```
SELECT   DISTINCT dName
FROM     Department
WHERE    mgrSSN = ANY
         (SELECT   ssn
          FROM     Employee
          WHERE    name LIKE 'Jennifer');
```

題目是：列舉所有主管為Jennifer的部門
=ANY的意思是子查詢的結果中有符合一個就可以

```
SELECT     *
FROM       Employee
WHERE      salary > ALL(SELECT salary
             FROM Employee
             WHERE dNum = 5);
```

題目是：列舉所有薪水比第五部門高的employee
=ALL的意思是要全部符合

```
SELECT      dNum, COUNT(*)
FROM        Employee E1
WHERE       salary > 20000
GROUP BY  dNum
HAVING 2 < (SELECT COUNT(*)
             FROM  Employee E2
             WHERE E1.dNum = E2.dNum)
```

Advanced Single Value Subquery Example
題目是：在至少三人的部門中,列舉出薪水大於兩萬的人

```
SELECT *
FROM Movie M1
WHERE NOT EXISTS
         (SELECT *
          FROM Movie M2
          WHERE M1.movieID <> M2.movieID AND M1.year = M2.year)
```

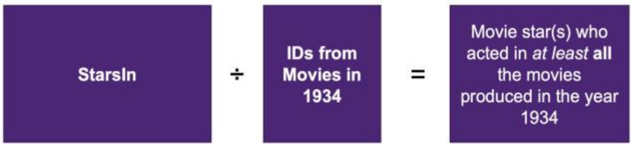
EXIST跟NOT EXIST的用法：EXIST就是後面至少有一個返回值就為T否則F.NOT EXIST反之
這題題目是：列舉出某個電影他是當年度唯一生產的電影
如果你打NOT拿掉 就變成列舉出某個電影他不是當年度唯一生產的電影

A practical example of a division problem may be as follows:

"Find the movie star(s) who acted in *at least* **all** the movies produced in the year 1934."

關聯除法的例子可以簡單理解為不能除以除術敘述的就會被淘汰

How can this problem be solved using division?


StarsIn
÷
IDs from Movies in 1934
=
Movie star(s) who acted in at least all the movies produced in the year 1934

Employee (ID, level, salary), F{ID→ level, level→ salary, ID→ name}

1. $ID^+ = \{ID\}$ X的封閉性的例子, 知道ID之後就能知道的所有其他屬性都稱為X的closure.
2. $ID^+ = \{ID, level\}$ using ID→ level
3. $ID^+ = \{ID, level, salary\}$ using level→ salary
4. $ID^+ = \{ID, name, level, salary\}$ using ID→ name

<u>4個 Design Guidelines</u>: 1.清晰定義屬性的語義,減少元組中的冗余值 2. 避免插入、刪除和修改異常 3. 減少元組中的空值 4. 確保關聯之間的屬性能夠進行關聯操作，並且保證不會生成虛假的元組(也就是 loseless Join，你把一個 relation 拆成兩個他照理來說應該要回復原狀不多不少，否則就是 lossy join). <u>Functional Dependency:</u>即：t1[X] = t2[X]，則 t1[Y] = t2[Y]例如當兩個員工為同一階級,則其薪水必相同.

<u>＊FD(依賴函數)中,</u> primarykey 是指能夠一個屬性包辦所有屬性的屬性, superkey 是指能夠唯一辨別元組的集合,例：R(A,B,C,D): B->C, C->B, D->ABC 中，D 是 PK 而 BD, CD 是(CANDIDATE)key 就是一個集合但也可以包辦所有屬性. Superkey 就是 key 的父集合，你也可以把所有 attribute 都放到 superkey 裡面說他是 superkey

<u>4個 Normalization:</u> 1NF, 2NF, 3NF, BCNF.

Examples of Non-1NF Relations

1NF就是要求每個屬性都只能有原子值就是不能再分割,item就不是原子值

customerName	orderNum	items
Tom Jones	123	Hat
Sri Gupta	876	Glass, Pencil

customerName	orderNum	item
Tom Jones	123	Hat
Sri Gupta	876 這個就是原子值1NF的結果	Glass
Sri Gupta	876	Pencil

Example Normalized to 2NF

Example

2NF就是把所有非主要屬性(沒畫底線的)完全依賴於畫底線的, 這個例子中, F提及的是{state, value}依賴{houseNum, street, postcode}, 但事實上後面那句"postcode->state"表明了state只部分依賴{houseNum, street, postcode}, 因為只需要postcode就可以得知state, 不需要houseNum, street

- Address [houseNum, street, postcode, state, value]
- F = {{houseNum, street, postcode} → {state, value},
  postcode → state}

所以我們多寫一個relation, Postcode[postcode, state]
讓state完全依賴於Postcode的PK, postcode
原來的Address就可以把state刪除了
下面再次提及postcode->state就好
但記得是用新的relation.

