บทที่ 5
รูปแบบฐานข้อมูลเชิงสัมพันธ์
(Relational Database Models)

5.1 เอนทิตี้และลักษณะประจำ (Entities and Attributes)

Entity หมายถึง คน, สถานที่, เหตุการณ์ หรือ สิ่งใดก็ได้ที่เราต้องการเก็บข้อมูล ตัวอย่างเช่น

ใน University Environment

Entity อาจได้แก่ ครู, นักเรียน, วิชาเรียน เป็นต้น

โดย Entity แต่ละตัวจะมีลักษณะกำหนด (Characteristics) ที่แน่นอน เรียกว่า **Attributes** (ลักษณะประจำ)

ตัวอย่าง Entity Set (กลุ่มของเอนทิตี้) ที่ชื่อว่า Product จะประกอบด้วย Entity ที่ชื่อว่า Pen กับ Book

Student Entity

Attribute - SId, SName, SAddress, SMajor, SGPA.

โดย Attribute ควรมีการตั้งชื่อให้เหมาะสม

EntitySet คือ กลุ่มของ Entities ที่สัมพันธ์กัน และควรมีการตั้งชื่อที่สัมพันธ์กับ ข้อมูลที่ อยู่ใน Entity Set นั่นเช่น

- Student เป็นชื่อของ Entity Set ที่เก็บข้อมูลของ Entity ที่เป็น Student

ตาราง (Tables)

Table เป็น Logical View ของ Relational Database จะแสดงได้ในรูปของ Table โดย Table นี้จะเก็บ กลุ่มของ Entity ที่สัมพันธ์กัน นั่นคือ Entity Set นั่นเองดังนั้น Table ก็คือ Entity Set นั่นเอง และ Table เรียกอีกอย่างหนึ่งว่า Relation (มาจาก CODD)

ลักษณะของตาราง

1. เป็นโครงสร้าง 2 มิติ ประกอบด้วย Row และ Column
2. แต่ละ Row (Tuple) แสดงถึง Entity แต่ละข้างที่มีใน Entity Set นั้น

CT 316 (S) 33

2. แต่ละ Row (Tuple) แสดงถึง Entity แต่ละข้างที่มีใน Entity Set นั้น

3. แต่ละ Column จะแสดงถึง Attribute ซึ่งแต่ละตัวจะต้องมีชื่อที่แตกต่างกัน
4. แต่ละ Row และ Column ที่ตัดกันจะแสดงค่าของข้อมูล 1 ตัว
5. แต่ละ Table ต้องมี Primary Key ที่ใช้ระบุแต่ละ Row ได้โดยไม่มีการซ้ำกัน
6. ข้อมูลของแต่ละ Column ต้องมีรูปแบบ (Format) เกี่ยวกันคือถ้าเป็น Integer ต้อง Integer หมด
7. แต่ละ Column ต้องมีการกำหนดขอบเขตของข้อมูลที่เรียกว่า Attribute Domain
8. แต่ละ Row จะเก็บข้อมูลที่เกี่ยวกับ Entity 1 ตัวเท่านั้น
9. ลำดับของ Row และ Column ไม่มีความจำเป็นต่อ User

Student

SId	SName	SAddress	SPhone	Smajor	SGPA.

Customer

CId	CName	CAddress	CPhone

5.2 Keys

การควบคุมการเกิด Redundancy เป็นหลักสำคัญของ Relational Database Table ในฐานข้อมูลจะมีการใช้ Attribute ร่วมกันเพื่อทำให้เกิดการเชื่อมความสัมพันธ์ระหว่าง Table เข้าด้วยกัน

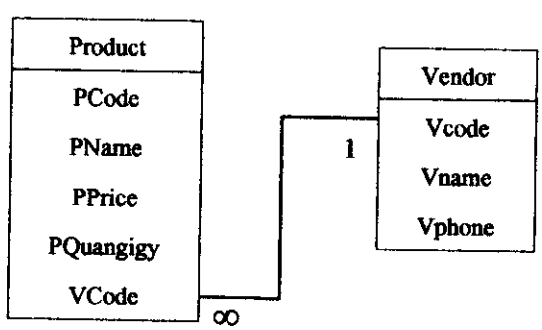


Table PRODUCT (PCode, PName, PPrice, PQuantity, VCode)

Primary Key PCode

Foreign Key VCode

Table Vendor (VCode, VName, VPhone)

Primary Key VCode

Foreign Key Non.

Key คือสิ่งที่ช่วยในการระบุ Entity และ Entity Relationship

การใช้ Key นั้นเป็นการใช้แนวความคิดเรื่อง Determination

- "A Determines B" หมายความว่า การรู้ Attribute Value ของ A จะ ทำให้เราสามารถรู้ Attribute Value ของ B ได้

A → B

ตัวอย่างเช่น การรู้ PId จะทำให้เราสามารถค้นหา PName. ได้

นั่นคือ PId → PName, PPrice, PQuantity, VCode.

