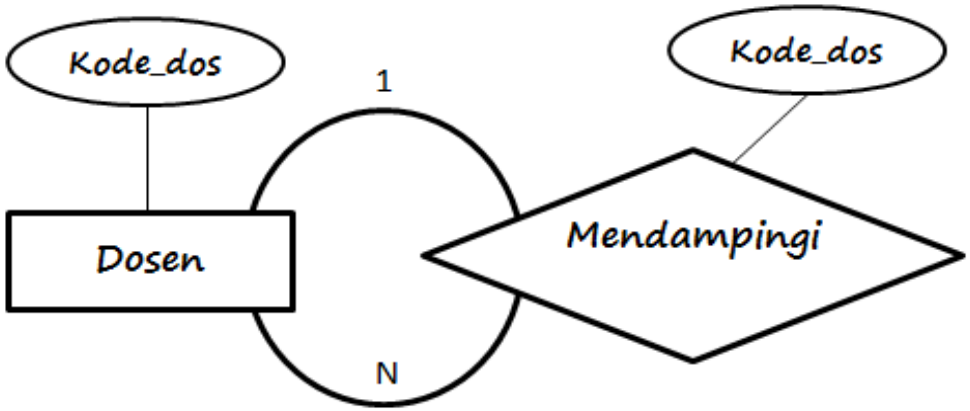
Tabel Dosen Tidak Tetap

Kode_dos	Nama_kantor	Alamat_kantor

Kode_dos : key yang diambil dari key himpunan entitas utamanya

Implementasi Relasi Tunggal (Unary Relation)

- Implementasi relasi tunggal dari/ke himpunan entitas yang sama dalam diagram E-R tergantung pada derajat relasinya.
- Jika ada himpunan A dengan dua atribut x dan y dengan x sebagai key, maka relasi tunggal terhadap himpunan entitas tersebut diwujudkan dengan menambah kembali field tersebut ke tabel A.
- Karena x (kode_dos) sebagai field unik, maka field x yang kedua harus diganti namanya sesuai dengan fungsinya

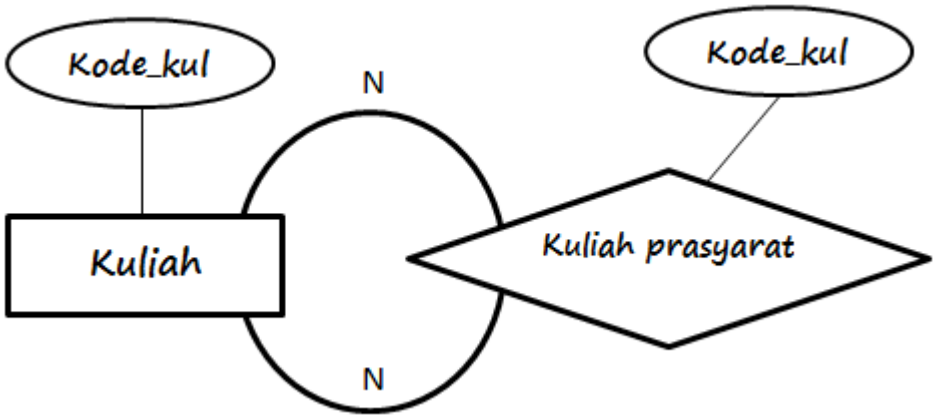


Tabel Dosen

Kode_dos	Nama_dos	Alamat_dos	Kode_dos_pend

Implementasi Relasi Tunggal (Unary Relation)

- Untuk relasi yang derajatnya banyak ke banyak akan diimplementasikan dengan membentuk tabel baru yang merepresentasikan tabel tersebut .
- Tabel baru mendapatkan field dari semua atribut relasi dan ditambah dengan atribut key dari himpunan entitasnya.