Normalizing Address
- Address [<u>houseNum, street, postcode</u>, value]
- Postcodes [<u>postcode</u>, state]
  Address.postcode → Postcodes.postcode

Example Normalized to 3NF

Example

3NF就是要消除非主要屬性對非主要屬要的依賴, 例子中, salary依賴於level, 但level並不是主要屬性, 所以我們另起一個relation, 將level跟salary放置在裡面並令level為這個relation的主鍵而且salary依賴於他, 那麼, 就完成了3NF

- Employee [<u>ID</u>, name, level, salary]
  - level → salary (transitive dependency)

注意在這個例子中如果ID->salary那他就已經是3NF了因為ID是主要屬性, 另一種常見表示方法為:ID->level, level->salary 那他就不是3NF因為ID->level, level->salary中的level不是主要屬性

Normalizing Employee
- StaffAppointment [<u>ID</u>, name, level]
- StaffIncome [<u>level</u>, salary]
  StaffAppointment.level → StaffIncome.level

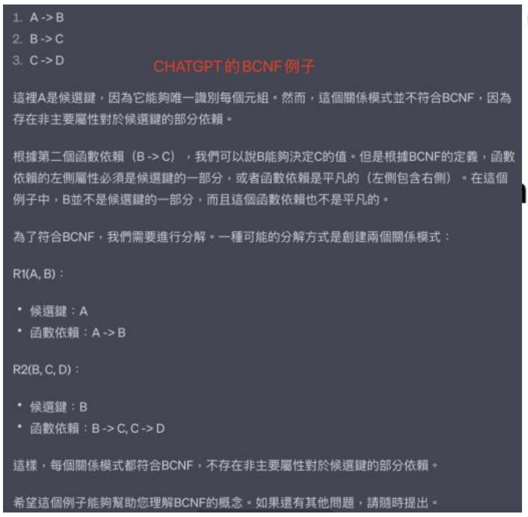
2NF: X is not a proper subset of a candidate key for R, or A is a prime attribute.

3NF: X is a superkey for R, or A is a prime attribute.

BCNF:X is a superkey for R.

逐個檢查依賴函數若都為 True,則視為符合.

Informally: Whenever a set of attributes of R determine another attribute, it should determine all the attributes of R. 這個是BCNF的定義, 亦即, 依賴函數的左側必須要能夠完全唯一定義所有屬性



1. A -> B
2. B -> C
3. C -> D

CHATGPT的BCNF例子

這裡A是候選鍵，因為它能夠唯一識別每個元組。然而，這個關係模式並不符合BCNF，因為存在非主要屬性對於候選鍵的部分依賴。

根據第二個函數依賴（B -> C），我們可以說B能夠決定C的值。但是根據BCNF的定義，函數依賴的左側屬性必須是候選鍵的一部分，或者函數依賴是平凡的（左側包含右側）。在這個例子中，B並不是候選鍵的一部分，而且這個函數依賴也不是平凡的。

為了符合BCNF，我們需要進行分解。一種可能的分解方式是創建兩個關係模式：

R1(A, B)：

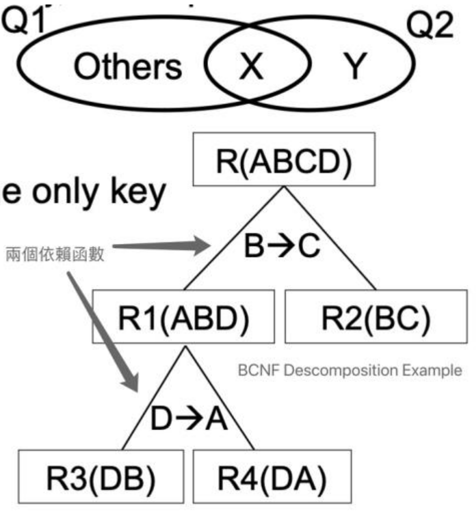
- 候選鍵：A
- 函數依賴：A -> B

R2(B, C, D)：

- 候選鍵：B
- 函數依賴：B -> C, C -> D

這樣，每個關係模式都符合BCNF，不存在非主要屬性對於候選鍵的部分依賴。

希望這個例子能夠幫助您理解BCNF的概念。如果還有其他問題，請隨時提出。



BCNF Descomposition Example

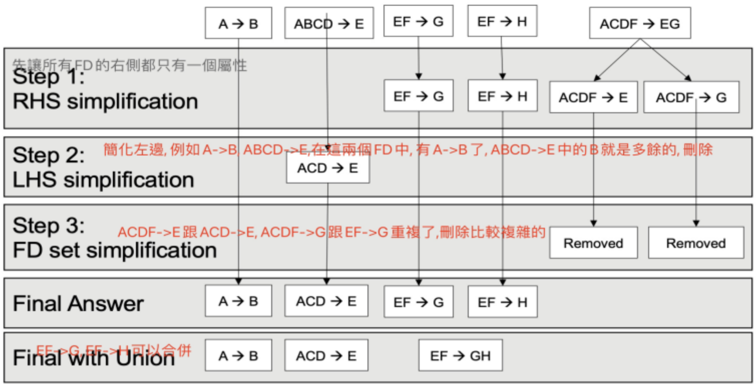
Consider relation R(ABCDE) with functional dependencies AB → C, BC → D, CD → E, DE → A, and AE → B. Project these FD's onto the relation S(ABCD).

