

日期: /

Attribute (/column) Eg: name

Schema: table with list of attributes



Tuple: (/ row) Eg: (1, James, --)

Relation: (/ table)

Super key  $\supseteq$  Candidate key  $\supseteq$  Primary key  
(uniquely identify tuples) (minimal) (one of candidate key)

Schema diagram

? Foreign key : { one to one  
one to many  
many to many

Relational algebra:

Relational algebra expression :  $\prod_{\substack{\text{brand} \\ \text{price} > 8.0}} \text{name} (\text{Epicarso} (\text{Product}))$   
tree :

Query

Select 6:  $\sigma_{\text{price} > 8.0} (\text{Product})$   $\sigma_{(\text{price} > 8.0) \vee (\text{brand} = 'DP')} (\text{Product})$   
and  $\wedge$ , or  $\vee$ , not  $\neg$

Project  $\prod$ :  $\prod_{\text{name}} (\text{Product})$ , show 'name' attribute.

Cartesian product  $\times$ :  $\sigma_{\text{customer. cust-id} = \text{invoice. cust-id}} (\text{Customer} \times \text{Invoice})$

{ Union  $\cup$  :  $\boxed{1}$   $\cup$   $\boxed{2}$  =  $\boxed{1 \atop 2}$ , union the tuples.

Set difference  $-$  :  $\boxed{1}$   $-$   $\boxed{2}$  =  $\boxed{1}$ , tuples - tuples.

Set intersection  $\cap$  :  $\boxed{1}$   $\cap$   $\boxed{2}$  =  $\boxed{1}$ , tuples

日期:

Natural join  $\bowtie$ : equal to  $\Pi_R \text{ attributes} \times S \text{ attributes} (G_R \text{ after } A_S \text{ after } (R \times S))$

Rename  $f$ :  $P_{TEMP} (\text{Product})$ ,  $\text{Product} \times P_{TEMP} (\text{Product})$

Assignment  $\leftarrow$ : (like variable, for simplification)

Eg:  $\text{Temp 1} \leftarrow \Pi_{-} (G \leftarrow)$

$\text{Temp 2} \leftarrow \Pi_{-} (e \leftarrow)$

$\text{Temp 1} \cup \text{Temp 2}$ .

Eg: Database modification

$\text{Project} \leftarrow \Pi_{-} (G_n (\text{Project}))$

Generalized projection: Eg:  $\Pi_{\text{price} * 80\% \text{ as discount price}} (\text{Product})$

Aggregate function: (Function: sum, avg, count, min, max)

Eg:  $G_{\text{max(price)}} (\text{Product}) \rightarrow \boxed{\text{max(price)}}$

Aggregate function + Group: first Group  $\rightarrow$  Aggregate function based on group

Eg:  $\text{cust-id } G_{\text{sum(amount)}} (\text{Invoice})$

$\uparrow$   
Group by  
 $\uparrow$   
Function on each group

$\rightarrow$ 

cust-id	sum( $\rightarrow$ )

Null value:

null	2
1	null

Insertion:  $\text{Product} \leftarrow \text{Product} \cup \{(5, 'soda', -)\}$

insert new tuple.

Deletion:  $\text{Product} \leftarrow \text{Product} - G_p (\text{Product})$ .

delete tuple.

# SQL / Clause

create table Invoice (inv\_id int,  
primary key (inv\_id))

alter table Invoice add staff\_id int  
(add column)

① select \* (all)  
select A<sub>1</sub>, A<sub>2</sub> from r<sub>1</sub>, r<sub>2</sub> where P;

$\Leftrightarrow \Pi_{A_1, A_2} (G_P(r_1 \times r_2))$ .

② select . . . from . . . where price > 8.0 and price < 9.0  
                  | where price between 8.0 and 9.0

③ select I.inv\_id, C.name  
from Customer as C, Invoice as I  
where C.cust\_id = I.cust\_id

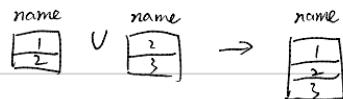
$\Pi_{\text{name}} (\text{customer} \bowtie \text{Invoice})$

④ select \* from Product order by price asc (desc)

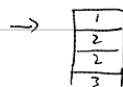
⑤ select max(price) from Invoice

⑥ select \* from Invoice group by cust\_id

⑦ (select name from A)  
union (intersect, except)  
(select name from B)



union all



⑧ insert into Product values (5, 'soda' --)

日期:

delete from Product where prod\_id=3)

update Product set price = price \* 2 where brand = 'AIS'

## ⑨ inner join

select \* from  
A inner join B on  
A.id = B.id

⑧ left outer join : All attribute in left table appear in result.  
fill Null.

select \* from  
A left outer join B on  
A.id = B.id

⑨ full outer join : left and right all appear.