ทำให้เกิด Functional Dependence

"The Attribute B is Functionally Dependence on a if a Determines B"

PId → PPrice

PPrice ↛ PId เพราะสินค้า 2 ตัวอาจมีราคาเท่ากันได้

Hours	Classification
0 – 29	FR
30 – 59	SO
60 – 89	JU
90 – 124	SR

Hours = Class

Class ↛ Hour

จากแนวความคิดเรื่อง Functional Dependence เราได้ว่า Key ก็คือ Attribute ที่เราใช้ในการพิจารณาข้อมูลของ Attribute อื่นๆ ที่มีใน Entity

ซึ่งบางกรณีอาจใช้มากกว่า 1 Attribute ก็ได้ ในการพิจารณา

Key Attribute หมายถึง Attribute ใดๆ ที่ทำหน้าที่เป็น Key

Composite Key คือ Attribute หลายๆ ตัวที่ร่วมกันทำหน้าที่เป็น Key

ถ้า Attribute (B) เป็น Functional Dependent กับ Composite Key (A) แต่ไม่เป็นกับ Subset ใด ของ A เรียกว่า B เป็น Fully Functionally Dependent on A.

Superkey คือ Key ที่แสดง Entity โดยไม่มีการซ้ำกัน → SId

นั่นคือ Superkey Functionally Determines all on the Entity's Attributes

<u>ตัวอย่าง</u> Student (SId, SLname , SFname, SPhone, SMajor, SGPA)

Superkey ได้แก่ SId

SId, SLname - Composit Key

SId, SLname, SFname.

จะเห็นว่า SId เป็น Superkey ที่ดีที่สุด

Candidate Key คือ Superkey ที่ไม่มีการซ้ำ

SId, SLname เป็น Superkey แต่ไม่เป็น Candidate Key เพราะ

SId เป็น Candidate Key แล้ว

SLname, SFname, SPhone เป็น Candidate Key ได้ ถ้าไม่มีการเกิดการซ้ำกันของ ข้อมูล

Primary Key คือ Superkey และเป็น Candidate Key

Secondary Key คือ Key ที่กำหนดขึ้นมาใช้ในการทำ Data Retrieval เท่านั้น

Relational Database Keys

Superkey : An Attribute (Or Combination of Attribute) That Uniquely Identifies Each Entity in a Table

: ก็คือการที่ Attribute ใด Attribute หนึ่ง หรือ กลุ่มของ Attribute ที่ใช้ระบุ Entity ในตารางได้โดยไม่มีการซ้ำกัน นั่นคือ Superkey จะเป็น Functionally Determination สำหรับ Attribute ทุกๆ ตัวที่มีอยู่

เช่น จาก STUDENT (SId, SName, SAddress, SPhone, SGrade)

Superkey : SId

SId , SName

SId , SName , SAddress

SId , SAddress

SId , SAddress , SPhone

Candidate Key : A Minimal Superkey a Superkey That Does not Contain a Subset of Attributes That is Itself a Superkey

: Superkey ที่สั้นที่สุดที่ Subset ของตัวมันเองไม่เป็น Superkey นั่นก็คือ Superkey ที่ไม่มีการซ้ำกันเกิดขึ้น

เช่น Candidate Key : Sid, SName , SAddress

Primary Key : A Candidate Key Selected to Unuely Identify All Other Attribute Values in any Given Row Cannot Contain Null Entries.

: Superkey ที่เป็น Candidate Key ต้อง

: Candidate Key ใด Candidate Key หนึ่ง ที่ถูกเลือกขึ้นมาจาก Candidate Key ทั้งหมดที่สามารถระบุค่าของ Attribute อื่นๆ ทั้งหมดในแถวได้ โดยค่าของมันในแต่ละแถวของตาราง จะมีค่าไม่ซ้ำกัน และไม่สามารถจะมีค่าเป็น Null หรือการไม่มีค่าได้

เช่น ในที่นี้เลือก Primary Key : Sid, SName , SAddress

จะสังเกตได้ว่า Primary Key จะต้องเป็นทั้ง Candidate Key และ Superkey

Secondary Key : An Attribute (Or Combination Of Attributes) Used Strictly For Data Retrieval Purpose.

: Attribute ใด Attribute หนึ่ง หรือ กลุ่มของ Attribute ที่ถูกกำหนดขึ้นมาใช้ในการทำ Data Retrieval (การดึงข้อมูล) เท่านั้น

Foreign Key : An Attribute (Or Combination Of Attributes) In One Table Whose Values Must Either Match The Primary Key In Another Table Or Be Null

: Attribute ใด Attribute หนึ่ง หรือ กลุ่มของ Attribute ในตารางหนึ่งที่ค่าของมันมีค่าเข้ากันได้กับ Primary Key ในตารางอื่น ซึ่งอาจมีค่าเป็น Null ได้คือไม่มีค่าได้

ตัวอย่าง

Product

PId	PName	Price	SId
P001	~~~~~~~~	~~~~	S002
P002	~~~~~~~~	~~~~~	S001
P003	~~~~~~~~	~~~~~	
P004	~~~~~~~~	~~~~~	S003

Sale

SId	SName	Area
S001	~~~~~~~~	~~~~~~~
S002	~~~~~~~~	~~~~~~~
S003	~~~~~~~~	~~~~~~~

ไม่มีค่ากำหนด แสดงว่ามีค่าเป็น Null

ในตาราง Product นี้มี Foreign Key : SId

ส่วนในตาราง Sale มี Primary Key : Sid

5.3 กฎความมั่นคง (Integrity Rules)

ในการออกแบบ Relational Database ที่ดีนั้นเราพบว่าเรื่องของ Integrity Rules มีความสำคัญอย่างยิ่ง โดยปกติ RDBMS จะมีการสนับสนุนเรื่อง Integrity Rules อยู่แล้ว อย่างไรก็ตามในการออกแบบ Application Program ที่จะมาใช้กับ RDBMS นั้นเราควรนำหลักการ 2 ประการเกี่ยวกับ Integrity Rules มาพิจารณาในการออกแบบนั่นคือ

1. Entity Integrity
2. Referential Integrity

Entity Integrity : (ความถูกต้องมั่นคงในเรื่องของ Entity) All Entries Are Unique And No Null Entries In a Primary Key

จุดประสงค์ : เพื่อให้แน่ใจว่า Entity แต่ละตัวเป็น Unique Identity.

