



UNIVERSITAS  
INDONESIA

*Veritas, Probitas, Justitia*

FAKULTAS  
ILMU  
KOMPUTER

Slide 4

# Pemodelan Basis Data dengan EER

---

CSF2600700 - BASIS DATA

SEMESTER GENAP 2018/2019



---

**The main reference of this presentation is the textbook and PPT from : Elmasri & Navathe, Fundamental of Database Systems, 7<sup>th</sup> edition, 2015, Chapter 3**

**Additional resources: presentation prepared by Prof Steven A. Demurjian, Sr  
(<http://www.engr.uconn.edu/~steve/courses.html>)**



# Tujuan Pemelajaran

Setelah mengikuti pemelajaran pada topik ini, jika diberikan *requirement* basis data, Anda diharapkan dapat memodelkan basis data dengan tepat menggunakan *Enhanced Entity Relationship Diagram*



# Outline

1. Latar Belakang

2. *Superclass/Subclass Relationship*

3. Spesialisasi dan Generalisasi

4. *Hierarchy* dan *Lattice*

5. Pemodelan dengan *Categories*

6. *Higher Degree Relationship*

7. Kapan Kita Menggunakan EER?

# Latar Belakang

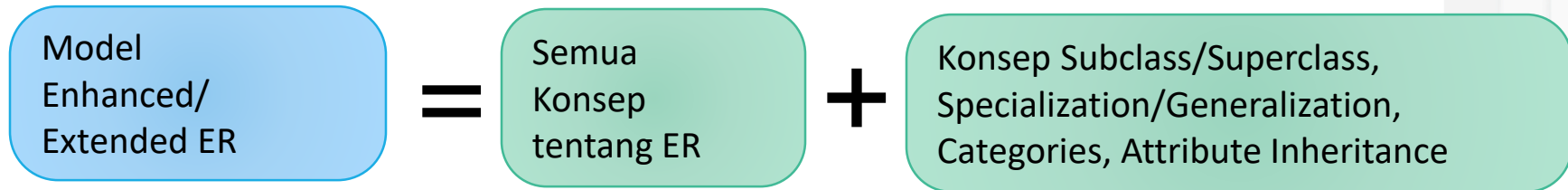


# Mengapa Perlu *Enhanced* ER?

- ER cukup digunakan untuk memodelkan skema basis data 'tradisional' (aplikasi pemrosesan data pada bisnis dan industri pada umumnya)
- Sejak akhir tahun 70-an, dirasakan perlu untuk merancang skema dapat merepresentasikan sifat-sifat dan batasan-batasan data dengan lebih tepat, terutama untuk aplikasi-aplikasi baru di berbagai bidang (CAD, CAM, GIS, dll)
- Hal ini memacu perkembangan konsep-konsep *semantic data modeling* yang ditambahkan ke model ER yang telah ada



# Konsep-Konsep Model EER



- ◆ Model EER digunakan untuk merepresentasikan aplikasi dengan lebih lengkap dan lebih akurat, **jika diperlukan**
- ◆ Model EER mengandung beberapa konsep *object oriented*, misal: *inheritance*



---

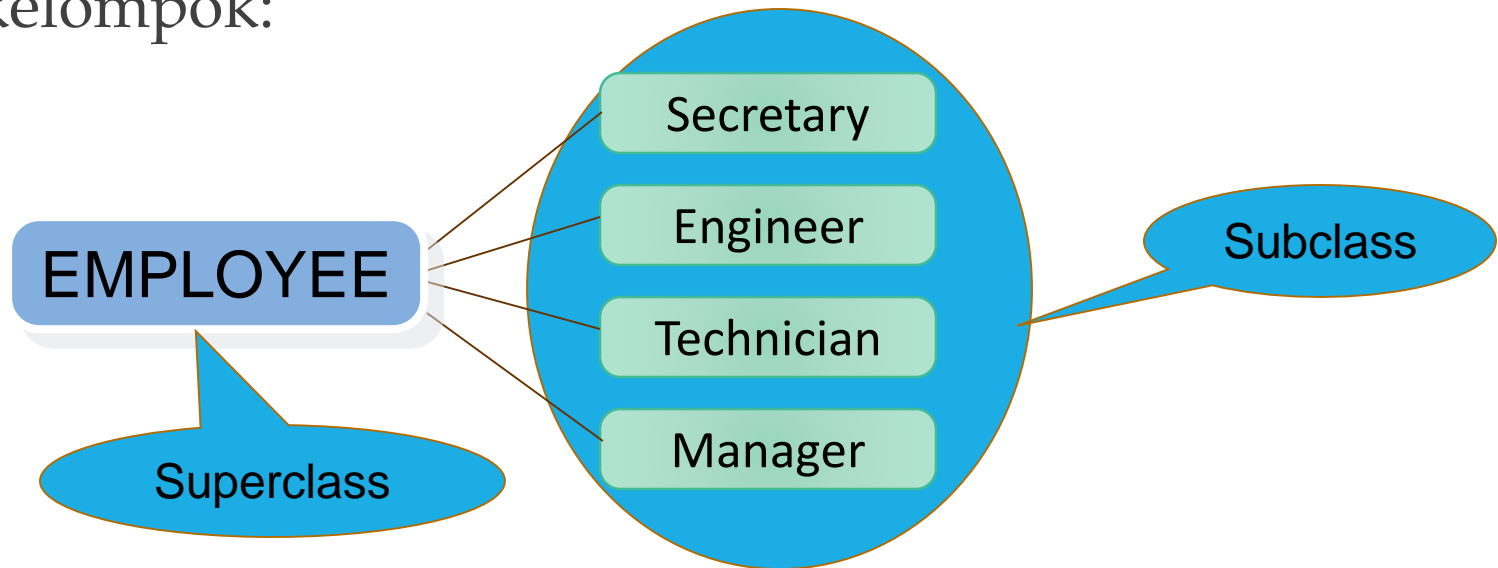
# *Superclass/Subclass Relationship*





# Subclass dan Superclass

- Misal EMPLOYEE dapat dikategorikan menjadi 4 kelompok:



- ◆ *Subclass* merepresentasikan *entity* yang sama dengan *superclass*, namun memiliki peran spesifik tertentu.
- ◆ *Entity* dalam *subclass* merupakan anggota *superclass*, namun tidak sebaliknya



# *Superclass/Subclass Relationship*

- *Superclass/ Subclass Relationship* adalah *relationship* antara sebuah *superclass* dengan salah satu *subclass*-nya.
- Contoh:  
Employee/Secretary, Employee/Technician
- Disebut juga dengan IS-A *relationship*
  - SECRETARY IS AN EMPLOYEE
  - TECHNICIAN IS AN EMPLOYEE



# Type Inheritance

- Suatu *entity* yang merupakan anggota sebuah *subclass* mewarisi (*inherits*)
  - semua *attribute* dan
  - semua *relationship*dari *entity* yang merupakan anggota *superclass*.

# Spesialisasi dan Generalisasi

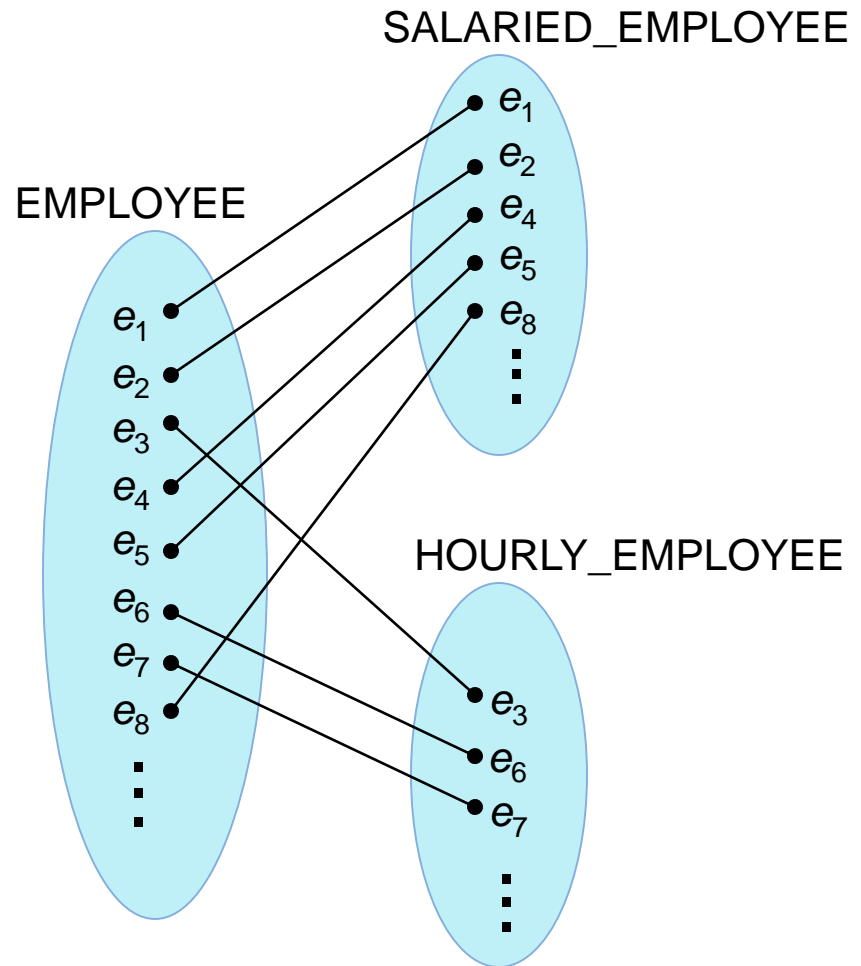
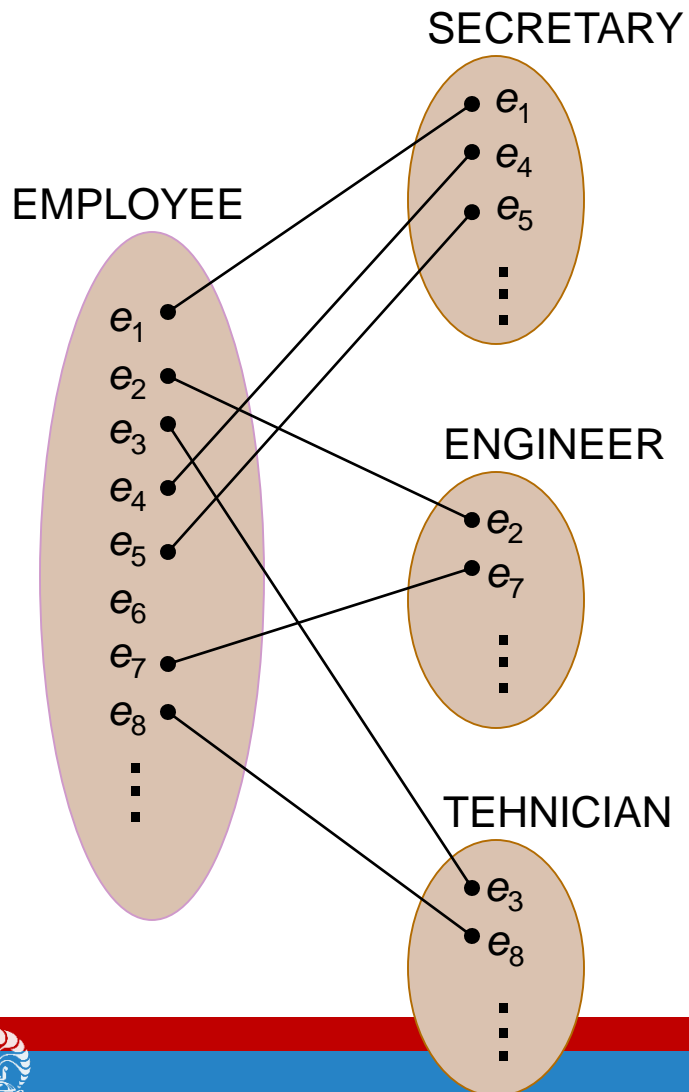


