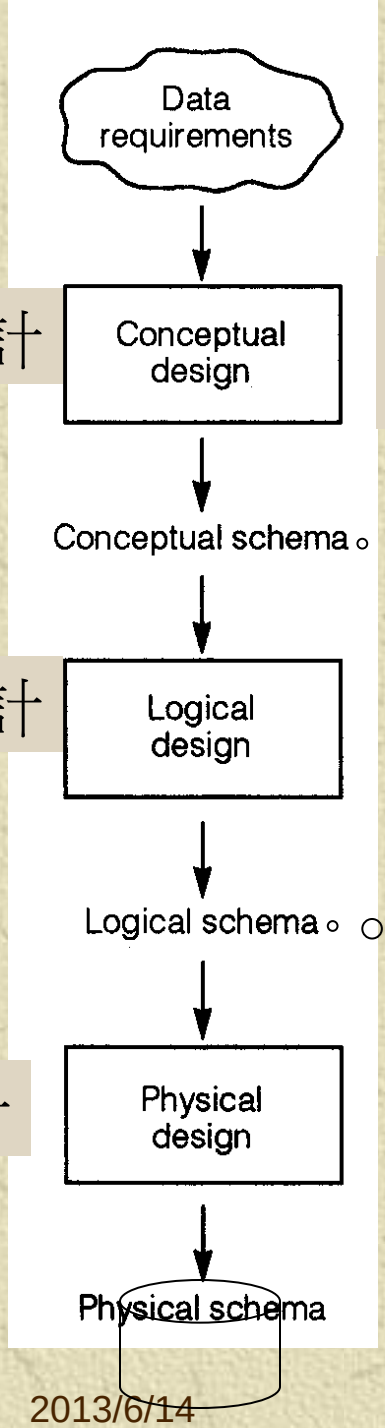
# Unit 9

## High-level Logical Design

# Phases of Database design




概念設計

在需求分析階段，清楚地描述出系統所包含的資訊之內容、意義及其間的關係。

用 ER model 繪出

邏輯設計

把 Conceptual schema 轉為關連式資料庫。

用 DDL 定義出

實體設計

描述 access method, storage structure, 考慮將 relations 存放入輔助記憶體。

# 本單元目的

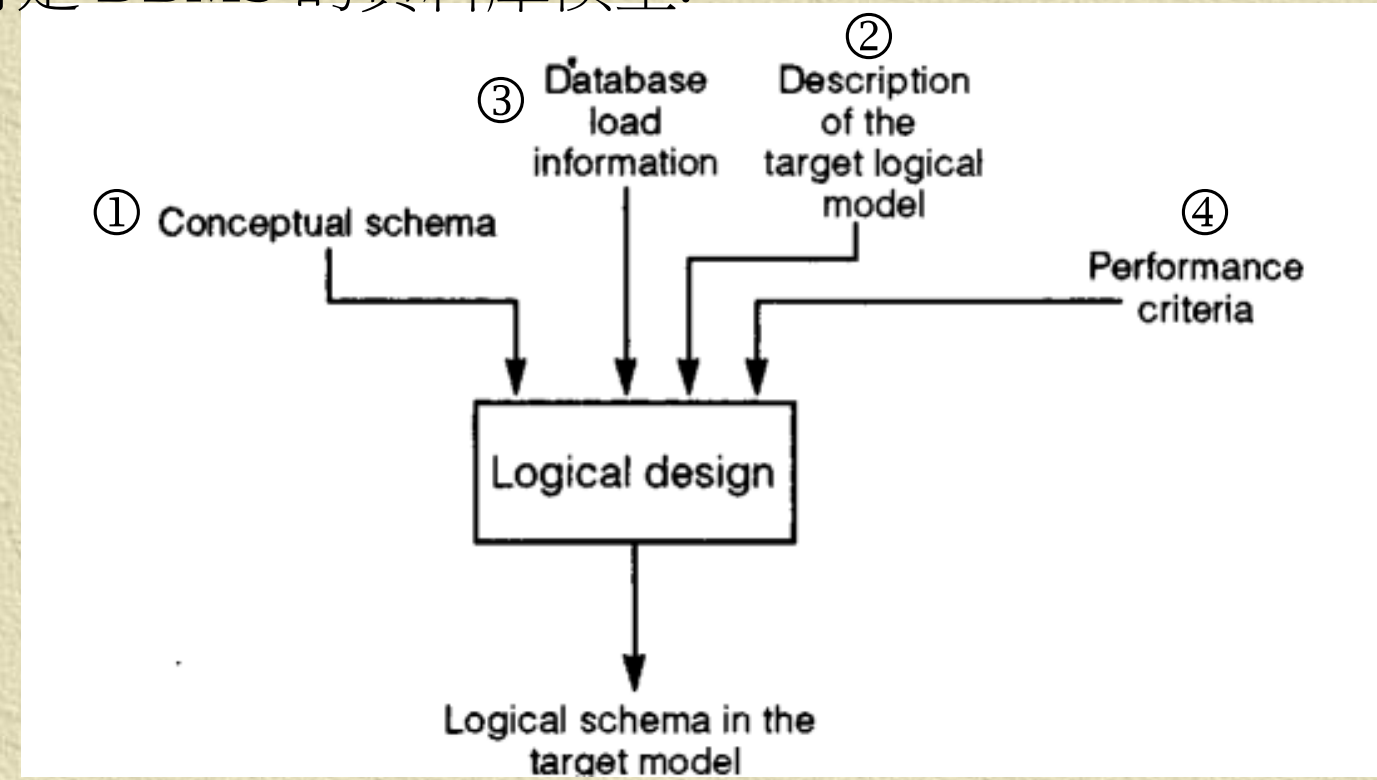
- 介紹兩階層的邏輯資料庫設計
- 估算資料庫負載量
- 根據負載量簡化 conceptual schema
- 建議 model independent logical design 決策事項

# Outlines

- An Overall Approach to Logical Design
- Modeling of the Database Load
- Decisions about Derived Data
- Removing Generalization Hierarchies
- Partitioning of Entities
- Merging Entities and Relationships
- Primary Key Selection

# Logical DB Design

把 conceptual data schema 轉為 logical schema, 以符合某一特定 DBMS 的資料庫模型.



# Inputs of logical design

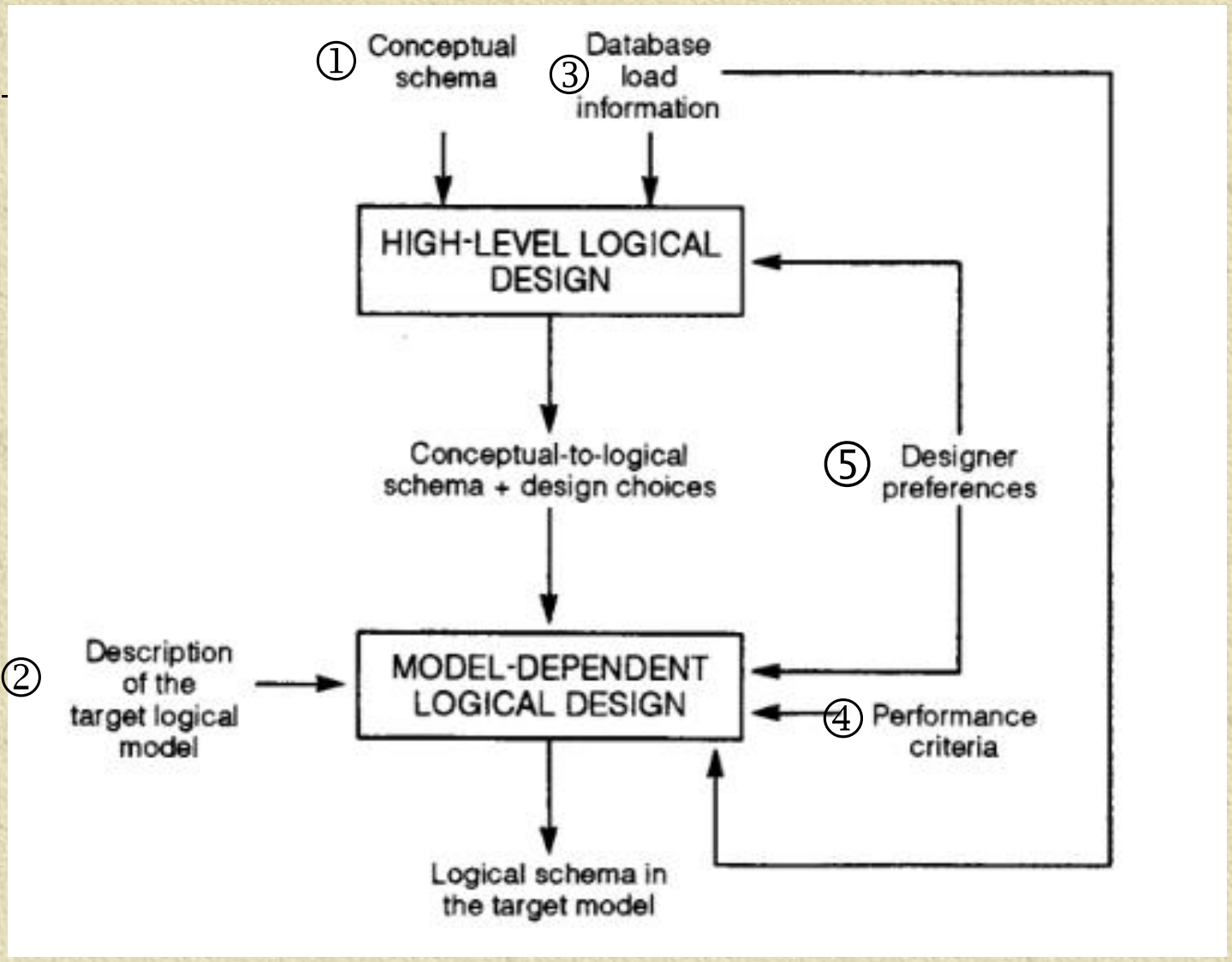
① Conceptual schema
- 表達出整個應用系統的資料概念.

