**Introduction**

**Data** = fact ใดๆ

**Database** = data เรื่องเดียวกันที่ถูก organize

**Entity** = สิ่งของจับต้องได้ / ไม่ได้ - ข้อมูลที่ entity เหมือนกันจะมีข้อมูลแบบเดียวกัน

**Relationship** = ความสัมพันธ์ระหว่าง 2 entity

**Database machine** = Backend processor = hardware ที่เก็บ data - storage อาจเป็น tape, disks, optical disks

**DBMS** = Database management system = software เพื่อเก็บและจัดการ database

* ลักษณะของ DBMS
  + **ACID** property support
    - **Atomicity** - แต่ละ transaction ถ้าไม่สำเร็จจะถอยหลังได้
    - **Consistency** - ข้อมูลล่าสุดเหมือนกัน
    - **Isolation** – concurrent transaction ไม่ effect each other
    - **Durability** – ไฟดับก็ไม่หาย
  + **Data security**
  + **Data consistency**
  + **Less redundancy** - ไม่ซ้ำซ้อน

**CAP theorem** - ไม่มีทางมีทั้ง 3 อย่าง

* **Consistency**
* **Availability** - ทุก user เห็น same data
* **Partition-tolerance** - แม้มี network partition ก็ทนได้

**3 Schema architecture**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ใคร work ใน level นี้ |
| 1. Internal level | Physical schema | อธิบาย structure ของ data เลยว่าเก็บที่ไหน (Physical data model) | Admin database |
| 1. Conceptual level | Logical schema | อธิบาย design (ลักษณะ ความสัมพันธ์) | Programmer |
| 1. External level | View schema | แต่ละ user เห็น view ไม่เหมือนกัน | End user |

**Logical data independent** = เปลี่ยนระดับน้อย ไม่กระทบระดับมาก

* **Physical** = เปลี่ยน physical ไม่กระทบ logical
* **Logical** = เปลี่ยน logical ไม่กระทบ view

\*metadata = อธิบาย data ว่าหมายถึงอะไร\*

**SQL** = ภาษาที่จัดการ data แบบ relational คือมีรูปแบบความสัมพันธ์ชัดเจน

|  |  |  |  |
| --- | --- | --- | --- |
| **DDL** | **Data definition** language | สร้างข้อมูล | create, drop, alter, truncate, comment |
| **DML** | **Data manipulation** language | แก้ไขข้อมูล | select, update, insert, delete, merge, call, lack table, explain plan |
| **DCL** | **Data control** language | ควบคุมการ access | grant, revoke |
| **TCL** | **Transaction control** language | ควบคุม transaction | commit, rollback, savepoint, set transaction |

**NoSQL** = Not only SQL - เก็บข้อมูลที่ไม่ได้มีรูปแบบความสัมพันธ์ที่ชัดเจน

* Document database : mongoDB
* Wide-column DBMS - กระจายข้าม server/node ได้ : assandra
* Key-value database – ข้อมูลเล็กๆ IoT : redis
* Graph database : neo4j
* Time series DBMS : inflexdb

|  |  |  |
| --- | --- | --- |
|  | RDBMS | NoSQL |
| ชนิด | Relational | Non-relational |
| รูปแบบชุดข้อมูล | เป็นโครงสร้าง เก็บใน table | ไม่เป็นโครงสร้าง เก็บใน json/text |
| การ scale | Vertical = เพิ่ม spec server | Horizontal = เพิ่มจำนวน server |
| Schema | เปลี่ยนไม่ได้ | เปลี่ยนได้ ยืดหยุ่น |
| ตัวอย่าง | Oracle, MySQL | MongoDB, Cassandra |
| ใช้เมื่อ | * Workload คงที่ ต้องการพื้นที่ปายกลางถึงมาก * รูปแบบข้อมูลมีโครงสร้างชัดเจน * ข้อมูลมี relation เชื่อมกัน * ข้อมูลเป็นไปตาม condition * ข้อมูล complex ต้องการ report * ให้ user ควบคุม * Deploy ใส่ hardware ขนาดใหญ่ / ของตัวเอง | * Workload มหาศาล * ข้อมูลแบบ dynamic เปลี่ยนแปลงบ่อย * ข้อมูลไม่จำเป็นต้องมี relation * เน้นเขียนไว ไม่เน้น condition * ข้อมูล simple * ต้องการให้ข้อมูลกระจายให้เข้าถึงได้ทุกส่วน * Deploy บน cloud |

