Database Technology

Pemodelan Entiry Relationalship

TEKNIK INFORMATIKA

KODE MK : TIF19410

SKS : 2/2

Pemodelan data adalah langkah pertama dalam perancangan *database*, berfungsi sebagai jembatan antara obyek dunia nyata dan model *database* yang diimplementasikan dalam perangkat komputer.

Detil pemodelan data diekspresikan secara grafis melalui diagram *entity relationship* (ERD)

Capaian Pembelajaran

Mahasiswa/i mampu membangun sebuah model *Entity Relationship* (ERM) melalui diagram *Entity Relationship* (ERD) dengan memenuhi kaidah-kaidah pemodelan E-R serta sesuai dengan kebutuhan bisnis & organisasi.

MODEL *ENTITY RELATIONSHIP*

Entity Relationship Model (ERM)

- ERM menjadi landasan pembuatan ERD
- ERD merupakan perwujudan *database* konseptual yang terlihat langsung oleh user
- ERD menggambarkan komponen utama *database*: *entity*, *attribute*, dan *relationship*

Notasi

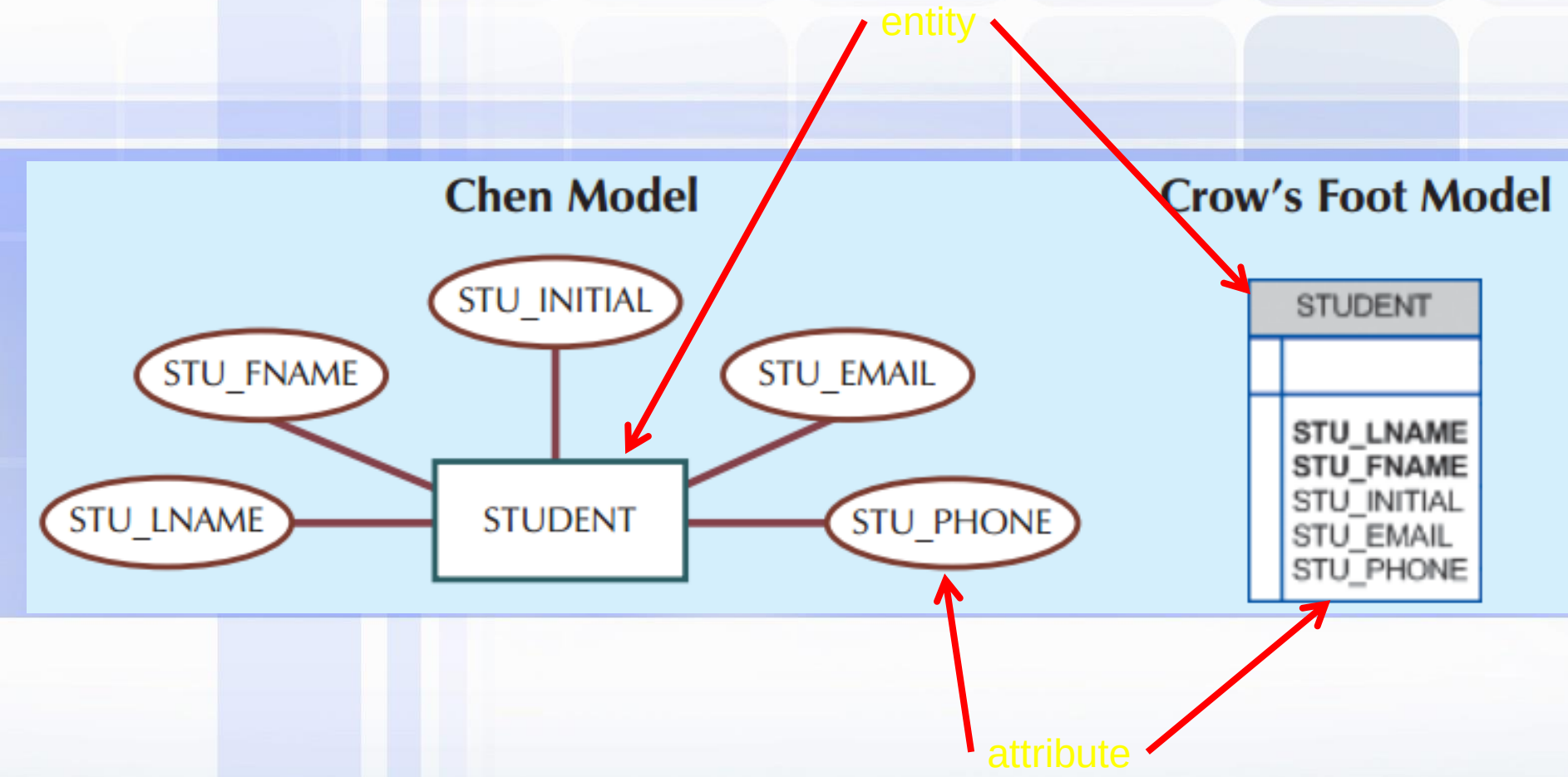
1. Chen lebih cocok digunakan untuk pemodelan konseptual
2. *Crow's foot* lebih cocok digunakan untuk pendekatan berorientasi implementasi
3. UML (*unified modelling language*) dapat digunakan untuk pemodelan konseptual maupun implementasi

ENTITY

- *Entity* adalah sebuah obyek yang menjadi perhatian *user*
- Setiap baris pada tabel *entity* adalah sebuah instan *entity*
- Pada notasi Chen and Crow's Foot, sebuah *entity* digambarkan dengan simbol persegi empat berisi nama *entity*
- Nama *entity*, dalam bentuk kata benda, biasanya ditulis seluruhnya dengan huruf kapital

