

Question 1

1. $\pi_{\text{appearsin.movieID}} [\sigma_{\text{actors.name} = \text{'Leonardo DiCaprio'}} \wedge \text{appearsin.actorID} = \text{actors.actorID} (\text{actors X appearsin})]$
2. $\pi_{\text{actors.name}} [(\sigma_{\text{appearsin.movieID} = \text{movies.movieID}} (\text{actors X appearsin})) \div (\pi_{\text{appearsin.movieID}} \sigma_{\text{actors.name} = \text{'Leonardo DiCaprio'}} \wedge \text{appearsin.movieID} = \text{movies.movieID} (\text{actors X appearsin}))]$
3. $\pi_{\text{actors.name}} [\sigma_{\text{actors.name} \neq \text{'Leonardo DiCaprio'}} \wedge \text{appearsin.actorID} = \text{actors.actorID} (\text{actors X appearsin})] \wedge [\text{Movie} \bowtie \pi_{\text{appearsin.movieID}} \sigma_{\text{actors.name} = \text{'Leonardo DiCaprio'}} \wedge \text{appearsin.movieID} = \text{movies.movieID} (\text{actors X appearsin})]$

Question 2

1. The minimal cover for F_m for F :

$A \rightarrow H$

$G \rightarrow A$

$E \rightarrow D$

$D \rightarrow G$

$E \rightarrow I$

$AB \rightarrow C$

$AB \rightarrow E$

$CD \rightarrow K$

2. The candidate keys are:

$\{J, B, A\}, \{J, B, D\}, \{J, B, E\}, \{J, B, G\}$

J needs to be present in every candidate key, because it's not available anywhere in the functional dependency (FD). Same goes with B, because it cannot be derived from any of the FD provided.

3. The decomposition is lossy.

Justification:

Let $R_1 = \{ABC\}$, $R_2 = \{DEKG\}$, $R_3 = \{HIJ\}$

$R_1 \cap R_2 \cap R_3$ has no common attributes. According to the lecture notes, the decomposition of $\{R_1, \dots, R_n\}$ of R is lossless if the common attributes of $R_1 \cap R_n$ form a superkey for either R_1 , or R_n .

By creating a table, not even one row has the same value in the same attribute

4. R is in 1NF, and it doesn't pass the requirements to be 2NF because it was found that at least 1 non-prime attribute (H) is not dependent on the whole of every candidate key. The attribute H only depends on A.

5. and 6.

R is in 3NF if all non-trivial if every non prime attribute is fully functionally dependent on the keys and not transitively dependent on any key.

In order to do that, we need to use our minimal cover, and merge the values with same left hand side as well as $J \rightarrow J$.

Therefore we have

$A \rightarrow H$

$G \rightarrow A$

$E \rightarrow D \mid I$

$D \rightarrow G$

$AB \rightarrow C \mid E$

$CD \rightarrow K$

$J \rightarrow J$

And lastly 3NF

R1: A H ($A \rightarrow H$, key A)

R2: G A ($G \rightarrow A$, key G)

R3: E D I ($E \rightarrow D$ and $E \rightarrow I$, key E)

R4: D G ($D \rightarrow G$ key D)

R5: A B C E ($AB \rightarrow C$ and $AB \rightarrow E$, key A, B)

R6: C D K ($CD \rightarrow K$, key C, D)

R7: J ($J \rightarrow J$, key J)

Question 3

Part 1


| | | | | |
|------------|---|---|---|---|
| Data Pages | 1 | 2 | 3 | 4 |
|------------|---|---|---|---|

| | |
|----|---|
| Q1 | 1 |
| Q2 | 2 |
| Q3 | 3 |
| Q4 | 4 |
| Q5 | 3 |
| Q6 | 4 |
| Q7 | 3 |
| Q8 | 4 |

| MRU | | |
|-----|---|---|
| 1 | | |
| 1 | 2 | |
| 1 | 2 | 3 |
| 1 | 2 | 4 |
| 1 | 2 | 3 |
| 1 | 2 | 4 |
| 1 | 2 | 3 |
| 1 | 2 | 4 |

| | | | |
|----|---|---|---|
| Q1 | 1 | | |
| Q2 | 1 | 2 | |
| Q3 | 1 | 2 | 3 |
| Q4 | 4 | 2 | 3 |
| Q5 | 4 | 2 | 3 |
| Q6 | 4 | 2 | 3 |
| Q7 | 4 | 2 | 3 |
| Q8 | 4 | 2 | 3 |

LRU outperforms MRU by far in this case.

 means the block that needs to be deleted in order to store a value from the next query

Question 3

Part 2


| | | | | | |
|------------|---|---|---|---|---|
| Data Pages | 1 | 2 | 3 | 4 | 5 |
|------------|---|---|---|---|---|

| | |
|-----|---|
| Q1 | 1 |
| Q2 | 2 |
| Q3 | 1 |
| Q4 | 3 |
| Q5 | 5 |
| Q6 | 1 |
| Q7 | 4 |
| Q8 | 5 |
| Q9 | 2 |
| Q10 | 1 |
| Q11 | 5 |
| Q12 | 1 |

| FIFO | | | |
|------|---|---|--|
| 1 | | | |
| 1 | 2 | | |
| 1 | 2 | | |
| 1 | 2 | 3 | |
| 5 | 2 | 3 | |
| 5 | 1 | 3 | |
| 5 | 1 | 4 | |
| 5 | 1 | 4 | |
| 2 | 1 | 4 | |
| 2 | 1 | 4 | |
| 2 | 5 | 4 | |
| 2 | 5 | 1 | |

| LRU | | | |
|-----|---|---|--|
| 1 | | | |
| 1 | 2 | | |
| 1 | 2 | | |
| 1 | 2 | 3 | |
| 1 | 5 | 3 | |
| 1 | 5 | 3 | |
| 1 | 5 | 4 | |
| 1 | 5 | 4 | |
| 2 | 5 | 4 | |
| 2 | 5 | 1 | |
| 2 | 5 | 1 | |
| 2 | 5 | 1 | |

In this case, FIFO needs 6 deletion to complete the sequence of queries 1 - 12, where LRU only needs 4 deletion.

 means the block that needs to be deleted in order to store a value from the next query