Assignment 5

Tong Zhou(tz13) 533

1.

CREATE TABLE Organization (

id VARCHAR(50) NOT NULL,

name VARCHAR(50) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE University (

id VARCHAR(50) NOT NULL,

name VARCHAR(50) NOT NULL,

rank INT,

PRIMARY KEY (id)

);

CREATE TABLE Meet (

name VARCHAR(50) NOT NULL,

start\_date DATE NOT NULL,

num\_days INT NOT NULL,

org\_id VARCHAR(50) NOT NULL,

PRIMARY KEY (name),

FOREIGN KEY (org\_id) REFERENCES Organization (id) );

CREATE TABLE Stroke (

stroke VARCHAR(50) NOT NULL,

PRIMARY KEY (stroke)

);

CREATE TABLE Distance (

distance INT NOT NULL,

PRIMARY KEY (distance) ;

);

CREATE TABLE Event (

id VARCHAR(50) NOT NULL,

gender CHAR(1) NOT NULL,

distance INT,

PRIMARY KEY (id),

FOREIGN KEY (distance) REFERENCES Distance (distance)

);

CREATE TABLE Heat (

id VARCHAR(50) NOT NULL,

meet\_id VARCHAR(50) NOT NULL,

event\_id VARCHAR(50) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (meet\_id) REFERENCES Meet (id),

FOREIGN KEY (event\_id) REFERENCES Event (id)

);

CREATE TABLE Participant (

id VARCHAR(50) NOT NULL,

gender CHAR(1) NOT NULL,

univ\_id VARCHAR(50) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (univ\_id) REFERENCES University (id) );

CREATE TABLE Leg ( leg INT NOT NULL, PRIMARY KEY (leg)

);

CREATE TABLE Swim (

participant\_id VARCHAR(50) NOT NULL,

heat\_id VARCHAR(50) NOT NULL,

time TIME NOT NULL,

rank INT,

leg INT NOT NULL,

PRIMARY KEY (participant\_id, heat\_id),

FOREIGN KEY (participant\_id) REFERENCES Participant (id),

FOREIGN KEY (heat\_id) REFERENCES Heat (id),

FOREIGN KEY (leg) REFERENCES Leg (leg)

);

CREATE TABLE StrokeOf (

event\_id VARCHAR(50) NOT NULL,

leg INT NOT NULL,

stroke VARCHAR(50),

PRIMARY KEY (event\_id, leg),

FOREIGN KEY (event\_id) REFERENCES Event (id),

FOREIGN KEY (leg) REFERENCES Leg (leg),

FOREIGN KEY (stroke) REFERENCES Stroke (stroke)

);

2. In swim table, rank is determined by participate\_id, heat\_id and time. But (participate\_id, heat\_id and time) is not key. So this table does not satisfy BCNF.

3.

A->BE => A->B Decomposition

A->B => A->AB Union

A->AB AB->C => A->C Transitivity

A->C C->D => A->D Transitivity

4.

SELECT \* FROM R r1

 CROSS JOIN R r2

 WHERE r1.B = r2.B

 AND r1.C != r2.C

This will return empty table if the dependency holds.

5.

This won’t change this algorithm’s correctness because once F=>C has been applied, Attributes C has already in results. So applying F=>C again wont’ add new attributes in results.

This version’s asymptotic running time is O(n\*(n+1)/2), because for the ith search, it needs (n – i - 1)times to search in the result sets with size of (n-i-1). So the total running time is (n + n -1 + n-2+…+1)= O(n\*(n+1)/2).

6.

(1)

a.

Closure{AB}={A,B,C,D}

Closure{A}={A}

Closure{B}={B}

Closure{C}={A,C,D}

Closure{D}={A,D}

Closure{BD}={ABCD}

So keys are AB,BD and BC.

b.

closure{B}={B,C,D}

so key is AB

(2)

a. none avoids 3NF.

b. none

(3)

a. C->D because it’s not trivial and C is not a super key of R

D->A because it’s not trivial and D is not a super key of R.

b. both because they are not trivial and B is not a super key of R.

(4)

a.

X=C

X+=(A,C,D)

D=X+-X=(A,D)

N=R-X+=(B)

R1=X ∪D=(A,C,D)

R2=X∪N=(B,C)

For R1

X=D

X+=(A,D)

D=X+-X=(A)

N=R1-X+=(C)

R11=X∪D=(A,D)

R12=X∪N=(C,D)

So it can be composed into R11(A,D), R12(C,D), R2(B,C).

b.

X=B

X+=(B,C,D)

D=X+-X=(C,D)

N=R-X+=(A)

R1=X∪D=(B,C,D)

R2=X∪N=(A,B)

So composition is R1(B,C,D), R2(A,B).