CS30202: Database Management Systems

Mid-semester Examination Model Solution, Spring 2023

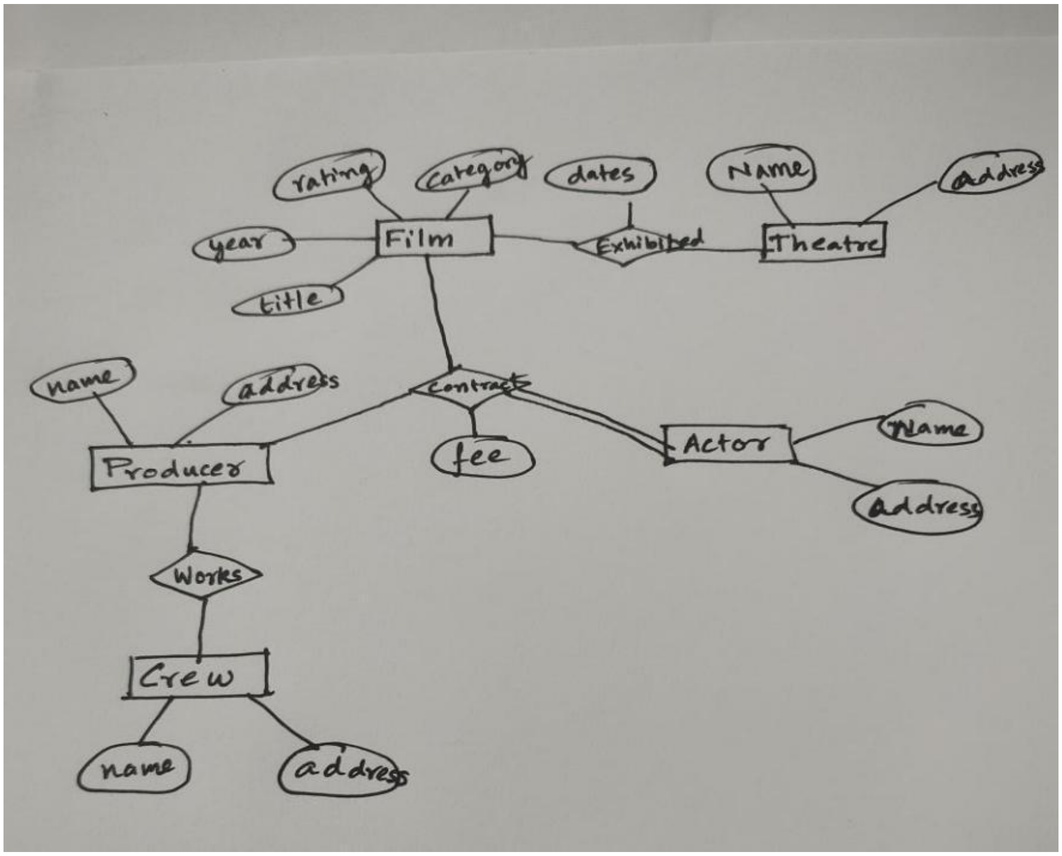
Time: 2 hrs. Total Marks: 60. Answer all three questions.

Please answer all parts of a question together.

1 A. We want to build an Indian movie database including information about: Films, Actors, Crew, Producers, and Theatres. A Film has four attributes: title, year, rating (U, A, UA, S), and category (e.g., action, comedy, romance, fantasy etc.). Actors, Crew, Producer, and Theatres, have a name and address. A Contract is signed between an Actor and a Producer against a fee for making a Film. A Film may be exhibited in multiple Theaters over various date intervals. An actor has signed at least one contract during her/his career. A crew may work for single/none/multiple Producer(s). No two Films, Actors, Producers, and Theatres, have the same name.

Draw the E-R diagram. Mark all participation and cardinality constraints for the relations. [10]

Solution:



1 B. Convert the E-R diagram to a set of equivalent relational tables. [4]

Solution:

```
Film(title, year, rating, category)
Actor(name, address)
Theatre(name, address)
Producer(name, address)
Crew(name, address)
Contract(Filmtitle, Actorname, Producername, fee)
Exhibited(Theatre.name, Film.title, dates)
Works(Crew.name, Producer.name)
```

1 C. Write the relational algebra expressions for the following queries:
[3 X 2= 6]

i) The relational algebra expressions for the following queries: i. Name of all Producers who have produced both "action" and "romance" films having "SRK" as an actor.

