Database Management System

Case Studies

Case Study 1

Hospital Management System

Aim: XYZ hospital is a multi specialty hospital that includes a number of departments, rooms, doctors, nurses, compounders, and other staff working in the hospital. Patients having different kinds of ailments come to the hospital and get checkup done from the concerned doctors. If required they are admitted in the hospital and discharged after treatment.

The aim of this case study is to design and develop a database for the hospital to maintain the records of various departments, rooms, and doctors in the hospital. It also maintains records of the regular patients, patients admitted in the hospital, the check up of patients done by the doctors, the patients that have been operated, and patients discharged from the hospital.

Description: In hospital, there are many departments like Orthopedic, Pathology, Emergency, Dental, Gynecology, Anesthetics, I.C.U., Blood Bank, Operation Theater, Laboratory, M.R.I., Neurology, Cardiology, Cancer Department, Corpse, etc. There is an OPD where patients come and get a card (that is, entry card of the patient) for check up from the concerned doctor. After making entry in the card, they go to the concerned doctor's room and the doctor checks up their ailments. According to the ailments, the doctor either prescribes medicine or admits the patient in the concerned department. The patient may choose either private or general room according to his/her need. But before getting admission in the hospital, the patient has to fulfill certain formalities of the hospital like room charges, etc. After the treatment is completed, the doctor discharges the patient. Before discharging from the hospital, the patient again has to complete certain formalities of the hospital like balance charges, test charges, operation charges (if any), blood charges, doctors' charges, etc.

Next we talk about the doctors of the hospital. There are two types of the doctors in the hospital, namely, *regular doctors* and *call on doctors*. Regular doctors are those doctors who come to the hospital daily. Calls on doctors are those doctors who are called by the hospital if the concerned doctor is not available.

Table Description:

Following are the tables along with constraints used in *Hospital Management* database.

1. **DEPARTMENT:** This table consists of details about the various departments in the hospital. The information stored in this table includes department name, department location, and facilities available in that department.

   Constraint: Department name will be unique for each department.

2. **ALL_DOCTORS:** This table stores information about all the doctors working for the hospital and the departments they are associated with. Each doctor is given an identity number starting with DR or DC prefixes only.

   Constraint: Identity number is unique for each doctor and the corresponding department should exist in DEPARTMENT table.

3. **DOC_REG:** This table stores details of regular doctors working in the hospital. Doctors are referred to by their doctor number. This table also stores personal details of doctors like name, qualification, address, phone number, salary, date of joining, etc.

   Constraint: Doctor's number entered should contain DR only as a prefix and must exist in ALL_DOCTORS table.

4. **DOC_ON_CALL:** This table stores details of doctors called by hospital when additional doctors are required. Doctors are referred to by their doctor number. Other personal details like name, qualification, fees per call, payment due, address, phone number, etc., are also stored.

   Constraint: Doctor's number entered should contain DC only as a prefix and must exist in ALL_DOCTORS table.

5. **PAT_ENTRY:** The record in this table is created when any patient arrives in the hospital for a check up. When patient arrives, a patient number is generated which acts as a primary key. Other details like name, age, sex, address, city, phone number, entry date, name of the doctor referred to, diagnosis, and department name are also stored. After storing the necessary details patient is sent to the doctor for check up.

   Constraint: Patient number should begin with prefix PT. Sex should be *M* or *F* only. Doctor's name and department referred must exist.

6. **PAT_CHKUP:** This table stores the details about the patients who get treatment from the doctor referred to. Details like patient number from patient entry table, doctor number, date of check up, diagnosis, and treatment are stored. One more field status is used to indicate whether patient is admitted, referred for operation or is a regular patient to the hospital. If patient is admitted, further details are stored in PAT_ADMIT

table. If patient is referred for operation, the further details are stored in PAT_OPR table and if patient is a regular patient to the hospital, the further details are stored in PAT_REG table.

Constraint: Patient number should exist in PAT_ENTRY table and it should be unique.

7. **PAT_ADMIT:** When patient is admitted, his/her related details are stored in this table. Information stored includes patient number, advance payment, mode of payment, room number, department, date of admission, initial condition, diagnosis, treatment, number of the doctor under whom treatment is done, attendant name, etc.