# Spesialisasi

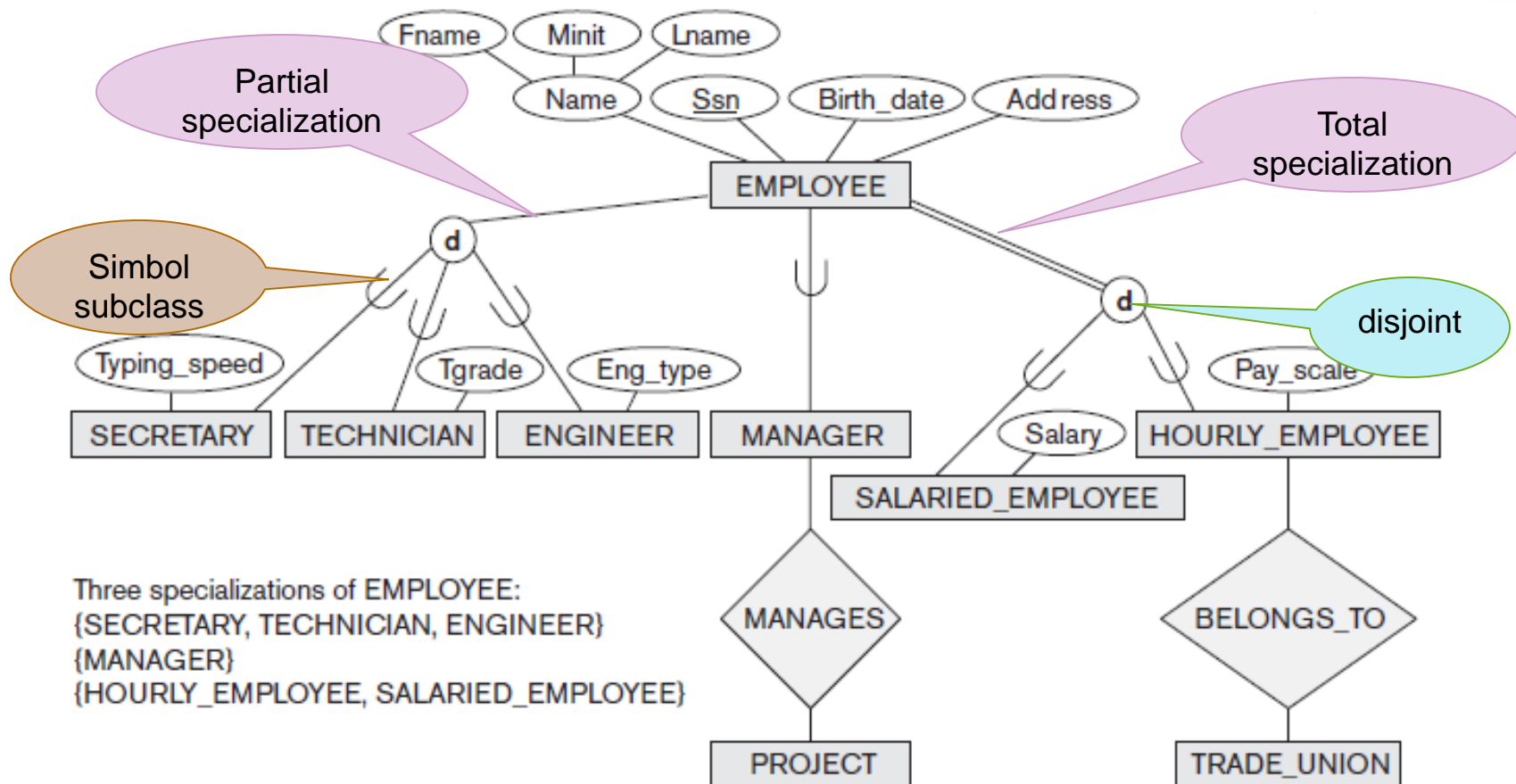
- Spesialisasi adalah proses mendefinisikan himpunan *subclass-subclass* dari sebuah *entity type* (*superclass*)
- Dilakukan berdasarkan karakteristik tertentu yang dapat membedakan *entity* pada *superclass*
- Suatu *superclass* dapat memiliki beberapa spesialisasi berdasarkan karakteristik yang berbeda
- Contoh:
  - SECRETARY, ENGINEERS, TECHNICIAN adalah spesialisasi dari EMPLOYEE berdasarkan attribute *job\_type*
  - SALARIED\_EMPLOYEE dan HOURLY\_EMPLOYEE adalah spesialisasi dari EMPLOYEE berdasarkan metode pembayarannya.



# Contoh Spesialisasi



# Notasi Spesialisasi dalam EER





# Manfaat Spesialisasi

- Mendefinisikan himpunan *subclass-subclass* dari suatu *entity type*
- Menggambarkan *attribute* spesifik untuk tiap *subclass*
- Menggambarkan *relationship* spesifik antara suatu *subclass* dengan *entity type* lain atau dengan *subclass* lain





# Generalisasi

---

- Kebalikan dari proses spesialisasi
- Dilakukan dengan mengidentifikasi *attribute-attribute* yang sama dan melakukan generalisasi ke sebuah *superclass*
- Contoh:
  - TRUCK & CAR dapat digeneralisasi menjadi VEHICLE

# Contoh Generalisasi

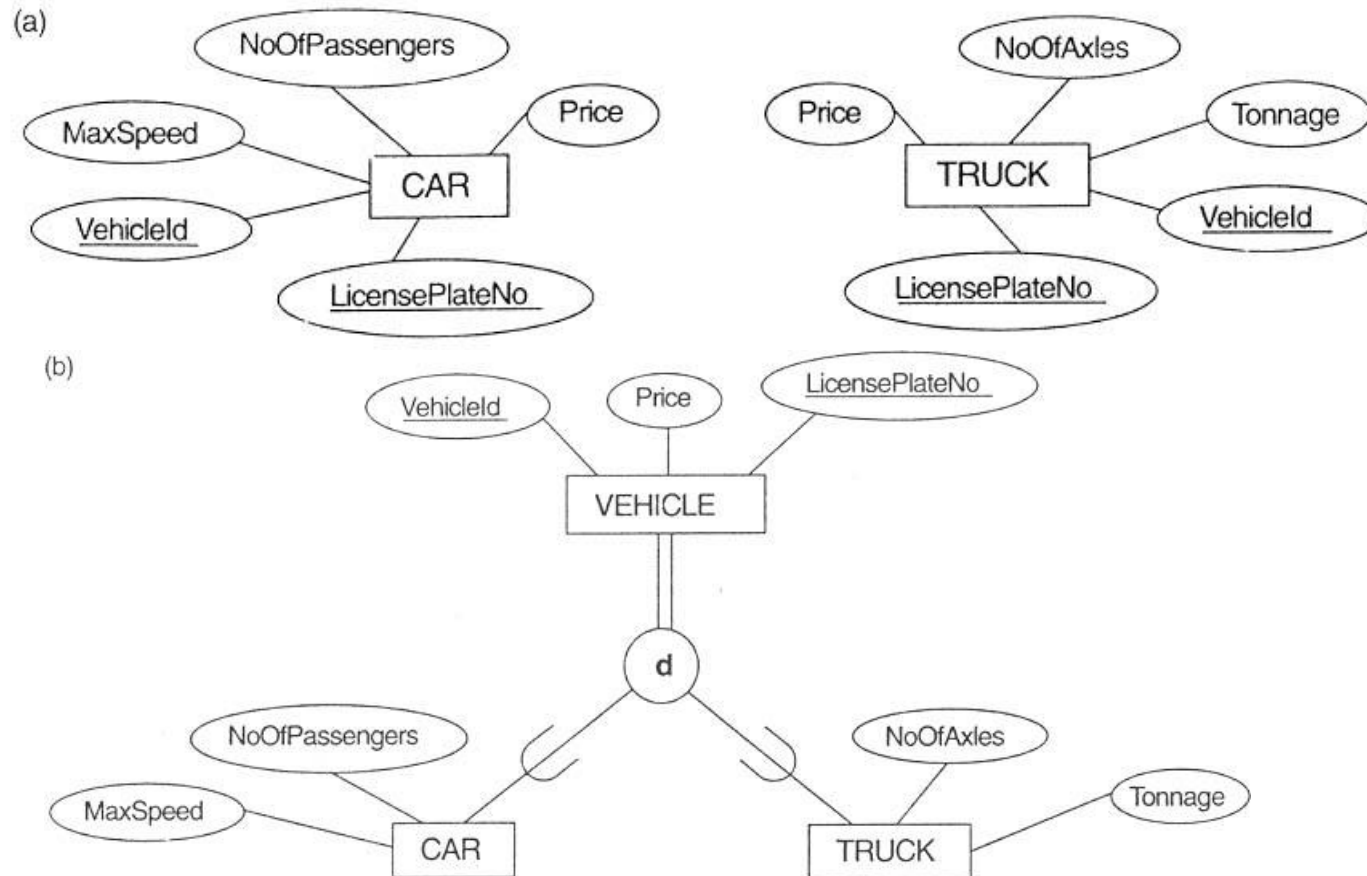


Figure 4.3 Examples of generalization. (a) Two entity types CAR and TRUCK. (b) Generalizing CAR and TRUCK into VEHICLE.

