# CS 411 Database Systems – HW 2 Submitted by: Payal Mantri(payaljm2)

## Q1 – Parking Lot Management System ER Diagram

Diagram

Description automatically generated

## Q2 – Theatre Management System ER Diagram\\\Diagram Description automatically generated

## Q3 – Restaurant DB System Schema

Solution:

-- Restaurant Management System DDL

-- customer table with id , name and birthdate

create table customer

(

id int not null,

name varchar(50) not null,

birthdate date not null,

primary key (id)

);

-- restaurant table with name , type , size and location

-- location is a foreign key to building table as one restaurant can only be in one building

-- restaurant is a weak entity of building

create table restaurant

(

name varchar(50) not null,

type varchar(50) not null,

size int not null,

location int not null,

primary key (name),

foreign key (location) references building(id) on delete cascade

on update cascade

);

-- 1 building can have many restaurants

-- 1 restaurant can only be in 1 building

-- 1 to many relationship

-- building table with name , address and phone

-- address is a primary key to address table

create table building

(

id int not null,

name varchar(50) not null,

address int not null,

phone varchar(50) not null,

primary key (id),

foreign key (address) references address(id)

);

-- chef with ssn, name, salary, restaurant

-- ssn is primary key

-- restaurant is foreign key to restaurant table as one chef can work in only one restaurant

create table chef

(

ssn int not null,

name varchar(50) not null,

salary int not null,

restaurant varchar(50) not null,

primary key (ssn),

foreign key (restaurant) references restaurant(name)

);

-- Create entity with name , popularity, flavor, price, restaurant

-- name is primary key

-- restaurant is foreign key to restaurant table as one dish can only be in one restaurant

-- however, one restaurant can have many dishes

-- dish is a weak entity of restaurant

create table dish

(

name varchar(50) not null,

popularity int not null,

flavor varchar(50) not null,

price int not null,

restaurant varchar(50) not null,

primary key (name, restaurant), -- same named dish can be in different restaurant

foreign key (restaurant) references restaurant(name) on delete cascade

on update cascade

);

-- 1 chef can have many dishes

-- 1 dish can be cooked by only many chefs

-- many to many relationship

-- create chef\_dish table with chef\_ssn and dish\_name

-- chef\_ssn is foreign key to chef table

-- dish\_name is foreign key to dish table

create table chef\_dish

(

chef\_ssn int not null,

dish\_name varchar(50) not null,

restaurant varchar(50) not null,

foreign key (chef\_ssn) references chef(ssn),

foreign key (dish\_name, restaurant) references dish(name, restaurant)

);

-- 1 customer can have many orders

-- 1 order can only be made by 1 customer

-- 1 order can have many dishes

-- create table order with id , date , customer\_id , dish\_name , restaurant

-- id, customer\_id, restaurant are primary key

-- dish\_name is foreign key to dish table

-- customer\_id is foreign key to customer table

create table orders

(

id int not null,

date date not null,

customer\_id int not null,

dish\_name varchar(50) not null,

restaurant varchar(50) not null,

primary key (id, customer\_id, restaurant),

foreign key (customer\_id) references customer(id),

foreign key (dish\_name, restaurant) references dish(name, restaurant)

);

* 1. R (A, B, C, D, E)   
     F= {B → CDE, CD → AE, E→A}

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| B | A | CDE |  |

B+= {B, C, D, E, A}

Candidate keys = {B}

* 1. R (A, B, C, D, E)   
     F= {AB→CD, E→C, BC→A}

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| BE | D | AC |  |

BE+ = {B, E, C, A, D}

Candidate Keys = {BD}

* + 1. R (A, B, C, D, E, F, G)   
       F= {ABC → DEF, B → AEG, C → EF, F → AE, DG → CE}

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| B | E | ACDFG |  |

B+ = {B, A, E, G}

BA+ = {B, A, E, G}

BC+ = {B, A, E, G, C, F, D} \*

BD+ = {B, A, E, G, D, C, E, F} \*

BF+ = {B, A, E, G, F}

BG+= { B, A, E, G}

BAF = {B, A, E, G, F}

Candidate Keys = {BC, BD, BCA, BCF, BCG, BDA, BDF,BDG, BCDA, BCDF, BCDG, BCDAF, BCDAG, BCDAFG}

* + 1. Non trivial functional dependencies
* BC → DF
* BF → A
* C → A
* BD → C

## 

## 

## R1 = (A, B, C, D, E) with a set of functional dependencies FD = {BC→A, DE→AC, C→BD}

**Solution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| E | A | BCD |  |

* E+ ={E}
* (BE)+ = {B, E}
* (CE)+ = {C, E}
* (DE)+ = {D, E, A, C, B}

So, DE is candidate key.

For a relation to be in BCNF, all functional dependencies X->Y, should have X as superkey.

In above relation, the below two FDs violate the rule and **hence R1 is not in BCNF**.

