Practical No. 1

Aim: Given the following relation schemas from part of an Accommodation database held in Relational DBMS:

HOTEL (hotelNo, hotelName, city) with primary key {hotelNo}

ROOM (roomNo, hotelNo, type, price) with the primary key {roomNo, hotelNo}

GUEST (guestNo, guestName, guestAddress) with primary key {guestNo}

BOOKING (hotelNo, guestNo, dateFrom, dateTo, roomNo) with primary key {hotelNo, guestNo, dateFrom}

Additional constraints:

(1) There cannot be two bookings for the same room in the same hotel on the same day.
(2) For every booking, date From must be specified and must be before dateTo, if specifed.
(3) A booking cannot be in joint names. In other words a booking can only be held in the name of one guest.
(4) A guest cannot book two different rooms in the same or different hotels for the same period.
(5) dateTo may not be specified in some bookings.

Problem Statements:

1. Find out alternate keys for BOOKING if any.

Solution:
As per the definition of Alternate Keys the values of the attributes in the alternate keys have to be unique even in the case of foreign attribute they must have value which is unique in their respective tables.
The given primary key for BOOKING already has attributes of hotelNo, guestNo and dateFrom where individual attributes of a primary key itself are not unique.
As far as attribute of dateTo is considered it can have NULL values and/or duplicate values.
And as per the constraint a room cannot be shared by two guests' name hence for a given hotelNo and roomNo and dateFrom we can for another Primary key: {hotelNo, roomNo, dateFrom} where guestNo can be replaced by roomNo as twos guests' name can never have same room booked.
Apart from that we have supersets of the given Primary Key and above mentioned Primary Key but then they are not candidate keys hence not Alternate Keys.

2. Identify foreign keys, if any, in Hotel, Room, Booking and Guest relations.

Solution:

HOTEL: No foreign keys.
ROOM: hotelNo is a foreign key from HOTEL relation.
BOOKING: hotelNo from HOTEL, guestNo from GUEST and roomNo from ROOM are the foreign keys.
GUEST: No foreign keys.

3. Is it possible to make a booking in the Accommodation database in the name of a person who is not listed in the Guest relation? Give reasons for your answer.

Solution:

The relation BOOKING has a primary key {hotelNo, guestNo, dateFrom} where the guestNo is a foreign key from relation GUEST.

The foreign key guestNo can be inherited into BOOKING relation only if it present in the GUEST relation.

Therefore if the guest's name is not in the list that means not in the GUEST table then it cannot be in BOOKING relation and hence booking of such unlisted person is impossible.

4. Is it possible to have same room numbers in two different hotels of the Accommodation database? Justify your answer.

Solution:

Logically it is both possible to have same roomNo in two different hotels and same goes with the Accommodation Database as the relation ROOM has Primary Key: {roomNo, hotelNo} which means that roomNo is not a uniquely valued attribute and hotelNo is because hotelNo is primary key for HOTEL individually which means it has unique values thus we can conclude that there can me same room numbers in different hotels but not in same hotels.

5. Suppose that a guest (`G1', `Smith', `14 Jickell Street') in the relation Guest has made several bookings stored in the relation Booking. When deleting the record of this guest from Guest, all the bookings he made should also be deleted. How would you ensure that this can happen when applying the SQL statement DELETE FROM GuestWHERE guestNo=`G1'?

Solution:

As the demand is to delete all the records from BOOKING made in the name of guest with guestNo - 'G1' so the attribute guestNo (which is primary key of GUEST table) is sufficiently unique to search the specific records in the BOOKING relation and we can directly use the SQL query DELTE FROM Booking WHERE guestNo="G1" along with the query DELETE FROM GuestWHERE guestNo=`G1' so that all the records of the guest 'G1' will be delete from both GUEST as well as BOOKING table.

Conclusion: Thus, from the given relation schemas from part of an Accommodation database held in a Relational DBMS, we have performed the given operations.