## ⑩ subqueries :

```
select * from Product  
where price <  
( select avg(price) from Product )
```

⑪ 78 in  $\exists$ , Eg. where price < all

8.0 > come  , (Select price from Product  
where brand = 'CO') ;  
8.0 > all  .

⑫	<u><u>exists</u></u>	<u><u>unique</u></u>
≡	exists in (table)	no duplicates

⑭ View    Create View    Project  
Select \* from Project;

⑯ group by ~. having ~; select cust-id, sum(amount)  
from ~ group by ~;

日期: /

⑭ create view DPC as

view = temp(table)

select —

from —

group by —

having — ;



⑮ select \* from — where

{
not exists
— < some
— < all

subquery ( select ~ from ~ where ~ )

⑯ select — from A (only support A.—, not B.—)  
 inner join B on B.— = A.— (B must in front)  
group by A.attr; (B must in front)  
having AVG(A.—) = ~; (A.attr must in select A.attr)

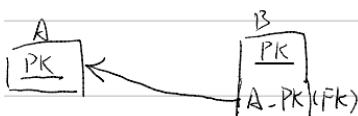
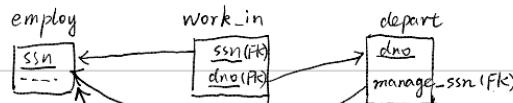
△ B to A: many to one Def: B A .

B to A : is many to one. (or one to one)

C (B-PK, A-PK)  $\leftarrow$  (B-PK)  
superkey

schema diagram (relational schema)

if then

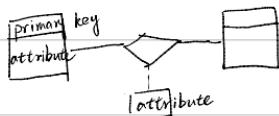


FK: Foreign key

日期:

ER Diagram

UML notation



Entity set : (tuple)

Relationship set

Multi-valued attributes :



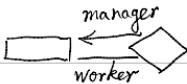
have many values

Eg: one man has 2 phones

{ one to one relationship

one to many relationship one many

Role

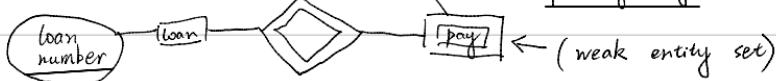


{ Total participation :   
Partial   
partial total

Degree : number of entity sets

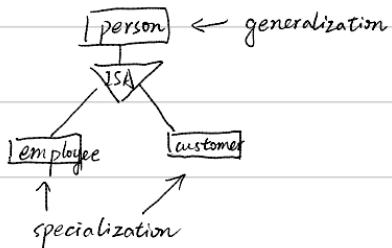


Weak entity sets : (identifying relationship)  
  
(discriminator or partial key)

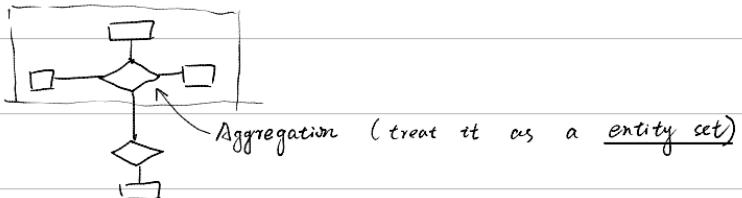


日期: /

specialization, generalization :



Aggregation : eliminate redundancy

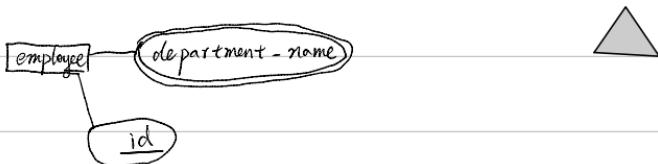


Representing entity set :

strong entity set:  $\text{loan} = (\underline{\text{loan number}}, \underline{\text{amount}})$

weak entity set:  $\text{payment} = (\underline{\text{loan number}}, \underline{\text{payment number}})$

multivalued attribute :  $\text{employee-depart-name} = (\underline{\text{id}}, \underline{\text{depart-name}})$



Specialization :

(generalization)  $\text{person} = (\underline{\text{id}}, \rightarrow)$

(specialization)  $\text{customer} = (\underline{\text{id}}, \rightarrow), \text{employee} = (\underline{\text{id}}, \rightarrow)$

日期： /  
First Norm

Def: ① attribute is atomic : cannot be divided into parts  
(CS101)  $\neq$  (CS, 101)

② no duplicates

good form : ① lossless-join decomposition, ②  $R_1, \dots, R_n$  is good form  
 split  $R \rightarrow \underbrace{\{R_1, \dots, R_n\}}_{(\text{Good Form})}$ , e.g. BCNF, 3NF

$\alpha \rightarrow \beta \rightsquigarrow \underline{\text{rule}} : \alpha \text{ determines uniquely } \beta$

$$\begin{array}{l} \text{(well defined)} \\ \text{(default: surjective)} \end{array} \quad \left( \begin{array}{cc} 1 & 4 \\ 1 & 5 \\ 3 & 7 \\ \hline A & B \end{array} \right) \quad B \rightarrow A \quad \checkmark, \quad A \rightarrow B \quad \times$$

$\text{tuple } t_1, t_2 : \text{ if } \exists \beta \Rightarrow t_1(\beta) = t_2(\beta) \Rightarrow t_1(\beta) = t_2(\beta)$   
 $(\text{key})$

functional dependencies :  $\alpha \rightarrow \beta$

Super key :  $K \rightarrow R$  (relation schema R)

candidate key:  $K \rightarrow R$ ,  $\# \neq K$ , s.t.  $K \rightarrow R$ .