ตัวอย่าง เช่น

ถ้าเป็น Primary Key จะมีค่าเป็น Null (การไม่มีค่า) ไม่ได้

PRODUCT

Pid	Pname	PPrice	SId
P001	Pen	15.00	S01
P002	Book	50.00	S02
P003	Chair	500.00	

ไม่ได้เป็น Primary Key จึงไม่มีค่าก็ได้

Referential Integrity : (ความมั่นคงในเรื่องของการอ้างอิงผ่าน Foreign Key)

Foreign Key Must Have Either a Null Entry Or An Entry That Matches The Primary Key Value In a Table To Which It Is Related.

จุดประสงค์ : ทำให้เกิดกรณีที่เป็นไปได้สำหรับ Attribute ที่ไม่มีค่า (Null) ได้ และไม่เกิดกรณีที่มีการอ้างถึง Invalid Entry ทั้งยังเป็นการป้องกันการลบข้อมูล Row ที่มี Primary Key ไปตรงกับ Foreign Key ของอีก Table หนึ่ง

ตัวอย่าง เช่น

SALE

Sid	SName	Area
S01	A	West
S02	B	East
S99	ZZZ	ZZZZ

โดย SId เป็น Primary Key ของ Table SALE แต่จะเป็น Foreign Key ของ Table PRODUCT ซึ่งมันสามารถมีค่าเป็น Null ได้ และค่าของมันจะต้องไม่ตรงกันกับ Primary Key ของอีก Table

Dummy Data : เป็นข้อมูลที่สร้างขึ้นมาเพื่อเอาไปใช้แทนในค่าตารางตรงที่มีค่าเป็น Null

จุดประสงค์ คือ ทำให้เกิดกรณีที่เป็นไปได้สำหรับ Attribute ที่ไม่มีค่า และ ไม่เกิดกรณีที่มีการอ้างอิงถึง Invalid Entity (ข้อมูลที่ไม่ถูกต้อง) ทั้งยังเป็นการป้องกันไม่ให้เกิดการลบข้อมูลของ Row ที่มี Primary Key ที่ไม่ตรงกันกับ Foreign Key ของอีก Table หนึ่ง

5.4 ตัวดำเนินการกับฐานข้อมูลเชิงสัมพันธ์ (Relational Database Operators)

Relational Database Operators ใช้ Relational Algebra กำหนดแนวทฤษฎีในการจัดการข้อมูลใน Table ซึ่งประกอบด้วย 8 Functions

Union : เป็น Operator ที่ใช้แสดงรวมทุก Row ที่มีใน Table ที่ 1 และ Table ที่ 2 โดย Table ที่จะเอามา Union กันได้นั้นจะต้องมี Attribute และ Domain ที่เหมือนกัน ซึ่งเราเรียกว่าเป็นคุณสมบัติของ Union Compatible คือ คุณสมบัติที่บอกว่า Table ที่จะมา Union กันจะต้องมี Attribute เหมือนกัน และมี Domain ที่เข้ากันได้ด้วย

(Domain: เป็นลักษณะของโครงสร้างของข้อมูลที่กำหนดให้กับ Attribute หรือ ก็คือขอบเขตนั่นเอง)

ตัวอย่าง ถ้ามี Set 2 Set คือ

$A = \{ a, b, c \}$

$B = \{ c, d, e \}$

$A \cup B = \{ a, b, c, d, e \}$

แต่ถ้าเรามี Table 2 Table ดังนี้ คือ

PRODUCT

P_ID
1111
2222
3333

Domain = 1 – 9999

PRODUCT1

P_ID
3333
4444
5555

Domain=1- 9999

จะเห็นได้ว่าทั้ง 2 Table นี้มันมีทั้ง Attribute ที่เหมือนกันคือ P_ID และ Domain ที่เหมือนกัน คือ 1 – 9999

ดังนั้น มันจึงสามารถ Union กันได้ ได้เป็น

PRODUCT ∪ PRODUCT1

P_ID
1111
2222
3333
4444
55555

Intersect : จะเป็น Operator ที่จะใช้ในการแสดงเฉพาะ Row ที่มีในทั้ง 2 Tables เท่านั้น ซึ่งมันต้องมีลักษณะของ Union Compatible ด้วยเช่นกัน

ตัวอย่าง PRODUCT ∩ PRODUCT1 จะเป็น

P_ID
3333

Difference : จะเป็น Operator ที่แสดงทุก Row ที่มีใน Table ที่ 1 แต่ไม่มีใน Table ที่ 2 และจะต้องเป็น Union Compatible

ตัวอย่าง PRODUCT - PRODUCT1 จะเป็น

P_ID
1111
2222

Product : เป็น Operator ที่ใช้แสดงทุกความเป็นไปได้ของ Row ที่จะจับคู่กันใน Table ทั้ง 2 Table โดย Table ทั้ง 2 ไม่จำเป็นต้องเป็น Union Compatible กันก็ได้ ซึ่งจะเรียกว่าเป็นการทำงานแบบ Cartisian Product คือความเป็นไปได้ในทุกๆ กรณี