Which of the following hold in S?

A. A→B
B. AB→E
C. AE→B
[✓ D.] BCD→A

A.沒有辦法由A定義B
B.S之中沒有E,且從依賴函數方面思考也得不到AB->E
C.S之中沒有E,當然不可能用E去定義別人
D.雖然S之中沒有E,但他可以被儲存並引用

Minimal Cover Example Visualised



3NF Example

3NF Synthesis Example

R(CSJDPQV), F = {SD→P, JP→C, J→S}

- Key: JDQV
- F is already minimal

先確認key然後再把依賴函數縮到最簡

Create tables based on F = {SD→P, JP→C, J→S}

- R1(SDP), R2(JPC), R3(JS)

像這樣子用依賴函數創立所有relation

if (any candidate key is missing from the relations)
add a relation with all prime attributes

- R4(JDQV)

如果key沒有被包含到, 就再多創立一個只有key的relation (prime attribute)

Remove redundant relations

如果有redundant, 就自動刪除
何謂redundant?就是現有relation的真子集
例如R5(JP), R6(J)那他們就是redundant
因為他們已經有一個更大的集合提及他們了

- Nothing is redundant.
- Final answer is R1(SDP), R2(JPC), R3(JS), R4(JDQV)

<u>數據安全的 3 個威脅：1.</u>Loss of confidentially(機密性的損失): 未經授權的揭露機密性資訊 2. Loss of integrity(完整性的缺失):不正當的修改數據 3. Loss of availability(可用性的缺失):合法用戶但不被允許放問數據. <u>隱私及安全性</u>:安全性: 控制什麼樣的數據可以被訪問, 剩下的全是隱私性, 安全性是隱私性的地基. <u>4 個 Database Control Measures:</u> 1.Access control: Provide access only to users who have the right access authority. 2. Inference control: Must ensure information about individuals cannot be accessed (applies to statistical databases)3. Flow control: Prevents information from flowing to unauthorized users 4. Data encryption(加密): Used to protect sensitive transmitted data. <u>Basic DBMS Authentication Security Functions(DBMS 的基本安全認證功能):</u>user account:使用時需要使用者登入帳號密碼. Login session: 將使用者的操作記錄在日誌中, 不只是為了安全性, 也可以用於分析跟備份. DB audit(數據審計): 檢查日誌以查看在某個時間段內進行的所有存取和操作. <u>Access Control Mechanisms(訪問的控制機制):</u> 1.Discretionary Access Control (DAC):使用者可以自由無限制訪問使用數據. 2. Mandatory Access Control (MAC):將使用者以及數據分級,什麼樣的使用者可以訪問什麼樣級別的數據.3. Role-based Access Control (RBAC):將使用者分級, 什麼樣級別的使用者被分配到 MAC 或是 DAC.

Discretionary Access Control

Based on Granting and Revoking privileges

```
CREATE USER <username> IDENTIFIED BY "<password>";

GRANT <privilege> TO <user>;
```

```
GRANT privilegeName
ON      objectName
TO      {userName |PUBLIC |roleName}
[WITH GRANT OPTION];
```

這是授予一個使用者權限的代碼, REVOKE代表撤銷

```
REVOKE privilegeName
ON       objectName
FROM    {userName |PUBLIC |roleName}
```

Example: CREATE SCHEMA

```
CREATE SCHEMA AUTHORIZATION oe
  CREATE TABLE Product
    (color VARCHAR2(10) PRIMARY KEY, quantity NUMBER)

  CREATE VIEW RedProduct AS
    SELECT color, quantity FROM Product WHERE color = 'RED'

  GRANT select ON RedProduct TO hr;
```

The above statement

- creates a schema named *oe*
- creates the table *Product;*
- creates the view *RedProduct;*
- grants the SELECT object privilege on *RedProduct* to another user *hr.* ← 作用在這裡

Suppose that A1 wants to give A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to retrieve only the name, dob, and address attributes and only for the tuples with dNum = 5.

```
CREATE VIEW A3EMPLOYEE AS
  SELECT name, dob, address
  FROM EMPLOYEE
  WHERE dNum = 5
```

```
GRANT SELECT ON A3EMPLOYEE TO A3
```

<u>MAC:</u>共有五個層級:top secret TS 最高機密, secret S 秘密, Confidential C 機密, Unclassified U 非機密. 2 個 <u>The Bell-LaPadula Model property:</u> 1. Simple security property(NRU):級別低的看不到級別高的數據 2. Star property(NWD): 級別高的不可以創立級別低的數據,避免有洩漏之虞.