② Description of the target logical model
- DBMS 所提供的資料庫模型.

③ Database load information
- 資料庫的負載量, 亦即資料儲存量與使用頻率.

④ Performance criteria
- 執行效率的要求水準, 例如: 反應時間.

# Two-Phase Logical DB Design



# Two-Phase Logical DB Design

- Phase 1: High-level logical design

  Model independent

  - 分析資料庫負載量
  - 簡化 conceptual schema
  - 建議 logical design 決策事項

- Phase 2: Model-dependent logical design

  ⑤ Designer preferences

  輸入系統設計者的個人喜好, 把 Phase 1 的結果轉為某一特定 DBMS 所提供的資料庫架構.

# Modeling of the Database Load

- 負載量的表達方式
  - The volume of data

    找出 conceptual schema 中各概念的資料出現量
  - The description of applications

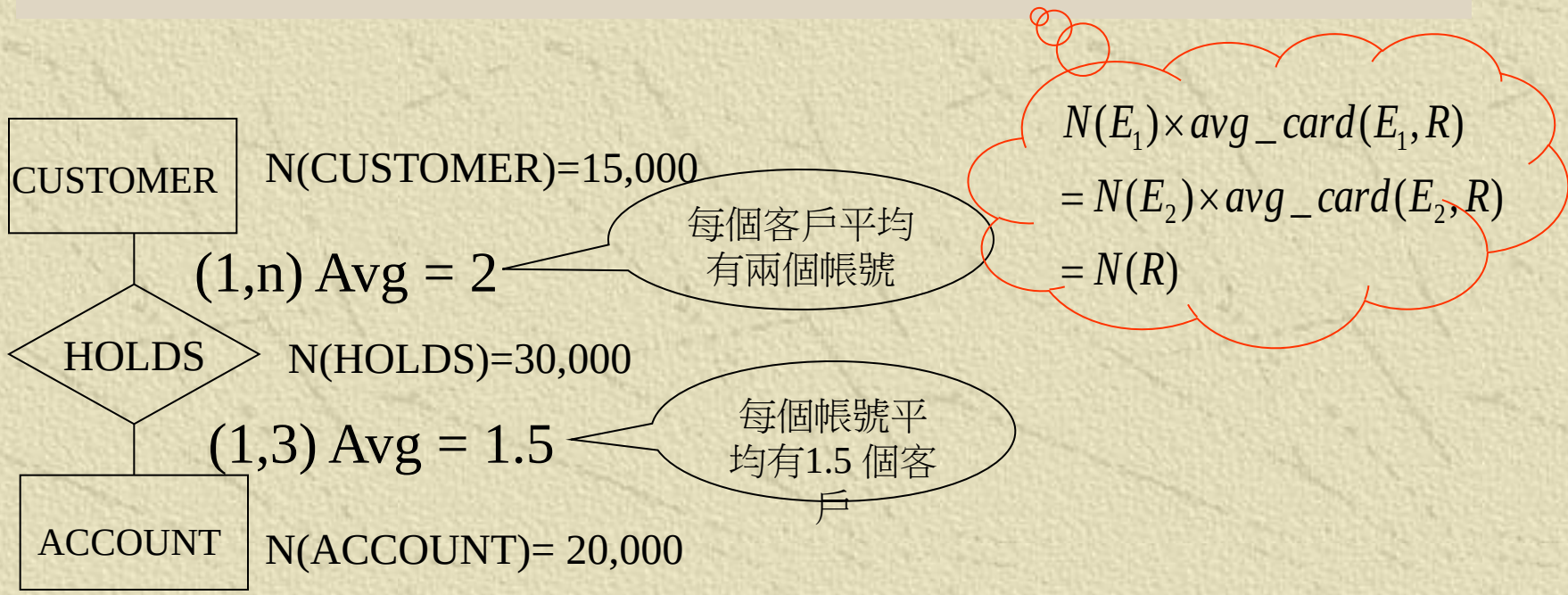
    找出重要的 operations, 分析其發生頻率與涉及的資料量
- 使用的模型
  - Schema with data-volume information
  - Data-volume table
  - Operation frequency table
  - Operation access-volume table

# The volume of data

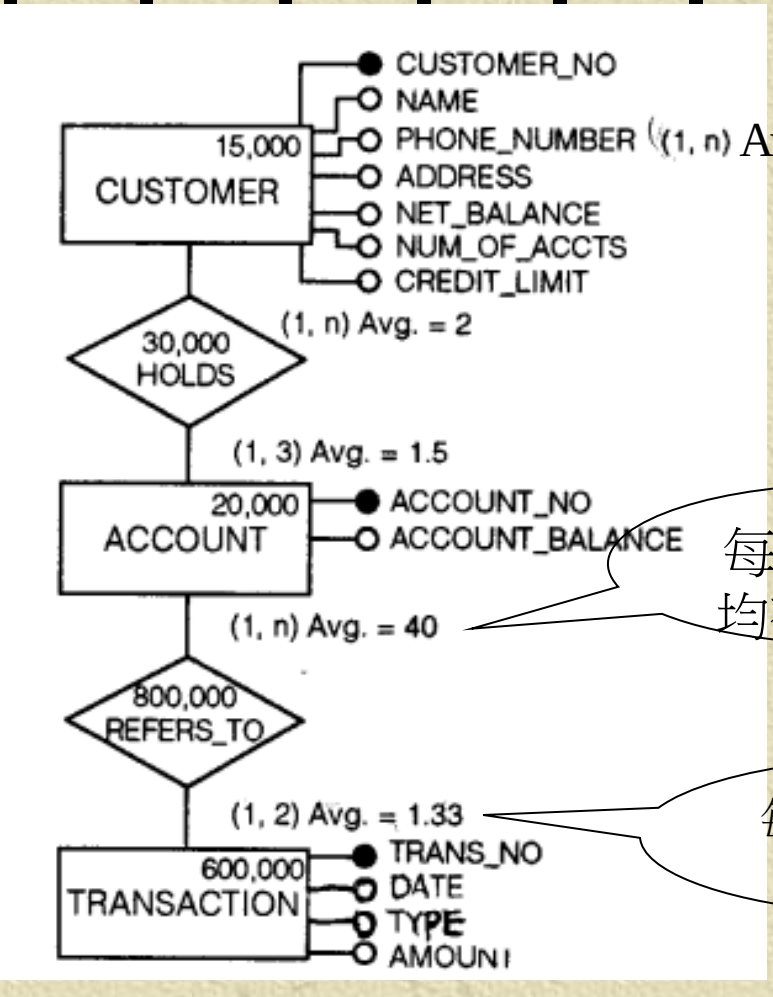
N(E)　表示 entity E 的資料平均發生次數(data instances).

N(R)　表示 relationship R 的資料平均發生次數.

Avg_card(E, R)　表示每個 E 的entry 存在 R的次數之平均值, 用 ( *min-card*, *max-car*) avg = *avg-card* 表示之.



# An example of a schema with data-volume information



# Data-Volume Table

| Concept | Type | Volume |
|---|---|---|
| CUSTOMER | E | 15,000 |
| ACCOUNT | E | 20,000 |
| TRANSACTION | E | 600,000 |
| HOLDS | R | 30,000 |
| REFERS_TO | R | 800,000 |
| CUSTOMER_NO | A | 15,000 |
| NAME | A | 15,000 |
| PHONE_MUMBER | A | 22,500 |
| ADDRESS | A | 15,000 |
| NET_BALANCE | A | 15,000 |
| NUM_OF_ACCTS | A | 10 |
| CREDIT_LIMIT | A | 50 |
| ACCOUNT_NO | A | 20,000 |
| ACCOUNT_BALANCE | A | 20,000 |
| TRANS_NO | A | 600,000 |
| DATE | A | 730 |
| TYPE | A | 10 |
| AMOUNT | A | 600,000 |