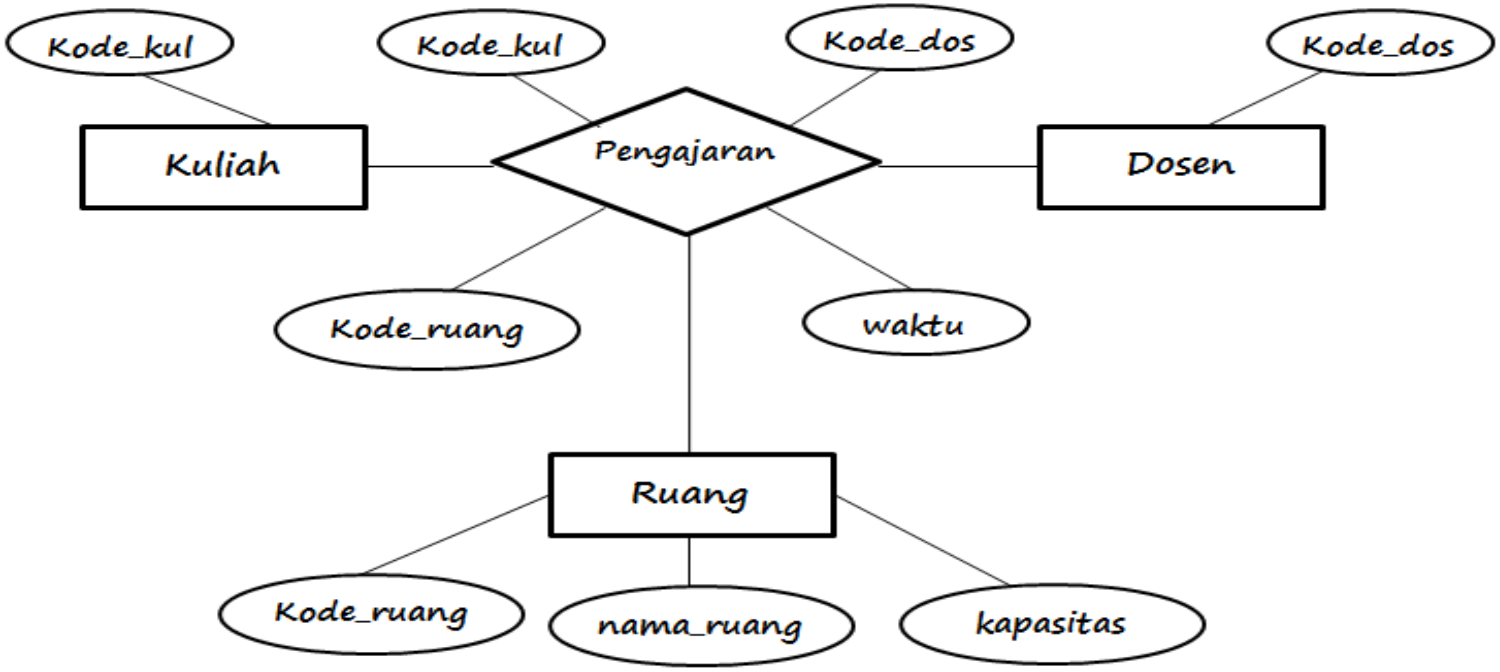
Tabel Kuliah

Kode_kul	Nama_kul	sks	semester

Tabel Prasyarat Kuliah

Kode_kul	Kode_kul_prasyarat

Implementasi Relasi Multi Entitas (N-ary Relation)



- Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B dan sebaliknya dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

Implementasi Relasi Multi Entitas (N-ary Relation)