   Constraint: Patient number should exist in PAT_ENTRY table. Department, doctor number, room number must be valid.

8. **PAT_DIS:** An entry is made in this table whenever a patient gets discharged from the hospital. Each entry includes details like patient number, treatment given, treatment advice, payment made, mode of payment, date of discharge, etc.

   Constraint: Patient number should exist in PAT_ENTRY table.

9. **PAT_REG:** Details of regular patients are stored in this table. Information stored includes date of visit, diagnosis, treatment, medicine recommended, status of treatment, etc.

   Constraint: Patient number should exist in patient entry table. There can be multiple entries of one patient as patient might be visiting hospital repeatedly for check up and there will be entry for patient's each visit.

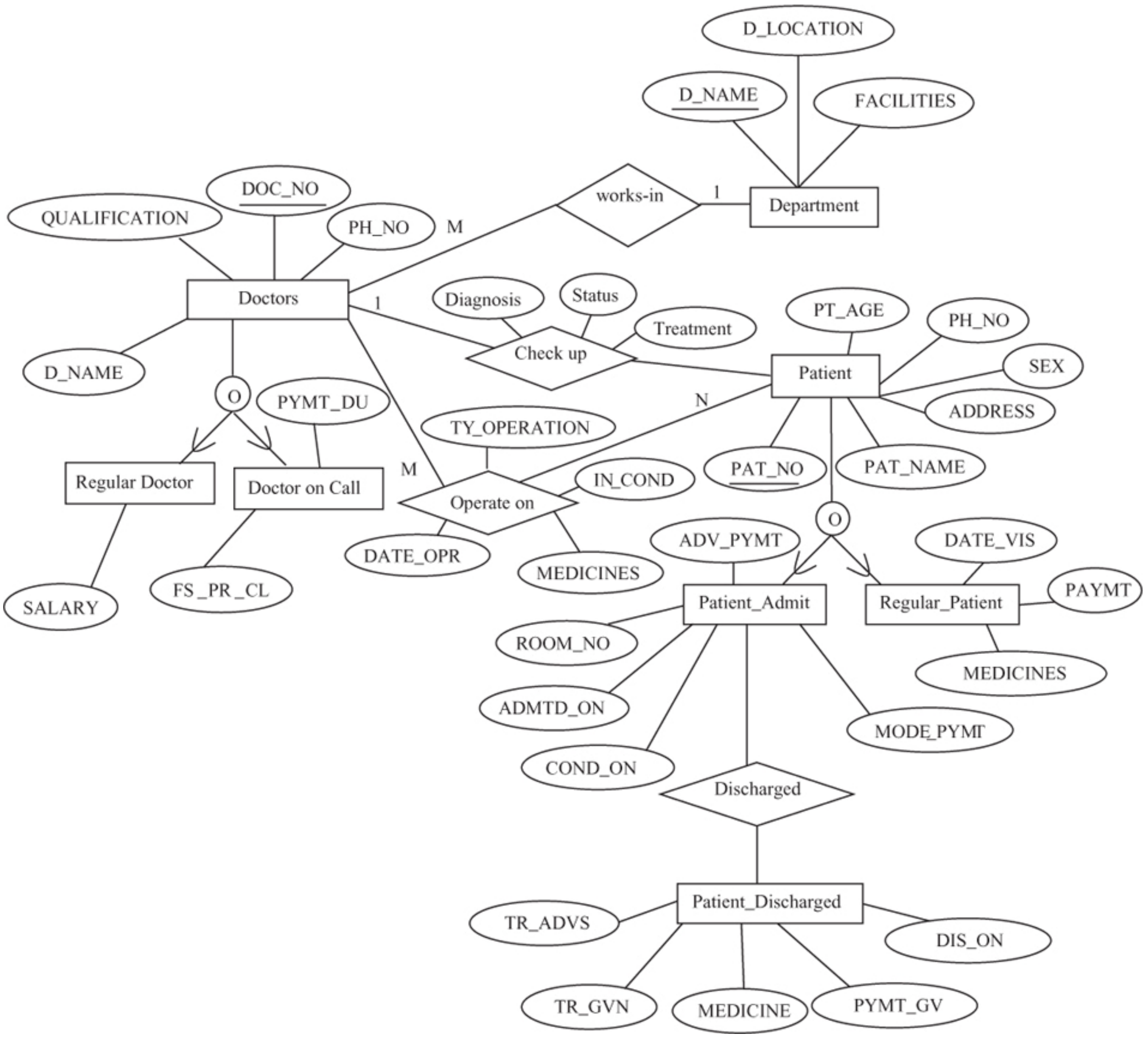
10. **PAT_OPR:** If patient is operated in the hospital, his/her details are stored in this table. Information stored includes patient number, date of admission, date of operation, number of the doctor who conducted the operation, number of the operation theater in which operation was carried out, type of operation, patient's condition before and after operation, treatment advice, etc.

