Atal Bihari Vajpayee

Indian Institute of Information Technology

And Management (ABV-IIITM), Gwalior

DBMS PROJECT

Topic: Movie Theatre Database Management

Instructor Name: Dr. Debanjan Sadhya

Group Members:

Anurag Srivastava (2018IMT-019)

Mehul Jain (2018IMT-051)

Parth Arora (2018IMT-063)

Rishab Singla (2018IMT-082)

Sayantan Banerjee (2018IMT-093)

Vinay Pratap (2018IMT-105)

INTRODUCTION

Multiplex is a movie theatre complex with multiple screens within a single complex. They are usually housed in a specially designed building.

- Movie Theatre Database System is an application of database management system which provides the user to book movie tickets and manager to keep their records.

- Movie Theatre Database system revolutionized the trivial system of distribution of Paper tickets and made it easy for users to book tickets from home.

- This system provides secure, organized and efficient storage of huge data. One does not have to be physically present at the theatre to get their jobs done.

- With the advent of various online platform like BookMyShow and PayTm use type of movie theatre database system, the process became very simple, hustle free and user-friendly.

- This Project aims to provide an insight of such Movie Theatre Database System.

ENTITY SET

AND ITS ATTRIBUTES

An entity set is a set of entities of the same type. Entity sets need not be disjoint. An entity is represented by a set of attributes.

We will be dealing with seven entity sets in this section:

- **People:** It is the set of all the people who book the tickets. Each Person is described by People ID, Name, Age and phone number.
  Attributes : people_id, Name, Age, phone_number.

- **Movie:** This entity stores the details of movies to be shown on the screen in the theatre. Each movie is described by its meta-data Movie Id, movie title, movie director, movie length and movie genre.
  Attributes: movie_id, movie_title, movie_director, movie_length, movie_genre.

- **Booking:** Booking stores data of booked tickets containing details of price, booking Id.
  Attributes: booking_id, price.

- **Screening:** It is the combined data of the Movie and hall in which it is shown. It describes screening id, screen time, screen data.
  Attributes: screening_ID, screen_time, screen_date.

- **Hall:** This entity stores the data of hall and seats combined described by hall id, number of seats and hall name.
  Attributes: hall_ID, no_of_seats, hall_name.

- **Seat Reserved:** Based on the booking, this entity stores the data of seat reserve id.
  Attributes: seat_Reserv_ID.

- **Seat:** This entity contains the information of seat in a hall and is described by seat id, seat row and seat number.
  Attributes: seat_ID, seat_row, seat_number.

RELATIONSHIP SET

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- N-ary = degree n

We will be dealing with 8 relationship set :

- **MADE:** This set establishes relationship between People and Booking entities with one to many relationship between them. By this relationship using people_ID attribute (as foreign key) of booking one can find details of the person who has booked ticket.

- **OF:** Relationship between Booking and Seat Reserved entities is one to one relationship, as one booking can only confirm one seat. Seat Reserved entity contains booking_ID as foreign key to get details of booking.

- **TO WATCH:** It is the relationship between Movie and Screening entities. It has one to many relationship as one movie can be shown at different screens but a screen cannot show more than one movie at a given time. Movie_ID is kept as foreign key in screening entity.

- **ON:** It is the relationship between Screening and Hall and has many to one relationship as many screenings can be presented in one hall at different times and dates. Hall_ID is kept as foreign key in Screening entity.