Type:
E is entity.
R is relationship.
A is attribute

=15,000x 1.5

交易種類
共有十種

# The description of application

固定時段發生頻率

**Operation Frequency Table**

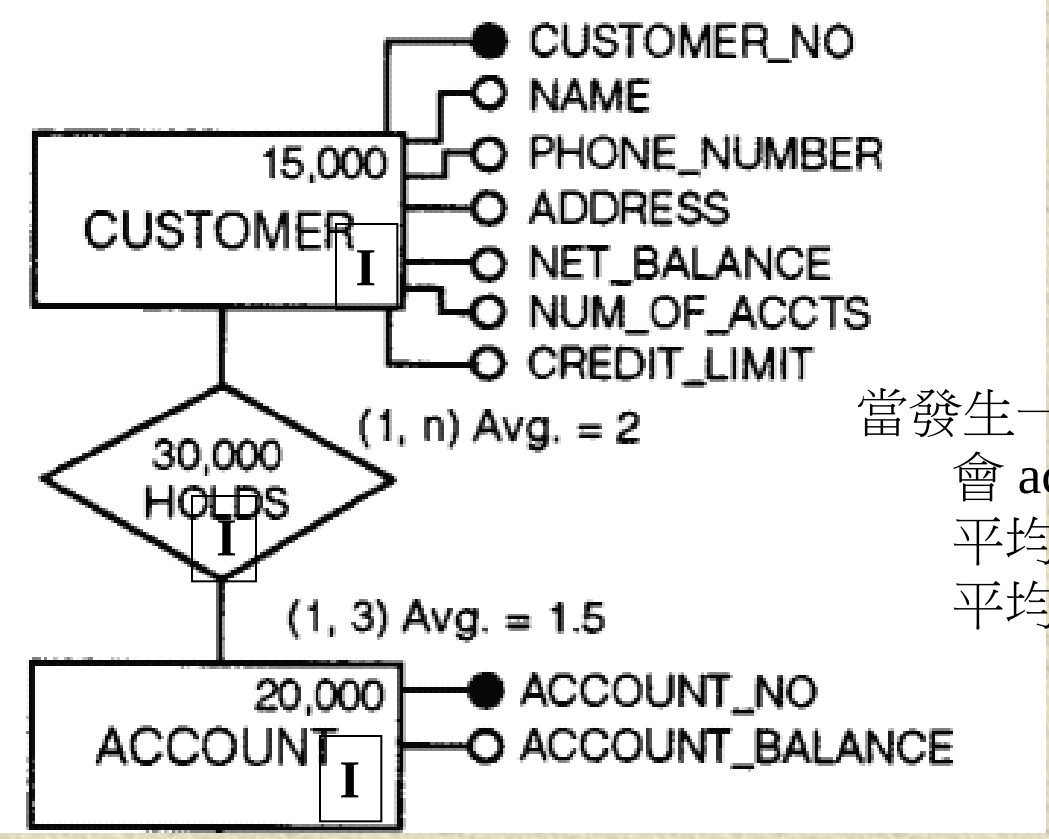
| Operation Name/Description | Frequency | Type (On-line/Bat |
|---|---|---|
| 01 OPEN AN ACCOUNT | 100 times a day | OL |
| 02 READ THE BALANCE | 3000 times a day | OL |
| 03 DISPLAY LAST 10 TRANSACTIONS | 200 times a day | OL |
| 04 WITHDRAW MONEY | 2000 times a day | OL |
| 05 DEPOSIT MONEY | 1000 times a day | OL |
| 06 PREPARE A MONTHLY STATEMENT | 1 time a month | B |
| 07 REPORT NO. OF ACCOUNTS HELD BY A CUSTOMER | 75 times a day | OL |
| 08 SHOW TRANSACTIONS FOR NEGATIVE- BALANCE ACCOUNTS | 20 times a day | OL |

集中考慮 **20%** 最重要的 **operation**

# 20-80 Rule

- 20% of the operations produce 80% of the load.
- One should concentrate at least on the important 20% of the operations.

# Navigation schema for O1: 為一個新客戶開一個新帳號




當發生一次 O1 時
會 access 1 次 ACCOUNT
平均 access 1.5 次 HOLDS
平均 access 1.5 次 CUSTOMER

該 operation 動到的 entity or relationship

## Operation Access-Volume Table

| Operation Name/Description | Concept | Concept Type | Read/ Write | Avg. Occurrences Accessed |
|---|---|---|---|---|
| 01 OPEN AN ACCOUNT | ACCOUNT | E | W | 100 |
| | CUSTOMER | E | W | 100 x 1.5 = 150 |
| | HOLDS | R | W | 100 x 1.5= 150 |
| 02 READ THE BALANCE | ACCOUNT | E | R | 3000 |
| 03 DISPLAY LAST 10 | ACCOUNT | E | R | 200 |
| TRANSACTIONS | REFERS_TO | R | R | 200 x 40 = 8000 |
| | TRANSACTIONS | E | R | select 2000 out of 8000 |
| 04 WITHDRAW MONEY | ACCOUNT | E | R | 2000 |
| | | | W | 2000 |
| | CUSTOMER | E | W | 2000 x 1.5 = 3000 |
| 05 DEPOSIT MONEY | ACCOUNT | E | R | 1000 |
| | | | W | 1000 |
| | CUSTOMER | E | W | 1000 x 1.5=1500 |

# Decision about Derived Data

- Derived data (衍生資料項)
  - 由別的 attributes 計算出來的 attribute.
  - 例如 (見 p.13 圖)
    - CUSTOMER 中的 NET_BALANCE 是由該客戶的所有帳號下之餘額累加結果.
    - CUSTOMER 中的 NUM_OF_ACC 指每個客戶有幾個帳戶, 由 HOLDS 即可計算出.
- 保留衍生資料項
  - 優點: 在執行查詢時不必重複計算該值.
  - 缺點:
    - 要維護資料的一致性, 增加資料存取量
    - 占儲存空間

# 考慮 NET_BALANCE 的存廢

- 就 p. 12 operation frequency table 考慮
- 優缺點分析
  1. O1, O3, O6 沒有用到 NET_BALANCE
  2. 留下 NET_BALANCE 的好處:
     O2 省下 6000個 access.
     說明: O2 每天 3000次, 每次要透過 HOLD 找 ACCOUNT, 每個 CUSTOMER平均有 2個 ACCOUNT, 所以是 3000 x 2.
  3. 留下 NET_BALANCE 的成本:
     O4 和 O5 合計多了 4500個 write 的動作.
     說明: O4 每天 2000次, 每次平均更新 1.5個 CUSTOMER中的 NET_BALANCE, 所以有 2000 x 1.5
     O5 每天 1000次, 每次平均更新 1.5個 CUSTOMER中的 NET_BALANCE, 所以有 1000 x 1.5
     多占了 90K bytes
     說明: 每個 CUSTOMER 占 6 bytes, 有 15000個CUSTOMER, 6 x 15000 = 90,000 bytes.

# 考慮 NET_BALANCE 的存廢

- 結論　**‘不要有 NET_BLANCE’**
  - 省了 6000個 retrieve access
  - 多了 4500個 write access
  - 看起來仍是少了 6000 – 4500 = 1500 個 access
  - 但是 write access 比 retrieve access 成本高出許多, 所以並不見得是省下 1500 個 access.
- 問: 倘若 O2 每天發生次數高達 6000次?

# Removing Generalization Hierarchies

現行的資料庫結構 (relation database) 無法直接表達 generalization 的關係, 要把 conceptual schema 中 generalization 的關係用 relation 及 relationship 表達出