* BC-> A
* C->BD

For a relation to be in 3-NF, all functional dependencies X->Y, should have X as superkey or Y should be part of a key

Both BC→A and C→BD violates 3NF rule. Hence **R1 is not in 3-NF**

* + 1. **R1 = (A, B, C, D, E, F) with a set of functional dependencies FD = {A→BF, BC→E, CE→D, DEF→C}**

**Solution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| A |  | BCDEF |  |

* A+ = {A, B, F}
* (AB)+ = {A, B, F}
* (AC)+ = {A, B, F, C, E, D} \*
* (AD)+ = {A, B, F, D}
* (AE)+ = {A, B, F, E}

So **AC is candidate key** and all supersets of AC are superkeys

For a relation to be in BCNF, all functional dependencies X->Y, should have X as superkey

In above relation, the all the FDs violate the rule and **hence R2 is not in BCNF**.

For a relation to be in 3-NF, all functional dependencies X->Y, should have X as superkey or Y should be part of a key

The following FDs violate the 3-NF rule :

* A→BF
* BC→E
* CE→D

**Hence R2 is not in 3-NF**

* + 1. **R1 = (A, B, C, D, E, F) with a set of functional dependencies FD = {AB→CD, B→A, BCD→EF}**

**Solution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| B | EF | ACD |  |

* B+ = {B, A, C, D, E, F} \*

Candidate Key = {B}

For a relation to be in BCNF, all functional dependencies X->Y, should have X as superkey

In above relation, the all the FDs have left hand side as superkeys, i.e., AB, B, BCD are all superkeys.

**Hence R3 is in BCNF**

Since every relation in BCNF is also in 3NF

**Hence R2 is in 3-NF**

* 1. R (A, B, C, D, E, F) and functional dependencies FD = {C → F; CE → DF; D→ CF; E → CD; EF→ D}
     1. Find the minimal basis

**Solution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Left | Right | Middle | None |
| E |  | CFD | AB |

ABE+= {A, B, E, C, D, F}  
So candidate key is ABE.

**Step 1: Make all RHS as singleton**

FD = {C → F, CE →D, CE→F, D→C, D→F, E→C, E→D, EF→D}

**Step 2: Remove redundant attributes from LHS**

* CE →D  
  E+= {E, D, C}. So, we can omit the C in FD, resulting FD is E→D
* CE→F

C+ = {C, F} so can omit the E attribute. Resulting FD is C→F

* EF→D  
  E+= {E, D, C} so we can omit F from the FD. Resulting FD is E→D

Updated FDs set FD= {C → F, D→C, D→F, E→C, E→D}

Step 3: Remove redundant FDs

* Ignoring C → F, C+ ={C} hence rule cannot be dropped
* Ignoring D → C, D+= {D, F} hence rule cannot be dropped
* Ignoring **D → F**, D+= {D, C, F} hence **rule can be dropped**
* Ignoring **E → C**, E+= {E, D, C} hence **rule can be dropped**
* Ignoring E → D, E+= {E, C, F} hence rule cannot be dropped

**Minimal Basis = {C → F, D → C, E → D}**

* + 1. Decomposing into 3-NF relations. **ABE** is the key to the table  
       **P (C, F)  
       Q (D, C)  
       S (E, D)  
       T (A, B, E)**
    2. R1 = (A, B, C, D) with a set of functional dependencies FD = {AB→C; C→A; D→B}

CD, AD are the key to the relation R1.

**Step 1:** the RHS should be in singleton form. It is already in proper form

**Step 2:** Remove redundant LHS attributes

For AB→C,   
 A+= {A}

B+={B}

Hence there are no redundant attributes

**Step 3:** Remove redundant FDs

* Ignoring AB→C, AB+ ={AB} hence rule cannot be dropped
* Ignoring C→A, C+ ={C} hence rule cannot be dropped
* Ignoring D→B, D+ ={D} hence rule cannot be dropped

Minimal basis = {AB→C; C→A; D→B}

Decomposing into 3-NF form, we can have following Relations

S (A, B, C), T (C, A), U (D, B), V (C, D)  
   
 ---------------------------------------------------------------------------------------------------------------

* + 1. R1 = (A, B, C, D) with a set of functional dependencies FD = {A→BC; D→AB; BD→AC}

D+= {D, A, B, C}. Hence D is the candidate key of the relation

**Step 1:** the RHS should be in singleton form.

FD = {A→B; A→C; D→A; D→B; BD→A; BD→C}

**Step 2:** Remove redundant LHS attributes

For BD→A,   
 D+= {D, A, C, B} Hence it can be converted to D→A

For BD→C,   
 D+= {D, A, C, B} Hence it can be converted to D→C

FD = {A→B; A→C; D→A; D→B}

**Step 3:** Remove redundant FDs

* Ignoring A→B, A+ = {A, C} hence rule cannot be dropped
* Ignoring A→C, A+ = {A, B} hence rule cannot be dropped
* Ignoring D→A, D+ = {D, B} hence rule cannot be dropped
* Ignoring D→B, D+ = {D, A, B, C} hence rule is redundant and will be dropped

Minimal basis = {A→B; A→C; D→A}

Decomposing into 3-NF form, we can have following Relations

S (A, B), T (A, C), U (D, A)