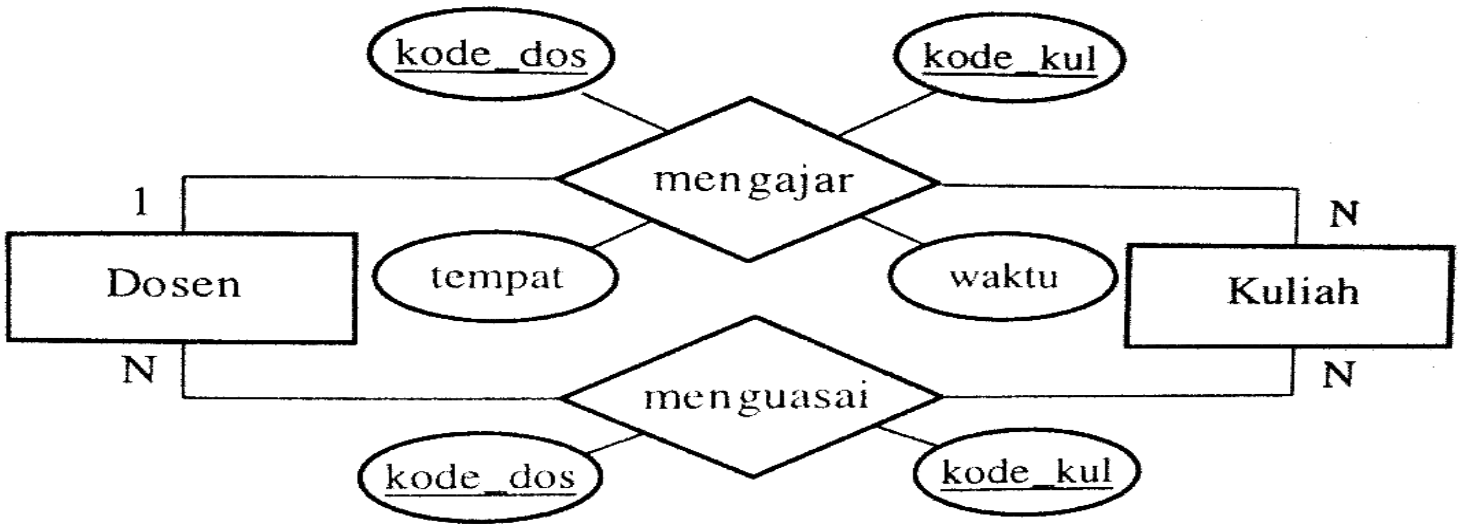
- Pada diagram ER diatas, derajat relasi parsial antar himpunan adalah :
  * Dosen – Kuliah : I – N
  * Ruang – Kuliah : I – N
  * Ruang – Dosen : N – N

Tabel Kuliah

Kode_kul	Nama_kul	sks	semester	Kode_dos	Kode_ruang	waktu

Kode_dos, kode_ruang dan waktu adalah 3 field yang mewakili relasi pengajaran.

Implementasi Relasi Ganda (Redundant Relation)



- Relasi Mengajar (1-N), field kode_dos dari himpunan entitas dosen ditambahkan ke tabel kuliah.

- Relasi Menguasai (N-N), relasi dinyatakan dalam tabel khusus dengan 2 buah field : kode_dos dan kode_kul.

Implementasi Spesialisasi dan Generalisasi

Tabel Dosen

kode_dos	nama_dos	alamat_dos

Tabel Kuliah

kode_kul	nama_kul	SKS	Trimester	Kode_dos

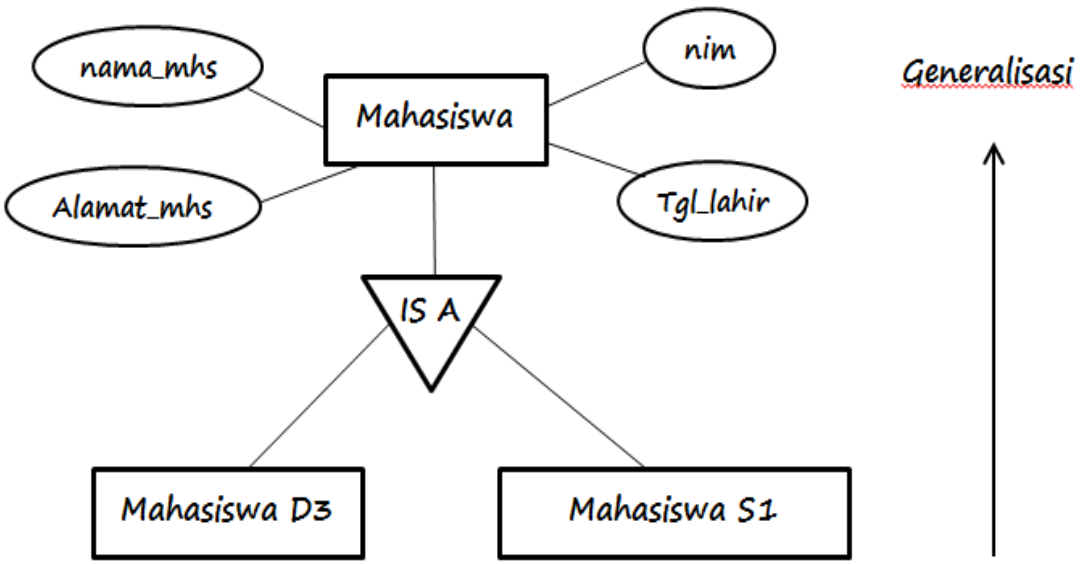
Kode_dos : atribut tambahan untuk merepresentasikan relasi mengajar