    Constraint: Patient number should exist in PAT_ENTRY table. Department, doctor number should exist or should be valid.

11. **ROOM_DETAILS:** It contains details of all rooms in the hospital. The details stored in this table include room number, room type (general or private), status (whether occupied or not), if occupied, then patient number, patient name, charges per day, etc.

    Constraint: Room number should be unique. Room type can only be *G* or *P* and status can only be *Y* or *N*

E-R Diagram



Relational Database Schema for Case Study

The relational database schema for *Hospital Management* database is as follows:

1. DEPARTMENT (D_NAME, D_LOCATION, FACILITIES)
2. ALL_DOCTORS (DOC_NO, DEPARTMENT)
3. DOC_REG(DOC_NO, D_NAME, QUALIFICATION, SALARY, EN_TIME, EX_TIME, ADDRESS, PH_NO, DOJ)

4. DOC_ON_CALL (DOC_NO, D_NAME, QUALIFICATION, FS_PR_CL, PYMT_DU, ADDRESS, PH_NO)

5. PAT_ENTRY (PAT_NO, PAT_NAME, CHKUP_DT, PT_AGE, SEX, RFRG_CSTNT, DIAGNOSIS, RFD, ADDRESS, CITY, PH_NO, DEPARTMENT)

6. PAT_CHKUP (PAT_NO, DOC_NO, DIAGNOSIS, STATUS, TREATMENT)

7. PAT_ADMIT (PAT_NO, ADV_PYMT, MODE_PYMT, ROOM_NO, DEPTNAME, ADMTD_ON, COND_ON, INVSTGTN_DN, TRMT_SDT, ATTDNT_NM)

8. PAT_DIS (PAT_NO, TR_ADVS, TR_GVN, MEDICINES, PYMT_GV, DIS_ON)

9. PAT_REG (PAT_NO, DATE_VIS, CONDITION, TREATMENT, MEDICINES, DOC_NO, PAYMT)

10. PAT_OPR (PAT_NO, DATE_OPR, IN_COND, AFOP_COND, TY_OPERATION, MEDICINES, DOC_NO, OPTH_NO, OTHER_SUG)

11. ROOM_DETAILS (ROOM_NO, TYPE, STATUS, RM_DL_CRG, OTHER_CRG)

Case Study 2

Railway Reservation

Aim: The railway reservation system facilitates the passengers to enquire about the trains available on the basis of source and destination, booking and cancellation of tickets, enquire about the status of the booked ticket, etc.

The aim of case study is to design and develop a database maintaining the records of different trains, train status, and passengers. The record of train includes its number, name, source, destination, and days on which it is available, whereas record of train status includes dates for which tickets can be booked, total number of seats available, and number of seats already booked. The database has been developed and tested on the Oracle.

Description:

Passengers can book their tickets for the train in which seats are available. For this, passenger has to provide the desired train number and the date for which ticket is to be booked. Before booking a ticket for a passenger, the validity of train number and booking date is checked. Once the train number and booking date are validated, it is checked whether the seat is available. If yes, the ticket is booked with confirm status and corresponding ticket ID is generated which is stored along with other details of the passenger. After all the available tickets are booked, certain numbers of tickets are booked with waiting status. If waiting lot is also finished, then tickets are not booked and a message of non-availability of seats is displayed.

The ticket once booked can be cancelled at any time. For this, the passenger has to provide the ticket ID (the unique key). The ticket ID is searched and the corresponding record is deleted. With this, the first ticket with waiting status also gets confirmed.