- 有三種方式

**方式一:** 合併為一個大的 entity

**方式二:** 把 super-entity 去掉

**方式三:** 用一個 relationship 連接 super-entity與 sub-entities 之間的關係.

# 合併為一個大的 entity

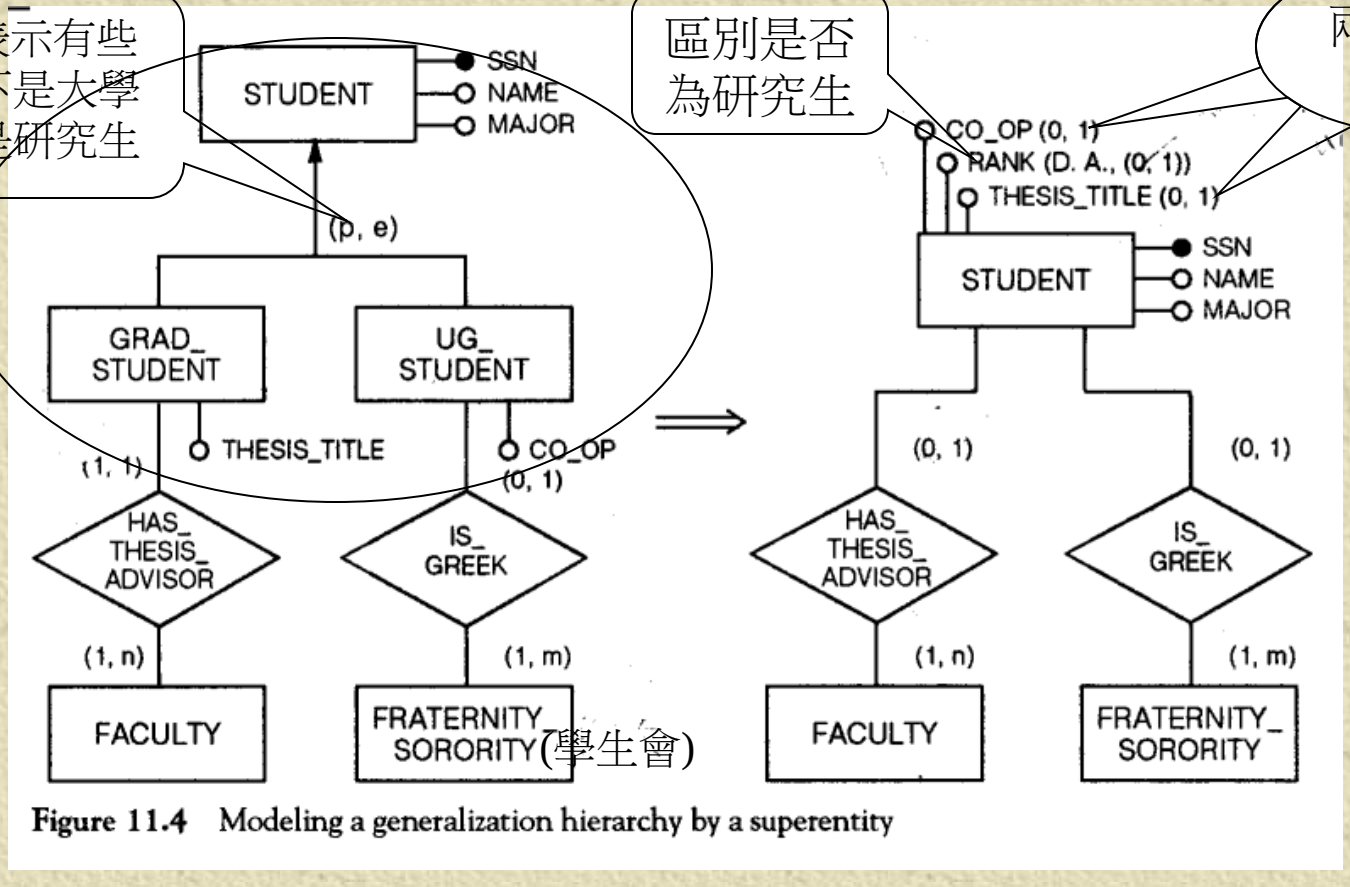


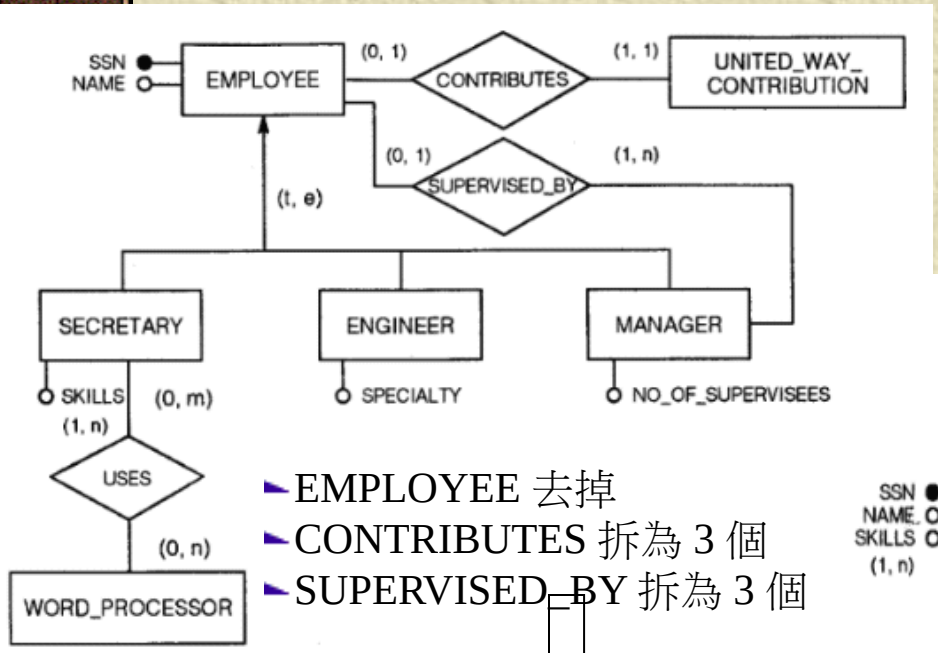

**Figure 11.4** Modeling a generalization hierarchy by a superentity

# 合併為一個大的 entity

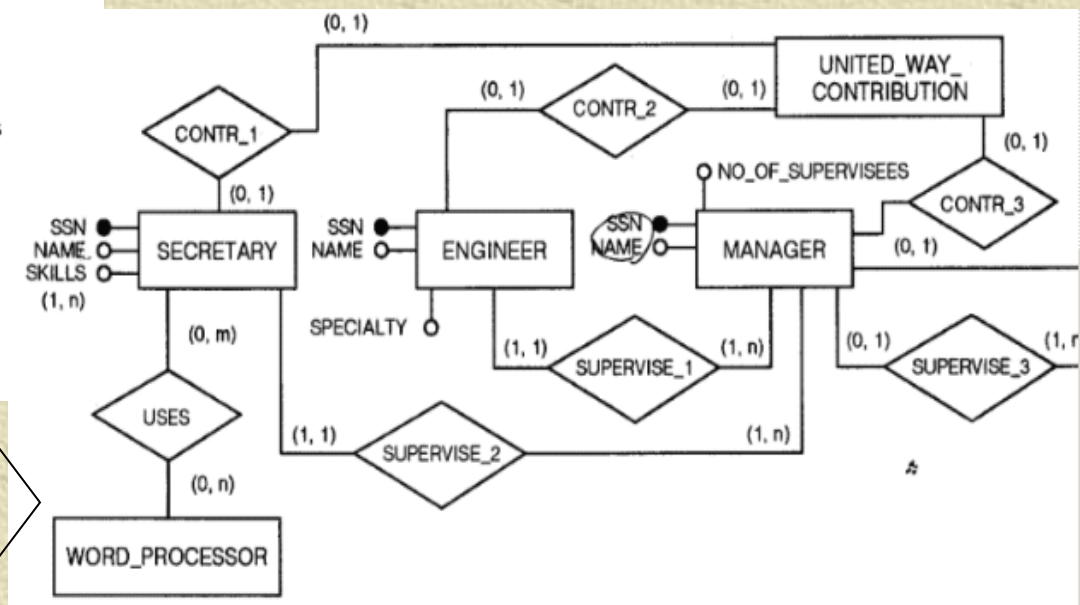
- 缺點
  - 會產生大量 null value attributes
  - 當搜尋 sub-entities 必得存取整個合併之 entities
- 優點
  - 最簡單
  - 當 coverage 是 partial or total, exclusive or overlap 皆可用之.

# 把 super-entity 去掉

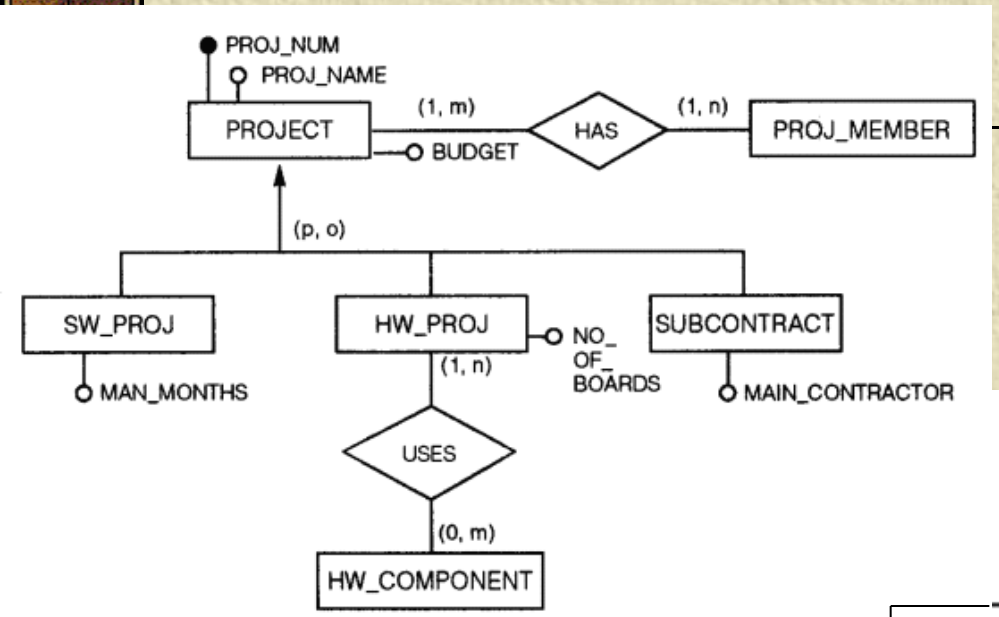
- 適用於 sub-entities 中各 attribute 很重要.
- Coverage 為 partial/overlap 時不能用.




- EMPLOYEE 去掉
- CONTRIBUTES 拆為 3 個
- SUPERVISED_BY 拆為 3 個

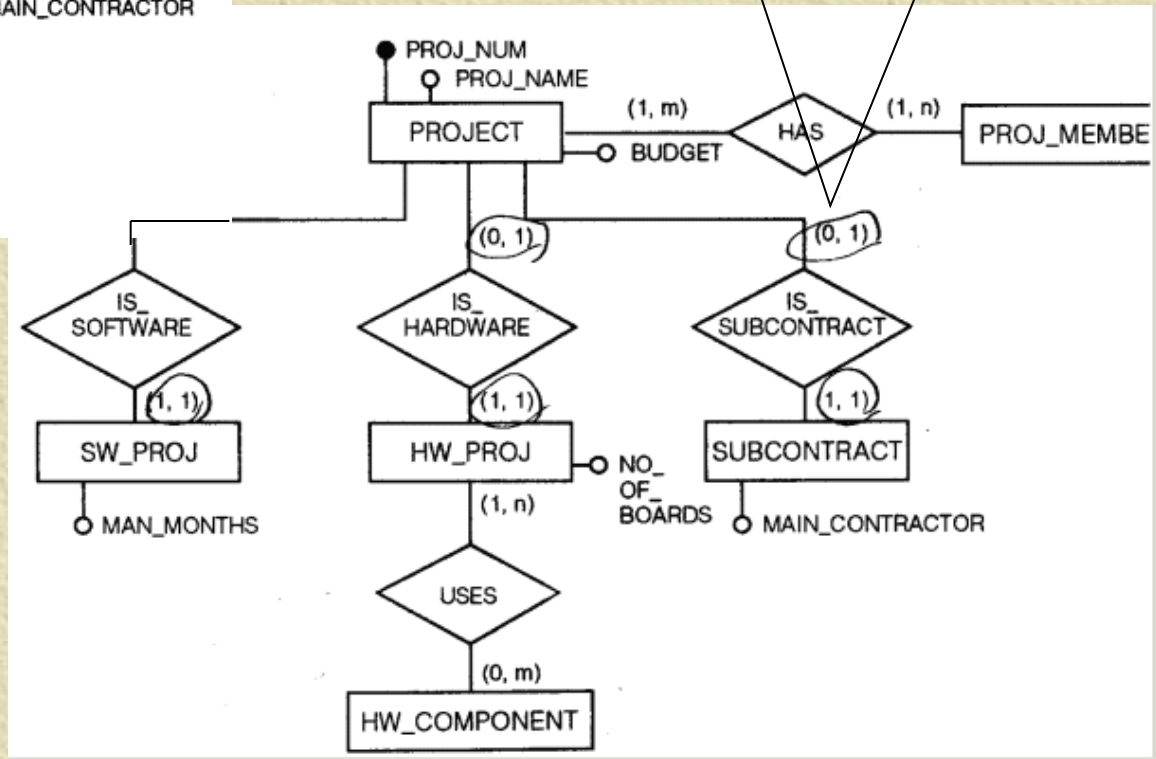


# 用 relationship 連接 (最常用)



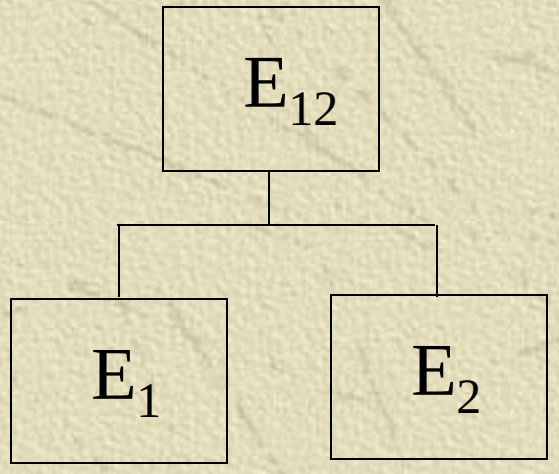

因為是 partial /overlap 所以 min_max card(0,1)
PROJECT 可以是 SW_PROJ or HW_PROJ or SUBCONTRACT or 都不是