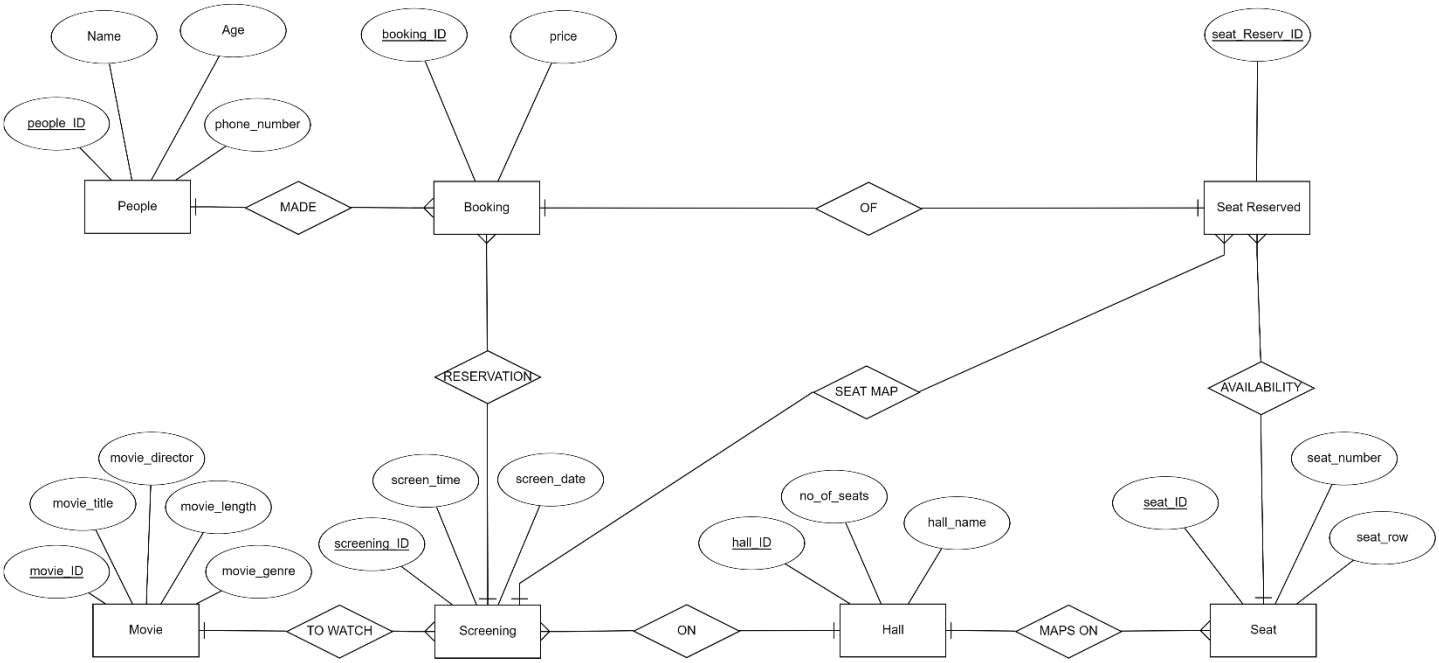
- **MAPS ON:** It maps relationship between Hall and Seat entity and has one to many relationship as one hall can have many seats. Seat contains Hall_ID as foreign key.

- **AVAILABILITY:** It is the relationship between Seat Reserved and Seat entity. It has many to one relationship as there can be multiple reservations on different times and dates of a particular seat (not more than one reservation of a seat at a given time and date). Seat Reserved entity has Seat_ID as foreign key.

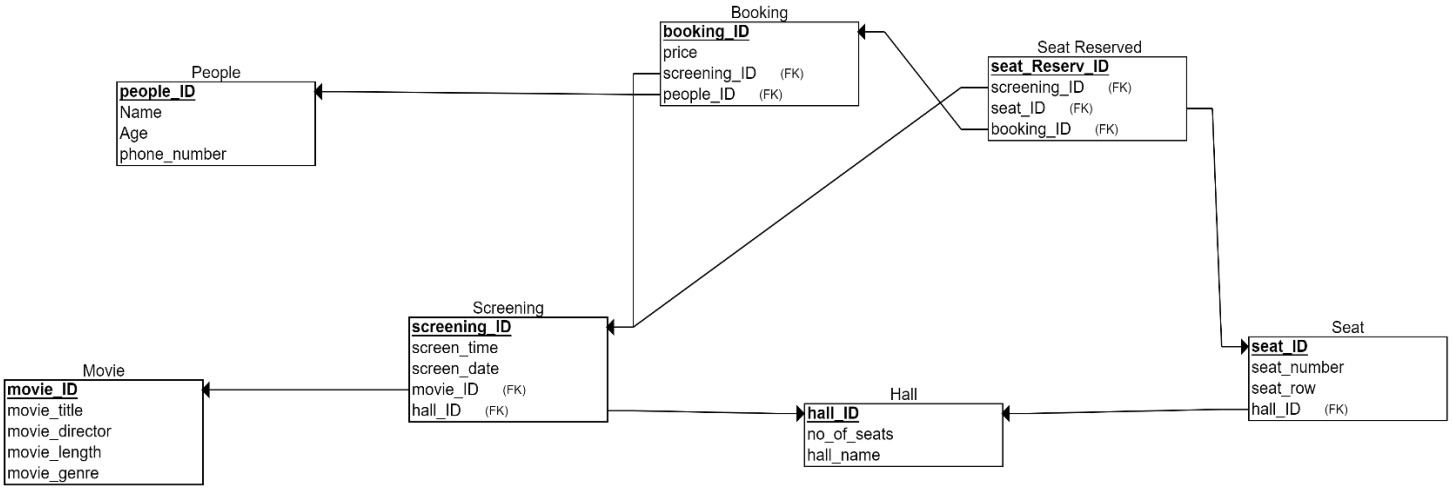
- **SEAT MAP:** It is the relationship between Screening and Seat Reserved entities. It has one to many relationship as there are multiple seats reserved for a screening. Seat reserved entity has screening_ID as foreign Key.