List of Assumption

Since the reservation system is very large in reality, it is not feasible to develop the case study to that extent and prepare documentation at that level. Therefore, a small sample case study has been created to demonstrate the working of the reservation system. To implement this sample case study, some assumptions have been made, which are as follows:

1. The number of trains has been restricted to 5.
2. The booking is open only for next seven days from the current date.
3. Only two categories of tickets can be booked, namely, *AC* and *General*.
4. The total number of tickets that can be booked in each category (AC and General) is 10.
5. The total number of tickets that can be given the status of waiting is 2.

6. The in-between stoppage stations and their bookings are not considered.

Description of Tables and Procedures

Tables and procedures that will be created are as follows:

1. **TrainList:** This table consists of details about all the available trains. The information stored in this table includes train number, train name, source, destination, fair for AC ticket, fair for general ticket, and weekdays on which train is available.

   Constraint: The train number is unique.

2. **Train_Status:** This table consists of details about the dates on which ticket can be booked for a train and the status of the availability of tickets. The information stored in this table includes train number, train date, total number of AC seats, total number of general seats, number of AC seats booked, and number of general seats booked.

   Constraint: Train number should exist in TrainList table.

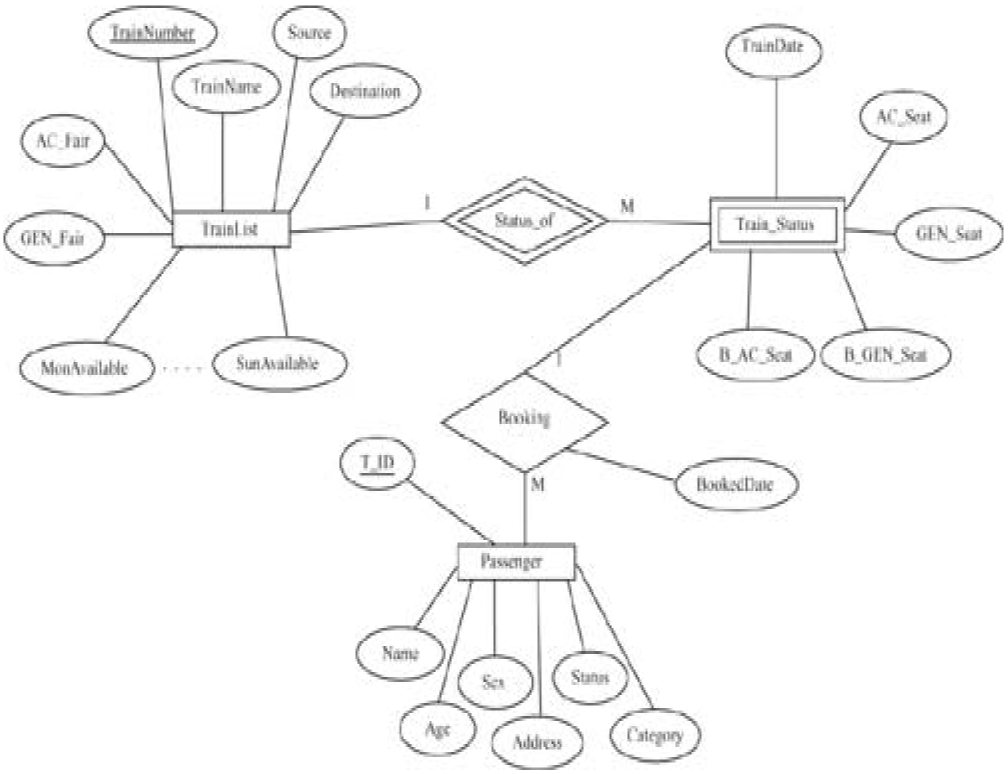
3. **Passenger:** This table consists of details about the booked tickets. The information stored in this table includes ticket ID, train number, date for which ticket is booked, name, age, sex and address of the passenger, status of reservation (either confirmed or waiting), and category for which ticket is booked.

   Constraint: Ticket ID is unique and the train number should exist in TrainList table.