# Generalisasi vs Spesialisasi

- Kadang-kadang notasi spesialisasi dan generalisasi dibedakan:
  - Arah panah menuju *superclass* menunjukkan generalisasi
  - Arah panah menuju *subclass* menunjukkan spesialisasi
- Di sini kita tidak membedakan notasi dengan arah panah, karena seringkali subyektif sesuai dengan proses yang dilakukan pada suatu situasi tertentu.

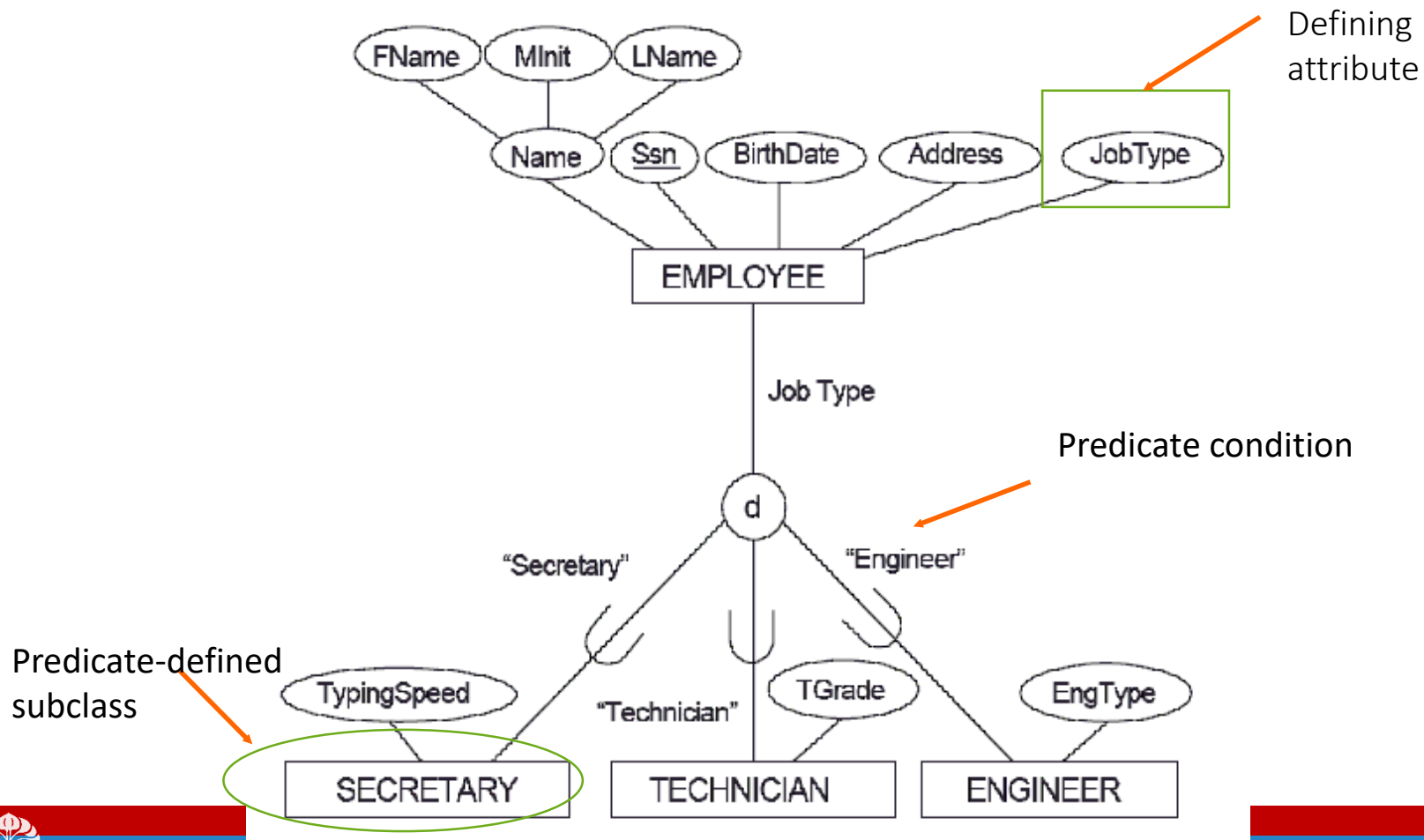
# *Constraints* untuk Spesialisasi dan Generalisasi

---

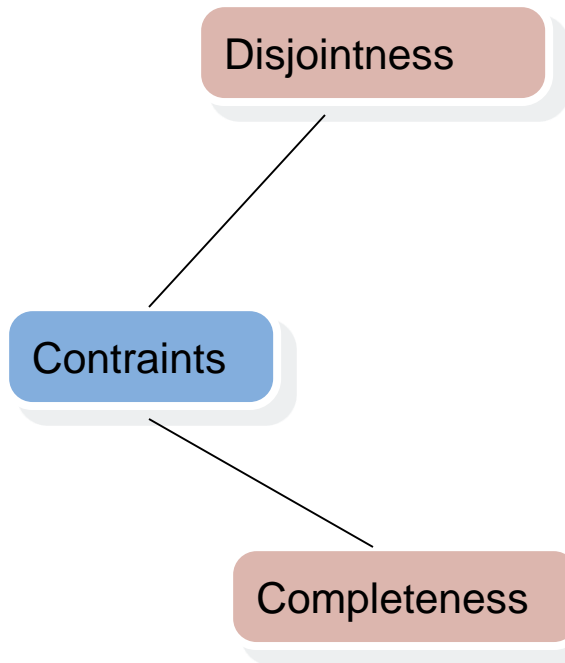
- Spesialisasi berdasarkan *attribute*
  - Spesialisasi dilakukan berdasarkan *attribute* dari *superclass* (*defining attribute*)
  - Contoh: *job\_type*
- *Subclass* yang ditentukan pengguna
  - Keanggotaan *entity* dalam suatu *subclass* ditentukan oleh pengguna

# Constraints untuk Spesialisasi dan Generalisasi

**Figure 4.4** An attribute-defined specialization on the JobType attribute of EMPLOYEE.



# Constraints untuk Spesialisasi dan Generalisasi



**d (disjoint)** Simbol **d (disjoint)** menyatakan bahwa sebuah *entity* hanya bisa menjadi anggota dari satu *subclass*.

**O (overlap)** Simbol **o (overlap)** menyatakan bahwa sebuah *entity* dapat menjadi anggota lebih dari satu *subclass*.

**= Total** Simbol **= Total** menyatakan bahwa setiap *entity* pada *superclass* menjadi anggota *subclass*. Dinyatakan dengan garis dobel.

**- Parsial** Simbol **- Parsial** menyatakan bahwa suatu *entity* boleh tidak merupakan anggota *subclass* manapun. Dinyatakan dengan garis tunggal.

# *Constraints* untuk Spesialisasi dan Generalisasi

---

- Dari *constraints* tersebut, ada 4 macam bentuk spesialisasi/generalisasi
  - *Disjoint*, total
  - *Disjoint*, parsial
  - *Overlap*, total
  - *Overlap*, parsial
- Generalisasi umumnya bersifat **total** karena *superclass* diturunkan dari *subclass-subclass*-nya.

# Constraints untuk Spesialisasi dan Generalisasi

## CONTOH KASUS:

- *Disjoint partial:*

Pada basisdata sebuah perusahaan, data EMPLOYEE yang memiliki subclass SECRETARY, TECHNICIAN, dan ENGINEER. Seorang pegawai boleh merupakan SECRETARY atau TECHNICIAN atau ENGINEER atau tidak ketiganya (misalnya ACCOUNTANT), namun tidak boleh memiliki pekerjaan lebih dari satu, misalnya SECRETARY sekaligus TECHNICIAN atau TECHNICIAN sekaligus ENGINEER.

- *Disjoint total:*

Merujuk pada contoh kasus sebelumnya, namun dalam hal ini seorang PEGAWAI HARUS memiliki SALAH SATU pekerjaan diantara SECRETARY, TECHNICIAN, atau ENGINEER. Jadi dalam kasus ini TIDAK ADA PEGAWAI yang memiliki pekerjaan selain ketiga pekerjaan tersebut (tidak ada pekerjaan ACCOUNTANT lagi).