ATTRIBUTE

- *Attribute* adalah karakteristik dari sebuah *entity*
- Pada notasi Chen notation, *attribute* dibuat dengan simbol oval yang dihubungkan dengan garis ke persegi empat *entity*
- Pada notasi *Crow's Foot*, *attribute* dituliskan dalam kotak *attribute* di bawah persegi empat *entity*



Catatan:

Tidak ada aturan baku untuk penulisan *attribute* selama konsisten

- Required attribute

  attribute yang harus memiliki sebuah nilai dan tidak boleh kosong (null)

  ditulis tebal pada notasi *Crow's foot*

- Optional attribute

  attribute yang tidak mengharuskan ada nilai dan boleh dikosongkan

- Domain

  kumpulan nilai yang dimungkinkan untuk sebuah *attribute* tertentu, misalnya:

  domain untuk *attribute* indeks prestasi kumulatif (IPK) adalah 0 sampai 4

  domain untuk *attribute* kelamin terdiri dari hanya 2 kemungkinan: L atau P

- Identifier (Primary Key)

  satu atau lebih *attribute* yang secara unik membedakan setiap instan pada *entity*

  identifier diberi garis bawah dalam ERD

  NAMA TABEL (<u>Attribute1</u>, Attribute2, Attribute3, ...Attribute ke-*n*)

  idealnya sebuah *identifier* hanya terdiri dari sebuah *attribute* tunggal

- Composite Identifier

- Composite attribute

  berbeda dengan *composite key*

  adalah sebuah *attribute* yang dapat dipecah menjadi beberapa *attribute*, misal alamat, nomor telpon, tanggal lahir

- Simple attribute

  is adalah *attribute* yang tidak dapat dipecah lagi, misal umur, kelamin

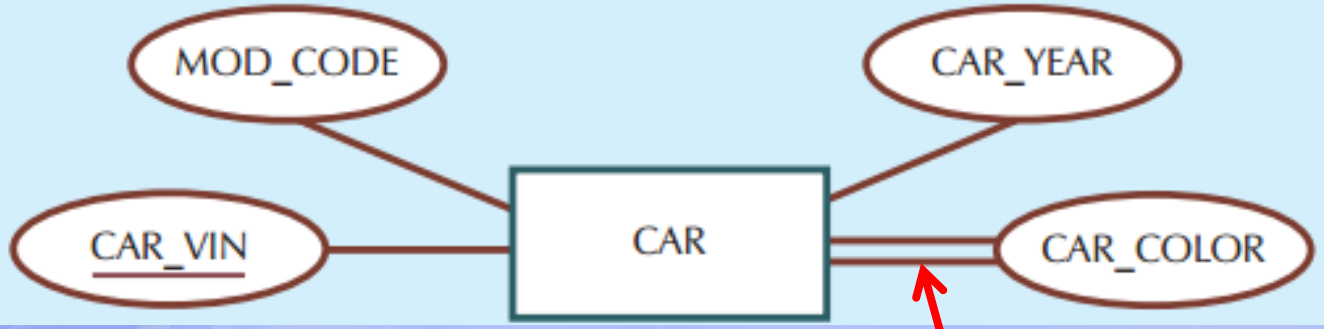
- Single-valued attribute

  adalah *attribute* yang hanya dapat berisi nilai tunggal, misal nomor KTP, nomor plat kendaraan
- Multivalued attribute

  adalah *attribute* yang dapat berisi banyak nilai, misal nomor telpon, gelar, warna mobil