4. **Booking:** In this procedure, the train number, train date, and category is read from the passenger. On the basis of the values provided by the passenger, corresponding record is retrieved from the Train_Status table. If the desired category is AC, then total number of AC seats and number of booked AC seats are compared in order to find whether ticket can be booked or not. Similarly, it can be checked for the general category. If ticket can be booked, then passenger details are read and stored in the Passenger table.

5. **Cancel:** In this procedure, ticket ID is read from the passenger and corresponding record is searched in the Passenger table. If the record exists, it is deleted from the table. After deleting the record (if it is confirmed), first record with waiting status for the same train and same category are searched from the Passenger table and its status is changed to confirm.

E-R diagram


TrainNumber
Source
TrainName
Destination
AC_Fair
GEN_Fair
TrainList
MonAvailable
. . . .
SunAvailable
1
Status_of
M
TrainDate
AC_Seat
Train_Status
GEN_Seat
B_AC_Seat
B_GEN_Seat
1
Booking
M
T_ID
BookedDate
Passenger
Name
Sex
Status
Age
Address
Category

Case Study 3

Painting Hire Business

System Description:

A local businesswoman has decided to start her own Internet business, called Masterpieces Ltd, hiring paintings to private individuals and commercial companies.

Because of your reputation as a database designer she has called upon your services to design and implement a database to support her new business. At the initial planning meeting, to discuss the design, the following user requirements were requested.

The system must be able to manage the details of customers, paintings and those paintings currently on hire to customers. Customers are categorized as B (bronze), S (silver), G (gold) or P (platinum). These categories entitle a customer to a discount of 0%, 5%, 10% or 15% respectively.

Customers often request paintings by a particular artist or theme (eg animal, landscape, seascape, naval, still-life, etc). Over time a customer may hire the same painting more than once.

Each painting is allocated a customer monthly rental price defined by the owner. The owner of the painting is then paid 10% of that customer rental price. Any paintings that are not hired within six months are returned to the owner. However, after three months, an owner may resubmit a returned painting.

Each painting can only have one artist associated with it.

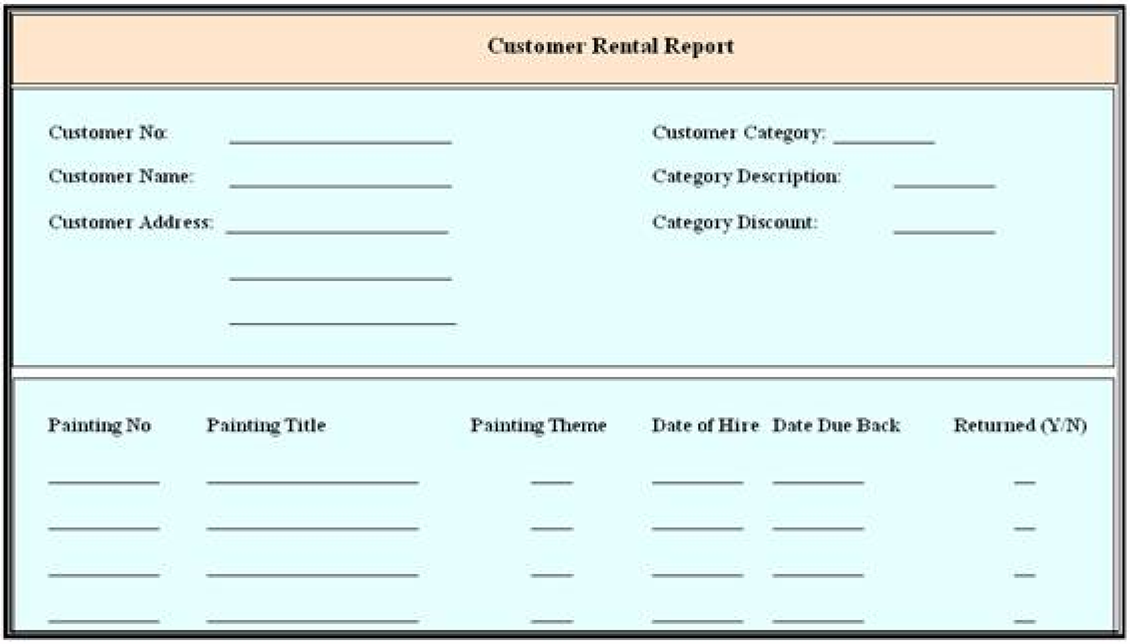
Several reports are required from the system. Three main ones are:

1. For each customer, a report showing an overview of all the paintings they have hired or are currently hiring
2. For each artist, a report of all paintings submitted for hire
3. For each artist, a returns report for those paintings not hired over the past six months

Remember to **identify key attributes** and any **foreign key attributes**.