$(ID, \text{name}) \rightarrow (ID)$

trivial:  $\lambda \rightarrow \beta$  is trivial if  $\beta \subset \lambda$ ,  $(\text{ID}, \text{name}) \rightarrow (\text{ID})$

Armstrong's axioms : { ①  $\beta \subset \alpha \Rightarrow \alpha \rightarrow \beta$  (reflexivity)       $\gamma \subset \alpha$ :  
 ②  $\alpha \rightarrow \beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma$  (augmentation) + attribute  
 ③  $\alpha \rightarrow \beta, \beta \rightarrow \gamma \rightarrow \alpha \rightarrow \gamma$  (transitivity)

日期:

 closure of  $f$ :  $(f^+)$

$$f = \{ \underbrace{A \rightarrow B, A \rightarrow C, \dots}_{\uparrow} \}$$

$$f^+ = \text{expand}(f)$$

functional dependencies

- ① union:  $\alpha \rightarrow \beta, \alpha \rightarrow \gamma \Rightarrow \alpha \rightarrow \beta\gamma$
- ② decomposition:  $\alpha \rightarrow \beta\gamma \Rightarrow \alpha \rightarrow \beta, \alpha \rightarrow \gamma$
- ③ pseudo-transitivity:  $\alpha \rightarrow \beta, \gamma\beta \rightarrow \delta \Rightarrow \gamma\alpha \rightarrow \delta$ .

closure of  $\alpha$  (i.e.  $\alpha^+$ ) under  $f$

① result :=  $\alpha$   
 while (result changed) do  
   for each  $\beta \rightarrow \gamma$  in  $f$  do  
     if  $\beta \subseteq \text{result}$  then  
       result := result  $\cup \gamma$

②  $f^+$  :  
 for each  $\gamma \in R$   
   find closure  $\gamma^+$   
 for each  $\beta \subseteq \gamma^+$   
   output  $\gamma \rightarrow \beta$ .  
 $f^+ = f \cup (\gamma \rightarrow \beta)$

find candidate key: 1个 {A}, 2个 {AB}, ---.

 canonical cover of  $f$  (minimal) :

$(f_c)$  means no redundant and extraneous.

extraneous: E.g. ①  $f = \{A \rightarrow C, AB \rightarrow CD\}$

B is extraneous in  $AB \rightarrow CD$

②  $f = \{A \rightarrow C, AB \rightarrow CD\}$      C is extraneous.

$$f_c = \{A \rightarrow C, AB \rightarrow D\}$$

Lossless-join decomposition:  $R = (R_1, R_2)$

r: relation    R: attr

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

 Thm:  $R_1, R_2$  is loss-less decomposition if  $\begin{cases} R_1 \cap R_2 \rightarrow R_1 \\ \text{or} \\ R_1 \cap R_2 \rightarrow R_2 \end{cases}$

dependency preserving: ①  $f$  must be canonical cover.  $f = f_c$

② E.g.  $f = \{A \rightarrow B, B \rightarrow C\}$      $R_1 = (A, B), R_2 = (B, C)$

all functional dependency can be splitted into  $R_1, \dots$

日期:

BCNF Normal Form :  $\forall$  functional dependency  $\alpha \rightarrow \beta \in F^+$   
at least one satisfies:  $\alpha \in R, \beta \in R$   
 $\left\{ \begin{array}{l} \alpha \rightarrow \beta \text{ is trivial (i.e. } \beta \subseteq \alpha) \\ \alpha \text{ is superkey for } R \end{array} \right.$

BCNF decomposition: each  $R_i$  is in BCNF, and  
decomposition is lossless-join.  
( preserve dependency)

3NF :  $\forall \alpha \rightarrow \beta \text{ in } F^+$

at least one satisfies :

$\left\{ \begin{array}{l} \alpha \rightarrow \beta \text{ is trivial (i.e. } \beta \subseteq \alpha) \\ \alpha \text{ is superkey for } R \end{array} \right.$

each attribute  $A$  in  $\beta - \alpha$ , is contained in a candidate key for  $R$ .

( maybe different candidate key )

3NF decomposition ensures : each  $R_i$  is in 3NF,

may not dependency preserving  
loss-less decomposition.

3NF  $\supset$  BCNF

日期: /

日期: /