# *Constraints* untuk Spesialisasi dan Generalisasi

---

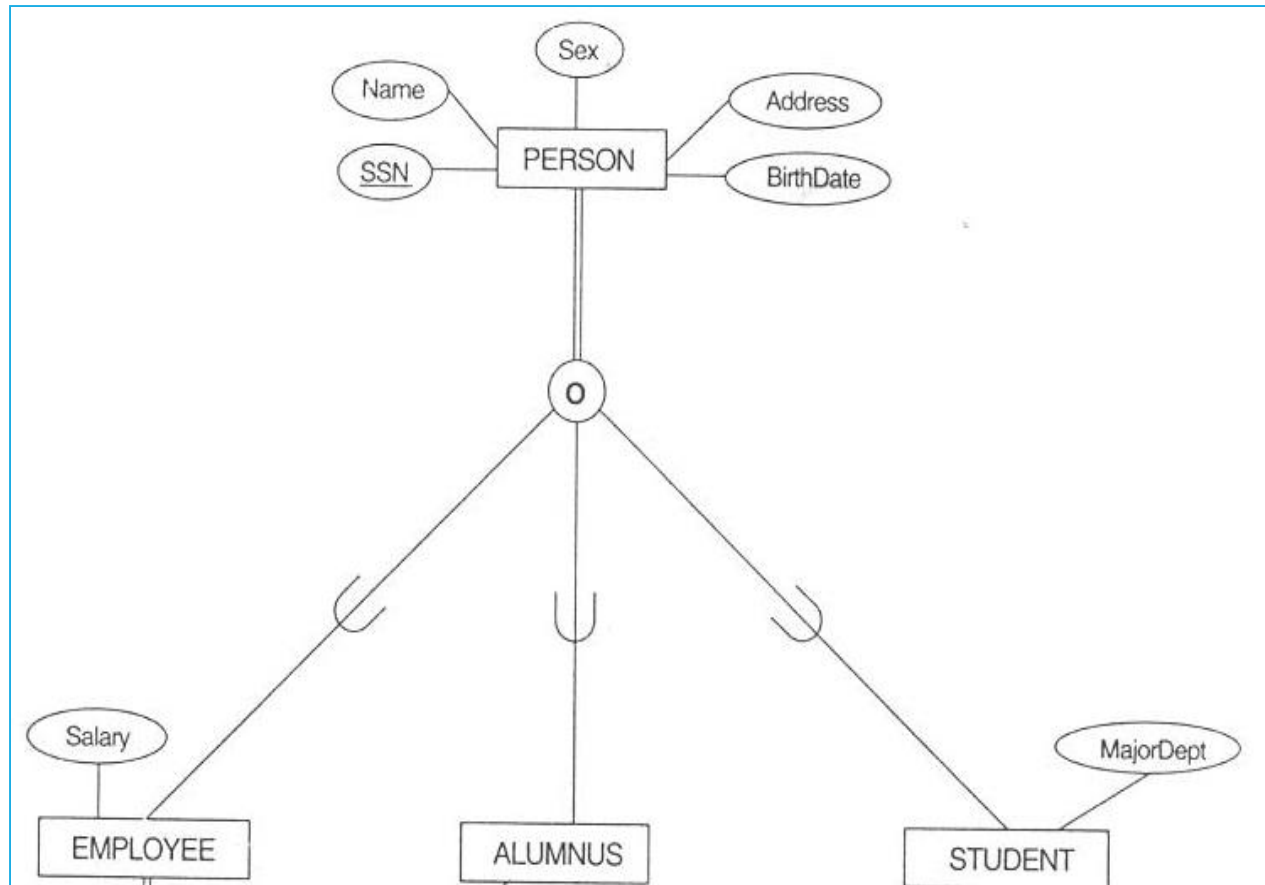
- *Overlap partial:*

Pada kasus ini, seorang EMPLOYEE dapat memiliki satu atau lebih jabatan, misalnya seseorang dapat bekerja sebagai TECHNICIAN sekaligus ENGINEER. Namun bisa pula seorang EMPLOYEE bukan merupakan anggota dari *subclass* manapun, misalnya ia merupakan seorang MANAGER.

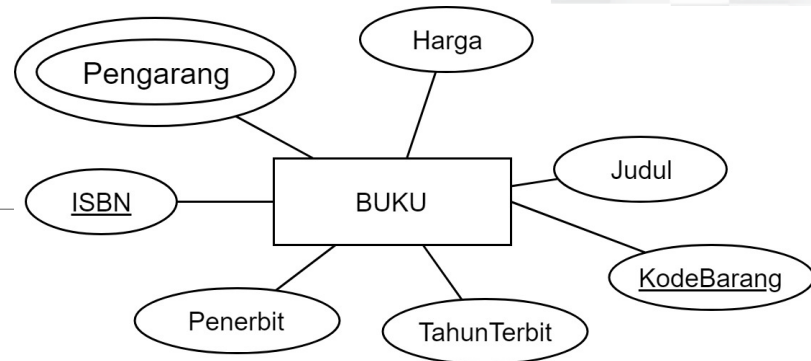
- *Overlap total:*

Pada kasus ini, seorang EMPLOYEE dapat terdaftar sebagai salah satu (MINIMAL), dua, atau ketiga pekerjaan antara SECRETARY, TECHNICIAN, dan ENGINEER.

# Contoh Spesialisasi *Overlap Total*

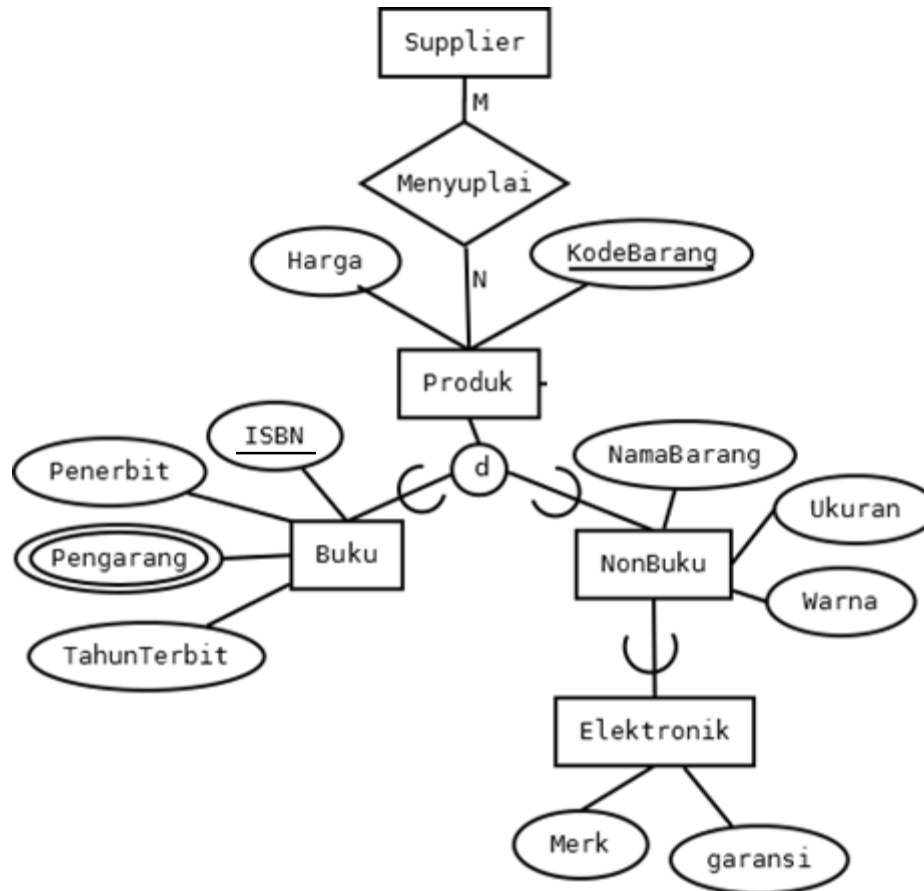


# Latihan



Berdasarkan potongan gambar ER di atas, Pemilik toko *eBuku* berencana menambahkan beberapa item yang dijual di toko, seperti *stationery* dan **barang elektronik**. Untuk *stationery* data yang perlu disimpan adalah kode barang yang unik, nama barang, harga, ukuran, dan warna. Sedangkan untuk barang elektronik data yang perlu disimpan adalah kode barang yang unik, nama barang, model, harga, warna, ukuran, garansi, dan merk. Ada juga beberapa item lain yang tidak termasuk buku, *stationery*, atau barang elektronik. Beberapa item, seperti kalkulator juga dapat dikategorikan sebagai *stationery* sekaligus barang elektronik. Beberapa benda yang dijual di *eBuku* ada yang memiliki supplier sehingga perlu dicatat juga suppliernya, seperti nama supplier (unik) dan alamat supplier. Ada supplier yang hanya menjual salah satu benda saja seperti buku, dan ada juga supplier yang menjual berbagai jenis benda.