PRODUCT

P_ID	P_NAME	PRICE
P001	PEN	10.00
P002	BOOK	80.00
P003	TABLE	500.00

Table PRODUCT มี 3 Attribute

3 Tuple

SALE

S_ID	S_NAME
S01	A
S02	B

Table SALE มี 2 Attribute

2 Tuple

ดังนั้นผลที่ได้จากการนำ ทั้ง 2 Table มาทำการ PRODUCT กัน จะทำให้ได้ Table ใหม่ ที่ประกอบด้วย Attribute ทั้งสิ้น (3 + 2) = 5 Attribute

Tuple ทั้งสิ้น (3 X 2) = 6 Tuple ดังนี้

P_ID	P_NAME	PRICE	S_ID	S_NAME
P001	PEN	10.00	S01	A
P002	BOOK	80.00	S01	A
P003	TABLE	500.00	S01	A
P001	PEN	10.00	S02	B
P002	BOOK	80.00	S02	B
P003	TABLE	500.00	S02	B

Select : เป็น Operator ที่จะใช้แสดงข้อมูลทุก Attributes ที่มีใน Row ที่เราสนใจ (แสดงข้อมูลในลักษณะของ Row)

ตัวอย่าง เช่น

Select ALL

ผลคือ จะแสดงข้อมูลทุก Tuple

Select Where PRICE = 10.00

ผลที่ได้คือ P001 PEN 10.00

Select Where PRICE > 10.00

ผลที่ได้คือ P002 BOOK 80.00

P003 TABLE 00.00

Project : เป็น Operator ที่ใช้แสดงข้อมูลของ Attributes ที่เราสนใจเท่านั้น (แสดงข้อมูลในลักษณะของ Column)

ตัวอย่าง เช่น

Project PRICE

จะได้

10.00
80.00
500.00

Projcct P_ID, PRICE

ก็จะได้

P001	10.00
P002	80.00
P003	500.00

โดย SQL จะนำมา Select , Project กับ Cartisial Product (PRODUCT) มาใช้ในลักษณะ

Select

From

[Where] [ ] คือจะมีหรือไม่มีก็ได้

เช่น

Select PRICE

From PRODUCT

Where PRICE > 10.00

ผลที่ได้ คือ

P002 BOOK 80.00

P003 TABLE 500.00

Join : เป็น Operator ที่ใช้รวมข้อมูลของ 2 Tables เข้าด้วยกัน แต่มีการทำงานที่ประกอบด้วย 3 ขั้นตอนย่อย คือ

1. ทำการ Product Table ทั้ง 2 ก่อน
2. ทำการ Select ผลที่ได้จากการ Product
3. ทำการ Project ผลที่ได้จากการ Select

โดย Join จัดว่าเป็น Operator ที่สำคัญที่สุดในการทำงานของ Relational Database

<u>ตัวอย่าง</u> ถ้ามี Table 2 Table คือ

PRODUCT

P_ID	P_NAME	PRICE	S_ID
P001	PEN	10.00	S01
P002	BOOK	80.00	S02
P003	TABLE	500.00	S03

SALE

S_ID	S_NAME
S01	A
S02	B

เราต้องการที่จะให้มัน Join กันเพราะอยากทราบว่าใครขายสินค้า ดังนั้นเราต้องทำการ Product มันก่อน ได้ Table ใหม่ดังนี้

P_ID	P_NAME	PRICE	S_ID	S_ID	S_NAME
P001	PEN	10.00	S01	S01	A
P002	BOOK	80.00	S02	S01	A
P003	TABLE	500.00	S01	S01	A
P001	PEN	10.00	S01	S02	B
P002	BOOK	80.00	S02	S02	B
P003	TABLE	500.00	S01	S02	B

ซึ่งมี 6 Attribute และ 6 Tuple

ต่อจากนั้น ก็นำผลที่ได้นี้มาทำการ Select โดยจะ Select เฉพาะ Row ที่มี S_ID ทั้ง 2 ที่ตรงกัน ทำให้ได้ Table ใหม่ ดังนี้

P_ID	P_NAME	PRICE	S_ID	S_ID	S_NAME
P001	PEN	10.00	S01	S01	A
P002	BOOK	80.00	S01	S01	A
P003	TABLE	500.00	S02	S02	B

ซึ่งเรามี 6 Attribute กัน 3 Tuple

สุดท้ายให้นำผลที่ได้มาทำการ Project ซึ่ง Row จะเท่ากัน แต่ Attribute จะเปลี่ยนไป โดยจะ Project เอาเฉพาะ Attribute ที่ไม่ซ้ำกันทำให้ได้ Table ใหม่ ดังต่อไปนี้

P_ID	P_NAME	PRICE	S_ID	S_NAME
P001	PEN	10.00	S01	A
P002	BOOK	80.00	S01	A
P003	TABLE	500.00	S02	B

ซึ่ง Table สุดท้ายที่ได้นี้ ก็คือ ผลที่ได้จากการ Join นั่นเอง แต่ถ้าใน SQL ก็จะใช้คำสั่งดังนี้ เช่น

```
Select      S_NAME
From        PRODUCT , SALE
Where       P_NAME = PEN
```