產生 3 個新的 relationship



# 選用哪一方式改 generalization

- 若 operations 多是單獨取用 $E_{12}$ 的 attributes 而少透過 $E_{12}$ 找 $E_1$ 或透過 $E_{12}$ 找 $E_1$ 則選方式一或方式三.
- 若 operations 多是取用$E_{12}$及$E_1$ 的組合結果或是取用$E_{12}$及$E_2$ 的組合結果, 則選用方式二.
- 從 access_volume table 找 access_volume 最大的 operation 來決定.



# Partitioning of Entities

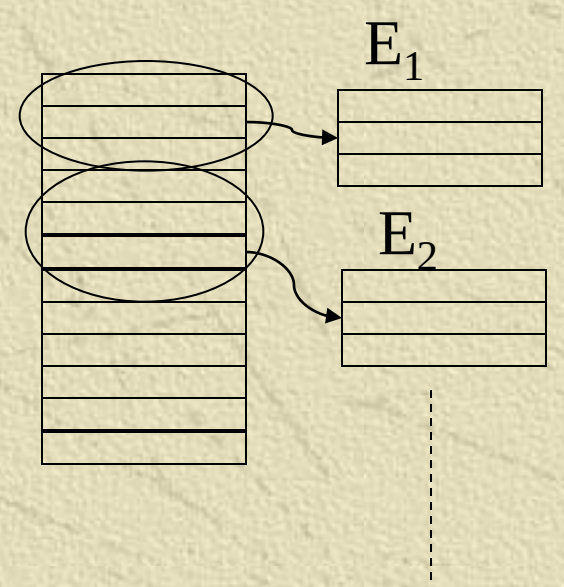
- 把 entities 以水平或垂直方式分割為兩個以上的 entities, 目的為
  - 方便經常一起存取的資料
  - 保密性

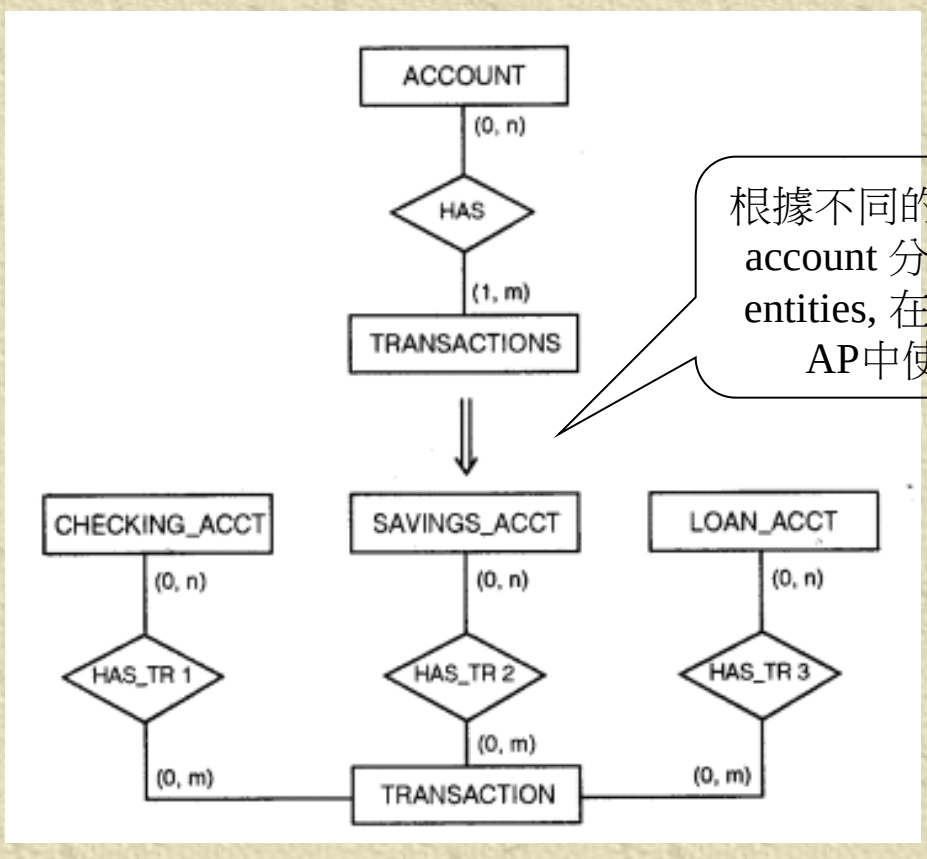
  在分散式資料庫上很管用
- Partitions of entities may be overlapping

# Horizontal partition (水平分割)

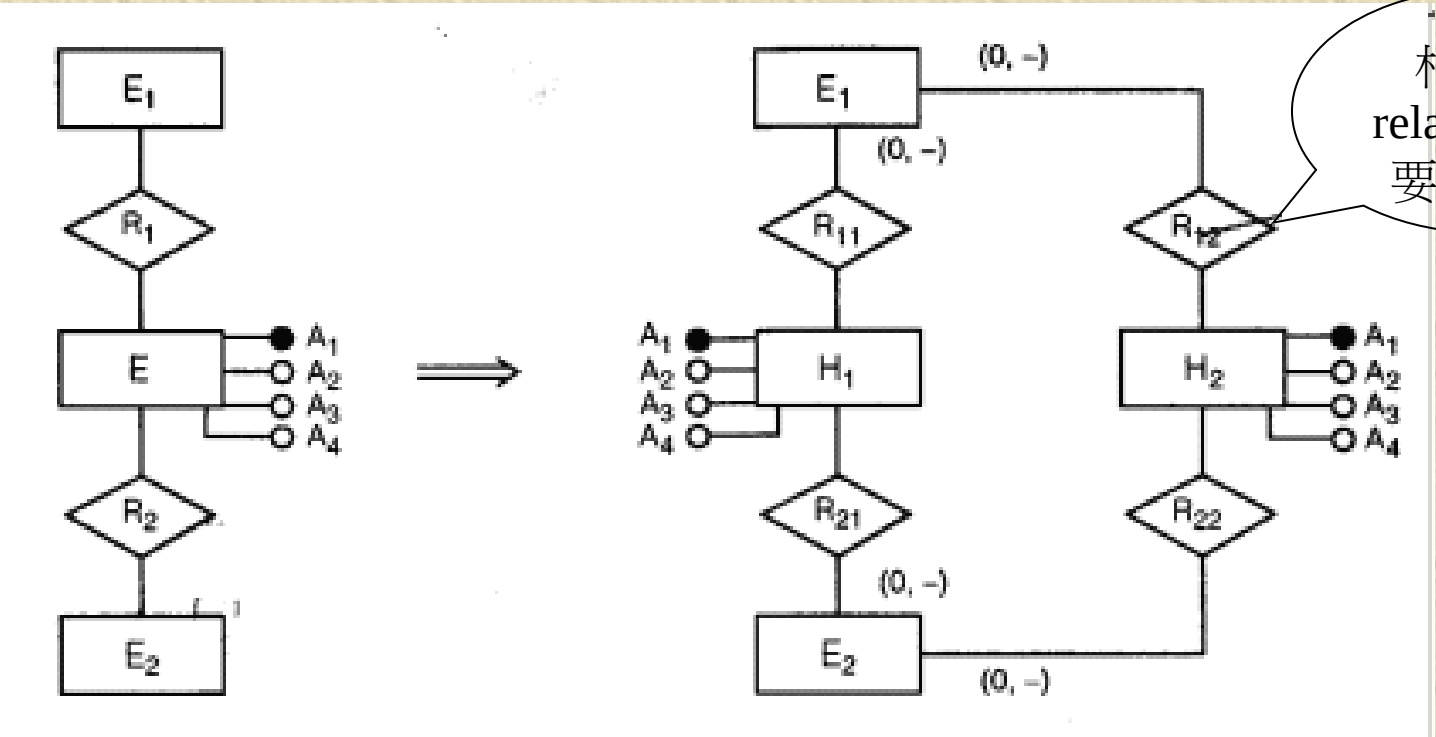
- E 分為 $E_1$, $E_2$, …, $E_n$, 每個 $E_i$的attributes 都和 E 一樣.