Multilevel Security Example

TC: Tuple Classification
(Highest classified attribute of the tupl

Example: A multilevel relation to illustrate multilevel security

1. The original EMPLOYEE tuples
2. Appearance of EMPLOYEE after filtering for classification C users
3. Appearance of EMPLOYEE after filtering for classification U users
4. Polyinstantiation of the Smith tuple (a database to maintain multiple records with the same key, in order to prevent inference attacks)

(a) EMPLOYEE (original) 原始資料,也可以說是擁有TC以上級別的使用者能看到的數據

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

(b) EMPLOYEE for classification C C級別使用者能看到的數據

Name	Salary	JobPerformance	TC
Smith U	40000 C	NULL C	C
Brown C	NULL C	Good C	C

(c) EMPLOYEE for classification U U級別使用者能看到的數據

Name	Salary	JobPerformance	TC
Smith U	NULL U	NULL U	U

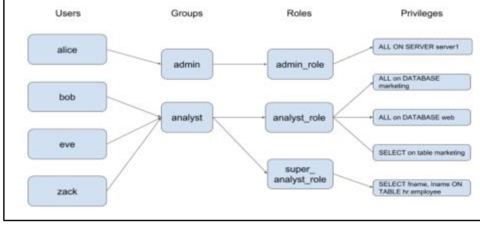
Polyinstantiation:混淆視聽, 魚目混珠

(d) EMPLOYEE Polyinstantiated 混淆攻擊者視聽

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Smith U	40000 C	Excellent C	C
Brown C	80000 S	Good C	S

Inference Attacks:回養有人要把 Smith 的數據從 NULL 修改, 但系統不應該拒絕否則他能推斷其事實上為非空.＊以上操作有 GRANT OPTION 的代表可以賦予他人相同等級的權限.當被賦予者被收回權限時,其賦予的人也會被收回權限.

An Example of Role-Based Access Control



Create/drop roles

```
CREATE ROLE manager;
DROP ROLE manager;
```

Grant/revoke privileges

```
GRANT privilegeName
ON       objectName
TO       {userName | PUBLIC |roleName}
[WITH GRANT OPTION];

REVOKE privilegeName
ON        objectName
FROM    {userName | PUBLIC |roleName}
```

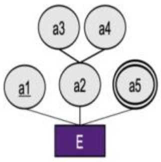
<u>其他的 Control Measures：</u>1. Inference Control(推斷控制): 防止推斷個人信息,在元組數量上設置最小閾值,亦即:意味著在每個子集中，至少包含 k 個相似的元組，以防止對單個個體的推斷。,禁止引用相同元組群體的查詢序列, 引入輕微的不準確性.2. Flow Control(流程控制): 禁止從機密到非機密的信息流動,例：所得稅計算服務被允許保留客戶的地址資料，但不允許保留其收入和扣除項目的資料。3. Data Encryption(數據加密):Performed by applying an encryption algorithm to data using a prespecified encryption key. Resulting data must be decrypted using a decryption key to recover original data(簡單講就是對文件加密)

<u>SQL 注入的風險(即常見的攻擊方法):</u> 1.數據庫指紋識別：攻擊者確認此數據庫的程式語言。2.拒絕服務攻擊：攻擊者拒絕對合法用戶提供服務。3.繞過身份驗證：攻擊者未經有效授權即可訪問數據庫。4. 確定可注入的參數：攻擊者確定後端結構的信息。5.執行遠程命令：攻擊者遠程執行有害命令。6.提升特權：攻擊者提升其訪問級別。

<u>保護技術:</u> 1. Bind variables (using parameterized statements) Protects against injection attacks, Improves performance. 2. Filtering input (input validation)Remove escape characters(轉譯字符) from input strings, Escape characters can be used to inject manipulation attacks.3. Function security(函數的使用)Standard and custom functions should be restricted(應限制自定義函數的使用以及新添標準)

database 的三個主要部件: database design, data model, physical storage.三個功能:data integrity, query processing, security management. Dbms 的主要部件: Database Administrators Database Designers End Users Application Programmers.三個 schema architecture: external 顯示 conceptual 來自 internal level. Data Independence: logical data independence: 可以改變 conceptual 而不動用 external. physical data independence:可以改變 physical 而不動用 conceptual. Composite key:這個 key attribute 是由許多 simple attribute 組成這些 simple attribute 可以重複但結合起來就會是一個 unique key.(Course is assigned a number by dep., 所以他的 primary key 就是 composite key, simple attribute number 和 dep.) 依賴 relationship: 兩條線的代表他至少要綁定一個一條線的. Partial key: 相當於 weak entity 的 primary key, 但必須要與 owner 的 primary key 搭配才能唯一識別,其用虛線底線表示.EER 中,父 entity 是子 entity 的 generalization, 子 entity 是父 entity 的 specialization, 而且 o 代表 overlapping 代表父 entity 會有這兩個子 entity, d 代表 disjoint, 代表父 entity 只會是其中一種 entity(用雙橫線連接父 entity 跟”d”). Relation 的定義:每個 relation 都是表格但不是每個表格都是 relation(relation 必須要一格一個 attribute,表格不用) Relation 中,直排叫 attribute, 橫排叫 tuple. Relation 中, x 個 attribute 記為 a relation scheme of degree x.同時亦有 x-tuple 的用法. * Integrity Constraints(完整性約束)共五個:1.domain constraint:資料欄位儲存了非指定的資料型態,例:ID 是 4 個數字結果你打 abcd.2.Key constraint:只要能唯一識別的都叫做 key 稱為 candidate key,其中有一個必不能為空的稱為 primary key. Key constraint 就是所有這些唯一識別的 key 中,有重複的數據出現.3. Entity Integrity Constraints: primary key 永遠不能為空, Entity Integrity Constraints 就是 primary key 中出現空值 4. Referential Integrity Constraints: foreign key 欄位中出現任何非法的數值(最常見就是 foreign relation 中沒有儲存應該要被引用的數值記住可以為 Null 亂引用不存在的才會報錯).relation 引用 foreign key 的寫法: 假設 Student [sid, name] Course [cid, department, manager] Enrollment [sid, cid, department, grade],則: Enrollment.sid references Student.sid. Enrollment.{cid, department} references Course.{cid, department}.5. Semantic integrity constraints:資料儲存與 UoD 矛盾,例:員工薪水永不高於主管然後你儲存了高於主管的薪水. * ER to Relation mapping 七個步驟:

ER Diagram



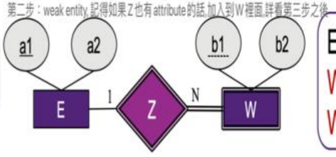
Mapping to Relation

第一步：先畫普通entity
記得derived, multivalued attribute不用畫入
如果ER中有多個key, 任選一個當PK

E [a1, a3, a4]

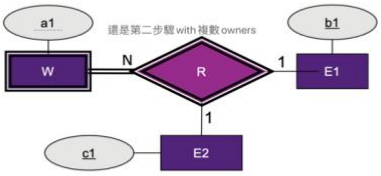
ER Diagram

第二步：weak entity, 記得如果Z也有attribute的話,加入到W裡面,詳看第三步之後



Mapping to Relation

E [a1, a2]
W [a1, b2, b2]
W.a1 references E.a1



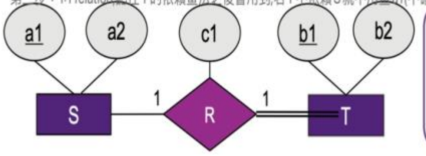
Mapping to Relation

W [a1, b1, c1]
- W.b1 references E1.b1
- W.c1 references E2.c1

E1 [b1]
E2 [c1]

ER Diagram

第三步：1:1 relation,記住T的依賴畫法之後會用到,若T不依賴S就不用畫a1(不確定)

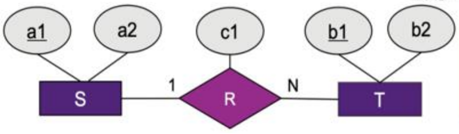


Mapping to Relation

S [a1, a2]
T [b1, b2, a1, c1]
T.a1 references S.a1

ER Diagram

recursive relation 算在這一步在自己的 relation 中提到引用自己的 PK 就可以



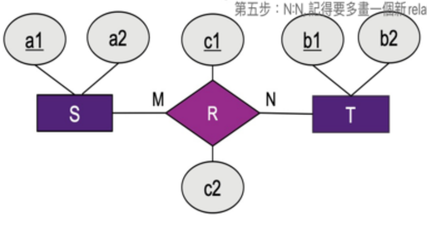
Mapping to Relation

第四步：1:N, 記住一定是N的那一方要提及另一方的attribute

S [a1, a2]
T [b1, b2, a1, c1]
T.a1 references S.a1

ER Diagram

第五步：N:N, 記得要多畫一個新relation "R"

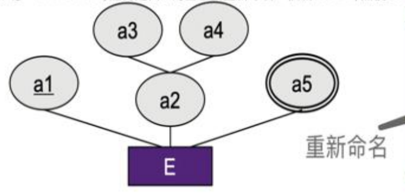


Mapping to Relation

S [a1, a2]
T [b1, b2]
R [a1, b1, c1, c2]
R.a1 references S.a1
R.b1 references T.b1

ER Diagram

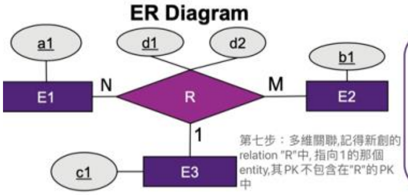
第六步：multivalued, 記得是由E為基底去重新命名一個新relation, 然後derived attribute就省略(哭哭



Mapping to Relation

E [a1, a3, a4]
E2 [a1, a5]
E2.a1 references E1.a1

ER Diagram

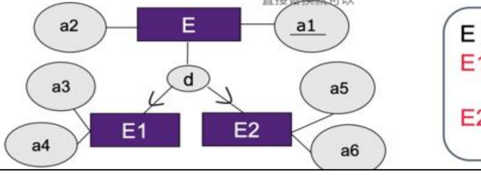


第七步：多維關聯,記得新創的relation "R"中, 指向1的那個entity,其PK不包含在"R"的PK中

Mapping to Relation

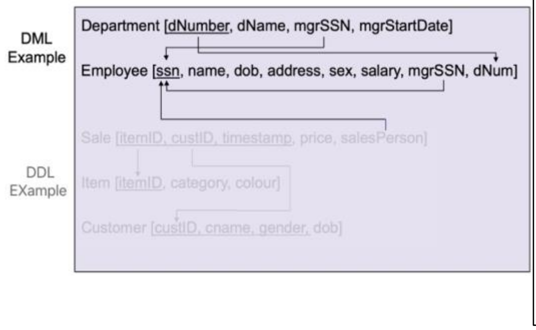
E1 [a1] E2 [b1] E3 [c1]
R [a1, b1, d1, c1 d2]
R.a1 references E1.a1
R.b1 references E2.b1
R.c1 references E3.c1