  notasi Chen memungkinkan hal ini, namun *Crow's foot* tidak dapat

Chen Model



Crow's Foot Model

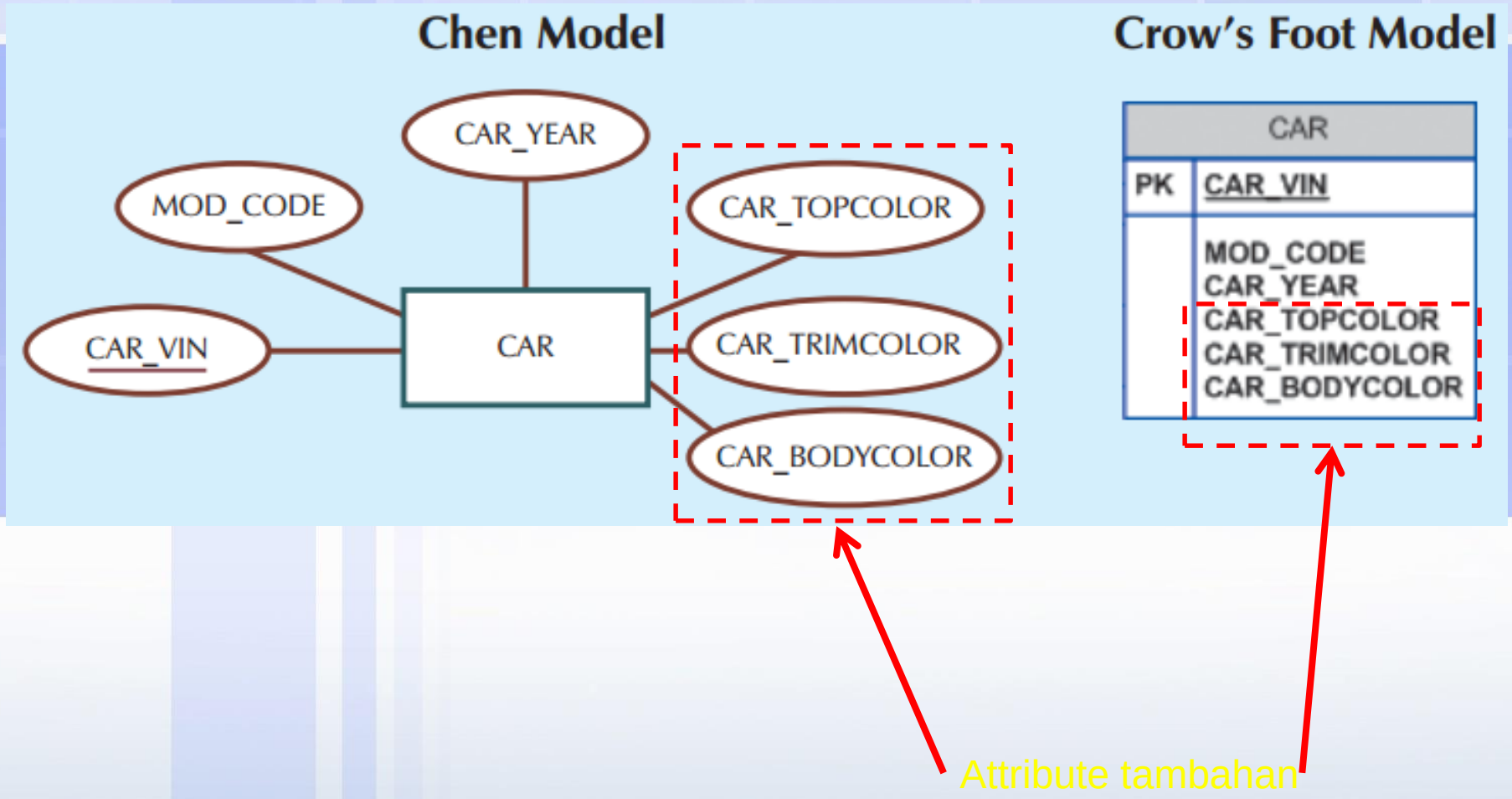
CAR	
PK	CAR_VIN
	MOD_CODE CAR_YEAR CAR_COLOR

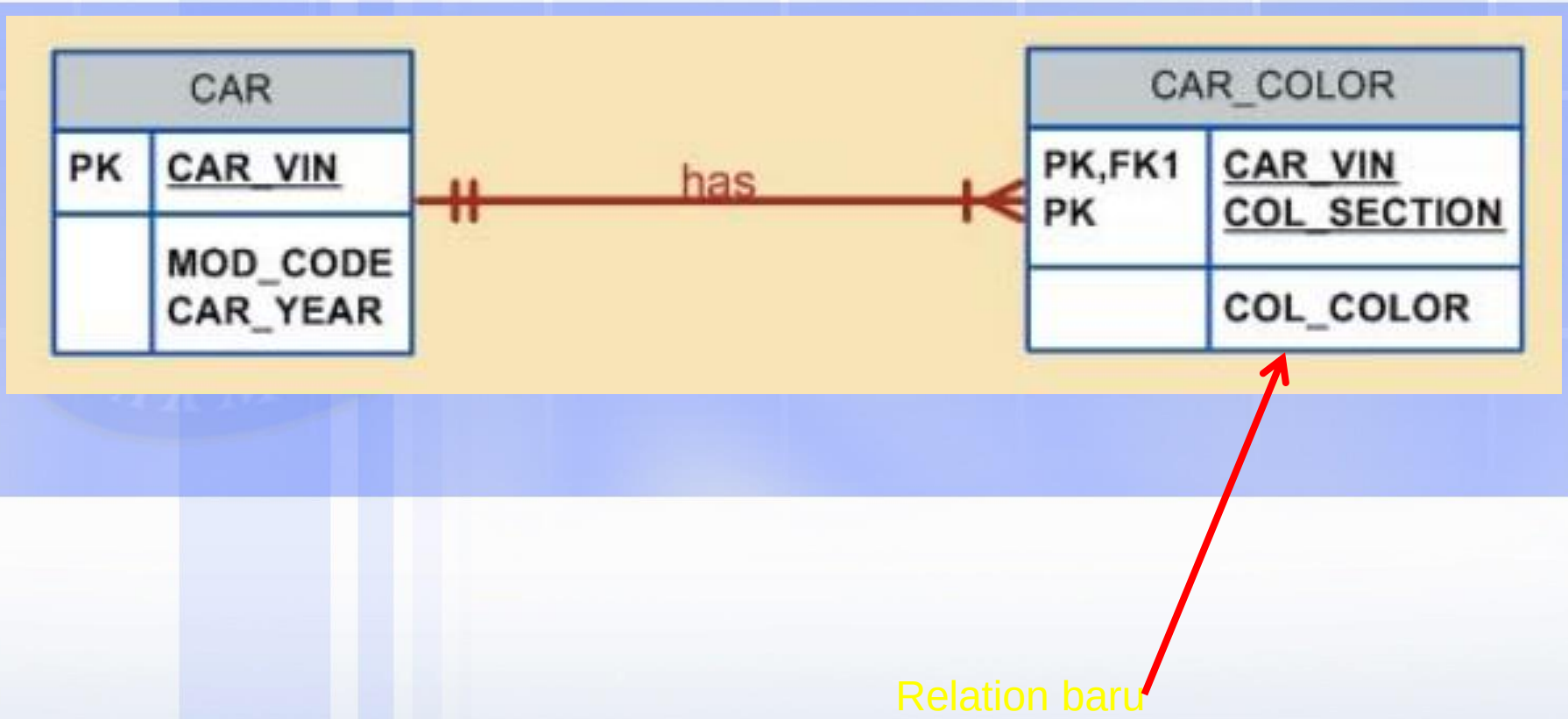
Multivalued attribute

Permasalahan implementasi *multivalued attribute* adalah satu *cell* bisa berisi lebih dari satu nilai