← บรรทัดนี้ DBMS จะทราบว่าต้องเอา Table ชื่อ PRODUCT กับ SALE มา JOIN กันก่อน แล้วจึงจะสามารถดึงเอาข้อมูลที่ต้องการจากคำสั่งในบรรทัดอื่นออกมาได้

ผลที่ได้คือ A

Divide :

<u>ตัวอย่าง</u> เช่น

P_ID	P_NAME	PRICE
P001	PEN	10.00
P002	BOOK	80.00
P003	TABLE	500.00

Divide →

P_ID
P001
P002

→

P_NAME	PRICE
PEN	10.00
TABLE	500.00

จากการทำงานของ Relational Algebra ทั้ง 8 Function และ กฎเกณฑ์ของ Integrity ทั้ง 2 ประการ ทำให้ได้มีการพัฒนาทางด้าน Software ทางด้าน Relational Algebra Database ออกมาหลายชนิด โดยแต่ละชนิดก็จะมีขีดความสามารถในการทำงานที่แตกต่างกัน

Relational Database Software Classification

จากการทำงานที่ได้รับความนิยมของ Relational Database ทำให้มีการพัฒนา software ที่ใช้กับ Relational Database Systems software C.J. DATES สรุปแบ่งประเภทของ Software ของ

Relational Database ออกได้เป็น 4 แบบ คือ

1. Fully Relational (ความสัมพันธ์เต็มรูปแบบ): Supports All Eight Relational Algebra Functions and Also Enforecis Both Entity and Referential Integrity Rules

: เป็น Software ทาง Relational Algebra Database ที่จะต้องทำงานทั้ง 8 Function ได้ครบ รวมทั้งสนับสนุนทั้ง Entity และ Relational Ingegrity

2. Relationally Complete (ความสัมพันธ์ที่สมบูรณ์): Suppotrs all Eighi Relation Algebra Functions but no the Integrity Rules.

: คือ Software ที่จะทำงานได้ครบทั้ง 8 Function ของ Relational Algebra แต่ไม่ตรวจสอบคุณสมบัติของ Integrity เลย

3. Minimal Relational (ความสัมพันธ์น้อยที่สุด): Supports Only Select Project and Join

: จะทำงานได้เฉพาะ Function Select ,Project และ Join เท่านั้น

4. Tabular (จัดระเบียบ): Support Only Select Project and Join and Requires That all Access Paths be Defined by The User.

: เป็น Software ประเภทที่จะสนับสนุนการทำงานของ Select, Project, Join เท่านั้น และผู้ใช้จะต้องเป็นผู้กำหนดแนวทางและเงื่อนไขในการเรียกข้อมูลด้วยตัวเอง

5.5 พจนานุกรมข้อมูลและระบบรายระเอียด (Data Dictionary and System Catalog)

Data Dictionary (พจนานุกรมข้อมูล): ทำหน้าที่เก็บรายลเอียดเกี่ยวกับ Tables ทั้งหมดที่มีอยู่ในฐานข้อมูล ที่มีผู้ใช้สร้างขึ้นมารายละเอียดที่ ฐานข้อมูล เก็บได้แก่ ชื่อ Table, ชื่อ Attribute, รายละเอียดของ Attribute เช่น ชนิด,ความยาว,Format, Key เป็นต้น หรือ ฐานข้อมูล เก็บ Metadata นั่นคือ Data About Data เช่น

Customer

C-Id	CNane	C-Address	C-Phone	S-Id
00001	XXX	XXXX	XXX-XXXXXX	01

Sale

S-Id	S-Name	Area
01	AAA	W
02	BBB	E

Data Dictionary

TABLE.NAME ATT.NAME CONTENTS TYPE LENGTH FORMAT

TABLE NAME	ATT.NAME	CONTENTS	TYPE	LENGTH	FORMAT	RANGE	KEY	PK,FK	FK TABLE
CUSTOMER	C-ID	CUSTOMER IDENTIFIER	FCHAR	5	99999	1-99999	Y	PK	
	C-NAME	CUSTOMER NAME	VCHAR	25	X(25)				
	C-ADDRESS		VCHAR	30	X(30)				
	C-PHONE		VCHAR	11	X(11)				
	S-ID	SALE IDENTIFIER	FCHAR	2	99		Y	FK	SALE
SALE	S-ID	SALE IDENTIFIER	FCHAR	2	99	1-99	Y	PK	
	S-NAME	SALE NAME	VCHAR	25	X(25)				

System Catalog : เก็บ Metadata เช่นเดียวกับ Data Dictionary แต่เปรียบได้เป็น Data Dictionary ที่เก็บข้อมูลที่ละเอียดมาก คือ เก็บรายละเอียดเกี่ยวกับ Object ทุกอย่างที่มีใน ฐานข้อมูล เช่น

- ชื่อ Table
- ผู้สร้าง Table
- วันที่สร้าง Table

จำนวน Column ใน Table

ชื่อ Column

Data Type ของ Column

ชื่อ Index File

ผู้สร้าง Index

Authorized User ฯลฯ

เนื่องจาก System Catalog และ Data Dictionary จะคล้ายๆ กันจึงมีการเรียกชื่อสลับกันไปมา ในปัจจุบัน RDBMS จะสร้าง System Catalog แต่เพียงอย่างเดียว เพราะสามารถสร้าง Data Dictionary ได้จาก System Catalog