ER Diagram 最後一步：EER,記得d跟o畫法一樣, 直接替換就可以



Mapping to Relation

E [a1, a2]
E1 [a1, a3, a4]
E1.a1 references E.a1
E2 [a1, a5, a6]
E2.a1 references E.a1



DML Example

Department [dNumber, dName, mgrSSN, mgrStartDate]

Employee [ssn, name, dob, address, sex, salary, mgrSSN, dNum]

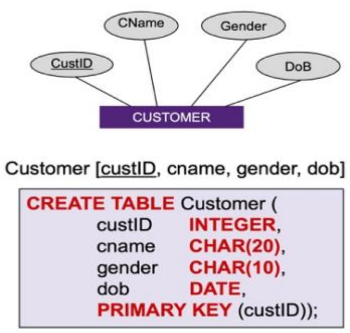
DDL EXample

Sale [itemID, custID, timestamp, price, salesPerson]

Item [itemID, category, colour]

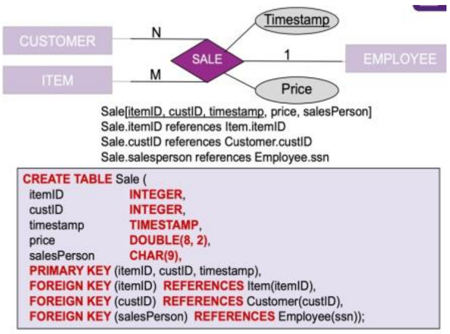
Customer [custID, cname, gender, dob]

Relational Query 的幾個必備功能:INSERT, DELETE, UPDATE, SELECT. SQL 的三個 statements: 1.Data Definition Language(也就是 CREATE, ALTER, DROP): Statements to define the database schema. 2. Data Manipulation Language(也就是 SELECT, INSERT , UPDATE, DELETE): Statements to manipulate the data. 3. Data Control Language(較難): a. Statements to specify transaction control, semantic integrity (triggers and assertions), authorization and management of privileges. b. Statements for specifying the physical storage parameters such as file structures and access paths (indexes). c. Statements to specify the role-based security controls. DDL 常用語法: DROP TABLE (刪除這個 TABLE 中的所有 constraint, 包括因此 TABLE 而產生的) 程式碼:



Customer [custID, cname, gender, dob]

```
CREATE TABLE Customer (
        custID      INTEGER,
        cname       CHAR(20),
        gender      CHAR(10),
        dob         DATE,
        PRIMARY KEY (custID));
```

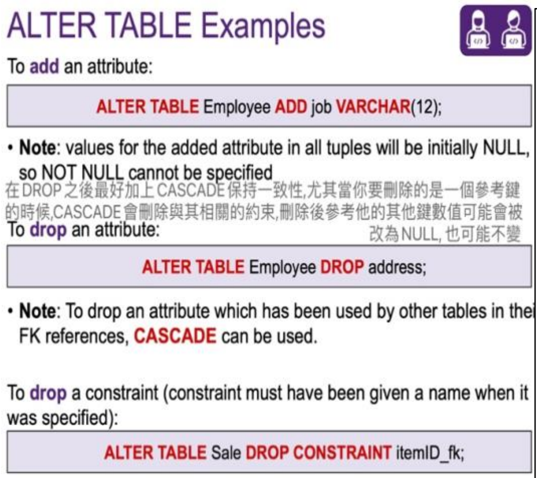


Sale[itemID, custID, timestamp, price, salesPerson]
Sale.itemID references Item.itemID
Sale.custID references Customer.custID
Sale.salesperson references Employee.ssn

```
CREATE TABLE Sale (
 itemID              INTEGER,
 custID              INTEGER,
 timestamp           TIMESTAMP,
 price               DOUBLE(8, 2),
 salesPerson         CHAR(9),
 PRIMARY KEY (itemID, custID, timestamp),
 FOREIGN KEY (itemID)  REFERENCES Item(itemID),
 FOREIGN KEY (custID)  REFERENCES Customer(custID),
 FOREIGN KEY (salesPerson)  REFERENCES Employee(ssn));
```

在第一張圖中,PRIMARY KEY (custID)跟定義一個新 Constraint 即: COUNSTRAINT item_pk PRIMARY KEY (itemID)是一樣的，只不過特別定義的話也代表特別好修改，例如 ALTER TABLE Item DROP CONSTRAINT item_pk;就可以輕易將他 DROP.報錯的時候也會報錯名字.