Diatasi dengan cara:

1. Membuat *attribute* tambahan yang nilai didalamnya sejenis
2. Membuat *relation* baru yang terhubung dengan *relation* asal


Chen Model
Crow's Foot Model
CAR_YEAR
MOD_CODE
CAR_TOPCOLOR
CAR_VIN
CAR
CAR_TRIMCOLOR
CAR_BODYCOLOR
CAR
PK
CAR_VIN
MOD_CODE
CAR_YEAR
CAR_TOPCOLOR
CAR_TRIMCOLOR
CAR_BODYCOLOR
Attribute tambahan


CAR
PK
CAR_VIN
MOD_CODE
CAR_YEAR
has
CAR_COLOR
PK,FK1
PK
CAR_VIN
COL_SECTION
COL_COLOR
Relation baru

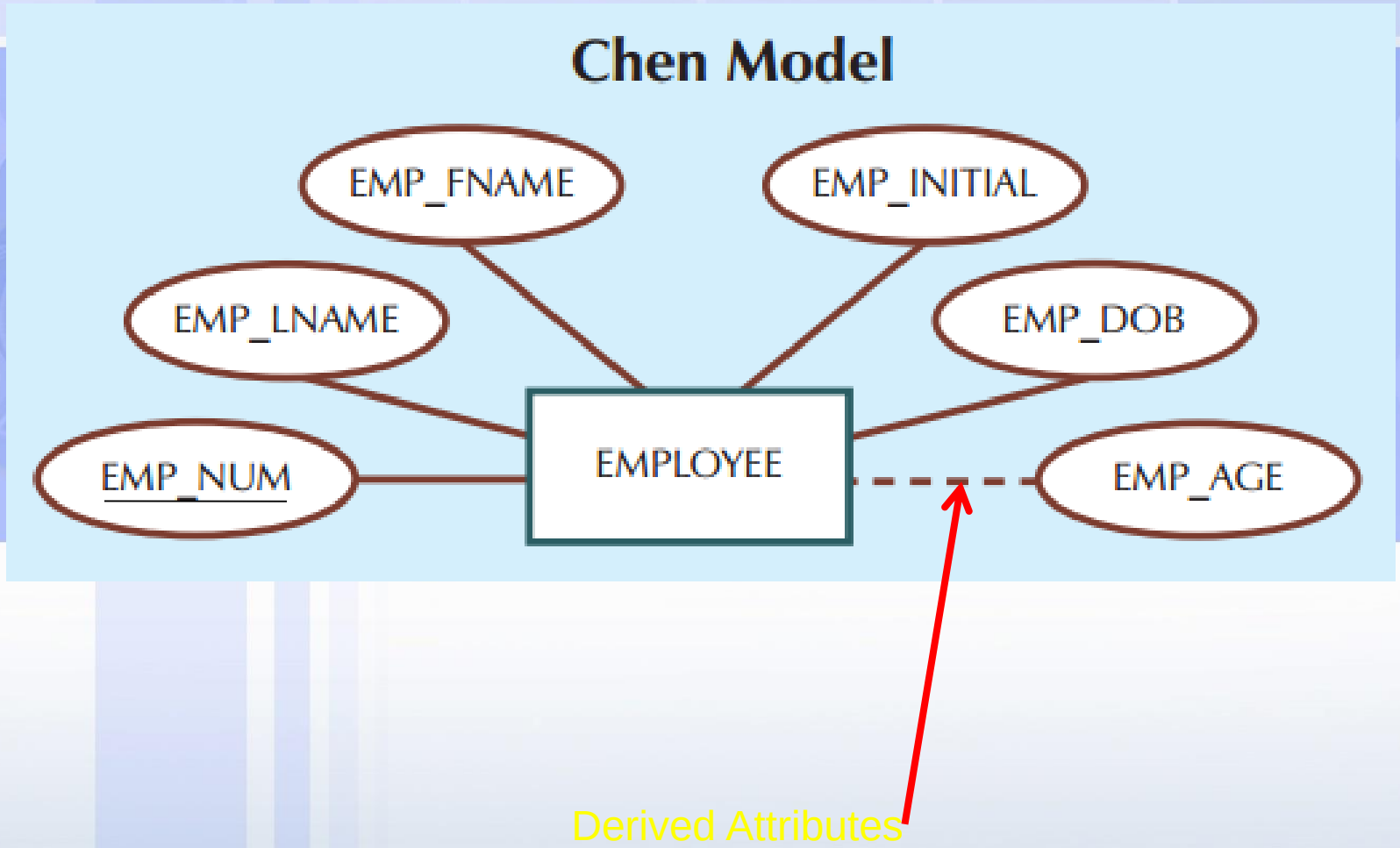
- Derived Attributes

  adalah *attribute* yang nilainya diperoleh dari hasil kalkulasi *attribute* lain

  dapat dihasilkan menggunakan algoritma

  sering disebut juga sebagai *computed attribute*

	STORED	NOT STORED
Advantage	Saves CPU processing cycles Saves data access time Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Increases data access time Adds coding complexity to queries


Chen Model
EMP_FNAME
EMP_INITIAL
EMP_LNAME
EMP_DOB
EMP_NUM
EMPLOYEE
EMP_AGE
Derived Attributes