Practical No. 2

Aim:

Dream Ride is an airline company that supplies on-demand charter fight services to people (customers) and uses a fleet of ten aircrafts. The data held about each customer is the customer identification number, customer name, customer initial, telephone identification number, balance, address. For the mechanical service of aircrafts the company has formed five service teams and each service team is made up of three service engineers. The employees of Dream Ride are classified into four categories namely, pilot, service engineer, flight attendant and administration staff. The data held for each employee includes employee identification number, first name, last name, date of birth, salary, telephone number, and address. For pilots some additional data is also recorded namely, license number, pilot rating and pilot check up date. The data held on aircrafts is aircraft registration number, model code, model name, number of seats, mileage, engine strength, and manufacturer. The data held on each aircraft's mechanical service is type of service, date of service, time of service, service detail sand comments. There are a dozen of specific paths for which a customer can charter an aircraft. The data held on each path is path identification number, path duration, cost and path description. The company maintains record of each charter trip. The data held on each charter trip is the date of trip, start time, customer identification number, path identification number and aircraft registration identification number. Some of the compulsory business assumptions are as follows:

(i) A service team can perform mechanical services on any of the aircrafts.
(ii) An aircraft can be chartered by only one customer for a specific charter trip.
(iii) Each charter trip involves just one path.
(iv) All aircrafts need a pilot and co-pilot for flying purposes.

Problem Statements:

1. Identify the entities of the DreamRide airline.

Solution:

Entities in DreamRide database are
Customer, Employee, Pilot, Aircraft, Aircraft_Service, Path and Trip.

2. Identify the relationships between the entities described above. Determine and represent the constraints for each relationship.

Solution:

Aircraft→PilotDuty←Pilot: A pilot and co-pilot are assigned an aircraft to form Duty relation where the LicenseNo of each and AircraftRegNo for the primary key, with additional attribute of Date, Start Time which are from Trip relation but Trip is not related to Duty but is similar to it.

Pilot→ISA←Employee: A pilot is an employee and hence it inherits all the attributes of Employee.

Service_Team derived from Employee table with some added attributes where we have attributes of TeamNo, TeamAssignDate and multivalued attribute TeamEngg, which has 3 employee IDs ‘ENo’.

Aircraft and Aircraft_Service→Maintenance_Schedule←Service_Team: This relationship which states that which aircraft is being serviced, its Aircraft_Service details clubbed together with the Team who is servicing it.

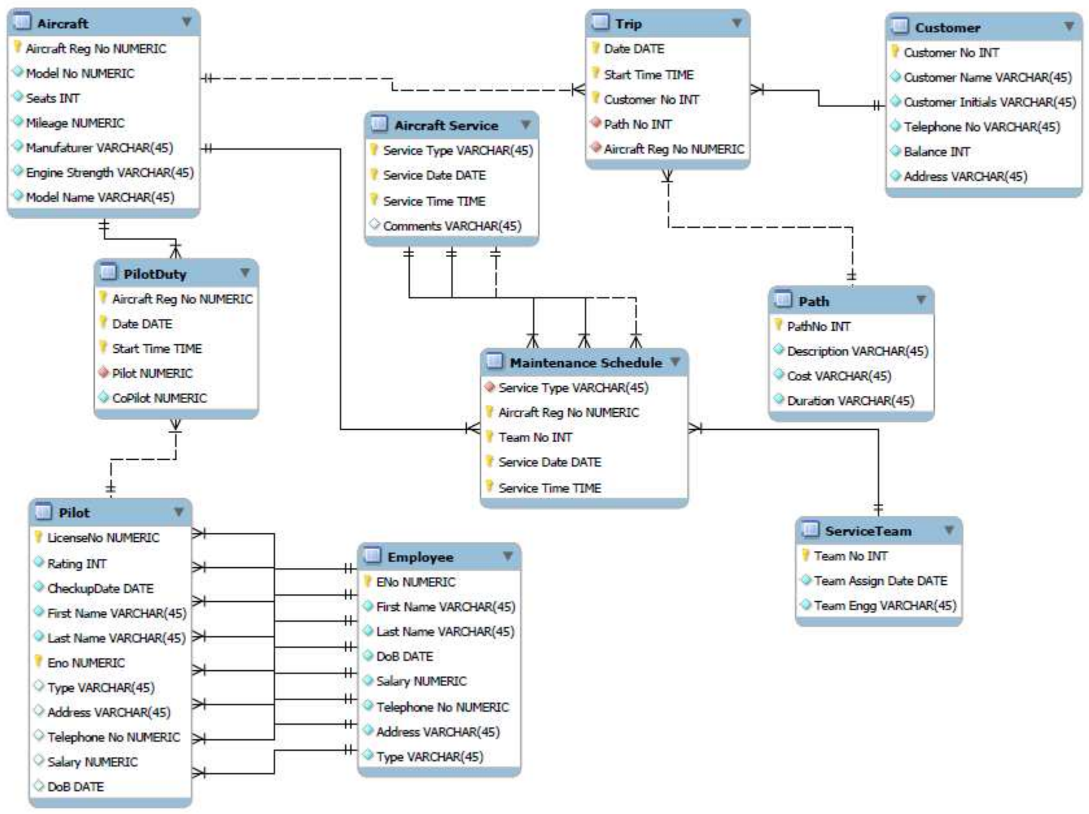
3. Identify attributes and associate them with the entities or relationships. Determine the key attributes for each entity.