- **RESERVATION:** It establishes relationship between Screening and Booking entities. It is one to many relationship. One screening can have multiple booking by people. Screening_ID is foreign key in Booking entity.

ER-DIAGRAM



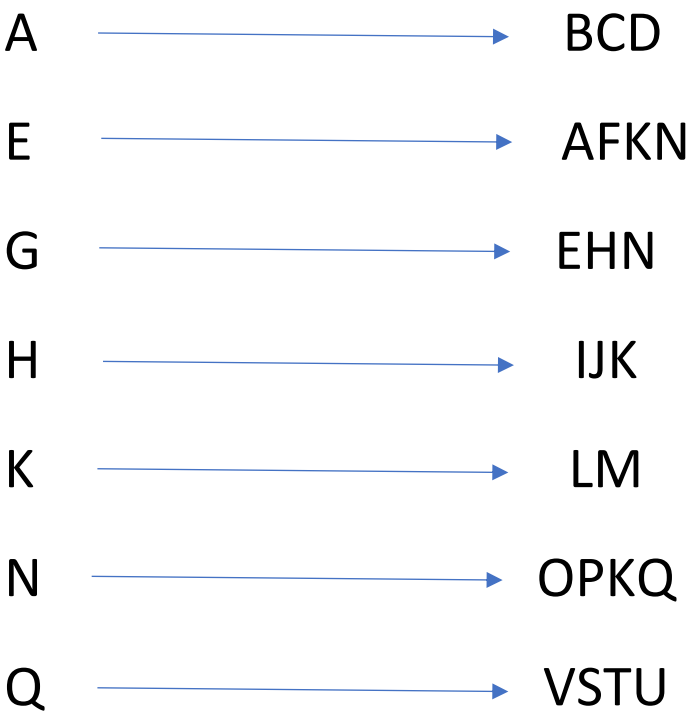
RELATIONAL SCHEMA



NORMALIZATION AND FUNCTIONAL DEPENDENCIES

Attributes	Represented As
people_ID	A
Name	B
Age	C
phone_number	D
booking_ID	E
price	F
seat_Reserv_ID	G
seat_ID	H
seat_number	I
seat_row	J
hall_ID	K
no_of_seats	L
hall_name	M
screening_ID	N
screen_time	O
screen_date	P
movie_ID	Q
movie_title	V
movie_director	S
movie_length	T
movie_genre	U

All the functional dependencies present in this database(R is relation) are as follows:



And it can be clearly stated from all the functional dependencies that the closure of G, that is

$(G)^+ = R$.

As there are no multivalued attribute in our database hence the given relation is in 1NF Form.

Now we can clearly see that there are no partial dependencies in our Relation R as partial dependencies occur when a subset of candidate key is capable to derive other non-prime attributes in the relation. But as in our Relation, there is only one attribute

hence there cannot be any proper subset of the candidate key capable to derive other attribute.

Hence this proves that our table is in 2NF Form.

Now, we can clearly notice from the dependencies that there are transitive dependencies present(as E,H and N being non-prime derives another non-prime attributes), hence, the table is not in 3NF Form.

After breaking the functional dependencies such that there are no non-prime to non-prime dependencies(transitive dependencies) table would come in 3NF Form and the functional dependencies now looks like:

G → EHN ...(i)

E → AFN ...(ii)

A → BCD ...(iii)

N → OPKQ ...(iv)

K → LM ...(v)

Q → VSTU ...(vi)

H → IJK ...(vii)

Now, in the above dependencies we can clearly notice that there are no dependencies that corresponds to a prime attribute (prime attribute is a proper subset of candidate key, in our case

only G is a prime attribute), hence it is safe to mention that our Relation is now in BCNF Form.