5.6 ความสัมพันธ์ในฐานข้อมูลเชิงสัมพันธ์ (Relationships Within the Relational Database)

Conceptual Relationship ใน Relational Database (RDB) มีอยู่ 3 รูปแบบคือ

1 : 1 หรือ One-To-One

1 : M หรือ One-To-Many

M : N หรือ Many-To-Many

ชื่อ 1 : M และ M : N เป็นความสัมพันธ์ที่สำคัญ เพราะนำมาใช้ในการสร้าง Relational Database ส่วน 1 : 1 นั้นจะมีใช้บ้างก็เฉพาะในบางกรณี เช่น ความสัมพันธ์ระหว่าง อธิการ กับ มหาวิทยาลัย

แต่ 1 : M เป็นรูปแบบความสัมพันธ์ที่สำคัญที่สุดในการนำมาใช้ออกแบบ Relational Database ดังนั้นเราต้องพยายามเปลี่ยนความสัมพันธ์ M : N ให้เป็น 1 : M ให้ได้

ความสัมพันธ์แบบ 1 : 1 นั้นทำให้เราทราบว่า Entity ที่เรากำหนดนั้นไม่ถูกต้อง และยังทำให้เราทราบว่า Entity ที่สัมพันธ์กันแบบ 1 : 1 นั้นแท้จริงแล้วอยู่ใน Table เดียวกันอย่างไรก็ตาม ความสัมพันธ์แบบ 1 : 1 ก็ยังคงมีอยู่ใน Relational Database ได้เช่นกัน เช่น ข้อมูลของ บุคลากร ของโรงพยาบาล

Employee

E-ID	E-NAME	E-ADDRESS	E-PHONE	E-POSITION	E-LICENSE	E-SPECIAL	E-PAGER

ที่จะกลายมาเป็น

EMPLOYEE

E-ID	E-NAME	E-ADDRESS	E-PHONE	E-POSITION

1 : 1 ↔

DOCTOR

E-ID	E-LLICENSE	E-SPACIAL	E-PAGER

<u>ตัวอย่าง</u> เช่น เหมือนกันทุกค่าจึงสัมพันธ์กันแบบ 1: 1

STUDENT1

SID	SNAME	SADDRESS
4105001	SOMCHAI	BANGKOK

STUDENT2

SID	SNAME	SCREDIT	SGPA
4105001	SOMCHAI	125	2.54

เนื่องจากมันมีความสัมพันธ์กับแบบ 1 : 1 จึงควรรวมเป็น Table เดียวกันจะดีกว่า จะได้ดังนี้

STUDENT

SID	SNAME	SADDRESS	SCREDIT	SGPA
4105001	SOMCHAI	BANGKOK	125	2.54

ดังนั้นถ้ามีการเกิด Relation แบบ 1 : 1 แสดงว่ามีการออกแบบที่ไม่ถูกต้องเกิดขึ้นแล้ว แต่ถ้าเป็นกรณีต่อไปนี้

STUDENT

SID	SNAME	SADDRESS	SGPA	SSPECAIL	SAREA
4105001	SOMCHAI	BANGKOK	2.54	นักกีฬา	FOOTBALL
4105002	SOMSEE	~~~~~~~~	~~		

ซึ่งไม่ใช่ว่าทุกคนจะมี

ไม่มี

เราสามารถแยกมันเป็นอีกตารางได้ คือ

SID	SNAME	SADDRESS	SGPA
4105001	SOMCHAI	BANGKOK	2.54
4105002	SOMSEE	~~~~~~~~	~~

SID	SSPECAIL	SAREA
4105001	นักกีฬา	FOOTBALL

ดังนั้นความสัมพันธ์ที่เราสนใจจริงๆ จึงเป็นความสัมพันธ์แบบ 1 : 1 ซึ่งเป็นรูปแบบความสัมพันธ์ที่สำคัญที่สุดในการนำมาใช้ออกแบบ Relational Database ดังนั้น ถ้ามีความสัมพันธ์แบบ M : N เกิดขึ้นเราจะต้องพยายามเปลี่ยนความสัมพันธ์ดังกล่าวให้กลายเป็นความสัมพันธ์แบบ 1 : M ให้ได้

<u>ตัวอย่าง</u>

SALE

SID	SALE	SAREA	CID	CNAME	CADDRESS	CPHONE
S01	A	BANGKOK	1001	XA	XAAA	9999999
S01	A	BANGKOK	1002	XB	XBBB	9999991
S01	A	BANGKOK	1003	XC	XCCC	9999992
S02	B	NONBURI	1004	YA	YAAA	2222222
S02	B	NONBURI	1005	YB	YBBB	2222221

จะเห็นว่าข้อมูลมีลักษณะที่ซ้ำๆ กันหลาย Column ทำให้เปลืองเนื้อที่ในหน่วยความจำเป็นอย่างมาก เราจึงต้องแยกออกเป็น 2 Table

SALE

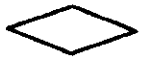
SID	SALE	SAREA
S01	A	BANGKOK
S02	B	NONBURI

CUSTOMER

CID	CNAME	CADDRESS	CPHONE	SID
1001	XA	XAAA	9999999	S01
1002	XB	XBBB	9999991	S01
1003	XC	XCCC	9999992	S01
1004	YA	YAAA	2222222	S02
1005	YB	YBBB	2222221	S02