Pages to be created

Customer Rental Report



Customer Rental Report

Customer No: ____________________

Customer Name: ____________________

Customer Address: ____________________

____

____

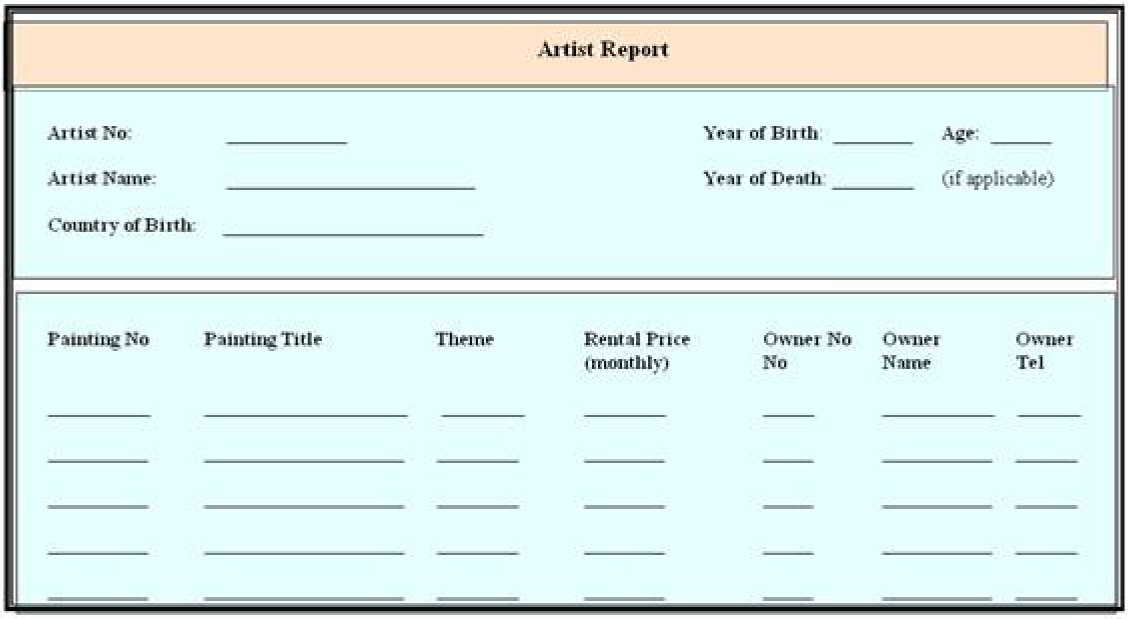
Customer Category: __________

Category Description: __________

Category Discount: __________

Painting No	Painting Title	Painting Theme	Date of Hire	Date Due Back	Returned (Y/N)
________	________________	___	_______	_______	_
________	________________	___	_______	_______	_
________	________________	___	_______	_______	_
________	________________	___	_______	_______	_

