Name: Arukala Mounika

ID: A02236773

HW1

## Operators used:

> Selection: $\sigma$

> Projection: $\Pi$

> Intersection: $\cap$

> Join: $\bowtie$

> And: $\wedge$

> Grouping: $\mathfrak{I}$

> Division: /

## Temporary relations used:

- $X_1$, $X_2$, $Y_1$, $Y_2$, $Y_3$, $Y_4$, $Y_5$, $Y_6$, $Y_7$, $Y_8$

## Results are stored in:

- $X_3$
- $Y_9$

Q1) $\Pi_{Age}(Actor)$

Q2) $\Pi_{Title}(\sigma_{WhenReleased >= 1940 \wedge WhenReleased < 1950}(Movie))$

Q3.a) $\Pi_{Title}(\sigma_{Cost>1000000 \wedge WhenReleased <1920}(Movie))$

Q3.b) $X_1 := \Pi_{Title}(\sigma_{Cost>1000000}(Movie))$

$X_2 := \Pi_{Title}(\sigma_{WhenReleased <1920}(Movie))$

$X_3 := X_1 \cap X_2$

Q4) $X_1 := \Pi_{Name}(\sigma_{WhenReleased >=1940 \wedge WhenReleased <1950}(Movie \bowtie CastIn))$

$X_3 := \Pi_{Age}(Actor \bowtie X_1)$

Q5) $X_1 := \Pi_{Name}(\sigma_{WhenReleased <1920}(Movie \bowtie CastIn))$

$X_3 := \Pi_{Age}(Actor \bowtie X_1)$

Q6) $X_1 := \Pi_{Name}(CastIn)$

$X_3 := \Pi_{Name}(Actor) - X_1$

Q7) $X_1 := {}_{Name}\Im_{COUNT\ Title}(CastIn)$

$X_3 := \Pi_{Name}(\sigma_{COUNT\_Title >=2}(X_1))$

Q8) $X_1 := {}_{Name}\Im_{COUNT\ Title}(CastIn)$

$X_3 := \Pi_{Name}(\sigma_{COUNT\_Title =1}(X_1))$

Q9) $Y_1 := (Movie \bowtie CastIn)$

$Y_2 := \rho_X(Y_1)$ , $Y_3 := \rho_Y(Y_1)$

$Y_4 := \Pi_{Name,WhenReleased}(Y_2 \bowtie_{(X.Name=Y.Name \wedge Y.WhenReleased=X.WhenReleased+1)} Y_3)$

$Y_5 := \Pi_{Name' \leftarrow X.Name,\ WhenReleased' \leftarrow X.WhenReleased}(Y_4)$

$Y_6 := \rho_X(Y_5)$ , $Y_7 := \rho_Y(Y_5)$

$Y_8 := (Y_6 \bowtie_{(X.Name'=Y.Name' \wedge Y.WhenReleased'=X.WhenReleased'+1)} Y_7)$

$Y_9 := \Pi_{X.Name'} (Y_8)$

Q10) $X_1 := \Pi_{Title}(Movie)$

$X_3 := CastIn/X_1$

## Exercise 2.14

I strongly feel the **Three-Tier Client/Server Architecture** would be the best for a Web-based System for few of the reasons listed below:

- The Client consists of Web User Interface
- The Web Server consists the application logic which includes all the rules and regulations related to the reservation process and the issue of tickets
- The Database Server contains the DBMS

By considering the drawbacks of the other architectures listed in this exercise, I don’t feel they would fit for a web based system.

**Centralized DBMS Architecture** can never be used for a web-based system since the server will be present on the same system on which the client is working on. Hence, this wouldn’t work for a web-based system where it is very much required for the server to be on a different machine from that of client’s.

**Two-Tier Client-Server Architecture** wouldn’t work for a web-based system since the business logic will reside on the DBMS Server which would over burden the server. This may result in errors when the computations to be done are high in number. This issue would be solved if the Business Logic can reside on server other than the DBMS Server. On the other hand, if the business logic resides on the web client, it will burden the communication network.

## Exercise 3.12

(a) Firstly, create an empty tuple with the following attributes as follows:

INSERT<A,B,C,D,E,F>into SEAT_RESERVATION;

Later, modify the LEG_INSTANCE tuple with the condition:

(FLIGHT_NUMBER=A AND LEG_NUMBER=B AND DATE=C) by setting changing the number of available seats to one unit less as follows:

NUMBER_OF_AVAILABLE_SEATS = NUMBER_OF_AVAILABLE_SEATS - 1;

And then, the above describes operations should be repeated for each LEG of the flight on which a reservation is made.

***NOTE***: Here I assumed that the reservation is requested only for one seat. If someone requests for more than one, you need to develop more complex operations for a better reservation.

b) On each LEG_INSTANCE of the flight, check if the SEAT_NUMBER to be reserved in SEAT_RESERVATION is available or not and the NUMBER_OF_AVAILABLE_SEATS is greater than one or equal to (>=1).

c) Insert operation into SEAT_RESERVATION will check the NUMBER_OF_AVAILABLE_SEATS on each LEG_INSTANCE of the flight and is greater than 1 is a **general semantic integrity constraint**, which doesn’t satisfy key, entity integrity and referential integrity constraints for the relation.

d)

>From the table, LEG_INSTANCE the attributes “FLIGHT_NUMBER, LEG_NUMBER” acts as the foreign keys to the relation FLIGHT_LEG.

>From the table, LEG_INSTANCE the attribute “DEPARTURE_AIRPORT_CODE” acts as the foreign key to the relation AIRPORT.

>From the table, LEG_INSTANCE the attribute “ARRIVAL_AIRPORT_CODE” acts as the foreign key to the relation AIRPORT.

>From the table, LEG_INSTANCE the attribute “AIRPLANE_ID” acts as the foreign key to the relation AIRPLANE.

>From the table, FLIGHT_LEG the attribute “FLIGHT_NUMBER” acts as the foreign key to the relation FLIGHT.

>From the table, FLIGHT_LEG the attribute “DEPARTURE_AIRPORT_CODE” acts as the foreign key to the relation AIRPORT.

>From the table, FLIGHT_LEG the attribute “ARRIVAL_AIRPORT_CODE” acts as the foreign key to the relation AIRPORT.

>From the table, CAN_LAND the attribute “AIRPLANE_TYPE_NAME” acts as the foreign key to the relation AIRPLANE_TYPE.

>From the table, CAN_LAND the attribute “AIRPORT_CODE” acts as the foreign key to the relation AIRPORT.

>From the table, SEAT_RESERVATION the attributes “FLIGHT_NUMBER, LEG_NUMBER, DATE” acts as the foreign keys to the relation LEG_INSTANCE.

>From the table, FARES the attribute “FLIGHT_NUMBER” acts as the foreign key to the relation FLIGHT.

>From the table, AIRPLANE the attribute “AIRPLANE_TYPE” acts as the foreign key to the relation AIRPLANE_TYPE.

**Exercise 3.14**

1) From the table, ORDER the attribute “Cust#” acts as the foreign key to the relation CUSTOMER.
2) From the table, ORDER_ITEM the attribute “Order#” acts as the foreign key to the relation ORDER.
3) From the table, ORDER_ITEM the attribute “Item#” acts as the foreign key to the relation ITEM.
4) From the table, SHIPMENT the attribute “Order#” acts as the foreign key to the relation ORDER.
5) From the table, SHIPMENT the attribute “Warehouse#” acts as the foreign key to the relation WAREHOUSE.