ในการกำหนดและ สร้างรูปแบบความสัมพันธ์ระหว่าง Entity เราจะใช้ Entity-Relationship Model (E-R Model) เป็นรูปแบบในการอธิบายความสัมพันธ์ดังกล่าว โดยจะใช้ E-R Diagram เป็นเครื่องมือในการอธิบายถึง E-R Model (ผลงานของ CHEN)

แทน Entity



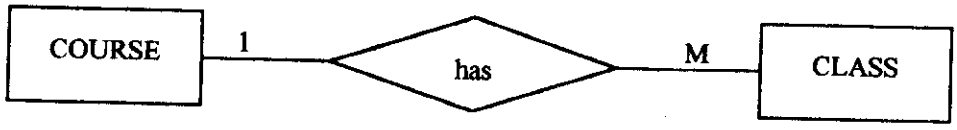
แทน Relationship ที่เกิดขึ้นระหว่าง Entities

1 แทน Entity ที่เป็นด้าน 1 ในความสัมพันธ์

M แทน Entity ที่เป็นด้าน M ในความสัมพันธ์

Advisor Advices Many Students.

Each Student is Adviced by One Advisor.



Course

CRS-CODE	DES-CODE	CRS-DESCRIPTION	CRS-CREDIT
AC211	ACCT	ACCOUNTING 1	3
AC212	ACCT	ACCOUNTING 2	3
CT105	COM	INTRO TO COMPUTER	3

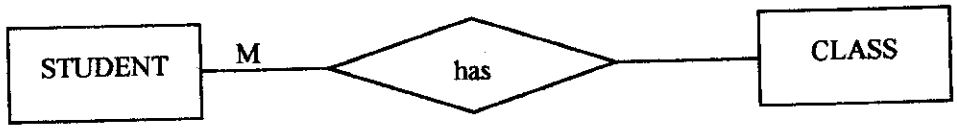
Class

C-CODE	CRS-CODE	C-SECTION	C-TIME	C-ROOM	T-CODE
1001Z	AC211	1	M0900-1130	SBB401	105
1001Z	AC211	2	W1300-1530	VPB701	105

จากตาราง Class เราพบว่า C-CODE สามารถใช้ระบุข้อมูลแต่ละ ROW ได้อย่างไม่ซ้ำกัน ดังนั้น C-CODE ซึ่งทำหน้าที่เป็น Primary Key อย่างไรก็ตาม CRS-CODE และ C-SECTION ร่วมกันใช้ในการระบุแต่ละ Row ได้อย่างไม่ซ้ำกันได้เช่นกัน ดังนั้นทั้ง 2 จึงถือว่าเป็น Composite Key ที่เป็น Candidate Key ด้วย

ความสัมพันธ์แบบ 1 : M สามารถนำมาใช้ได้ง่ายใน Relational Database เพียงแต่ต้องแน่ใจว่า Primary Key ของฝ่าย 1 ต้องไปเป็น Foreign Key ของฝ่าย M ตัวอย่างเช่น CRS-CODE ซึ่งเป็น Primary Key ของตาราง Course ถูกใช้เป็น Foreign Key ในตาราง Class

ส่วนความสัมพันธ์แบบ M : N นั้นกลายมาเป็นสิ่งยุ่งยากใน Relational Database แต่เราก็สามารถแก้ไขปัญหาที่เกิดขึ้นได้โดยการแตกให้เป็นความสัมพันธ์แบบ 1 : M



นั่นคือ Student 1 คน สามารถเรียนได้หลายๆ Class

และ Class 1 ชั้น สามารถที่มีนักเรียน เรียนได้หลายๆ คน

ตัวอย่าง เช่น

สมชาย	ลงเรียน	EN101	10014
		LW103	10018
		TH101	10021
สมหญิง	ลงเรียน	EN101	10014
		LW103	10018
		TH101	10021

เมื่อนำมาสร้างเป็น Table จะได้

Student

S-NUM	S-NAME	C-CODE
41050001	สมชาย	10014
41050001	สมชาย	10018
41050001	สมชาย	10021
41050002	สมหญิง	10014

Class

C-CODE	S-NUM	CRS-CODE	SECTION	TIME	ROOM
10014	41050001	EN101	1		
10014	41050002	EN101	1		
10018	41050001	LW103	2		
10018	41050002	LW103	2		
10021	41050001	TH101	1		
10021	41050002	TH101	1		

เราจะพบว่า ตารางจะมีข้อมูลซ้ำๆ กันเกิดขึ้น เช่น S-NUM ในตาราง Student รวมทั้งในตาราง Class ด้วย

วิธีการแก้ไขปัญหา M : N ก็คือการสร้าง "Composite Entity" หรือ " Bridge Entity" ซึ่งก็คือ Entity ที่ซึ่ง Primary Key ประกอบด้วย Primary Key ของ Entity ที่จะต้องนำมาเชื่อมกัน ดังนั้นเราจะได้ว่า