Artist Report



Artist Report

Artist No: __________

Artist Name: ____________________

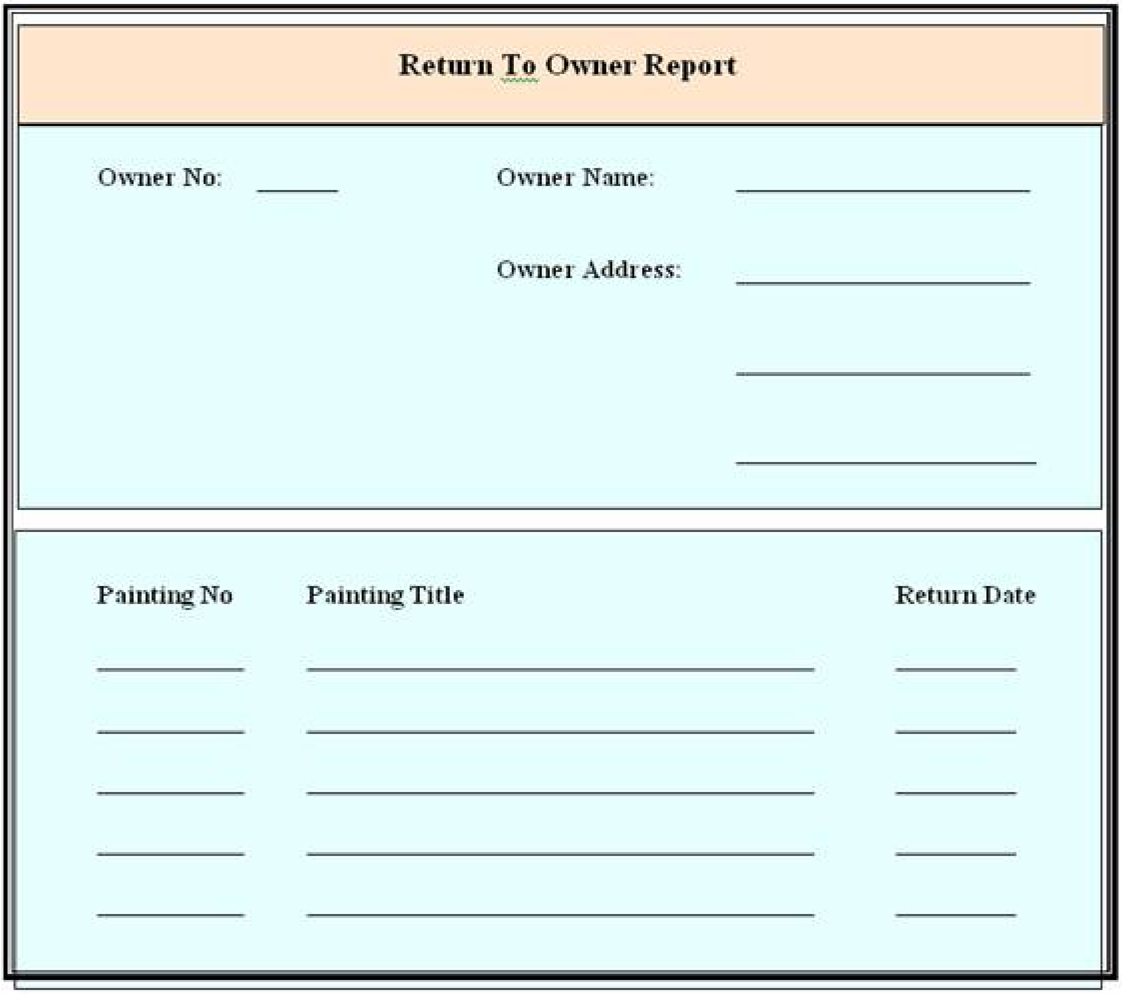
Country of Birth: ____________________

Year of Birth: _______ Age: _____

Year of Death: _______ (if applicable)

Painting No	Painting Title	Theme	Rental Price (monthly)	Owner No No	Owner Name	Owner Tel
________	________________	_______	_______	____	_________	_____
________	________________	_______	_______	____	_________	_____
________	________________	_______	_______	____	_________	_____
________	________________	_______	_______	____	_________	_____
________	________________	_______	_______	____	_________	_____

Return to Owner Report



Return To Owner Report

Owner No: ______

Owner Name: ______________________

Owner Address: ______________________

____

____

Painting No	Painting Title	Return Date

Case Study 4

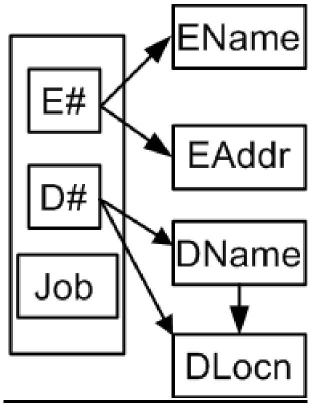
The WORK relation illustrates data about employees, their job title and the department they are assigned to. From examining sample data and discussions with management we have found that employees can have multiple job titles and can be assigned to more than one department. Each department is completely sited in a single location but a city could have more than one department at some time.