# Jawaban





---

# *Hierarchy dan Lattice*

# Hierarchy dan Lattice

## Hierarchy

- Satu *subclass* hanya berpartisipasi pada satu *class/subclass relationship* (satu *subclass* hanya memiliki satu *superclass* saja)
- Contoh: VEHICLE dengan TRUCK dan CAR

## Lattice

- ◆ Satu *subclass* dapat berpartisipasi pada lebih dari satu *class/subclass relationship*
- ◆ Contoh: seorang Engineering Manager, haruslah seorang Engineer dan juga seorang Manajer
- ◆ Mengandung konsep *multiple inheritance*

# Contoh *Lattice*

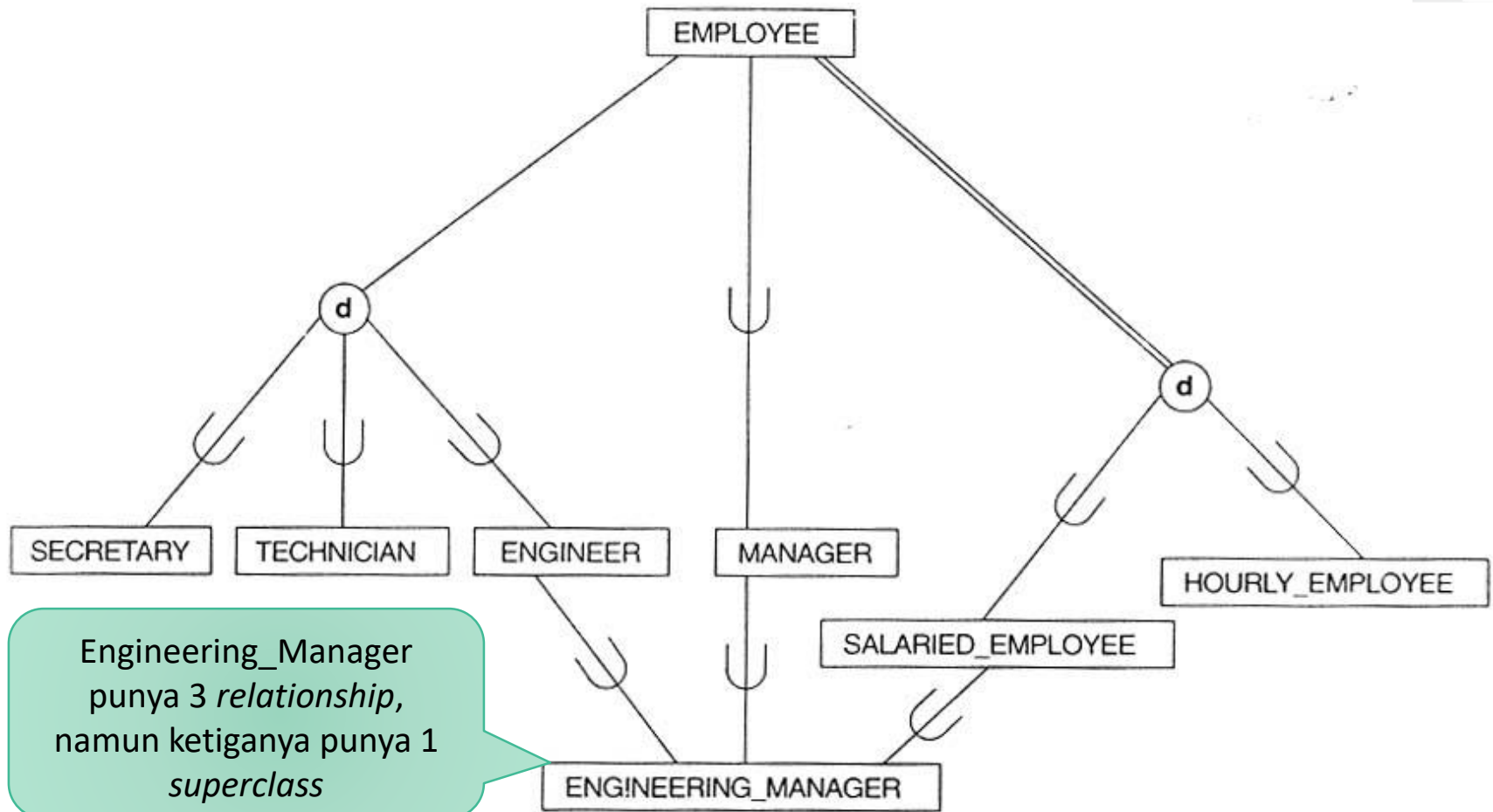
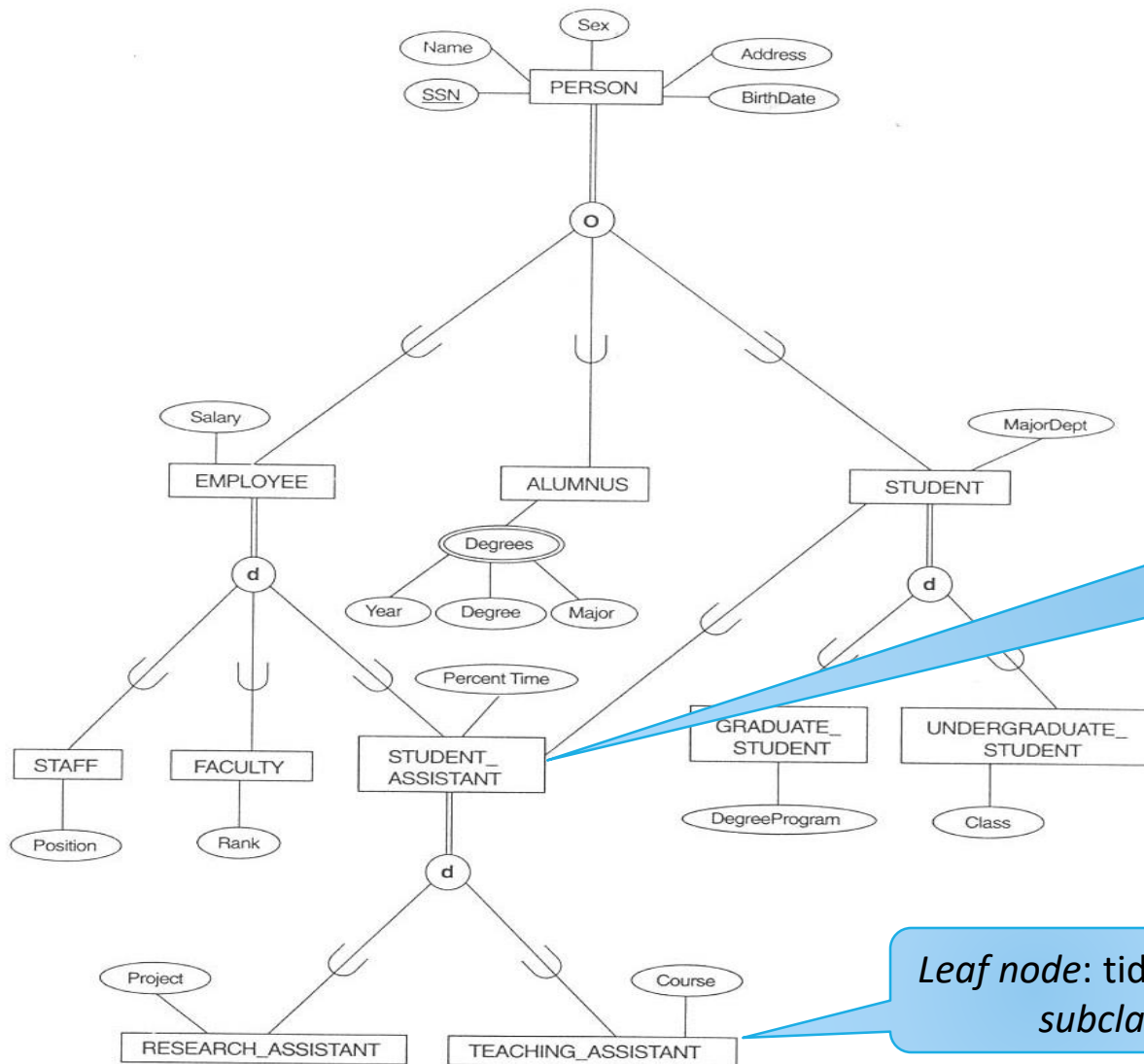


Figure 4.6 A specialization lattice with the shared subclass `ENGINEERING_MANAGER`.

# Contoh *Lattice*



Satu *entity* mungkin ada di beberapa *subclass*. Misal graduate student sekaligus teaching assistant

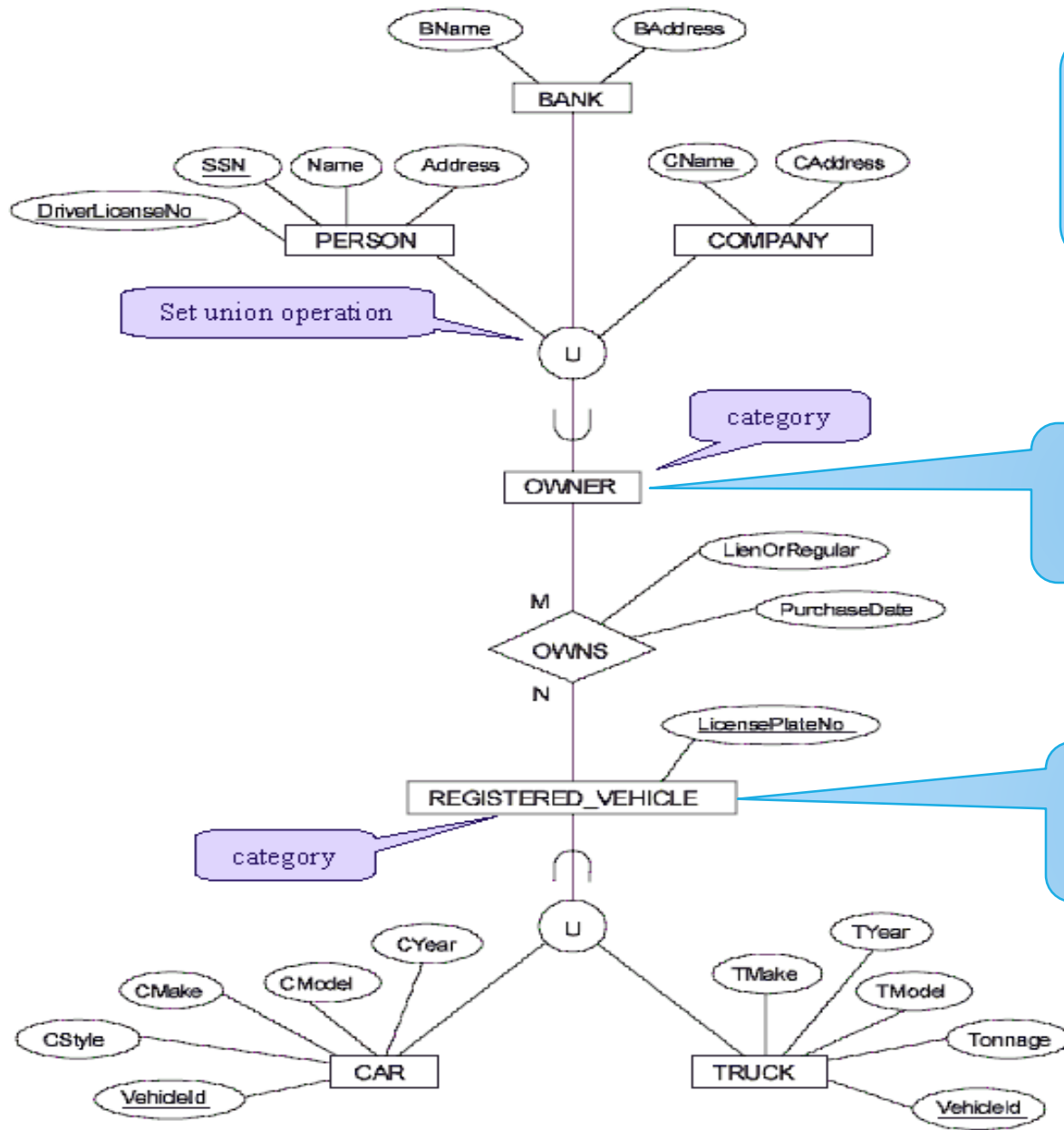
*Multiple inheritance!* Namun *attribute* dari PERSON hanya diwariskan 1 kali