**NewSQL** - ได้ทั้ง relational และ non-relational

**Entity-Relationship Model**

**Database schema** = database description

Database application

* ลด redundancy
* มี consistency

**Entity** = specific object

**Entity type** = entity ที่มี attribute เหมือนกัน

**Entity set** = entity ใน database ณ ขณะหนึ่ง

**Weak entity type** = entity ที่ไม่มี key ที่จะ unique identify

* ต้องมี identify relationship กับ identifying entity type
* identify โดยใช้ partial key กับ related entity’s key

**Attribute** = property ของ entity - แต่ละ attribute มี value set (data type)

* **Key attribute** = attribute ที่มี unique value - ใช้ในการ identify entity
* **Simple** **attribute** = เป็น atomic value – value แยกต่อไม่ได้

**Composite** **attribute** = มีหลาย field : ชื่อ(ชื่อจริง, นามสกุล)

* **Single-valued** **attribute** = มีอันเดียว : อายุ

**Multi-valued** **attribute** = มีหลายอัน : {เบอร์โทรศัพท์}

* **Stored** **attribute** = เก็บค่าจริงๆ

**Derived** **attribute** = คำนวณจากค่าอื่น

**Null value** = ไม่มีค่า / ไม่ทราบค่า

**Relationship** = การ relate 2 entity ขึ้นไป

**Relationship type** = การ relate entity type

* **Degree** = จำนวน entity ที่ relate
  + **Binary**
  + **Ternary**
  + **N-ary** - ไม่เหมือน n binary relationship เพราะ n-ary บังคับว่าต้องมีครบ n entity
* **Attribute**
* **Ratio constraint**
  + **Maximum cardinality**
    - 1:1
    - 1:N
    - N:N
  + **Minimum cardinality** = Participation constraint = Existence dependency constraint
    - Zero = Optional participation = Not existence-dependent - ไม่มีก็ได้
    - One / more = Mandatory = Existence-dependent – ต้องมี
* **Recursive** relationship type = relation ระหว่าง 1 entity type (แต่คนละ object)
* **Relationship structural constraint**
  + Cardinality ratio - เขียน 1 / N ไว้ที่เส้น

Participation constraint – เขียนเส้นเดี่ยว Partial / เส้นคู่ Total

* + (min, max) notation – บอกว่าเส้นนั้นมีได้เท่าไร

เส้นคู่ 🡪 min = 1

เส้นเดี่ยว 🡪 min = 0

เลขที่เส้น = max ของอีกข้าง

**Superclass/Subclass relationship**

* subclass เป็น subset ของ superclass
* **IS-A relationship** = ถ้า entity อยู่ใน subclass 🡪 entity อยู่ใน superclass ด้วย
* subclass inherit ทุก attribute และ relationship ของ superclass
* **Specific attribute** = attribute ของ subclass

**Specialization & Generalization**

* **Specialization** = การสร้าง subclass จาก superclass - แบ่งจากลักษณะของ entity ใน superclass
* **Generalization** = การสร้าง superclass จาก subclass - ดูที่ common feature ของทุก subclass

: {CAR, TRUCK} เป็น specialization ของ VEHICLE <--> VEHICLE เป็น generalization ของ CAR และ TRUCK

* Defining constraint
  + **Attribute-defined** = แบ่ง **Condition-defined subclass** = **Predicate-defined subclass** ด้วย **Defining attribute** : job type
    - เขียน defining attribute ไว้ที่เส้นฝั่ง superclass
    - เขียน ‘value’ ไว้ที่เส้นฝั่ง subclass
  + **User-defined** : เงื่อนไข honor degree
* Disjointness constraint
  + **Disjoint** = ห้ามอยู่หลาย subclass
  + **Overlap** = อยู่หลาย subclass ได้
* Completeness constraint
  + **Total** = ทุก entity ต้องอยู่ใน subclass
  + **Partial** = entity ไม่ต้องอยู่ใน subclass ไหนเลยก็ได้

Generalization ส่วนใหญ่เป็น total เพราะ superclass ถูก derive มาจาก subclass

* Hierarchy & Lattice - subclass มี subclass ย่อย – subclass inherit จาก predecessor superclass ของ direct superclass ด้วย
  + Hierarchy = มี single inheritance = แต่ละ subclass มี 1 superclass
  + Lattice = มี multiple inheritance = shared subclass inherit จากหลาย superclass
* **Shared subclass** = subclass ที่มีมากกว่า 1 superclass
  + เป็นหลาย superclass/subclass relationship ที่แต่ละ relationship มี 1 superclass
  + เป็น subset ของ superclass มา intersect กัน
  + member ของ shared subclass อยู่ในทุก superclass