RELATIONSHIP

- *Relationship* adalah sebuah hubungan/asosiasi di antara *entity*
- Nama *relationship* menggunakan kata kerja aktif atau pasif, misal:
  - satu STUDENT mengambil satu CLASS
  - satu DIVISION dikelola oleh satu
  - EMPLOYEE

- *Relationship* di antara *entity* selalu bersifat 2 arah, misal antara *entity* CUSTOMER dan INVOICE akan dijelaskan sebagai berikut:
  - satu CUSTOMER memiliki banyak INVOICE
  - satu INVOICE dimiliki oleh satu CUSTOMER

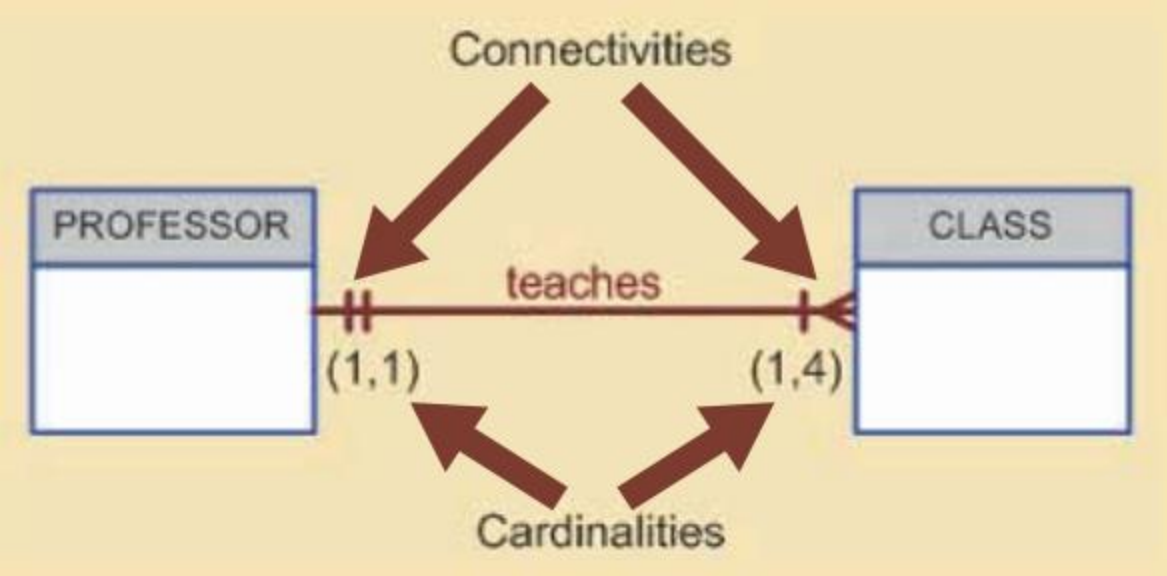
CONNECTIVITY & CARDINALITY

- Connectivity

  digunakan untuk menjelaskan hubungan dari *relationship*
- Cardinality

  menunjukkan jumlah minimum dan maksimum keberadan satu instan *entity* dengan instan *entity* lain yang berhubungan

  dalam ERD, *cardinality* dituliskan dalam format (x,y) di samping *entity*



Perhatikan *relationship* "PROFESSOR mengajar CLASS" dalam diagram di atas:

- *cardinality* (1,4) ditulis di samping CLASS berarti satu professor mengajar maksimum empat class
- *cardinality* (1,1) ditulis di samping PROFESSOR berarti satu class diajar oleh satu dan hanya satu professor

RELATIONSHIP STRENGTH

- Konsep *relationship strength* didasarkan pada bagaimana *primary key* dari *entity* yang berhubungan didefinisikan/dijelaskan
- Ada kemungkinan bahwa *foreign key* adalah juga menjadi komponen *primary key* pada *entity* yang berhubungan

Weak (Non-identifying) Relationships

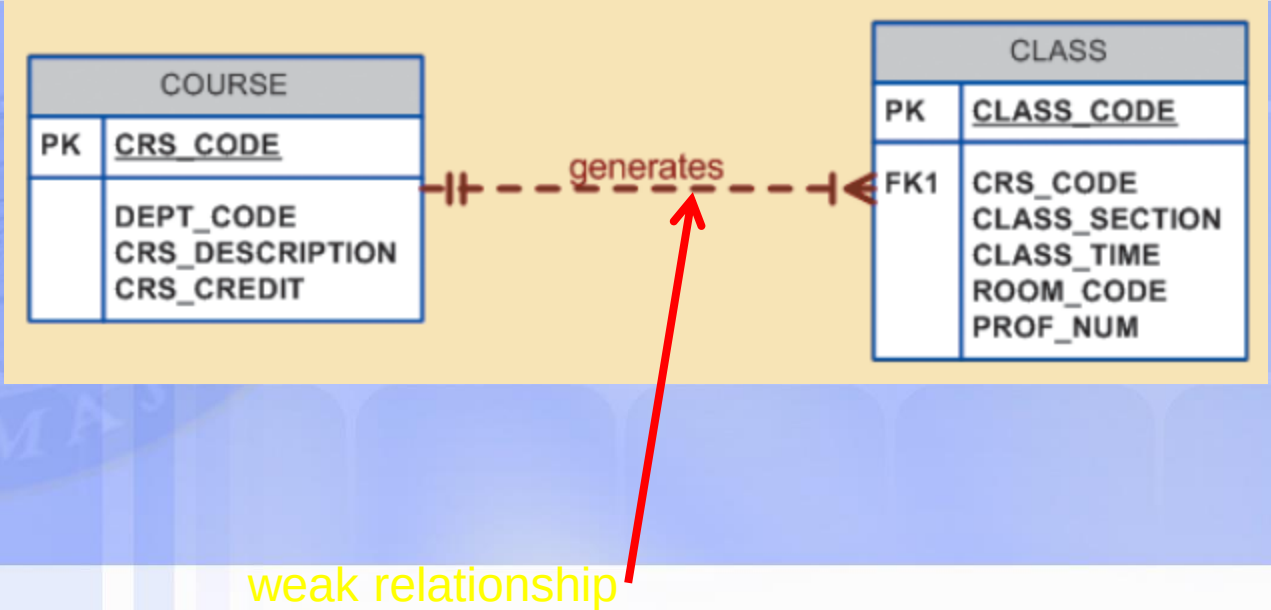
Sebuah *weak relationship*, disebut juga *non-identifying relationship*, terjadi pada saat PK *entity* yang berhubungan tidak mengandung komponen PK dari *entity* asalnya