Tabel Menguasai

Kode_dos	Kode_kul

Kode_kul : tabel khusus untuk merepresentasikan relasi menguasai

Implementasi Spesialisasi dan Generalisasi

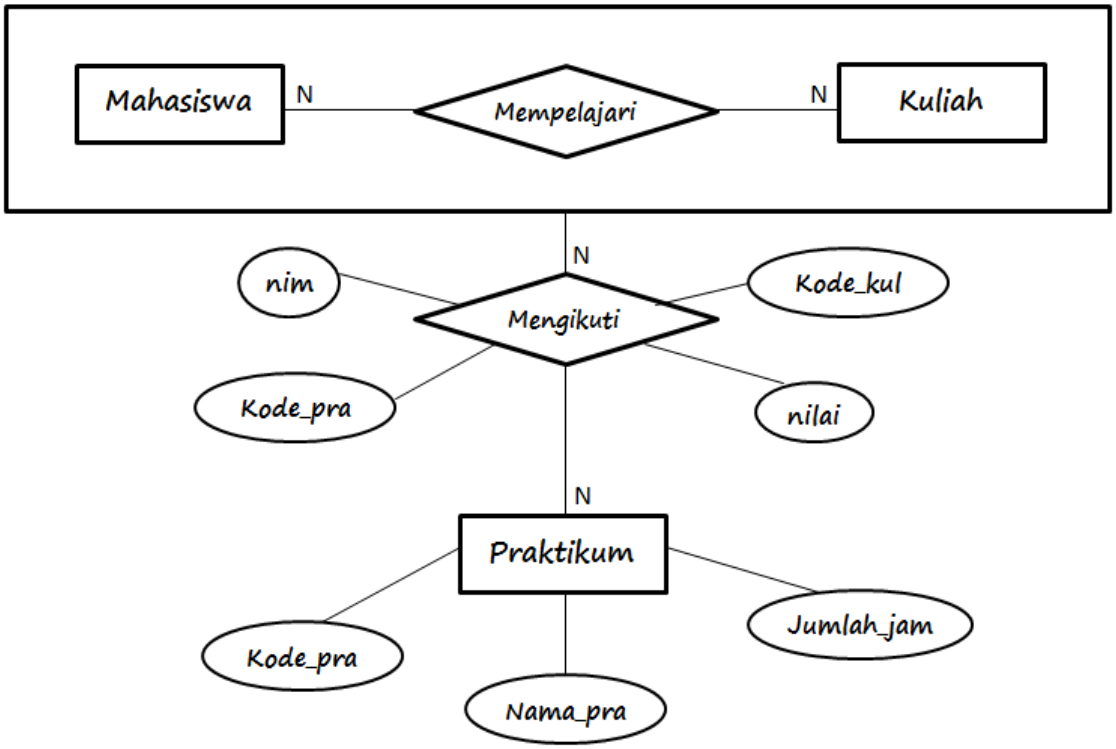


- Tahap Generalisasi menyebabkan penyusutkan jumlah himpunan entitas menjadi sebuah tabel saja.
- Terdapat penambahan atribut untuk mengakomodasikan perbedaan tabel dengan tabel sebelumnya.

Tabel Kuliah

nim	Nama_mhs	Alamat_mhs	Tgl_lahir	Program_studi

Implementasi Agregasi



Tabel Mempelajari / Nilai

nim	Kode_kul	Indeks_nilai

Tabel Praktikum

Kode_pra	Nama_pra	Jumlah_jam

Tabel Mengikuti

nim	Kode_kul	Kode_pra	nilai

- **Relasi dibentuk dari relasi lain (prasyarat)**
- **Pengimplementasian dilakukan setelah relasi prasyarat terimplementasikan.**

DBMS dan Struktur Tabel

- Dalam menentukan struktur tabel khususnya penetapan tipe data dan *feature* tambahan untuk setiap *field* terdapat pilihan :
  * Data angka (numerik atau alfanumerik (Teks))
  * Data numerik (integer, real)
  * Data Bilangan Bulat (integer)
  * Data Uang
  * Data teks
  * *feature* tambahan