**Category** = Union type = member ของ subclass เป็น subset ของ superclass ที่มา union กัน

* + เป็น 1 superclass/subclass relationship ที่มี > 1 superclass
  + เป็น subset ของ superclass มา union กัน
  + member ของ category อยู่ในอย่างน้อย 1 superclass

**Crow’s foot** notation

* Cardinality - อยู่ติด entity - บอกค่า max
  + Single
  + Multiple
* Modality - อยู่ด้านใน – บอกค่า min
  + Mandatory
  + Optional

|  |  |  |
| --- | --- | --- |
|  | Entity |  |
|  | Weak entity | Entity ที่ไม่มี key |
|  | Relationship |  |
|  | Identifying relationship | Relationship กับ weak entity |
|  | Attribute |  |
|  | Key attribute |  |
|  | Partial key attribute | เป็นของ weak entity |
|  | Multi-valued attribute |  |
|  | Composite attribute |  |
|  | Derived attributed |  |
| E1  R  E2 | Total participation | E2 ต้องมี E1 |
| E1  R  E2  1  N | Cardinality ratio | 1 E1 มีได้หลาย E2 |
| R  E  (min, max) | Structural constraint | ต้องมี E อย่างน้อย min อย่างมาก max |
| Subclass  Superclass | Superclass/Subclass relationship |  |
| Subclass  Superclass  Subclass  Subclass  Superclass  Subclass | Overlap specialization  Disjoint specialization |  |
| Superclass | Total specialization |  |
| Subclass  Superclass  Superclass  Superclass  Subclass  Superclass | Shared subclass  Category |  |

**Relational model**

Relation definition

* Informal
  + Relation = table of value / set of row
  + แต่ละ row = = entity / relationship - มี key
* Formal
  + Relation = set of tuple : <16753, “John”, 27>
  + attribute = column ใน table
  + **dom(Ai)** = domain ของ attribute i
  + R(A1, A2, …, An) = relational schema R of degree n
  + r(R) = relation ของ relation schema – specific value ของ R - เป็น subset ของผลคูณคาร์ทีเซียนของ dom(Ai)
    - r = **{** t1, t2, …, tm **}** 🡪 set ของข้อมูลขณะนั้น – แต่ละเวลามีไม่เท่ากัน
    - t = **<** v1, v2, …, vn **>** 🡪 list ของ value

n-tuple โดย vi คือ element ใน dom(Ai) / null value

* + - t[Ai] = vi
    - t[Au, Av, …, Aw] = vu, vv, …, vw
  + R = **intension** ของ relation

r = **extension** ของ relation

* + ลักษณะของ relation
    - Tuple ใน r(R) อาจไม่เรียงลำดับ 🡪 r เป็นเซ็ต
    - Attribute ใน relation schema & ค่าใน tuple ต้องเรียงลำดับ 🡪 t, R(A) เป็น list

|  |  |
| --- | --- |
| Informal | Formal |
| Table | Relation |
| Column | Attribute / Domain |
| Row | Tuple |
| Value in column | Domain |
| Table definition | Schema of relation |
| Populated table | Extension |

Relational database constraint

* Inherent model-based constraint - ห้ามมี tuple ซ้ำกันเป๊ะๆ 🡪 สมาชิกในเซ็ตห้ามซ้ำกัน
  + **Domain constraint**
    - Attribute ใน tuple เป็น atomic
    - Value มีค่าตาม dom(A) – data type
  + **Key constraint**
    - SK = **Superkey** = subset ของ attribute ของ R ที่ unique - มี attribute ซ้ำได้ 🡪 t1[SK] != t2[SK]

**Key** = minimal superkey

**Candidate key** = minimal superkey ที่จะใช้เป็น primary key ได้

**Primary key** = candidate key ที่ถูกเลือกมาเป็น key

* + - * ห้ามมี tuple ที่มี key ซ้ำ
      * เป็น minimal superkey - ถ้าเอา attribute สักตัวออกจะไม่ใช่ superkey แล้ว
    - ทุก relation ต้องมีอย่างน้อย 1 default superkey คือ set of all attribute
  + Entity integrity constraint
    - Relational database schema