Contoh:

COURSE (CRS_CODE, DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)

CLASS (CLASS_CODE, *CRS_CODE*, CLASS_SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

weak relationship terjadi antara COURSE dan CLASS karena CrsCode (PK di *entity* COURSE) tidak menjadi komponen PK di *entity* CLASS, tapi hanya menjadi FK



Notasi *Crow's Foot* menggambarkan *weak relationship* dengan garis putus-putus pada garis *relationship* antar *entity*

Strong (Identifying) Relationships

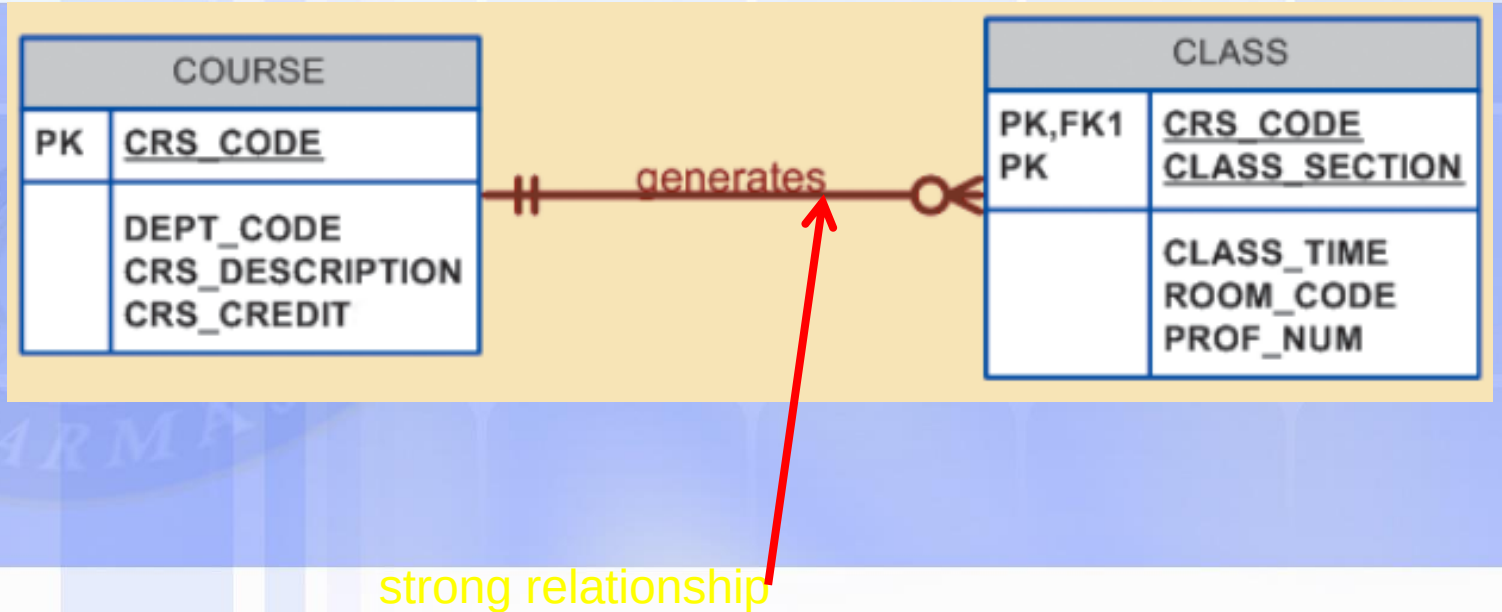
Sebuah *strong relationship*, disebut juga *identifying relationship*, terjadi saat PK dari *entity* yang berhubungan mengandung komponen PK dari *entity* asalnya

Contoh:

COURSE (CRS_CODE, DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)

CLASS (*CRS_CODE*, CLASS_SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

strong relationship terjadi antara *entity* COURSE dan CLASS, karena *composite* PK dari *entity* CLASS terdiri dari komponen CRS_CODE (PK dari *entity* COURSE) dan CLASS_SECTION



Notasi *Crow's Foot* menggambarkan *strong relationship* dengan garis utuh pada garis *relationship* antar *entity*