＊Semantic Constraints：用 Check，例如：CHECK (price >= 8 AND price < 10)或 CHECK (gender IN(“male”，“female”) ＊CONSTRAINT itemID_fk FOREIGN KEY (itemID) REFERENCES Item(itemID) ON DELETE RESTRICT ON UPDATE CASCADE，其中”ON DELETE RESTRAINT”代表 Item(itemID)要被刪除的請求會被拒絕,因為他是 itemID 的參考鍵.”ON UPDATE CASCADE”代表的是父子表保持一致,即：當你改變 Item(itemID), itemID 也會跟著改變.



ALTER TABLE Examples

To add an attribute:

```
ALTER TABLE Employee ADD job VARCHAR(12);
```

- **Note**: values for the added attribute in all tuples will be initially NULL, so NOT NULL cannot be specified

在DROP之後最好加上CASCADE保持一致性,尤其當你要刪除的是一個參考鍵的時候,CASCADE會刪除與其相關的約束,刪除後參考他的其他鍵數值可能會被改為NULL,也可能不變

To drop an attribute:

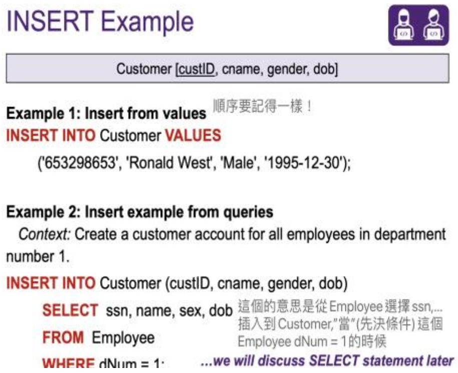
```
ALTER TABLE Employee DROP address;
```

- **Note**: To drop an attribute which has been used by other tables in thei FK references, CASCADE can be used.

To drop a constraint (constraint must have been given a name when it was specified):

```
ALTER TABLE Sale DROP CONSTRAINT itemID_fk;
```

DML 四個功能: 1.INSERT – insert data into a table. 2.UPDATE – updates existing data within a table. 3.DELETE – deletes records from a table. 4.SELECT – retrieve data from a database.



INSERT Example

Customer [custID, cname, gender, dob]

Example 1: Insert from values 順序要記得一樣！

```
INSERT INTO Customer VALUES
     ('653298653', 'Ronald West', 'Male', '1995-12-30');
```

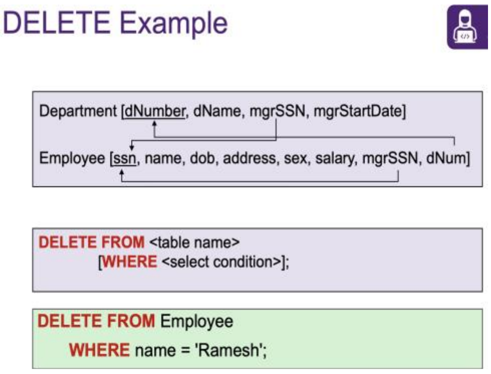
Example 2: Insert example from queries

Context: Create a customer account for all employees in department number 1.

```
INSERT INTO Customer (custID, cname, gender, dob)
     SELECT  ssn, name, sex, dob
     FROM  Employee
     WHERE dNum = 1;
```

這個的意思是從Employee選擇ssn,...插入到Customer,"當"(先決條件)這個Employee dNum = 1的時候

...we will discuss SELECT statement later



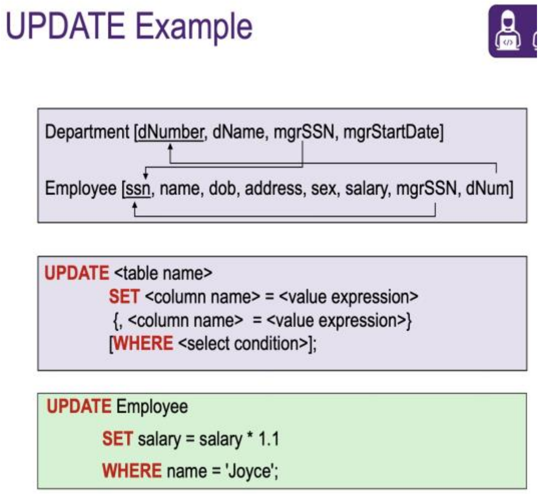
DELETE Example

Department [dNumber, dName, mgrSSN, mgrStartDate]

Employee [ssn, name, dob, address, sex, salary, mgrSSN, dNum]

```
DELETE FROM <table name>
        [WHERE <select condition>];
```

```
DELETE FROM Employee
    WHERE name = 'Ramesh';
```



UPDATE Example

Department [dNumber, dName, mgrSSN, mgrStartDate]