Instances of E



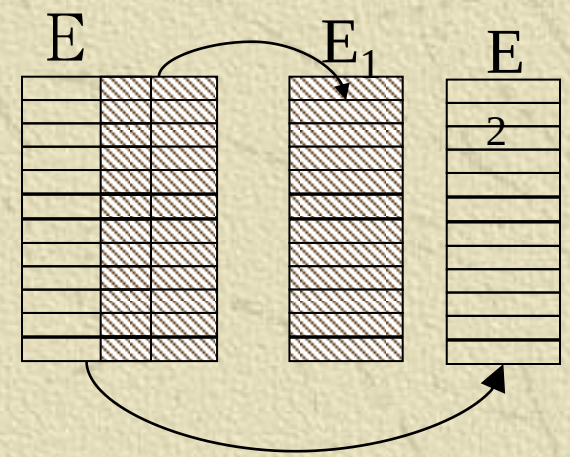



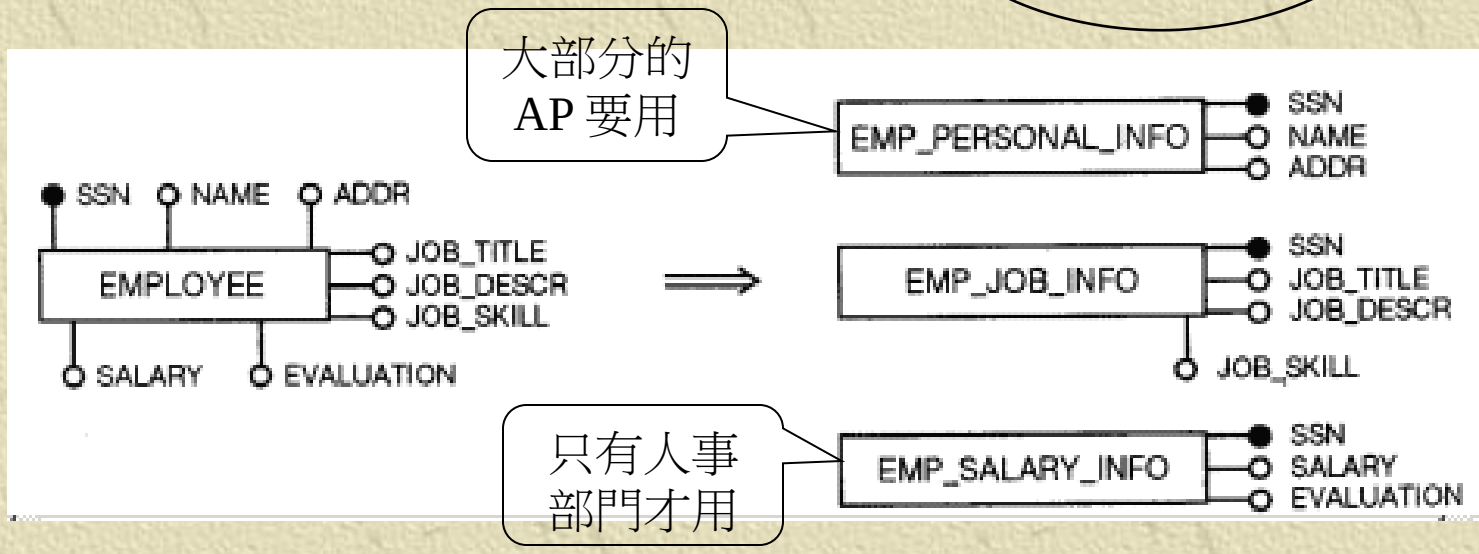
# 水平分割的一般原則


E1
R1
E
A1
A2
A3
A4
R2
E2
(0, –)
E1
(0, –)
R11
R12
A1
A2
A3
A4
H1
H2
A1
A2
A3
A4
R21
R22
(0, –)
E2
(0, –)
相連的
relationship
要跟著分
出

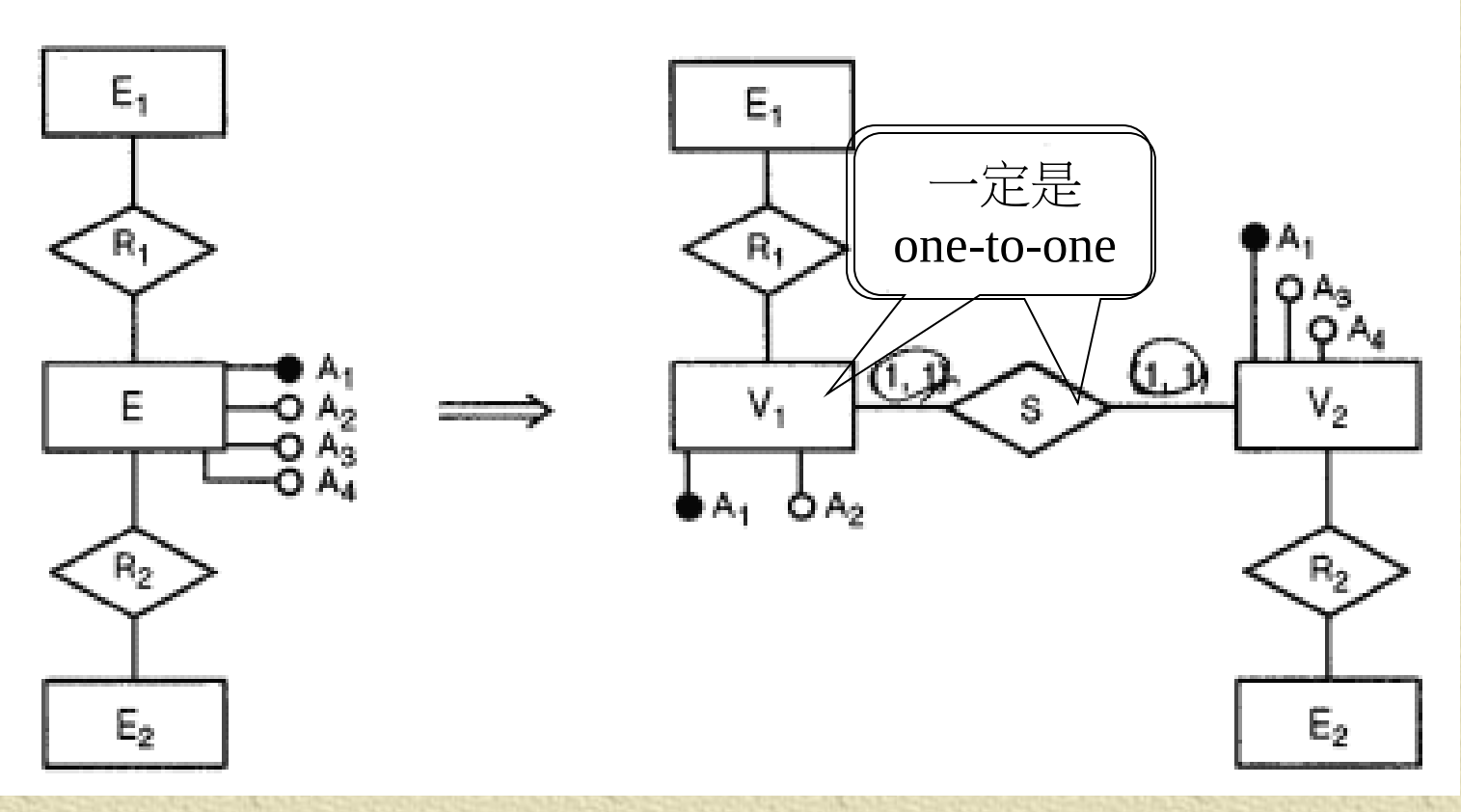
# Vertical partition (垂直分割)

- E 分為 $E_1$, $E_2$, …, $E_n$, 每個 $E_i$都和 E有一樣個數的data instances, 但是attributes 則不同.

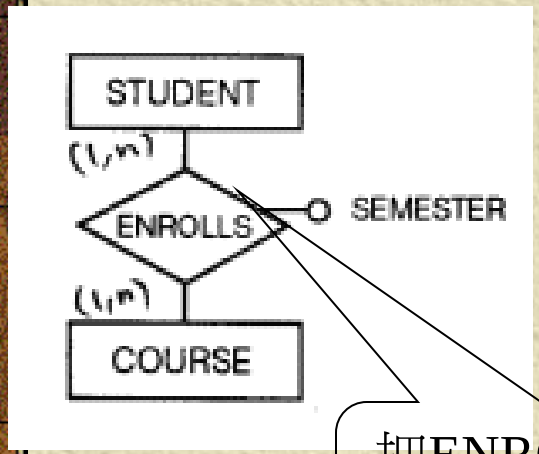


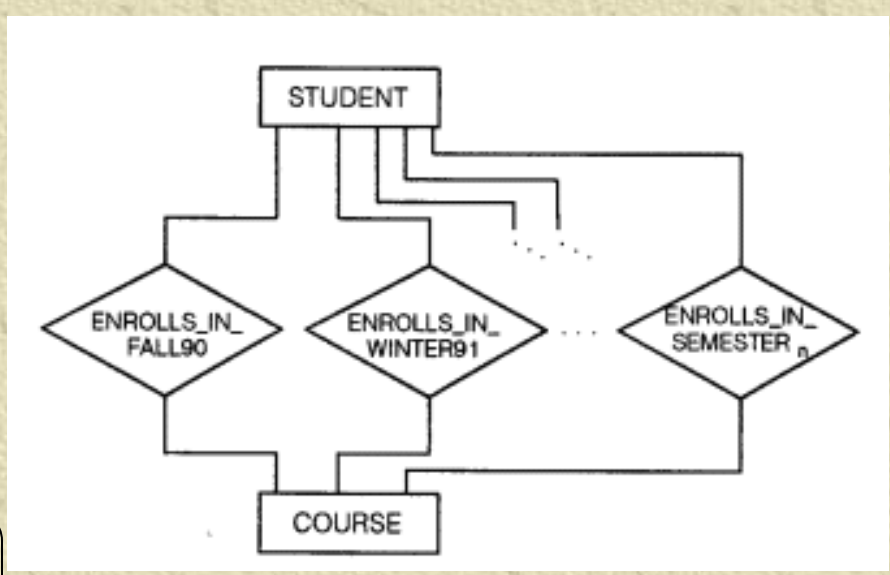
# 垂直分割的一般原則


E1
R1
E
A1
A2
A3
A4
R2
E2
E1
R1
一定是
one-to-one
V1
(1, 1)
S
(1, 1)
V2
A1
A3
A4
A1
A2
R2
E2