*Leaf node:* tidak punya *subclass*



# Pemodelan dengan *Categories*

# Union Type dengan Menggunakan Category



Satu subclass memiliki satu relationship dengan 3 buah superclass: disebut sebagai *union type* atau *category*

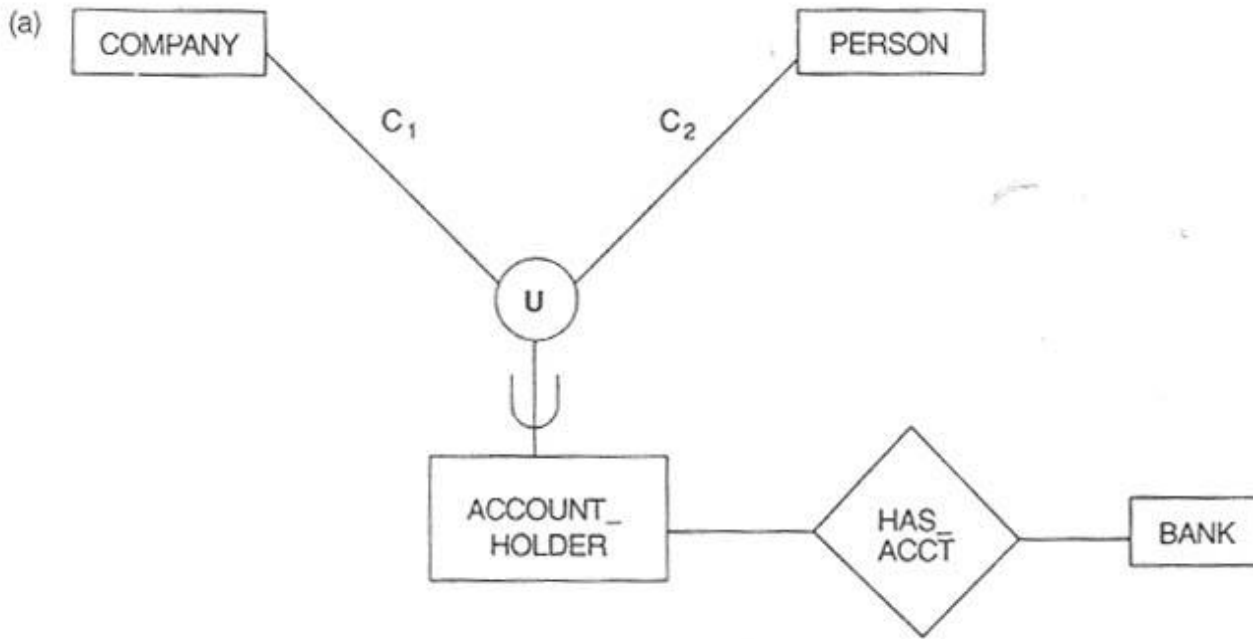
OWNER merupakan *union subclass* dari COMPANY, BANK, PERSON

REGISTERED\_VEHICLE merupakan *union subclass* dari TRUCK dan CAR

# Perbedaan *Category* dengan *Lattice*

- Engineering\_Manager harus ada pada semua *superclass*: Manager, Engineer, Salaried\_Employee
- Owner harus ada pada salah satu dari ketiga *superclasses*
- Engineering\_Manager: mewarisi semua *attribute* dari *superclasses*
- Owner mewarisi *attribute* tertentu saja, tergantung dari *superclass*-nya

# Partial Category



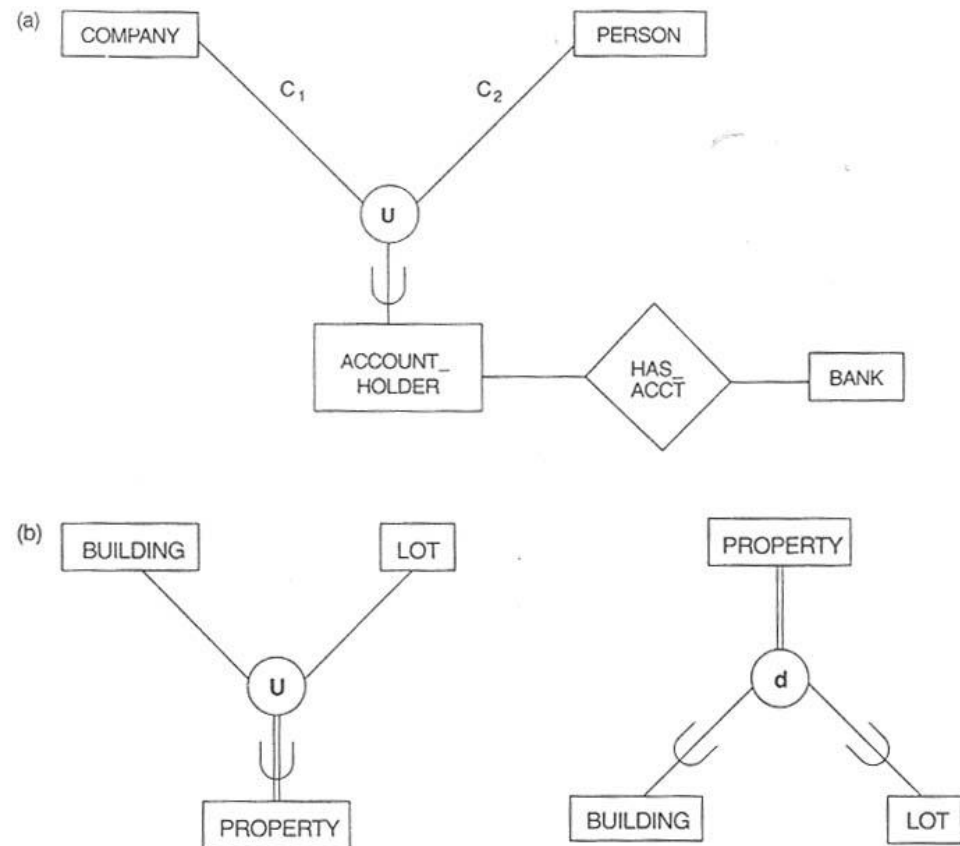
*Partial category: dapat berpartisipasi ataupun tidak pada relationship*

# Total Category

Harus merupakan salah satu *superclasses*

Contoh: *A building and a lot must be a member of PROPERTY*

Dapat direpresentasikan sebagai *generalization (d)*, khususnya jika kemiripannya banyak.



**Figure 4.9** Total and partial categories. (a) Partial category **ACCOUNT\_HOLDER** that is a subset of the union of two entity types **COMPANY** and **PERSON**. (b) Total category **PROPERTY** and a similar generalization.