Here, the functional dependencies after doing normalization till BCNF Form corresponding to the table present in our database are as followed:

Functional Dependency (i) corresponds to table Seat Reserved

Functional Dependency (ii) corresponds to table Booking

Functional Dependency (iii) corresponds to table People

Functional Dependency (iv) corresponds to table Screening

Functional Dependency (v) corresponds to table Hall

Functional Dependency (vi) corresponds to table Movie

Functional Dependency (vii) corresponds to table Seat .

ENTITY RECORDS

1. **MOVIE TABLE**:

Below is the SQL command to Create Movie Table:

```
CREATE TABLE Movie
(
  movie_ID VARCHAR(5) NOT NULL,
  movie_title VARCHAR(20) NOT NULL,
  movie_director VARCHAR(20) NOT NULL,
  movie_length INT NOT NULL,
  movie_genre VARCHAR(15) NOT NULL,
  PRIMARY KEY (movie_ID)
);
```

Sample Record of Movie Table:

movie_ID	movie_title	movie_director	movie_length	movie_genre
M1	Avengers Endgame	Russo Brothers	181	Superhero
M2	Avatar	James Cameron	162	Sci-Fi
M3	Titanic	James Cameron	195	Drama

2. **HALL TABLE**:

SQL command to create Hall table:

```
CREATE TABLE Hall
(
  hall_ID VARCHAR(4) NOT NULL,
  no_of_seats INT NOT NULL,
  hall_name VARCHAR(10) NOT NULL,
  PRIMARY KEY (hall_ID)
);
```

Sample Record of Hall Table:

hall_ID	no_of_seats	hall_name
H1	8	Platinum
H2	6	Executive
H3	4	Royal

3. PEOPLE TABLE:

SQL command to create People Table:

```
CREATE TABLE People
(
   people_ID VARCHAR(10) NOT NULL,
   Name VARCHAR(20) NOT NULL,
   Age INT NOT NULL,
   phone_number CHAR(8) NOT NULL,
   PRIMARY KEY (people_ID)
);
```

Sample Record of People Table:

people_ID	Name	Age	phone_number
P1	Anurag	19	12345678
P2	Mehul	18	15345278
P3	Parth	19	12045667
P4	Rishab	18	27345178
P5	Sayantan	19	92345608
P6	Vinay	21	87945678

4. SCREENING TABLE:

SQL command to create Screening Table:

```
CREATE TABLE Screening
(
  screening_ID VARCHAR(10) NOT NULL,
  movie_ID VARCHAR(5) NOT NULL,
  hall_ID VARCHAR(4) NOT NULL,
  screen_time TIME NOT NULL,
  screen_date DATE NOT NULL,
  PRIMARY KEY (screening_ID),
  FOREIGN KEY (movie_ID) REFERENCES Movie(movie_ID),
  FOREIGN KEY (hall_ID) REFERENCES Hall(hall_ID)
);
```

Sample Record of Screening Table:

screening_ID	movie_ID	hall_ID	screen_time	screen_date
SCR1	M1	H3	17:35:00	2019-10-05
SCR2	M1	H3	17:35:00	2019-10-06
SCR3	M2	H2	17:35:00	2019-10-05
SCR4	M3	H1	09:45:00	2019-10-06

5. SEAT TABLE:

SQL command to create Seat Table:

```
CREATE TABLE Seat
(
  seat_ID VARCHAR(7) NOT NULL,
  seat_number INT NOT NULL,
  seat_row CHAR(1) NOT NULL,
  hall_ID VARCHAR(4) NOT NULL,
  PRIMARY KEY (seat_ID),
  FOREIGN KEY (hall_ID) REFERENCES Hall(hall_ID)
);
```

Sample Record of Seat Table:

seat_ID	seat_number	seat_row	hall_ID
H1S1	1	A	H1
H1S2	2	A	H1
H1S3	1	B	H1
H1S4	2	B	H1
H1S5	1	C	H1
H1S6	2	C	H1
H1S7	1	D	H1
H1S8	2	D	H1
H2S1	1	A	H2
H2S2	2	A	H2
H2S3	3	A	H2
H2S4	1	B	H2
H2S5	2	B	H2
H2S6	3	B	H2
H3S1	1	R	H3
H3S2	2	R	H3
H3S3	3	R	H3
H3S4	4	R	H3