Ans:

```
π Producer.name ((σ Film.category = 'action' AND Film.title IN (σ
Film.category = 'romance' AND Actor.name = 'SRK' (Actor ⋈ Contract ⋈
Film ⋈ Producer))))
```

ii) Name of all Actors who have co-acted with Actor "SRK in a film.

Ans:

```
π Actor.name ((σ Film.title IN (σ Actor.name = 'SRK' (Actor ⋈
Contract ⋈ Film)) AND Actor.name!= 'SRK' (Actor ⋈ Contract ⋈ Film)))
```

iii) Name of all Actors whose Film was exhibited in at least 1000 Theatres.

Ans:

```
π Actor.name ((σ COUNT(Theatre.name) >= 1000 (Film ⋈ Exhibits ⋈
Theatre) ⋈ Contract ⋈ Actor))
```

2 A. Consider two relations *R* and *S* having *m* and *n* tuples respectively. Express the following operations in terms of (a) primitive relational algebra operators, (b) SQL queries. What is the minimum and maximum number of tuples that the result of the operations might have?.

[4 x 3 = 12]

i). $R \ltimes S$ is the collection of tuples *t* in *R* such that there is at least one tuple *s* in *S* that agrees with *t* in all attributes that *R* and *S* have in common.

Ans:

(a) If $a_1, \ldots, a_n$ are the attributes of R, then $R \ltimes S = \pi_{a1,\ldots,an}(R \bowtie S)$

(b) SELECT $a_1, \ldots, a_n$ FROM R NATURAL JOIN S or SELECT $a_1, \ldots, a_n$ FROM R WHERE $c_1, \ldots, c_k$ IN (SELECT $c_1, \ldots, c_k$ FROM S) where $c_1, \ldots, c_k$ are the common attributes of R and S.

(c) Min: *0*, Max: *m*

ii). $R \triangleright S$ is the collection of tuples *t* in *R* that do not agree with any of the tuples in *S* in the attributes common to *R* and *S*.

Ans:

(a) $R \triangleright S = R - R \ltimes S$

(b) (SELECT $a_1, \ldots, a_n$ FROM R) EXCEPT SELECT $a_1, \ldots, a_n$ FROM R NATURAL JOIN S or

SELECT $a_1, \ldots, a_n$ FROM R WHERE NOT EXISTS (SELECT $c_1, \ldots, c_k$ FROM S) where c_1,c_k are the common attributes of R and S.

(c) Min: 0, Max: m

iii). $R \div S$ is the restriction of tuples in *R* to the attribute names unique to *R*, (i.e., present in *R* but not in *S*), for which it holds that all their combinations with tuples in *S* are present in *R*.

Ans:

(a) $R \div S = \pi_{a1,\ldots,an}(R) \times S - R$, where $a_1, \ldots, a_n$ are the attributes of R

(b) SELECT * FROM R as sx WHERE NOT EXISTS ((SELECT p.y FROM S as p)

EXCEPT (SELECT sp.y FROM R as sp WHERE sp.x = sx.x))

(c) Min: 0 ,Max: floor(m/n)

2 B. Write SQL statements involving the following three relations answering the following queries. Two or more movies released in two different years can have the same title. [4 X 2 =8]

`Movie(title, year, rating, category, producerId)`

`ActedIn(movieTitle, movieYear, actorName)`

`Producer(id, name, address)`

i. Name of all the producers of the movie titled "Devdas".

Ans:

SELECT Producer.name FROM Movie, Producer WHERE Movie.title = "Devdas"

ii. Name of all producers of movies in which "SRK" acted.

Ans:

SELECT Producer.name FROM Movie, ActedIn, Producer WHERE actorName = "SRK" AND movieTitle = title AND id = producerId

iii. Number of "romance" movies in which "SRK" acted.

Ans:

SELECT COUNT(category) FROM Movie ActedIn

WHERE category = "romance" AND actorName = "SRK" AND title = moviTitle AND year = movieYear

GROUP BY category

iv. Titles that have been used for two or more movies.

Ans:

SELECT title FROM Movie GROUP BY title HAVING COUNT(title) >= 2

3. A. Define what is a functional dependency (FD). ?

Definition: A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets. It is denoted as X → Y, where X is a set of attributes that is capable of determining the value of Y. The attribute set on the left side of the arrow, X is called Determinant, while on the right side, Y is called the Dependent.

For any two instances t1,t2 in a Relation R & X,Y are a set of attributes AND

if t1[x]=t2[x] then t1[Y]=t2[Y].