Indeks dan Struktur Penyimpanan

1. Indeks Primer (*Primary Index*)
   - IP pada setiap tabel hanya ada satu dan hampir selalu berasal (ditentukan) dari kunci primer yang telah ditetapkan dalam sebuah entitas / relasi.
   - IP yang baik terdiri atas field-field dengan kriteria berikut :
     a. Field yang menjadi komponen IP harus bersifat mandatory (datanya tidak boleh kosong atau bernilai null)
     b. Keseluruhan nilai IP bersifat unik
     c. Nilai-nilainya lebih permanen (idealnya tidak pernah berubah)
     d. Berukuran kecil (pendek) dengan jumlah field minimal (sedikit)

2. Indeks Skunder

   Digunakan untuk mendukung keberadaan IP yang dibuat untuk suatu tabel dengan alasan untuk mempermudah berbagai cara pengaksesan ke suatu tabel.
   Misalnya : field Nama_Mahasiswa → untuk memudahkan pencarian data berdasar nama mahasiswa : disamping pencarian berdasar No_MHS
   Catatan :
   - Jumlah IS dalam sebuah tabel boleh lebih dari Satu
   - Nilai-nilai field yang menjadi pembentuk IS tidak harus bersifat unik

Struktur Penyimpanan

Dalam struktur penyimpanan terdapat enam pilihan dalam struktur penyimpanan dasar yang diterapkan pada suatu tabel (bergantung pada DBMS yang dipakai) yaitu : **Pile, Heap, hash, Sekuensial Berindeks, File Berindeks** dan **Multiring.**

1. **Heap**
   - Merupakan struktur penyimpanan yang paling sederhana dan paling hemat dalam kebutuhan ruang penyimpanan.
   - Setiap baris data disusun secara kronologis penyimpanannya.
   - Record yang pertama akan disimpan dan ditempatkan pada posisi awal ruang penyimpanan dan begitu seterusnya.
   - Pengubahan data tidak akan mengubah urutan record tersebut.
   - Pencarian data berjalan dengan lambat, karena dilakukan sekuensial baris demi baris.

Struktur Penyimpanan

2. Hash

- Baris – baris data ditempatkan berdasar nilai alamat fisik yang diperoleh dari hasil perhitungan (*fungsi hashing*) terhadap nilai *key-nya*.
- Memiliki performansi yang lebih baik dalam pencarian data tunggal berdasar kunci indeks.
- Struktur yang sangat cocok untuk menjadi acuan bagi tabel lainnya.

Kekurangannya :
Membutuhkan ruang penyimpanan awal yang besar, untuk menjamin agar *record- record* yang disimpan tidak menempati alamat yang sama→dibutuhkan alokasi ruang penyimpanan.

Struktur Penyimpanan

3. Sekuensial Berindeks

- Menempatkan data dengan urutan tertentu berdasar nilai indeks primernya. *Record* yang memiliki nilai IP paling kecil dibandingkan *record* yang akan ditempatkan diawal ruang penyimpanan tabel meskipun dimasukkan belakangan.

- Performansi turun pada saat terjadi penambahan atau perubahan data yang menyangkut nilai indeks primernya, karena perlu dilakukan penataan ulang.