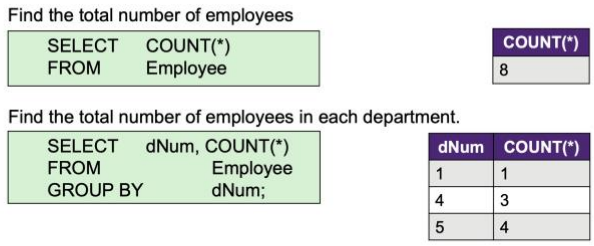
Employee [ssn, name, dob, address, sex, salary, mgrSSN, dNum]

```
UPDATE <table name>
         SET <column name> = <value expression>
         {, <column name>  = <value expression>}
         [WHERE <select condition>];
```

```
UPDATE Employee
         SET salary = salary * 1.1
         WHERE name = 'Joyce';
```

Select 的一些特殊用法:1. WHERE 中可以使用”LIKE”表查找關鍵字,例: WHERE location LIKE “%Gabba%”,意思就是找出所有 location 中含有”Gabba”的 tuple(看你 SELECT 什麼).”%Gabba%”代表模糊匹配有”Gabba”就可以被選上,”_Gabba_”代表只會選中有七個字符的 tuple 並且中間五個是”Gabba”.2. IN，例:WHERE lastName IN (‘Jones’, ‘Wong’, ‘Harrison’) 3. IS, WHERE dNum IS NULL(IS 通常用來判別是否是 NULL.) 4. +, -, *, /, date and time functions, etc.例:WHERE salary * 2 > 50000 以及 WHERE YEAR(GETDATE() - dob) > 55. 5. BETWEEN，例 WHERE salary BETWEEN 10000 AND 30000.複合條件記得用 AND 或 OR 連接. Select 的聚合函數: COUNT: Counts the number of tuples that the query returns, SUM/AVG: Calculates the sum/average of a set of numeric values, MAX/MIN: Returns the maximum/minimum value from a set of values which have a total ordering. Note that the domain of values can be non-numeric. (可以與 DISTINCT 一起使用, “＊”的意思是全部 tuple 的意思)

Group BY：例：



Find the total number of employees

```
SELECT    COUNT(*)
FROM      Employee
```

COUNT(*)
8

Find the total number of employees in each department.

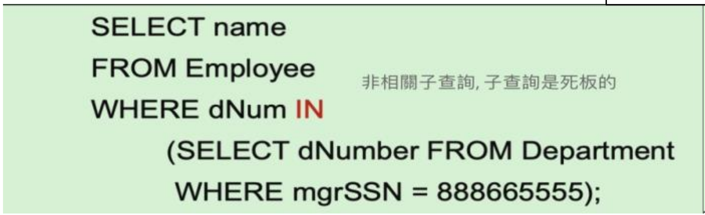
```
SELECT      dNum, COUNT(*)
FROM                  Employee
GROUP BY              dNum;
```

dNum	COUNT(*)
1	1
4	3
5	4

＊當你需要從多個表格中顯示結果時，使用聯結。＊當你需要比較聚合值與其他值時，使用子查詢。

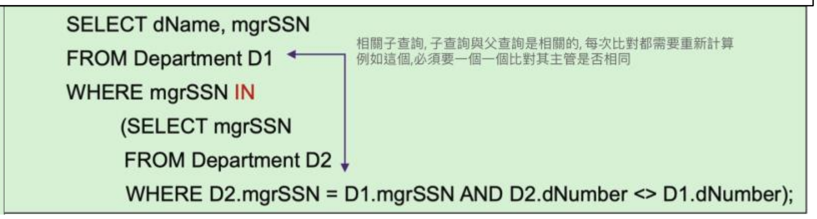
用了 GROUP BY 基本上就要用 HAVING 因為 GROUP BY 相當於是把個體變成群組，要對群組進行任合操作就得用 HAVING.例如：SELECT department, AVG(salary) FROM Employee GROUP BY department HAVING AVG(salary) > 5000; ＊複數的 GROUP BY 代表要這複數的 GROUP BY 參數都一樣才會是同一群組，只要有一個不一樣就是不同群組.

JOIN: INNER JOIN 跟 OUTER JOIN:INNER JOIN 基本就是普通 JOIN. OUTER JOIN 就是會顯示出沒被匹配到的行,取決於 LEFT 或是 RIGHT JOIN(LEFT JOIN 就會顯示沒配對到的 tuple 的左表格 attribute，右表亦然)且 LEFT OUTER JOIN 中，FROM 後面提及的表是左表，而 RIGHT OUTER JOIN 中，FROM 後面提及的表是右表。＊SQL 三個邏輯運算：UNION, INTERSEPT, EXCEPT(差)



```
SELECT name
FROM Employee
WHERE dNum IN
      (SELECT dNumber FROM Department
       WHERE mgrSSN = 888665555);
```

非相關子查詢, 子查詢是死板的



```
SELECT dName, mgrSSN
FROM Department D1
WHERE mgrSSN IN
      (SELECT mgrSSN
       FROM Department D2
       WHERE D2.mgrSSN = D1.mgrSSN AND D2.dNumber <> D1.dNumber);
```

相關子查詢, 子查詢與父查詢是相關的, 每次比對都需要重新計算
例如這個,必須要一個一個比對其主管是否相同