Eg.

roll_no	**name**	**dept_name**	**dept_building**
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3

46	mno	EC	B2
47	jkl	ME	B2

From the above table we can conclude some valid functional dependencies:

roll_no → { name, dept_name, dept_building }
roll_no → dept_name
dept_name → dept_building e.t.c

3. B. Consider the relation schema R(A, B, C, D) with a set of FD's : AB→C, C→D, D→A.

Taking Attribute Closure
{}+={}
{A}+={A}
{B}+={B}
{C}+={C,D,A}
{D}+={D,A}
{A,B}+={A,B,C,D} - Candidate Key
{A,C}+={A,C,D}
{A,D}+={A,D}
{B,C}+={B,C,D,A} - Candidate Key
{B,D}+={B,D,A,C} - Candidate Key
{C,D}+={C,D,A}
{A,B,C}+={A,B,C,D} - Super Key
{A,B,D}+={A,B,D,C} - Super Key
{A,C,D}+={A,C,D} - Super Key
{B,C,D}+={B,C,D,A} - Super Key
{A,B,C,D}+={A,B,C,D} - Super Key

(i) List all the nontrivial FD's that follow from the given FD's. Restrict the list to FD's with single attributes on the right side.

Ans:
C→D , C→A , D→A , AB→C , AB→D , AC→D , BC→D , BC→A , BD→A , BD→C, CD→A , ABC→D , BCD→A

(ii) What are the candidate keys of R?

Ans:

Essential attribute of the relation is : B , So, attribute B will definitely be a part of every candidate key.
Check closure of AB,BC and BD.
AB , BC and BD are the Candidate Keys.

3. C. Consider the schema R = (A, B, C, D, E, F, G) and the set F of functional dependencies

AB → CD
B→D
DE→B
DEG→ AB
AC→DE

R is not in BCNF for many reasons, one of which arises from the functional dependency AB→CD. Explain why AB→CD shows that R is not in BCNF and then use the BCNF decomposition algorithm starting with AB

Ans:

Essential attribute of the relation is : F,G , So, attribute F and G will definitely be a part of every candidate key.

Check closure of (FG) : (FG)+ = {F,G}
Check closure of (AFG,BFG,CFG,DFG,EFG)
{A,F,G}+={A,F,G}
{B,F,G}+={B,F,G,D}
{C,F,G}+={C,F,G}
{D,F,G}+={D,F,G}
{E,F,G}+={E,F,G}
None of them are candidate keys,
Check closure of (ABFG,ACFG,ADFG,AEFG,BCFG,BDFG,BEFG,CDFG,CEFG,DEFG)
{A,B,F,G}+={A,B,F,G,C,D,E} - Candidate Key
{A,C,F,G}+={A,C,F,G,D,E,B} - Candidate Key
{A,D,F,G}+={A,D,F,G}
{A,E,F,G}+={A,E,F,G}

{B,C,F,G}+={B,C,F,G,D}
{B,D,F,G}+={B,D,F,G}
{B,E,F,G}+={B,E,F,G,D,A,C} - Candidate Key
{C,D,F,G}+={C,D,F,G}
{C,E,F,G}+={C,E,F,G}
{D,E,F,G}+={D,E,F,G,B,A,C} - Candidate Key
ABFG , ACFG, BEFG, DEFG are the Candidate Keys
Checking whether any 5 attribute C.key exists or not
ABDFG - Super Key (ABFG is a C.Key)
ACDFG - Super Key (ACFF is a C.Key)
ADEFG - Super Key (DEFG is a C.Key)
ABEFG - Super Key (ABFG is a C.Key / BEFG is a C.Key)
ACEFG - Super Key
ADEFG - Super Key
BCEFG , BDEFG , CDEFG - Super Key

No 5 attribute C.keys exists!
Candidate keys are : ABFG , ACFG, BEFG, DEFG

The functional dependency AB → CD shows that R is not in BCNF because it violates the BCNF condition that every determinant must be a super key. In this case, AB is not a super key. Therefore, R is not in BCNF.

Now since we have to start the BCNF decomposition algorithm starting with AB.

R(ABCDEFG) - (AB->CD) =>R1(ABEFG) & R2(ABCD)
Now R1 : AB->E , BEG->A
R2 : AB->CD,B->D,AC->D
So again these FDs violate BCNF. Therefore again we have to apply BCNF decomposition Algo to R1 & R2. Now we can start with any one of the FDs.
So R1->R11(ABE),R12(ABFG)
R2->R21(ABC),R22(BD)

Final decomposition is R11(ABE), R12(ABFG) , R21(ABC), R22(BD).

Note: Multiple Final decomposition is possible.

And clearly it is not dependency preserving(these FDs are not satisfied : DEG->A,DE->B).