Solution:

List of Entities/Relations/Relationships and Attributes associated with them underlined being the Primary Keys:

Customer {CusNo, CusName, CusInitials, TeleNo, Balance, Address}
Employee {ENo, FirstName, LastName, DoB, Salary, TeleNo, Address, Type}
Pilot {LicenseNo, Rating, CheckupDate, ENo (Optional, as it is redundant)}
Aircraft {AircraftRegNo, ModelNo, ModelName, Seats, Mileage,Engine Strength, Manufaturer}
Aircraft_Service {SType, Date, Time, Comments}
Path {PathNo, Duration, Cost, Description}
Trip {Date, StartTime, CusNo, PathNo, AircraftRegNo}
PilotDuty {AircraftRegNo, Date, StartTime, Pilot (ENo/LicenseNo), CoPilot (ENo/LicenseNo)}
Service_Team{TeamNo, TeamAssignDate, TeamEngg (ENo) }
Maintenance_Schedule {TeamNo, Date, Time, AircraftRegNo, SType}

4. ER Diagram



Conclusion: Thus, from the given database of DreamRide airline Company, we have performed the given operations.

Practical No: 3

Aim: To study the Functional Dependencies and various Anomalies, which can be occur in given Database.

Problem Statement:
Consider the following relation Appointment held in a relational database for the Canberra Medical Centre, which has the primary key { DoctorID, PatientNo, Time}.

DoctorID	DoctorName	PatientNo	PatientName	Time	Room
D10	Peter Raman	P1	Lisa Hill	12/11/2011 10 AM	R1
D11	Sue Smith	P3	Mark Dunn	12/11/2011 10 AM	R2
D10	Peter Raman	P2	Jill Khan	12/11/2011 1 PM	R1
D11	Sue Smith	P3	Mark Dunn	14/11/2011 2 PM	R2
D12	Robin Duong	P2	Jill Khan	14/11/2011 4 PM	R3

Q1. What update anomalies may occur in the above relation?
Solution:

Firstly the DoctorID gives us DoctorName and Room hence every time we update DoctorID we have to update DoctorName and Room and this data is merely duplicated without any necessity.
The same applies to PatientNo and PatientName, PatientNo being unique gives PatientName and hence every time a PatientNo is updated we need to make changes to PatientName every time a patient has appointment with Doctor.

Q2. Give an example of a functional dependency which is violated in the given relation. Explain why the functional dependency is violated.
Solution:

The Primary Key: DoctorID, PatientNo, Time → PatientName, DoctorName, Room is the only given FD. The other FD which we can figure out is PatientNo,Time → Everything Else and same with DoctorID, Time → Everything Else, because at a time a single doctor treats a single patient and also a patient is treated by single doctor. So we can say that given PK is not minimal superkey and we can give that DoctorID,Time and PatientNo, Time are 2 candidate keys.

Q3. What is a minimal change that you can make to your functional dependency in 2 such that the resulting functional dependency is non-trivial and satisfied in the given sample relation?

Solution:

Instead of the given primary key DoctorID, PatientNo, Time we can drop either DoctorID or PatientNo from the attributes.
Alternatively we can also have Time, Room as a Primary key because they are sufficient to determine all other attributes unless there are some more specified constraints.

Conclusion: Thus, from given database, we have studied the Functional Dependencies and various Anomalies which can be occur in given database.

Practical No: 4

Aim: To study Functional Dependencies in detail

Problem Statement:

Consider a relation schema R(A,B,C,D,E) with the following set ∑ of functional dependencies:

BC → E, A → BE and ED → A.

Q1. List all candidate keys of R.

Solution:

In the above relation the C and D are independent determinants.

BC → E and ED → A therefore BCD → ABCDE
ED → A and A → BE therefore ECD → ABCDE
A → BE and C & D are independent determinants therefore ACD → ABCDE
Thus we have 3 candidate keys as above.

Q2. Does AE → D hold on any relation of R that satisfies ∑? If so, explain why; otherwise give a counter example.