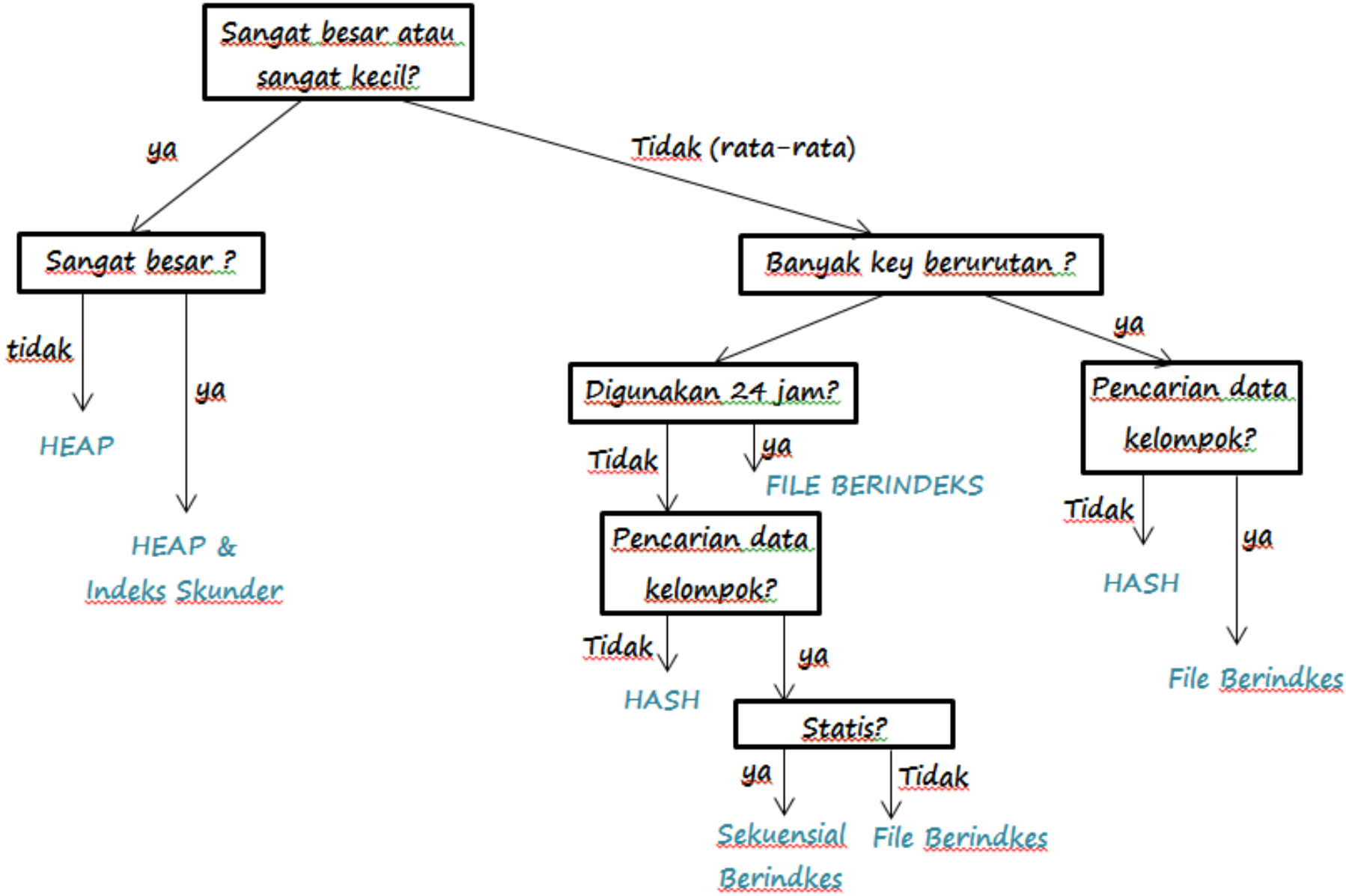
- Struktur ini cocok untuk tabel yang sifatnya statis dan untuk pencarian data kelompok dalam suatu tabel (lebih baik daripada hash).

Struktur Penyimpanan

4. File berindeks

- Dikembangkan dari struktur heap. *Record-record* disusun berdasar kronologis penyimpanannya (seperti heap). Namun disediakan pula file indeks yang disusun berdasar nilai key setiap record yang berguna untuk membantu proses pencarian data ke suatu tabel.
- Terdapat 2 komponen yaitu **komponen data** dan **komponen indeks**.
- Komponen data disusun dengan struktur *heap* dan komponen indeks disusun dengan struktur *sekuensial berindeks*
- Struktur ini sangat cocok untuk tabel yang dinamis dan berukuran besar.

Struktur Penyimpanan



LATIHAN

- Buat Contoh lain untuk transformasi model data ke basis data fisik, berikut ini:

1. Transformasi Umum
2. Implementasi Himpunan Entitas Lemah dan Sub Entitas
3. Implementasi Relasi Tunggal (*Unary Relation*)
4. Implementasi Relasi Multi Entitas (*N-ary Relation*)
5. Implementasi Relasi Ganda (*Redundant Relation*)
6. Implementasi Spesialisasi dan Generalisasi
7. Implementasi Agregasi