6. BOOKING TABLE:

SQL command to create Booking Table:

```
CREATE TABLE Booking
(
  booking_ID VARCHAR(6) NOT NULL,
  price INT NOT NULL,
  screening_ID VARCHAR(10) NOT NULL,
  people_ID VARCHAR(10) NOT NULL,
  PRIMARY KEY (booking_ID),
  FOREIGN KEY (screening_ID) REFERENCES Screening(screening_ID),
  FOREIGN KEY (people_ID) REFERENCES People(people_ID)
);
```

Sample Record of Booking Table:

booking_ID	price	screening_ID	people_ID
B1	1160	SCR1	P5
B2	150	SCR4	P1
B3	300	SCR3	P4
B4	1160	SCR2	P2
B5	150	SCR4	P3
B6	300	SCR3	P6

7. SEAT RESERVED:

SQL command to create Seat Reserved:

```
CREATE TABLE Seat_Reserved
(
  seat_Reserv_ID VARCHAR(6) NOT NULL,
  screening_ID VARCHAR(10) NOT NULL,
  seat_ID VARCHAR(7) NOT NULL,
  booking_ID VARCHAR(6) NOT NULL,
  PRIMARY KEY (seat_Reserv_ID),
  FOREIGN KEY (screening_ID) REFERENCES Screening(screening_ID),
  FOREIGN KEY (seat_ID) REFERENCES Seat(seat_ID),
  FOREIGN KEY (booking_ID) REFERENCES Booking(booking_ID)
);
```

Sample Record of Seat Reserved:

seat_Reserv_ID	screening_ID	seat_ID	booking_ID
SR1	SCR1	H3S3	B1
SR2	SCR4	H1S6	B2
SR3	SCR3	H2S1	B3
SR4	SCR2	H3S3	B4
SR5	SCR4	H1S1	B5
SR6	SCR3	H2S2	B6

SQL QUERIES

- Which movies are screened on 2019-10-06?

Syntax:

```
SELECT movie_title, movie_director, movie_length, movie_genre , hall_ID
FROM Movie
INNER JOIN Screening
ON Screening.movie_ID = Movie.movie_ID
WHERE screen_date = "2019-10-06";
```

Output:

movie_title	movie_director	movie_length	movie_genre	hall_ID
Avengers Endgame	Russo Brothers	181	Superhero	H3
Titanic	James Cameron	195	Drama	H1

- Show seat map of hall Royal.

Syntax:

```
SELECT seat_ID,seat_row, seat_number
FROM Seat
INNER JOIN Hall
ON Seat.hall_ID = Hall.hall_ID
WHERE hall_name = "Royal";
```

Output:

seat_ID	seat_row	seat_number
H3S1	R	1
H3S2	R	2
H3S3	R	3
H3S4	R	4

- Show the date, time and hall when the film “Avengers Endgame” will be screened

Syntax:

```
SELECT movie_title, hall_ID, screen_date, screen_time
FROM Screening
INNER JOIN Movie
ON Screening.movie_ID = Movie.movie_ID
WHERE movie_title = "Avengers Endgame";
```

Output:

movie_title	hall_ID	screen_date	screen_time
Avengers Endgame	H3	2019-10-05	17:35:00
Avengers Endgame	H3	2019-10-06	17:35:00

- Show all the details of people who booked for the film “Avengers Endgame”

Syntax:

```
SELECT Name, Age, phone_number
FROM People
INNER JOIN Booking
ON People.people_ID = Booking.people_ID
INNER JOIN Screening
ON Screening.screening_ID = Booking.screening_ID
INNER JOIN Movie
ON Screening.movie_ID = Movie.movie_ID
WHERE movie_title = "Avengers Endgame";
```

Output:

Name	Age	phone_number
Sayantan	19	92345608
Mehul	18	15345278

- Show details of all the seats booked at hall 2 on date 2019-10-05:-

Syntax:

```
SELECT Seat.hall_ID, movie_title , screen_time , seat_row, seat_number
FROM Seat
INNER JOIN Seat_Reserved
ON Seat.seat_ID = Seat_Reserved.seat_ID
INNER JOIN Screening
ON Screening.screening_ID = Seat_Reserved.screening_ID
INNER JOIN Movie
ON Screening.movie_ID = Movie.movie_ID
WHERE screen_date = "2019-10-05" AND Seat.hall_ID = "H2";
```