Solution:

The FD, AE → D does not hold on any relation in R that satisfies ∑ because the attribute D is prime attribute and does not appear on the right hand side, i.e. it is indeterminate.
We can consider an FD say ED → B which holds true because ED→A and A→BE, hence A→ B and by transitive rule ED → B.

Conclusion: Thus, from given relation schema, we have studied Functional Dependencies in detail.

Practical No: 5

Aim: To study the Normalisation and its implementation of given database.

Problem Statement:

Consider the relation Classtimetable with the attributes: **{CourseCode, CourseName, LecturerID, LecturerName, Day, Time, Room, SemesterYear**} and the primary key **{CourseCode, Day, Time, SemesterYear}**.The relation Classtimetable stores data about courses,lectures and classes. A coursecode (CourseCode) determines its course name (CourseName) that may not be unique. Eachlecturer ID (LecturerID) can determine the lecturer's name (LecturerName) that may not be unique. The data stored about classes include class day (Day), class time (Time), class room(Room), and semester and year (SemesterYear) treated as an atomic attribute. A class is a group of students, comprised of students from one or more courses that are taught together.

The following are some of the business rules given by the functional dependencies:

CourseCode, Day, Time, SemesterYear→ Room
LecturerID, Day, Time, SemesterYear → CourseCode
LecturerID, Day, Time, SemesterYear→ Room
CourseCode→CourseCode, CourseName
CourseCode, SemesterYear→ LecturerID
LecturerID → LecturerName
Room, Day, Time, SemesterYear→ CourseCode
Room, Day, Time, SemesterYear→ LecturerID

Q1. Normalisation with respect to the given primary key (i.e., 2NF and 3NF below are considered with respect to the given primary key).

(a) Is the relation Classtimetable in 2NF? If yes, explain why. If not, normaliseClasstimetable to 2NF and specify the primary keys of the decomposed relations.

Solution:

The given relation is clearly not in 2NF because attribute: LecturerName is dependent on attribute LecturerID which is not a primary key.

To decompose the given relation so as to have it in 2NF we need to split the relation such that all that no non-key attribute determines the other non-key attribute.

Thus we get:
R1: (CourseCode, Day, Time, SemesterYear, CourseName, Room)
FD: CourseCode, Day, Time, SemesterYear → Room, CourseName
*CourseName may not be unique hence we can't take CourseID→CourseName
R2: (CourseCode, SemesterYear, LecturerID, LecturerName)
FD: CourseCode, SemesterYear, LecturerID →LecturerName
*LecturerName may not be unique hence we can't just take LecturerID →LecturerName

(b) Are the relations obtained in (a) in 3NF? If yes, explain why. If not, normalise these relations to 3NF and specify the primary keys of the decomposed relations.

Solution:

Yes the relations R1 and R2 obtained in above solution are in 3NF because there are no transitive dependencies present in the set of FDs

Q2 Normalisation with respect to all candidate keys (i.e., 2NF, 3NF and BCNF below are considered with respect to all candidate keys).

(a) Is the given set of functional dependencies on Classtimetable minimal? If not, give an equivalent minimal set.

Solution:

No, the set of FDs given in the Classtimetable is not minimal.
We can eliminate the FDs like
CourseCode, SemesterYear→ LecturerID: because CourseCode and SemesterYear are a parts of key
CourseCode→ CourseCode, CourseName: because CourseCode is a part of key
LecturerID → LecturerName can be clubbed as
LecturerID, Day, Time, SemesterYear → CourseCode, CourseName, Room

Thus the minimal FD set is:
LecturerID, Day, Time, SemesterYear →CourseCode, CourseName, Room, LecturerName

We can skip the remaining FDs because the above FD is sufficient to determine all the attributes in the given relation.

(b) Find all the candidate keys and prime attributes of Classtimetable.
Solution:

From the set of FDs given we can have following Candidate Keys:
{CourseCode, Day, Time, SemesterYear}
{Room, Day, Time, SemesterYear}
{LecturerID, Day, Time, SemesterYear}

Hence CourseCode, Room, Day, Time, SemesterYear, LecturerID are prime attributes.

(c) Is the relation Classtimetable in 2NF? If yes, explain why. If not, normaliseClasstimetable to 2NF and identify the functional dependencies that can be preserved on these relations.