# Relationship 的分割



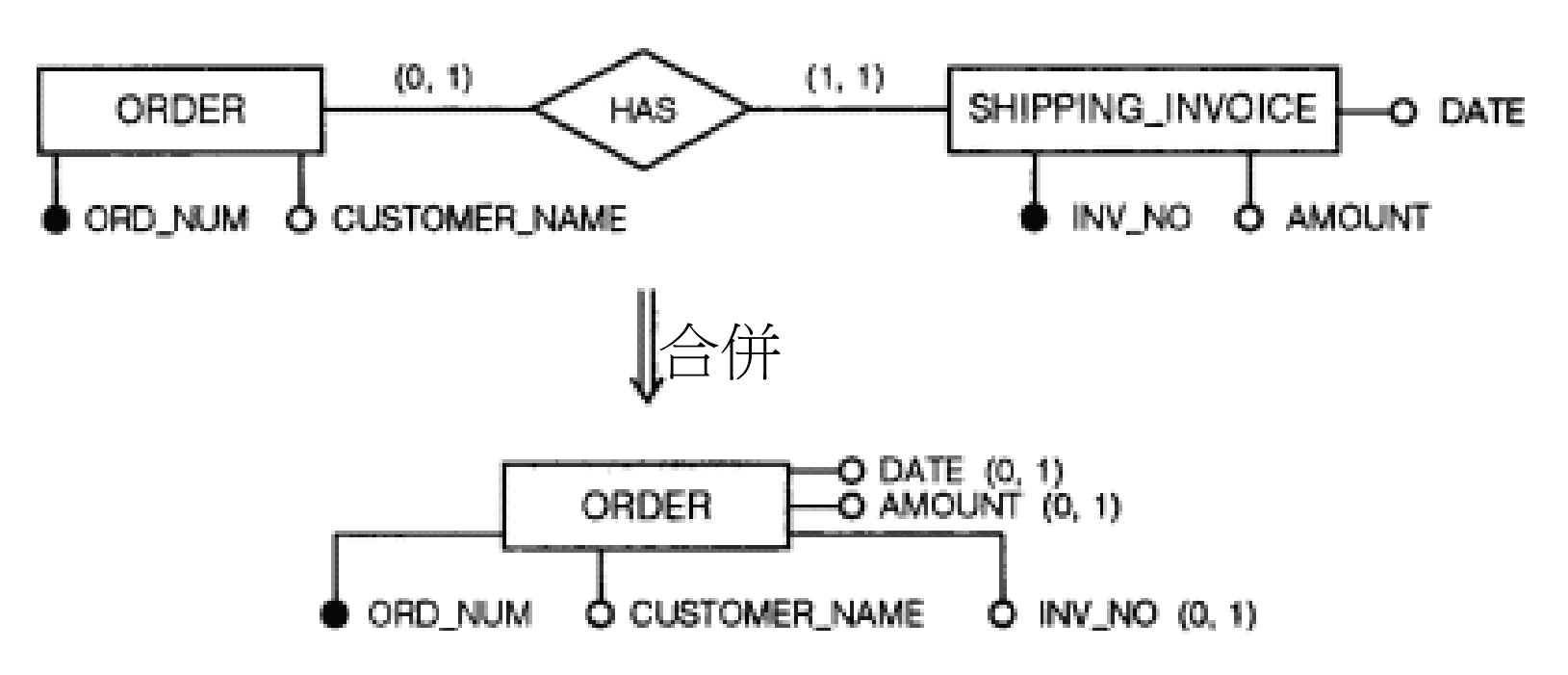

把ENROLLS 依註冊的學期分割



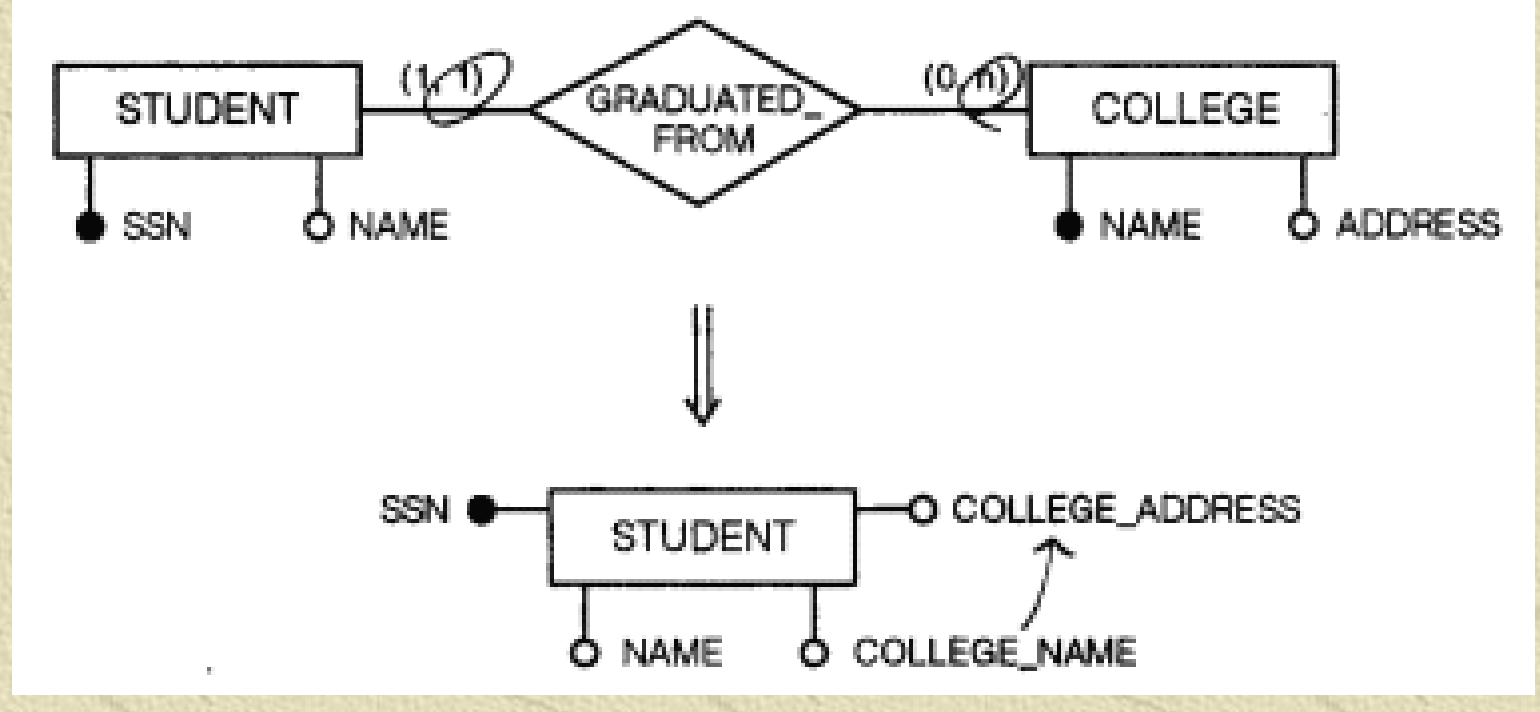
# Merging Entities and Relationships

- 當兩個或兩個以上的entities 必須經常被某些 operations 一起使用時, 則適合將之合併為一個 entity以簡化operation.
- 當欲合併的兩個entities 其關係為
  - One-to-one時, 合併之後不受影響
  - One-to-many時, 合併之後會違反第三正規化
  - Many-to-one時, 合併之後會違反第二正規化
- 是否該合併, 宜按 operations 之重要性仔細評估.

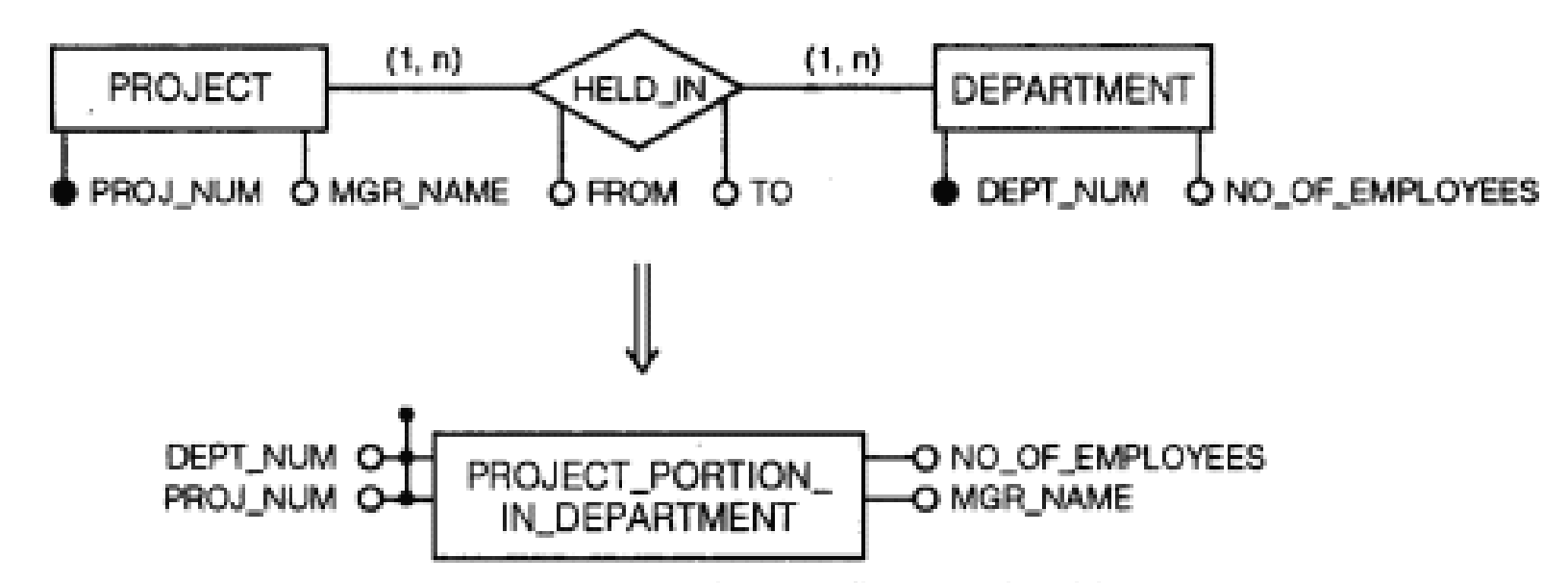
# Entity merging with a one-to-one intermediate relationship