Output:

hall_ID	movie_title	screen_time	seat_row	seat_number
H2	Avatar	17:35:00	A	1
H2	Avatar	17:35:00	A	2

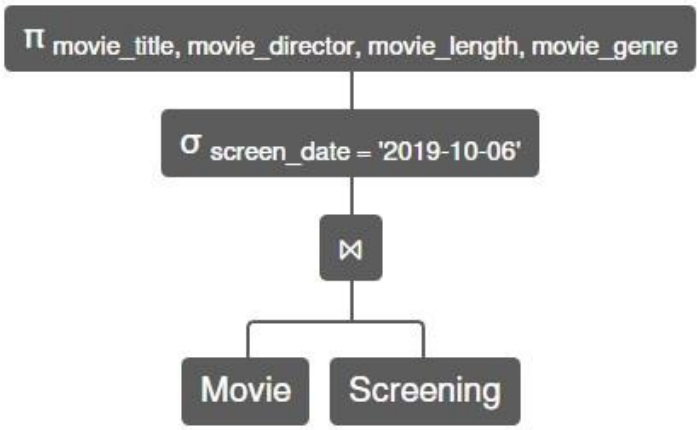
RELATIONAL ALGEBRA QUERIES

- Which movies are screened on 2019-10-06:-

  Syntax:

  π movie_title , movie_director, movie_length, movie_genre (σ screen_date = '2019-10-06' (Movie $\bowtie$ Screening))

  Output:



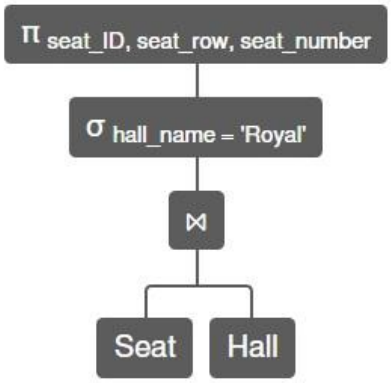
Movie.movie_title	Movie.movie_director	Movie.movie_length	Movie.movie_genre
Avengers Endgame	Russo Brothers	181	Superhero
Titanic	James Cameron	195	Drama

- Show seat map of hall Royal:-

  Syntax:

  $\pi_{\text{seat\_ID, seat\_row, seat\_number}} (\sigma_{\text{hall\_name = 'Royal'}} (\text{Seat} \bowtie \text{Hall}))$

  Output:



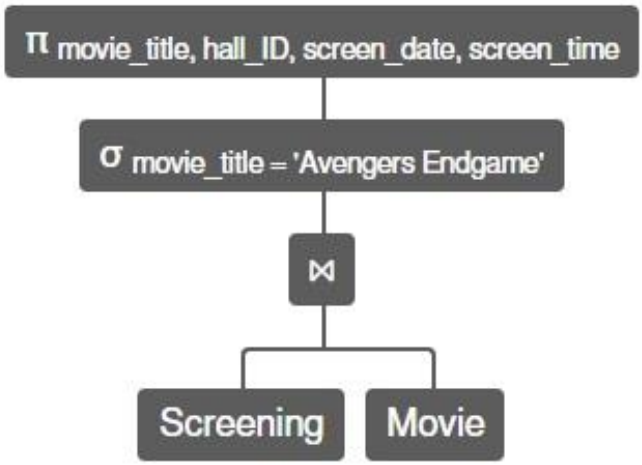
Seat.seat_ID	Seat.seat_row	Seat.seat_number
H3S1	R	1
H3S2	R	2
H3S3	R	3
H3S4	R	4

- Show the date, time and hall when the film “Avengers Endgame” will be screened:-

  Syntax:

  π movie_title, hall_ID, screen_date, screen_time (σ movie_title = 'Avengers Endgame' (Screening $\bowtie$ Movie))

  Output:



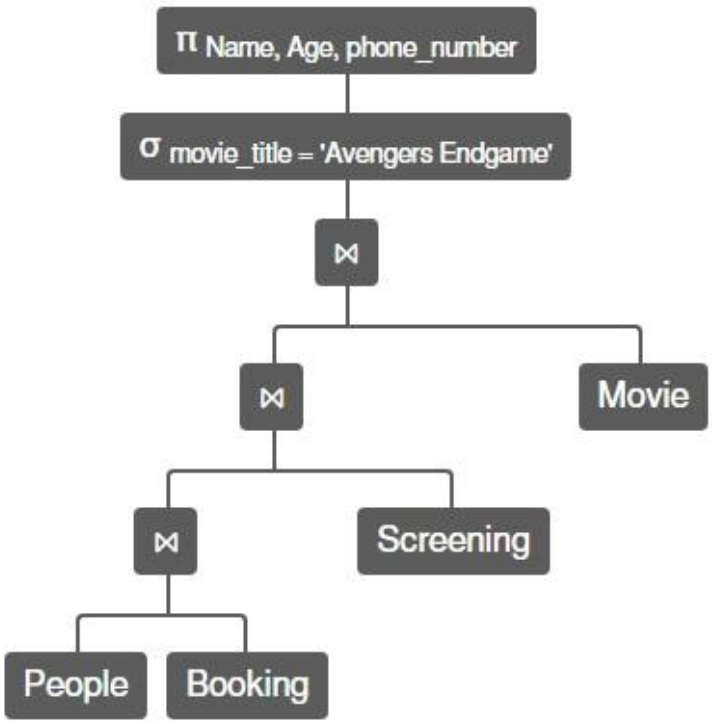
Movie.movie_title	Screening.hall_ID	Screening.screen_date	Screening.screen_time
Avengers Endgame	H3	2019-10-05	17:35:00
Avengers Endgame	H3	2019-10-06	17:35:00

- Show all the details of people who booked for the film “Avengers Endgame”:-

  Syntax:

  π Name, Age, phone_number (σ movie_title = 'Avengers Endgame' (People $\bowtie$ Booking $\bowtie$ Screening $\bowtie$ Movie))

  Output:



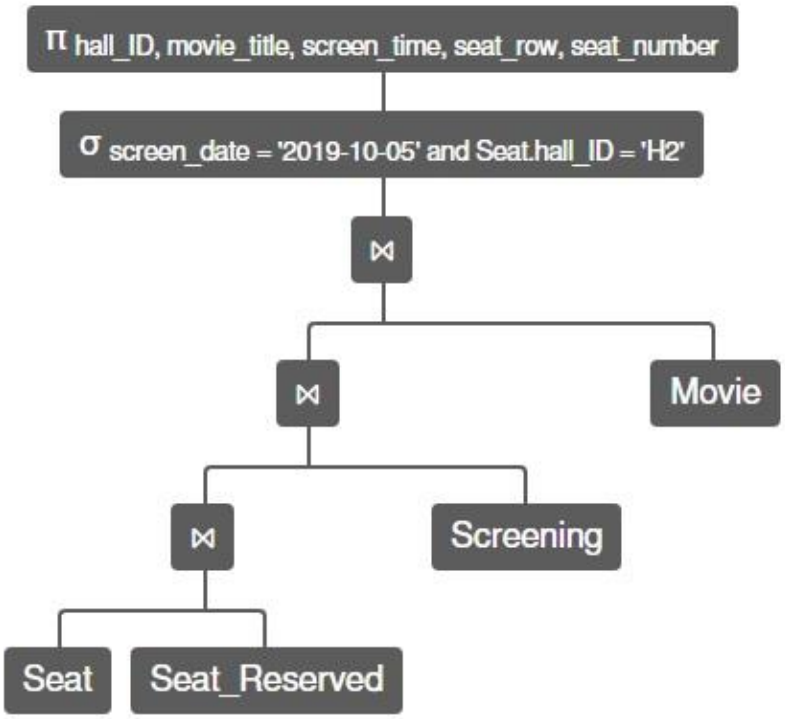
People.Name	People.Age	People.phone_number
Mehul	18	15345278
Sayantan	19	92345608

- Show details of all the seats booked at hall 2 on date 2019-10-05:-

  Syntax:

  π hall_ID, movie_title , screen_time , seat_row, seat_number (σ screen_date = '2019-10-05' $\wedge$ Seat.hall_ID = 'H2' (Seat $\bowtie$ Seat_Reserved $\bowtie$ Screening $\bowtie$ Movie))

  Output:



Seat.hall_ID	Movie.movie_title	Screening.screen_time	Seat.seat_row	Seat.seat_number
H2	Avatar	17:35:00	A	1
H2	Avatar	17:35:00	A	2