PARTISIPASI *RELATIONSHIP*

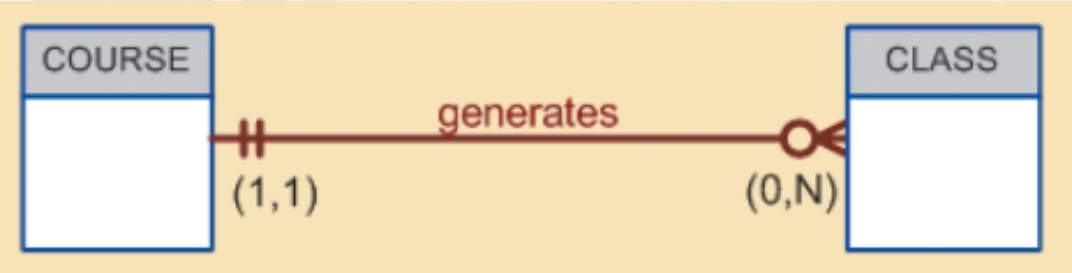
- Optional participation

  berarti bahwa satu instan *entity* tidak memerlukan adanya kaitan dengan instan pada *entity* lain yang berhubungan

- Mandatory participation

  berarti bahwa satu instan *entity* memerlukan adanya kaitan dengan instan pada *entity* lain yang berhubungan

CROW'S FOOT SYMBOL	CARDINALITY	COMMENT
	(0,N)	Zero or many. Many side is optional.
	(1,N)	One or many. Many side is mandatory.
	(1,1)	One and only one. 1 side is mandatory.
	(0,1)	Zero or one. 1 side is optional.



CLASS adalah *optional*

Asumsinya adalah kampus membuat *entity* COURSE baru kemudian membuat *entity* CLASS saat membuat jadwal kuliah

Kemungkinannya adalah ada course yang tidak dibuka class-nya setiap semester.

CLASS adalah *mandatory*

Asumsinya adalah mengikuti pernyataan *relationship* bahwa "satu COURSE menghasilkan satu atau lebih CLASS"

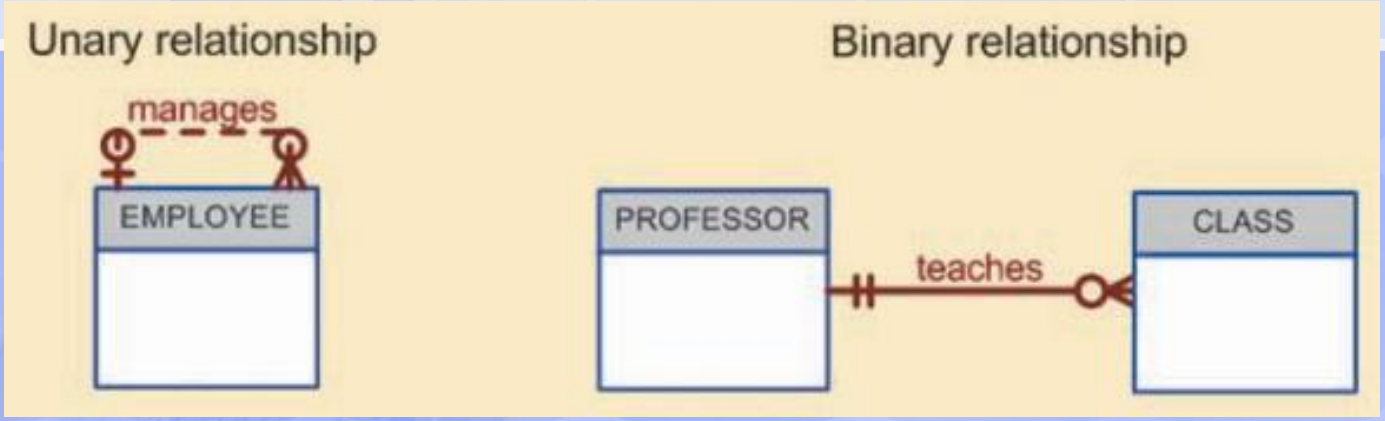
Kondisinya adalah *entity* CLASS harus dibuat saat *entity* COURSE juga dibuat. Sehingga satu COURSE harus memiliki sedikitnya satu CLASS.