Student

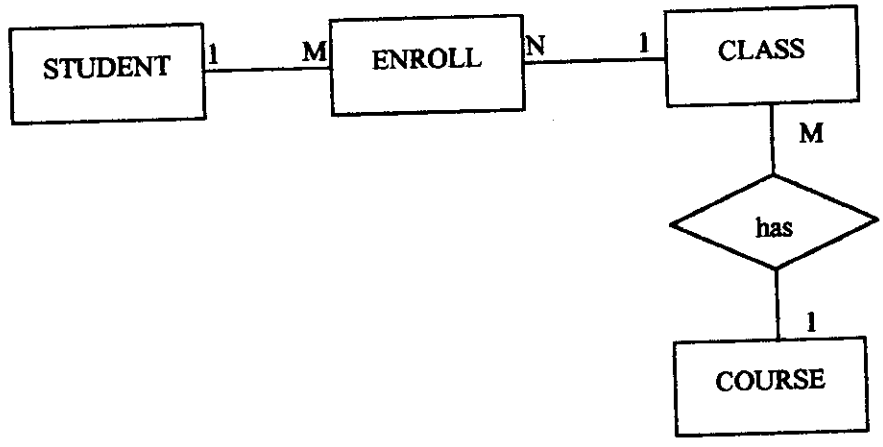
S-NUM	S-NAME
41050001	สมชาย
41050002	สมหญิง

Enroll

C-CODE	S-NUM	GRADE
10014	41050001	B
10014	41050002	A
10018	41050001	B
10018	41050002	B
10021	41050001	C
10021	41050002	B

Class

C-CODE	CRS-CODE	SECTION	TIME	ROOM
10014	EN101	1		
10018	LW103	2		
10021	TH101	1		



EX Invoicing Systems.

Customer

C-CODE	C-NAME	C-PHONE
10001		
10002		
:		

Invoice

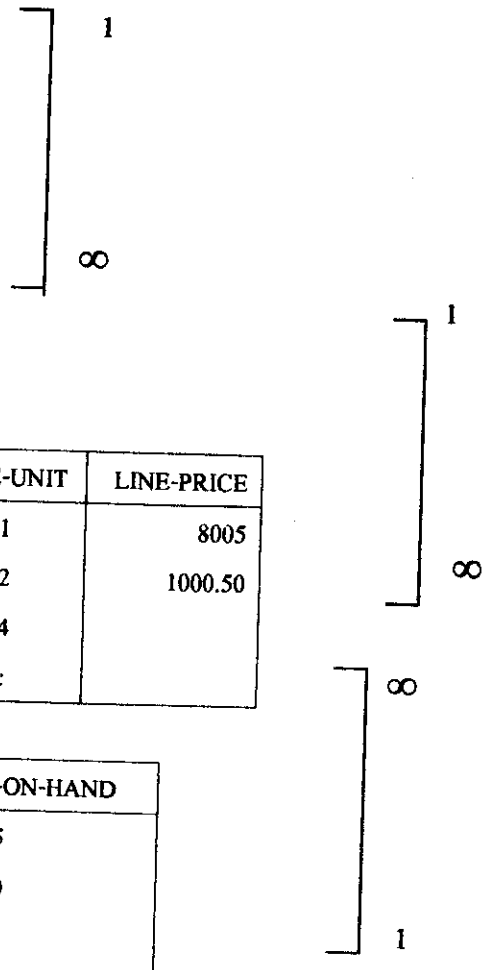
INV-CODE	C-CODE	DATE
5001	10001	10/2/99
5002	10006	10/2/99
:	:	:

Line

INV-CODE	LINE-NUM	P-CODE	LINE-UNIT	LINE-PRICE
5001	1	A001	1	8005
5002	2	A002	2	1000.50
5002	1	A001	4	
:	:	:	:	

Product

P-CODE	P-DESCRIPTION	P-PRICE	P-ON-HAND
A001	BOOK	800.00	25
A002	PEN	500.00	40
:	:		
:	:		



การสร้าง E – R Diagram คือ Iterative Processing คือไม่สามารถสร้างเสร็จภายในครั้งเดียว หลังจากได้ E – R Diagram ก็จะนำ E – R Diagram ที่ได้นี้มาสร้างเป็น Table ซึ่งในที่นี้จะได้ Table ถึง 8 Table ดังนี้

EMPLOYEE (EMP_NUM, EMP_NAME, EMP_ADD, EMP_JOB)

PROFESSOR (EMP_NUM, EMP_NAME, DEP_CODE)

SCHOOL (SCH_CODE, SCH_NAME, EMP_NUM)

DEPARTMENT (DEP_CODE, SCH_CODE, EMP_NUM, DEP_NAME)

STUDENT (S_NUM, S_NAME, S_ADD, DEP_CODE, EMP_NUM)

COURSE (CRS_CODE, CRS_DESCRITP, CRS_CREDIT, DEP_CODE)

CLASS (C_CODE, C_TIME, C_ROOM, CRS_CODE, EMP_NUM)

ENROLL (C_CODE, S_NUM)

ต่อมาก็ใช้คำสั่ง Create Table ต่อ จะได้

```
CREATE TABLE EMPLOYEE (
        EMP_NUM     INTEGER NOT NULL UNIQUE,
        EMP_NAME    VCHAR(20)   NOT NULL,
        EMP_ADD     VCHAR(30)   NOT NULL,
        EMP_JOB     FCHAR(10)   NOT NULL,
        PRIMARY KEY (EMP_NUM) );

CREATE TABLE PROFESSOR (
        EMP_NUM     INTEGER     NOT NULL,   UNIQUE,
        EMP_NAME    VCHAR(20)   NOT NULL,
        DEP_CODE    FCHAR(2)    NOT NULL,
        PRIMARY KEY (EMP_NUM),
        FOREIGN KEY (DEP_CODE) REFERENCES DEPARMENT
                ON DELETE       RESTRICT
                ON UPDATE       CASCADE );
```