S = **{** R1, R2, …, Rn **}** = เซ็ตของ relational schema ใน same database

* + - * S = name ของ database
    - Primary key ของ relation ห้ามเป็น null 🡪 t[PK] != null
  + Referential integrity constraint
    - Foreign key = attribute ของ referencing relation ที่อ้างถึง primary key ของ referenced relation
      * เป็น null ได้
      * จะเป็น / ไม่เป็น primary key ของ referencing relation ก็ได้

t1[FK] = t2[PK]

foreign key primary key

* Schema-based constraint - ใช้ภาษา DDL
* Application-based constraint – constraint ที่ต้องเอาค่าไปทำอะไรสักอย่างก่อน

Update operation on relation

|  |  |
| --- | --- |
| Operation | Constraint ที่อาจ violate |
| Insert | * Domain constraint – value ต้องอยู่ใน domain * Key constraint – key ต้องไม่ซ้ำ * Entity constraint – primary key ห้ามเป็น null * Referential constraint - ห้ามใส่ foreign ที่ refer ถึงค่าที่ไม่ใช่ primary key |
| Delete | * Referential constraint – ถ้าจะลบตัวที่ถูก foreign key refer 🡪 reject การลบ / ลบทั้งหมด / แก้ค่า / เป็น null ได้ |
| Update / modify | * Domain constraint * Key constraint * Entity constraint * Referential constraint |

ER-to-Relational mapping = การเปลี่ยน ER model เป็น relation

1. Regular entity type

* สร้างเป็น table ของ entity นั้น
* Composite attribute แยกเป็น attribute ย่อยเลย

1. Weak entity type

* สร้างเป็น table ที่มี primary key ของ related entity ด้วย 🡪 ขีดเส้นใต้ทั้ง primary key และ related key

1. Binary 1:1 relation type
   1. Foreign key approach
   * เลือกมา 1 relation แล้วเอา primary key ของอีกตัวมาใส่เป็น foreign key
   * ถ้ามีตัวที่เป็น total participation ให้เลือกตัวนั้น
   1. Merged relation option
   * เอามา merge เป็น table เดียวกัน
   * เหมาะกับ total participation ทั้ง 2 ฝั่ง
   1. Cross-reference = Relationship relation option
   * สร้าง table ใหม่ที่เก็บ primary key ของทั้ง 2 ฝั่ง
   * ใช้ตัวไหนเป็น key ก็ได้เพราะ unique ทั้งคู่
2. Binary 1:N relation type

* ใช้ foreign key approach - ให้ฝั่ง N เก็บ primary key ของฝั่ง 1

1. Binary M:N relation type

* สร้าง table ใหม่ที่เก็บ primary ของทั้งคู่
* ใช้ primary key ทั้ง 2 อันเป็น key

1. Multivalued attribute

* สร้าง table ใหม่ เก็บ primary key ของ entity คู่กับแต่ละ value

1. N-ary relationship type

* สร้าง table ใหม่ เก็บ primary key N ตัว
* ใช้ทุกอันเป็น key

1. Specialization and Generalization

ให้ subclass = S1, S2, …, Sm

และ superclass = C ที่มี attribute เป็น {k, a1, a2, …, an}

* 1. Multiple relation
     1. Multiple relation – Superclass and subclass
  + สร้างตาราง superclass กับแต่ละ subclass
  + ตาราง subclass เก็บ primary key ของ superclass ด้วย
  + ใช้ได้กับทุกแบบ
    1. Multiple relation – Subclass relation only
  + ไม่ต้องสร้างตาราง superclass - เก็บทุกอันไว้ในตาราง subclass เลย
  + ใช้กับ total participation เท่านั้น - บังคับว่าต้องอยู่ใน subclass
  1. Single relation
     1. Single relation with one type attribute
  + สร้างตาราง superclass เดียวที่มี attribute type และทุก attribute ของแต่ละ subclass 🡪 null เยอะ
  + ใช้ได้กับ disjoint, total, partial (NULL) แต่ใช้กับ overlap ไม่ได้
    1. Single relation with multiple type attribute
  + สร้างตาราง superclass เดียวที่มี attribute flag เก็บค่า True/False โดยมีจำนวนเท่ากับจำนวน subclass และทุก attribute ของแต่ละ subclass 🡪 null เยอะ
  + ใช้กับ overlap ได้
* Shared subclass
  + ใช้แบบ Single relation แบบไหนก็ได้ - สร้างตาราง subclass ตารางเดียวที่มี flag ว่าเป็น superclass ไหน

1. Union type / Category

* Superclass มี key ไม่เหมือนกัน
* สร้างตาราง subclass ที่มี primary key เป็น surrogate key
* ตารางแต่ละ superclass มี attribute เก็บ surrogate key

Schema diagram = ตารางที่มีเส้นชี้จาก foreign key ไป primary key