Solution:

Considering the candidate key {Room, Day, Time, SemesterYear} and given set of FDs, the relation is not in 2NF because the FD LecturerID → LecturerName violates as the Lecturer is a part PK and LecturerName is not a part of PK.
To normalize in 2NF we just need to take following relations:
R1: (Room, Day, Time, SemesterYear, CourseCode, LecturerID)
FD: Room, Day, Time, SemesterYear → CourseCode, LecturerID

R2: (LecturerID, LecturerName, CourseCode, SemesterYear)
FD: LecturerID, CourseCode, SemesterYear→ LecturerName
*LecturerName may not be unique hence we can't just take LecturerID →LecturerName

(d) Are the relations obtained in (c) in 3NF? If yes, explain why. If not, normalise these relations to 3NF and identify the functional dependencies that can be preserved on these relations.

Solution:

The relations R1 and R2 are in 3NF as there are no transitive dependencies in the defined FDs after normalisation.

(e) Are the relations obtained in (d) in BCNF? If yes, explain why. If not, normalise these relations to BCNF and identify the functional dependencies that can be preserved on these relations.

Solution:

The relation obtained for 2NF normalization are
R1: (Room, Day, Time, SemesterYear, CourseCode, LecturerID)
FD: Room, Day, Time, SemesterYear → CourseCode, LecturerID

R2: (LecturerID, LecturerName, CourseCode, SemesterYear)
FD: LecturerID, CourseCode, SemesterYear→ LecturerName

Where we have considered the (Room, Day, Time, SemesterYear) as primary key in R1 and there are no other FDs in the set hence the relation is in BCNF.

As far as the second relation R2 is concerned the FD is (LecturerID, CourseCode, SemesterYear) which determines LecturerName and hence it is the Primary Key of R2 so R2 is in BCNF.

Q3. Has the following business rule been provided by the given set of functional dependencies?

“A class in a semester is taught by only one lecturer".

If not, explain how the given set of functional dependencies should be changed to meet this business rule. If yes, explain how the given set of functional dependencies should be changed to meet the weaker business rule below:

“A class in a semester can be taught by two lecturers".

Solution:

We have a given FD: { Room, Day, Time, SemesterYear→LecturerID } & { Room, Day, Time, SemesterYear → CourseCode } which means that at given ‘Room’ on a ‘Day’, at a ‘Time’ in a particular semester a Lecturer teaches a particular course. That implies that a single Lecturer can teach a single Course in a semester. But that also means for other courses we need other lecturers.
So we can conclude that a single Lecturer does not teach whole class.

We need to add an FD: **LecturerID, SemesterYear → Room** which means that in a Room where a class is held in a specific semester only single Lecturer teaches.

There is a simple way to show that a class is taught by two lecturer in a given semester
LecturerID 1, LecturerID 2, SemesterYear → Roommeans that two lecturers are assigned to s where **LecturerID 1 =! LecturerID 2**

Conclusion: Thus, from given database, we have studied the Normalization and its implementation.

Practical No: 6

Aim:

Consider a simple hotel booking system with a relational database schema as follows:

_ Employee(eno, ename, ird) with the primary key fenog,

_ Room(rno, location, facilities) with the primary key frnog,

_ Guest(gname, contact, phone) with the primary key fgname, contactg,

_ Offer(rno, date, price) with the primary key frno, dateg, and with the foreign key [rno] _ Room[rno]

Booking(gname, contact, rno, date, eno) with the primary key(fgname, contact, rno, dateg), and with the foreign keys

[gname, contact] _ Guest[gname, contact]

[rno, date] _ Offer[rno, date]

[eno] _ Employee[eno]

1. List all the bookings handled by employee \Pepe" (assume that there is only one such employee) showing, for each such booking, the name of the guest, and the price of the offer.

Solution:

π gname ,price(σ ename=”pepe” (Guest >< officer) >< employee)

2. List the room numbers and prices of all offers for \31/12/2011" that have not been booked for that day.

Solution:

π rno, price(σ date=”31/12/2011”(OFFER >< ROOM))

3. List the names and phones of all those guests who have made a booking on either \31/12/2011" or \01/01/2012".

Solution:

π gname, phone(σ date=”31/12/2011” U “01/01/2012”(GUEST >< BOOKING))

4. List the names and contact details of all those guests who have been affected by adouble-booking.

Solution:

π gname,contact(σ b1.rno=b2.rno) ^ (b1.date=b2.date)(BOOKING)

5. List the names and contact details of all those guests who have never been affected bya double-booking.

Solution:

π gname,contact((σ gname,contact,rno.date(BOOKING))-¶(σ(b1.rno.=b2.rno.)^(b1.date=b2.date)(BOOKING)

Conclusion: Thus, from the given simple hotel booking system with a relational database schema, we have performed the given operations.