WORK JOB	ENAME	EADDR	E#	D#	DNAME	DLOCN
HELPER	DAVIS	111 FIRST ST	12	1	PRESSING	ALCOA
HELPER	SPENCE	222 SECOND ST	78	1	PRESSING	ALCOA
ELECTRICIAN	MURPHY	100 MAIN ST	66	2	WELDING	NIOTA
FOREMAN	SMITH	300 BROAD ST	77	9	PACKING	LOUDON
CLERK	WILSON	111 FIRST ST	99	7	PAYROLL	MEMPHIS
CLERK	DAVIS	111 FIRST ST	12	1	PRESSING	ALCOA
CLERK	SPENCE	222 SECOND ST	78	1	PRESSING	ALCOA
CLERK	DAVIS	111 FIRST ST	12	5	MAILROOM	ONEIDA

For this relation, a composed key is required as no one attribute is a candidate. It turns out that the following SRN depicts the situation:

WORK (Job, EName, EAddr, E#, D#, DName, DLocn)

and the functional dependency diagrams would be:



There are numerous problems with the data model as it currently stands. We cannot add new employees until they have a job title and a department assignment. We can easily lose department data by removing an employee who is the sole person assigned to a department. Certain updates require careful propagation of changes throughout the database. Careful decomposition can take care of these problems. The employee data makes an obvious grouping and should be decomposed the get it into at least 2NF. It will actually go to BCNF as there are no further problems. It is ready to become a table.

EMPLOYEE E#	ENAME	EADDR
12	DAVIS	111 FIRST ST
78	SPENCE	222 SECOND ST
66	MURPHY	100 MAIN ST
77	SMITH	300 BROAD ST
99	WILSON	111 FIRST ST

The Dept relation is another logical decomposition to remove the partial dependency and move to 2NF. Careful examination reveals the transitive dependency still exists so further decomposition is necessary.

DEPT D#	DNAME	DLOCN
1	PRESSING	ALCOA
2	WELDING	NIOTA
9	PACKING	LOUDON
7	PAYROLL	MEMPHIS
5	MAILROOM	ONEIDA

Job-Worked winds up looking like the original relation's key. All three attributes are still the composed key. Since there are no dependencies, there is nothing to prevent this relation from being BSNF so it is ready too.

JOB-WORKED E#	D#	JOB
12	1	HELPER
78	1	HELPER
66	2	ELECTRICIAN
77	9	FOREMAN
99	7	CLERK
12	1	CLERK
78	1	CLERK
12	5	CLERK

To remove the transitive dependency, we will decompose Dept into Department and Dept-Locn.

Each of these is now in BCNF.

DEPARTMENT D#	DNAME
1	PRESSING
2	WELDING
9	PACKING
7	PAYROLL
5	MAILROOM

DEPT-LOCN D#	DLOCN
1	ALCOA
2	NIOTA
9	LOUDON
7	MEMPHIS
5	ONEIDA

Case Study 5

A relational database is to be designed for a medium sized Company dealing with industrial applications of computers. The Company delivers various products to its customers ranging from a single application program through to complete installation of hardware with customized software. The Company employs various experts, consultants and supporting staff. All personnel are employed on long-term basis, i.e. there is no short-term or temporary staff. Although the Company is somehow structured for administrative purposes (that is, it is divided into departments headed by department managers) all projects are carried out in an inter-disciplinary way. For each project a project team is selected, grouping employees from different departments, and a Project Manager (also an employee of the Company) is appointed who is entirely and exclusively responsible for the control of the project, quite independently of the Company's hierarchy. The following is a brief statement of some facts and policies adopted by the Company.

- Each employee works in some department.
- An employee may possess a number of skills
- Every manager (including the MD) is an employee
- A department may participate in none/one/many projects.
- At least one department participates in a project.
- An employee may be engaged in none/one/many projects
- Project teams consist of at least one member.

For the above business stories you are expected to create the following.

1. Analyze the data required.

2. Normalize the attributes.

3. Create the logical data model (ER diagrams).