# Contoh Skema EER untuk Basis Data Universitas

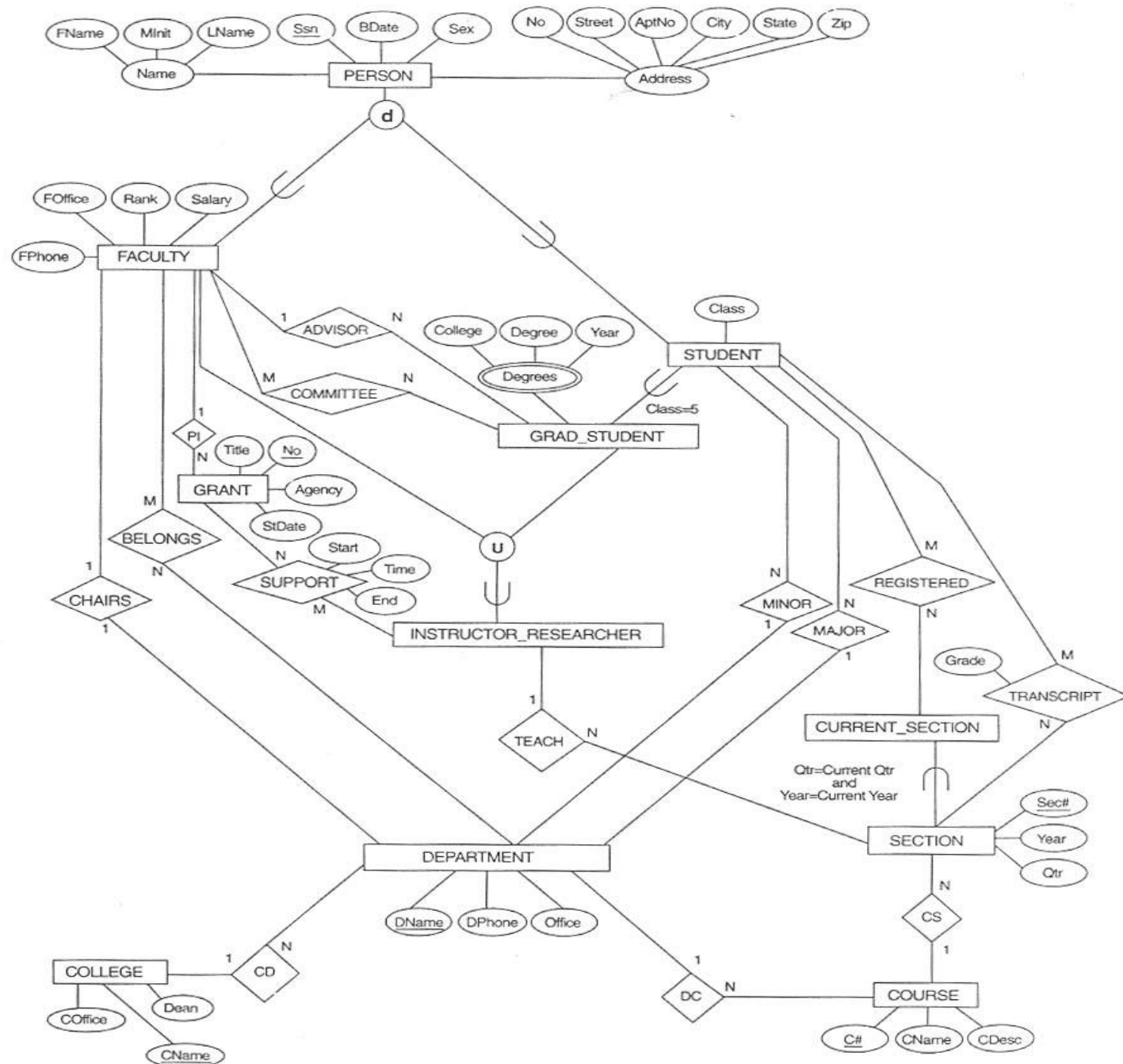
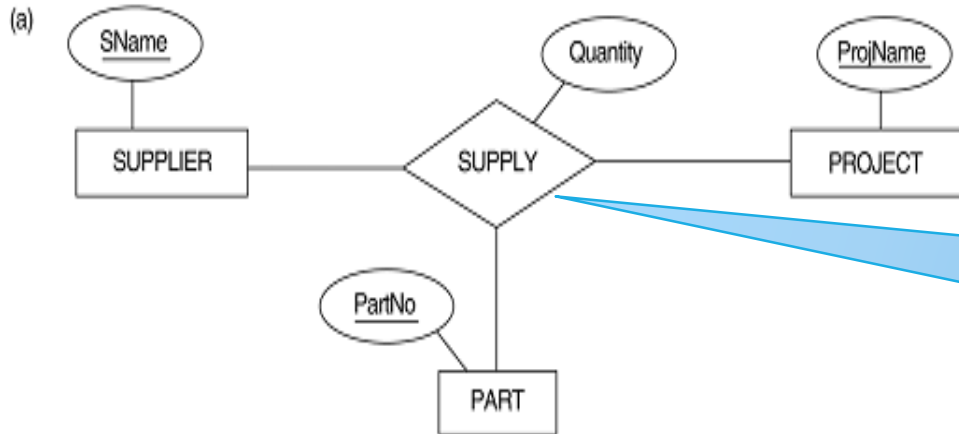


Figure 4.10 An EER conceptual schema for a UNIVERSITY database.

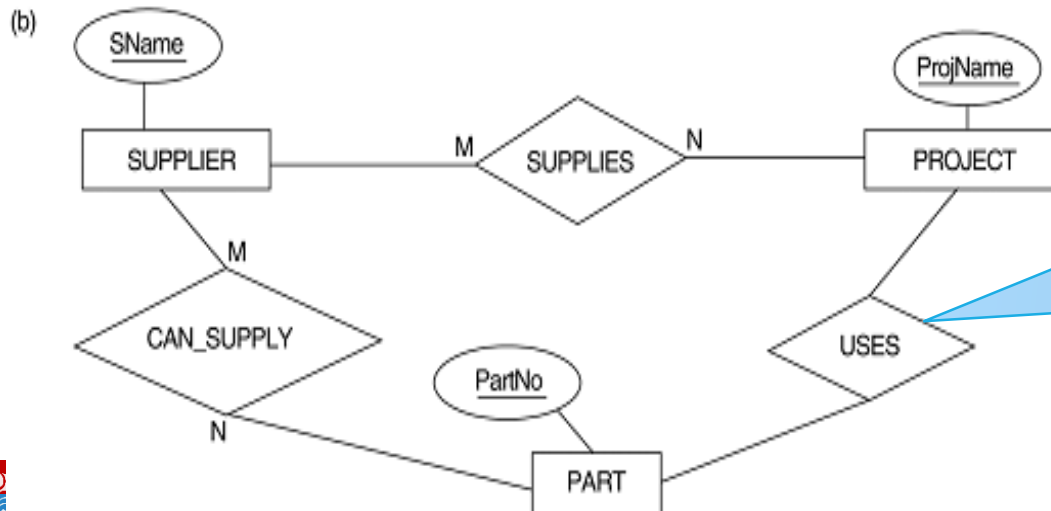
# *Higher Degree Relationship*

# Higher Degree Relationship



Dua skema ini beda maknanya!

*Ternary relationship type:* menghubungkan 3 entity types



Tiga *binary relationship type*: CAN\_SUPPLY, USES, SUPPLIES



# *Higher Degree Relationship*

- *Higher degree relationship* tampak kompleks, bagaimana menyederhanakannya?

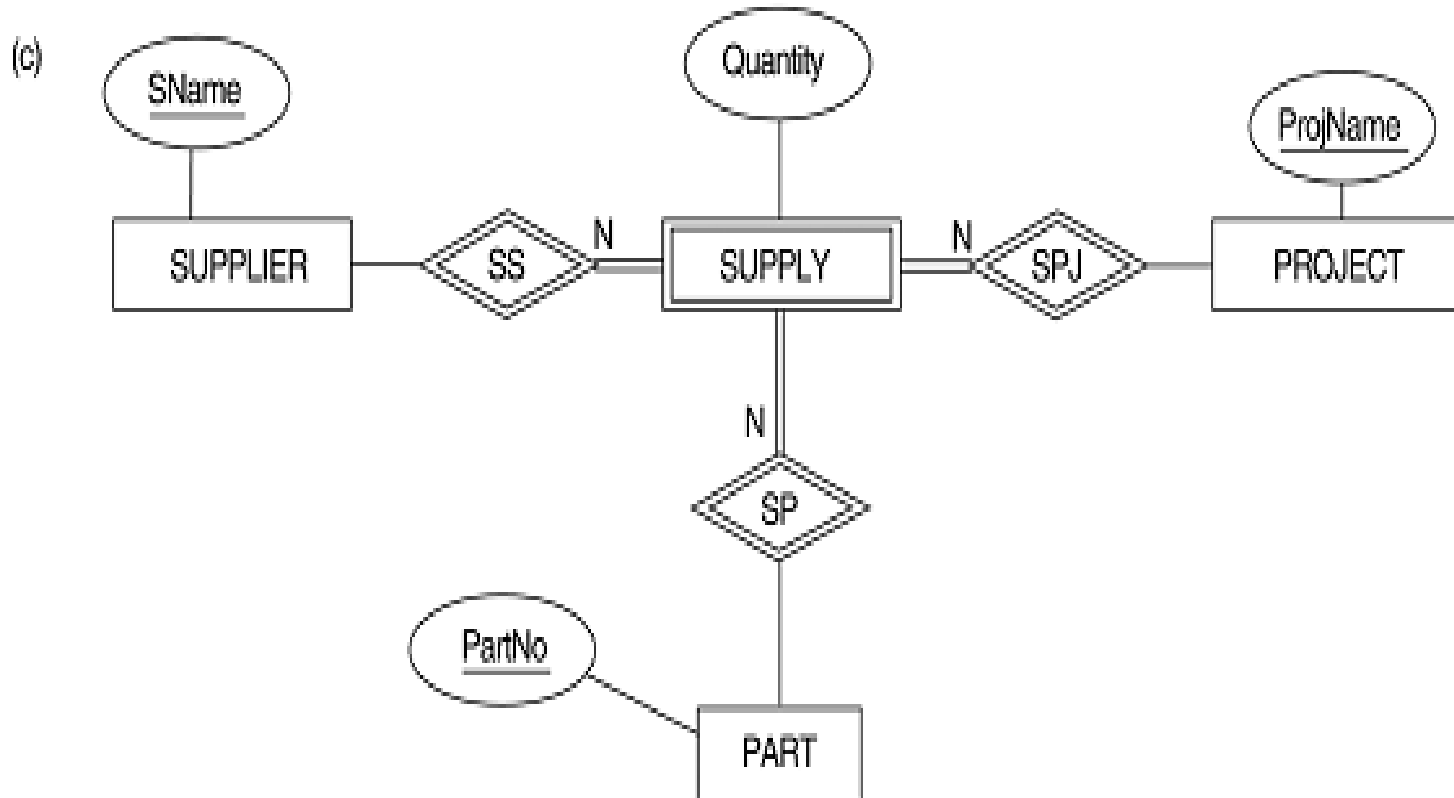
Opsi 1. *Higher degree relationship* sebagai *weak entity*

- Merepresentasikan *Higher degree relationship* sebagai *weak entity type* yang berhubungan ke *owner entity types*
- Mengandung *binary (identifying) relationship*