DERAJAT *RELATIONSHIP*

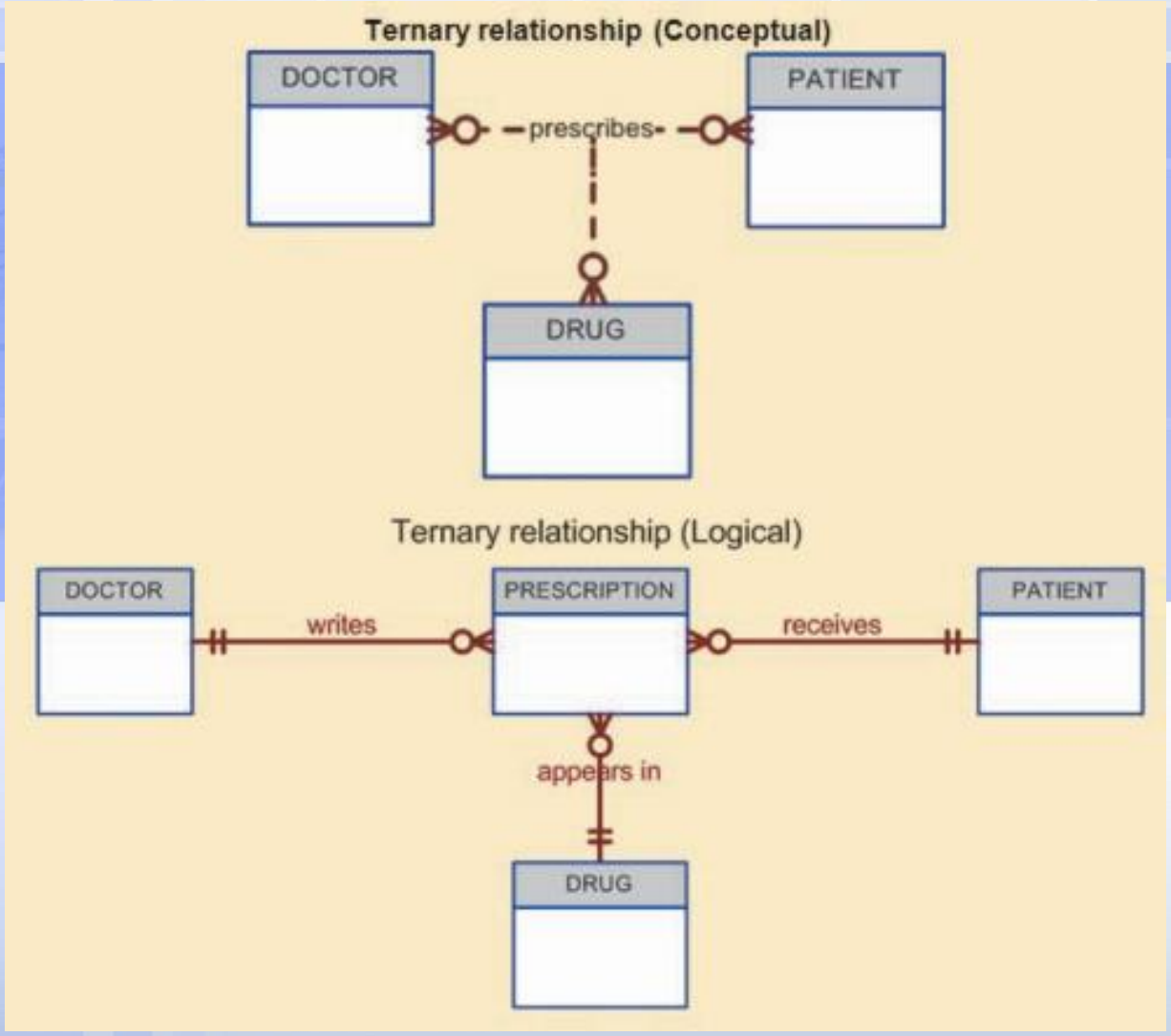
Sebuah *relationship degree* menunjukkan jumlah *entity* yang berhubungan dengan sebuah *relationship*

1. *Unary relationship* terjadi saat hubungan ada hanya pada *entity* tunggal. Disebut juga sebagai *recursive relationship*
2. *Binary relationship* terjadi saat dua *entity* saling berhubungan
3. *Ternary relationship* terjadi saat tiga *entity* saling berhubungan

Degree yang lebih tinggi bisa saja ada, namun sangat jarang dan belum ada istilahnya



- Seorang employee dalam *entity* EMPLOYEE adalah manager untuk satu atau lebih employee dalam *entity* tersebut adalah *unary relationship*
- Seorang PROFESSOR mengajar satu atau lebih CLASS adalah *binary relationship*


Ternary relationship (Conceptual)
DOCTOR
prescribes
PATIENT
DRUG
Ternary relationship (Logical)
DOCTOR
writes
PRESCRIPTION
receives
PATIENT
appears in
DRUG

Contoh *ternary relationship*

- seorang DOCTOR menulis satu atau lebih PRESCRIPTION
- seorang PATIENT menerima satu atau lebih PRESCRIPTION
- satu DRUG muncul di satu atau lebih PRESCRIPTION

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_SPOUSE
345	Ramirez	James	347
346	Jones	Anne	349
347	Ramirez	Louise	345
348	Delaney	Robert	
349	Shapiro	Anton	346

EMP_CODE	EMP_LNAME	EMP_MANAGER
101	Waddell	102
102	Orincona	
103	Jones	102
104	Reballoh	102
105	Robertson	102
106	Deltona	102

MENGEMBANGKAN ERD

Membangun sebuah ERD biasanya mengikuti aktifitas berikut ini:

1. Buat narasi detil dari operasi yang dijelaskan oleh organisasi
2. Identifikasi business rule berdasarkan deskripsi operasi di atas
3. Identifikasi entity utama dan relationship dari business rule
4. Bangun ERD awal
5. Identifikasi attribute dan primary key yang cocok dan tepat untuk mendeskripsikan entitie
6. Tinjau dan perbaiki ERD tersebut

TANTANGAN PERANCANGAN *DATABASE*

Seorang perancang *database* biasanya harus membuat rancangan dengan kompromi yang dipicu oleh kemauan yang bertentangan:

1. Standar perancangan
2. Kecepatan proses
3. Kebutuhan informasi

1# standar perancangan

- Rancangan *database* harus memenuhi standar perancangan.
- Standar ini membantu dalam mengembangkan struktur lojik yang mengurangi redundansi data, sehingga mengurangi kemungkinan terjadinya *modification problems*.
- Standar juga membantu sebisa mungkin untuk menghindari terjadinya *null*.

2# kecepatan proses

- Kecepatan proses yang tinggi biasanya menjadi prioritas utama dalam perancangan *database*.
- Kecepatan proses yang tinggi berarti waktu akses yang cepat, yang mungkin bisa tercapai dengan mengurangi jumlah dan kompleksitas *relationship*.

3# kebutuhan informasi

- Kebutuhan informasi yang kompleks dapat mempengaruhi transformasi data, dan menambah jumlah *entity* dan *attribute* dalam rancangan.
- Sehingga, database mungkin harus mengorbankan beberapa struktur rancangan yang "clean" dan/atau sejumlah kecepatan transaksi untuk memastikan informasi dihasilkan lebih banyak.

TERIMA KASIH