Practical No: 7

Aim: Translate the given relational algebra queries into SQL statements.

1) π _eno(employee)
Solution:

SELECT enoFROM employee;

2)1 π _rno;date(_price>200(offer ./ booking))

Solution:

select rno,date from OFFICER >< BOOKING where price>200

3) 2 _e1:eno;e2:eno(_e1(employee) ./e1:ename=e2:ename _e2(employee))

Solution:

select ename,e1.name or ename,e2.name or pname from EMPLOYEE

Where e1.name=e2.name

Conclusion: Thus, from above, we have translated the given relational algebra queries into SQL statements.

Practical No: 8

Aim: Consider the following database state:

EMPLOYEE		
Eno.	ename	IRD
1	Daffy	875-649-322

ROOM		
Rno.	Location	Facilities
11	Level 0	Spa,Tennis court,
13	Level 13	13 Bowling lanes
17	Level 2	Pool, Pool table

GUEST		
gname	contact	phone
Pepe	Hostsforskunksville	0800443344
Taz	Bunkerhill	0800232323
Coyote	Ringroad	0800838383

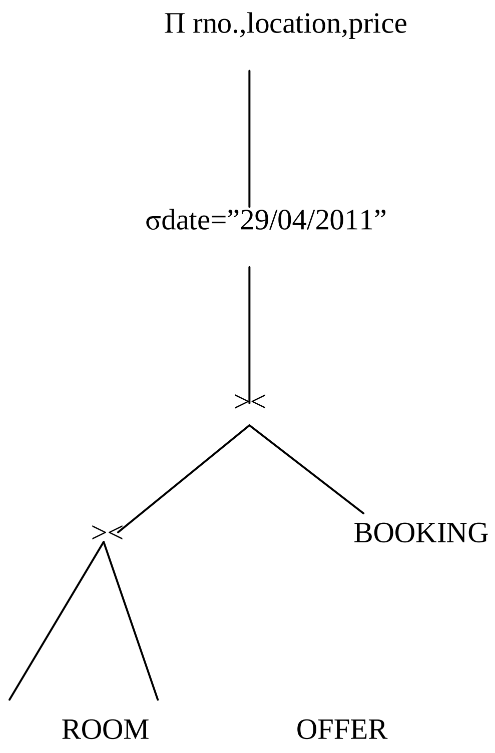
OFFER		
rno.	date	price
11	29/04/2011	749.00
13	29/04/2011	1313.13
17	29/04/2011	235.00
11	13/04/2011	749.00
13	13/04/2011	1313.13

BOOKING				
gname	contact	Rno.	date	eno
Pepe	Hostsforskunksville	11	29/04/2011	1
Coyote	Ringroad	17	29/04/2011	1
Pepe	Hostsforskunksville	13	13/04/2011	1

The following query retrieves the room number, location and price of offers for rooms that were booked for \29/04/2011":

_rno, location, price(_date=\29/04/2009"((Room ./ Offer) ./ Booking))

1. Draw the query tree for this query.



2. Evaluate the query and show your answer as a table.
a) Show the result of Room ><Offer

rno	location	facilities	date	Price
11	Level 0	Spa, tennis court	29/04/2011	749.00
11	Level 0	Spa, tennis court	13/04/2011	749.00
13	Level 13	13 bowling lanes	29/04/2011	1313.13
13	Level 13	13 bowling lanes	13/04/2011	1313.13
17	Level 2	Pool, pool table	29/04/2011	235.00

b) Show the result of (Room ><Offer) ><Booking

rno	location	facilities	date	price	gname	contact	Eno
11	Level 0	Spa, tennis court	29/04/2011	749.00	Pepe	Hostforskunkville	1
13	Level 13	13 bowling lanes	13/04/2011	1313.13	Pepe	Hostforskunkville	1
17	Level 2	Pool , pool table	29/04/2011	235.00	coyote	Ring road	1

c) Show the result of _date=\29/04/2009"((Room ><Offer) ><Booking)

rno	**location**	**facilities**	**date**	**price**	**gname**	**contact**	**Eno**
11	Level 0	Spa, tennis court	29/04/2011	749.00	pepe	hostforskunkville	1
17	Level 2	Pool, pool table	29/04/2011	235.00	coyote	Ring road	1

d) Show the result of _rno, location, price(_date=\29/04/2009"((Room ><Offer) ><Booking)

rno	**location**	**Price**
11	Level 0	749.00
17	Level 2	235.00

Conclusion: Thus, from the given database, we have Drown the query tree and Evaluated the given query