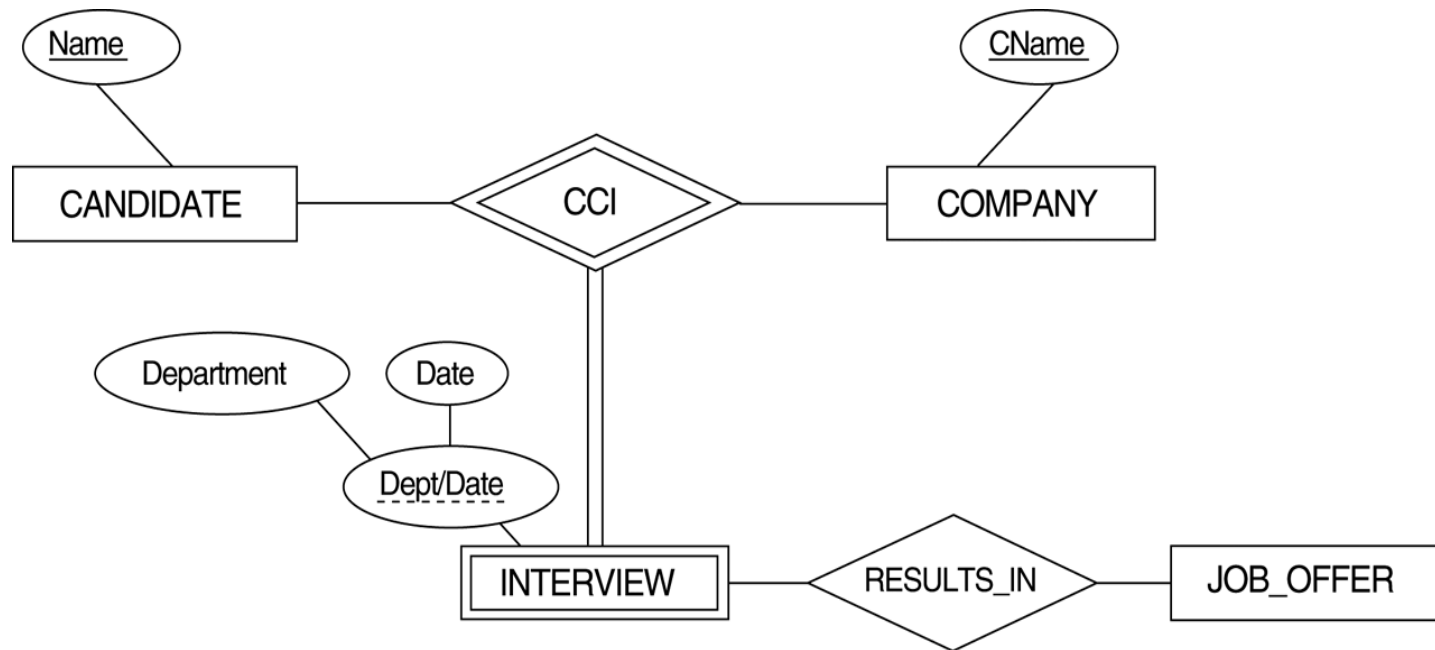
Opsi 2. *Higher degree relationship* sebagai *identifying relationship type*

- Sebuah *ternary relationship type* dengan sebuah *weak entity type* dan dua buah *owner entity type*

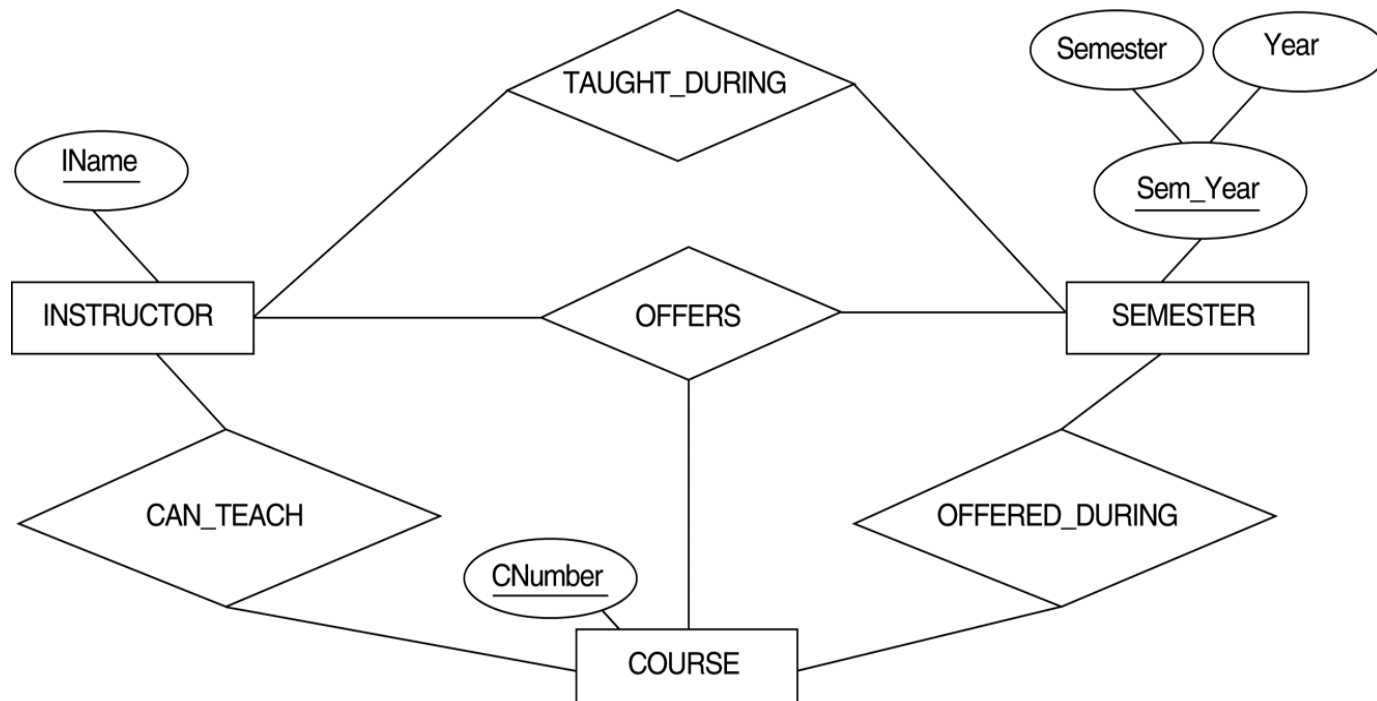
# *Ternary Relationship sebagai Weak Entity Type*



# *Ternary Relationship sebagai Identifying Relationship Type*



# Contoh: *Ternary vs Binary Relationship Type*



# Kapan Kita Menggunakan Model EER?

# Kapan Kita Menggunakan Model EER?

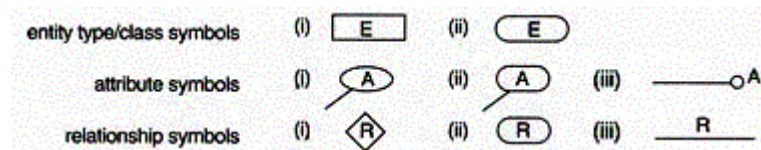
- Sebagian besar proyek basis data tidak perlu fitur-fitur model berorientasi obyek yang ada pada EER
- Tujuan pemodelan data konseptual adalah untuk menghasilkan sebuah model yang sederhana dan mudah dimengerti
- Jangan menggunakan *class/subclass relationship* yang kompleks jika tidak diperlukan
- Penggunaan model EER menawarkan keuntungan dibandingkan model ER jika digunakan pada kondisi yang tepat

# Kapan Kita Menggunakan Model EER?

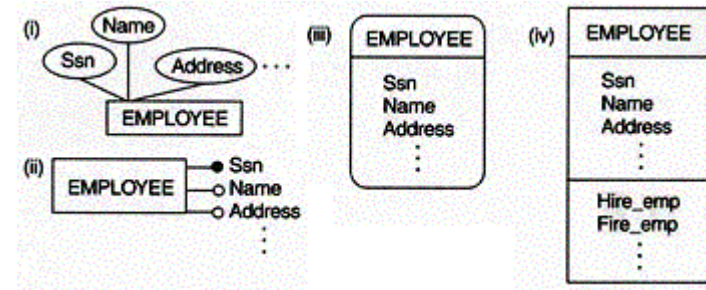
- Model EER perlu digunakan jika domain yang dimodelkan secara alamiah bersifat *object-oriented*, *inheritance* akan mereduksi kompleksitas perancangan
- Gunakan EER pada situasi:
  - Ketika penggunaan *attribute inheritance* dapat mereduksi penggunaan *null* pada suatu *single entity relation* (yang mengandung *multiple subclasses*)
  - *Subclass* dapat digunakan untuk secara eksplisit memodelkan dan menamai *subset* dari *entity* yang berpartisipasi pada *relationship*-nya sendiri (dimana *subclass* lain dalam *superclass* yang sama tidak berpartisipasi pada *relationship* tersebut)

# Alternative Diagrammatic Notations

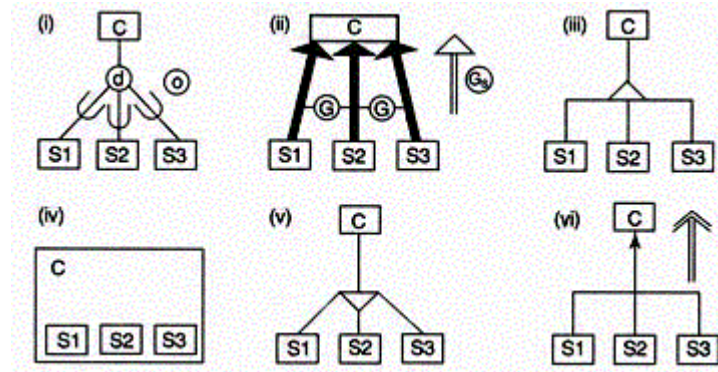
*Symbols for entity type / class, attribute and relationship*



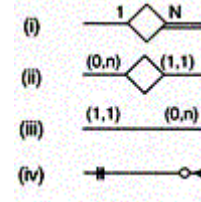
*Displaying attributes*



*Notations for displaying specialization / generalization*



*Various (min, max) notations*



*Displaying cardinality ratios*