# Entity merging with a one-to-many intermediate relationship



# Entity merging with a many-to-many intermediate relationship

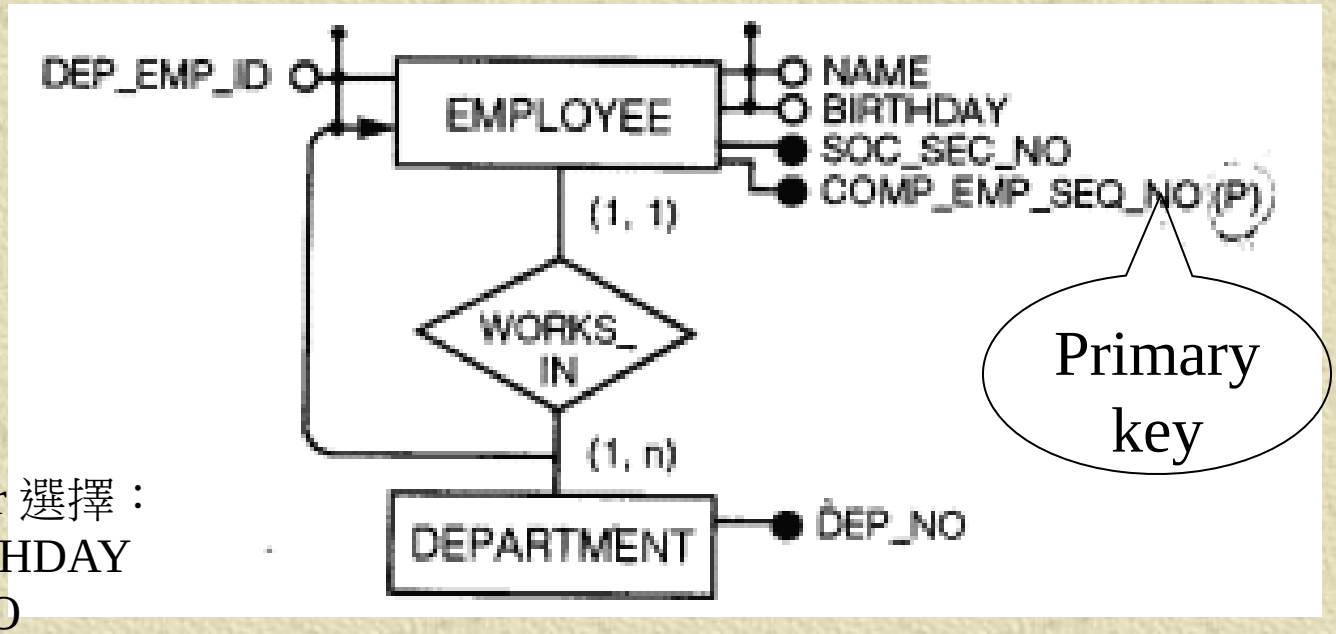



PROJ_NUM→ MGR_NAME
DEPT_NUM → NO_OF_EMPLOYEE

# Primary Key Selection

- 從數個 identifier中選一個為 primary key
- 選 primary key 的原則:
  - 最多 operations 拿來當 direct access 的 identifier
  - Simple identifier 優於　multiple identifier
  - Internal identifier 優於　external identifier

# Selection of the primary key for the entity EMPLOYEE




四個 identifier 選擇：
NAME+BIRTHDAY
SOC_SEC_NO
COMP_EMP_SEQ_NO
DEP_EMP_ID+DEP_NO

# Exercises

1. Assume that the following operations are defined in the schema of next page

01: CREATE A NEW PROJECT AND ASSIGN EMPLOYEES TO THE PROJECT.

02: CHANGE THE MANAGER OF THE PROJECT.

03: CHANGE A GIVEN EMPLOYEE'S ASSIGNMENT FROM ONE PROJECT TO A DIFFERENT PROJECT.

04: FIND PROJECTS WITH MORE THAN 10 EMPLOYEES ASSIGNED.

05: FIND MANAGERS WHO MANAGE MORE THAN ONE PROJECT.

Cardinality information is provided on the schema. Assume reasonable data volumes and operation frequencies for the above operations. Then draw navigation schemas and set up an operation access-volume table for each of the above operations.

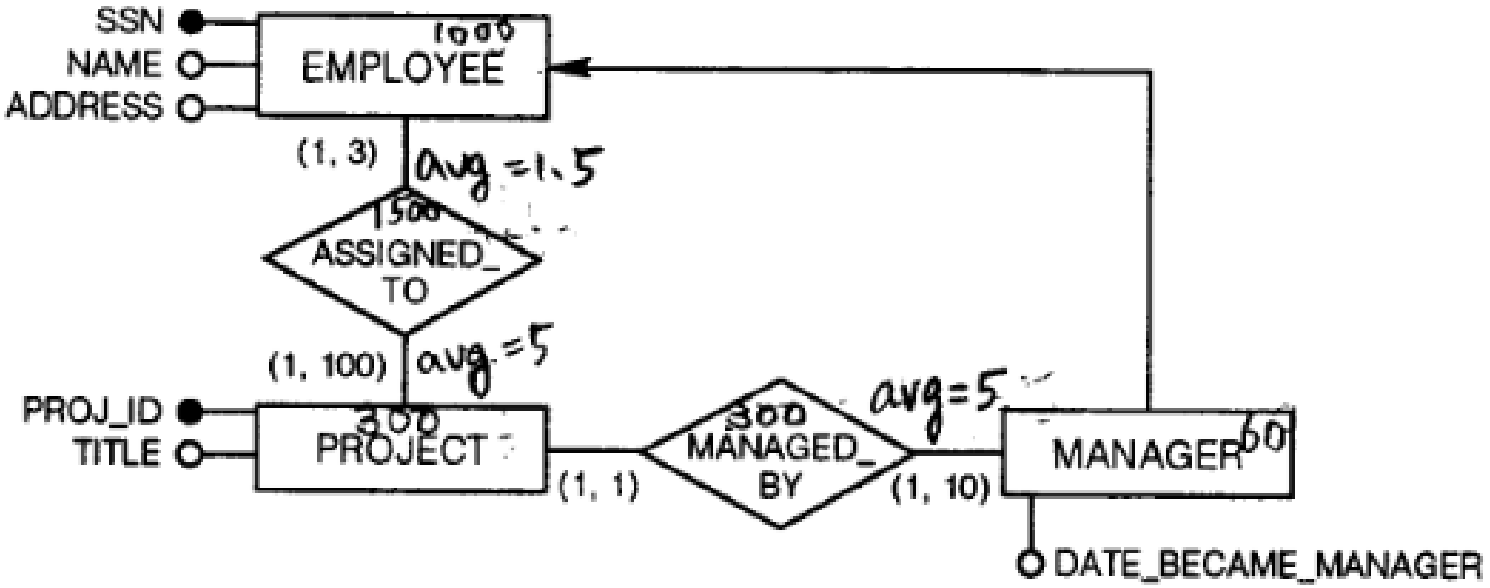



**Figure 11.16** A project database schema

2. Design the information system of a medical diagnosis laboratory. Several types of persons are of interest: doctors, hospital attendants, and patients. For each of them we represent the last name, age, and a code. Patients (approximately 60,000) need medical examinations that must be reserved in advance. The history of the examinations of the last 180 days is stored in the system. Examinations are of certain types, identified by a code, a description, and a price. The price of the examination depends also on the type of patient. Each doctor (500 of them) and hospital assistant (1100 of them) is able to perform only certain types of examinations. Examinations are done in specific rooms. Each examination must be approved with a doctor's name.

Together with examinations (200 a day), visits (50 a day) must also be scheduled. Examinations can be assigned either as sequels to visits, or independently. Every examination has a result, and the results of both examinations and visits must be stored in a log for the patient, which should store the history of the last 30 visits or examinations. Examinations may be done by doctors and hospital assistants, but visits are made only by doctors.

The main operations on the database follow:

01: CREATE A NEW PATIENT.
02: SCHEDULE AN EXAMINATION ON THE FIRST AVAILABLE DAY.
03: PRINT THE MEDICAL HISTORY OF A PATIENT.
04: COMPUTE THE STATISTICS OF A NUMBER OF PATIENTS VISITED BY EACH DOCTOR AND EACH HOSPITAL ASSISTANT EVERY MONTH.
05: CHANGE A SCHEDULED VISIT.
06: CHANGE THE APPOINTMENT OF A PATIENT FROM ONE DOCTOR TO ANOTHER.
07: CHANGE THE PRICES OF EXAMINATIONS.
08: COMPUTE THE TOTAL AMOUNT TO BE PAID BY A PATIENT.
09: PREPARE A RECEIPT FOR A PATIENT.
010: CHANGE THE TYPE OF A PATIENT.

Complete the specifications with reasonable attributes for entities and provide load data for concepts in the schema. Add new reasonable operations and assign frequencies to them. Develop all three required tables. Then go through the decisions discussed in this chapter. (You may want to perform a logical design of the database for a certain model by consulting an appropriate subsequent chapter.)