Project

Databases are an integral part of an organization. Aspiring database developers and administrators should be able to efficiently implement, maintain and administer databases. Knowledge of these will enable the developers to build robust database solutions.

This module will enable students to demonstrate the skills that they have acquired on databases using SQL Server 2005 to provide a solution in a given scenario. In addition, it will enable students to implement the various design principles and guidelines for designing a database and database objects.

Objectives

In this project, you will learn to:

- Design a database and database objects
- Apply security on the database and the database server
- Implement backup, recovery, and mirroring techniques

Case Study 1: Shop Here

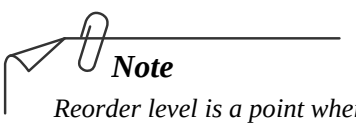
Shop Here is a leading chain of departmental stores in North America. It has various stores in the United States, Bahamas, Canada, Cuba, Costa Rica, and Mexico with its head office in Washington.

The chain is known for quality products, affordable prices, timely service, and a free home-delivery facility. The store procures goods from various countries. To ensure that all its products are available at all times, Shop Here maintains an optimum level of inventory.

Current System

There are various categories of products sold by Shop Here. Product categories include Household, Sports, Accessories, and Clothing. The details of all these categories are stored in the ProductCategory file. This file contains the ID, name, and description of the category.

The details of all the products are stored in the ItemDetails file. This file includes the ID of the product, the name of the product, the description of the product, the category ID, the unit price of each product, the quantity available of each product, the reorder level, the reorder quantity of each product, and the supplier ID.



Note

Reorder level is a point where the product needs to be ordered again. Reorder quantity refers to the quantity that needs to be ordered to maintain the optimum level of the stock.

Shop Here has some regular suppliers for various products. These suppliers are widespread over geographical boundaries. The details of all these suppliers are stored in the SupplierDetails file. This file contains the ID, name, address, phone, and country of the supplier.

When the quantity in hand reaches the reorder level for any product, an order needs to be placed with the supplier to send the required product. When an order is placed, a purchase order form needs to be filled by an employee. The format of the purchase order form is given in the following figure.

PURCHASE ORDER FORM

Purchase Order ID:	Supplier ID:
Employee ID:	Order Date:
Shipment Method:	Quantity:
Item ID:	Unit Price:

Format of the Purchase Order Form

The details of the purchase order form are stored in the OrderDetails file. The OrderDetails file stores the purchase order ID, the ID of the supplier, the ID of the employee who placed the order, the date of order, the expected date of receiving the goods, the ID of the item ordered, the unit price of the ordered item, the quantity ordered, the quantity received, the shipment method, and the current status of the order.

The value of the OrderStatus field will be InTransit while placing the order and it will be updated to Received when the order is received. The RecievingDate and QuantityReceived field is updated when the order is received. The QuantityInHand field is increased by the QuantityReceived value for a particular product after receiving the order.

Because the size of business is increasing, it is becoming difficult for the management to maintain the inventory manually.

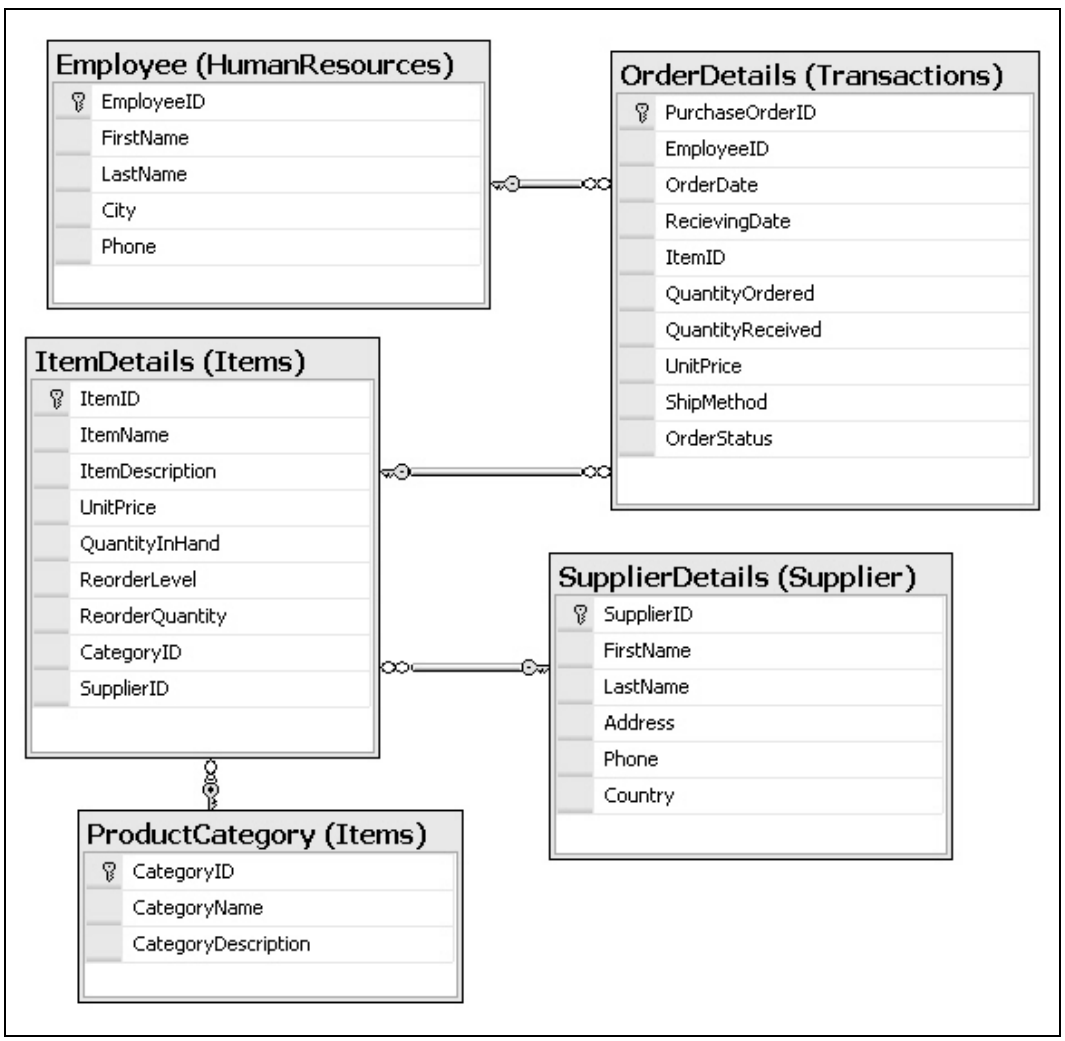
Envisioned System

To simplify the inventory management work of Shop Here, Inventory head Chris Jones has decided to implement the computerized transaction management and storing system. To accomplish this task, he has contacted BlackDB Inc. to develop the computerized system. Sam Williams, a project manager in BlackDB has been deputed to manage this assignment. Sam and his project team have:

1. Identified the various entities involved.
2. Identified the attributes of the various entities. These attributes completely define the entities.
3. Drawn an E/R diagram to demonstrate the relationships between the various entities.
4. Mapped the E/R diagram to tables.

5. Identified various schemas involved. These schemas are Items, Suppliers, HumanResources, and Transactions.
6. Normalized the tables to 3 NF. The normalized tables have:
   - Table names.
   - Field names, datatypes, and size of fields.
   - Primary and foreign key identification and integrity.
7. Drawn a diagram to show the relationships between the various tables.

The final relationship diagram created by the project team is shown in the following figure.



Relationship Diagram for Shop Here

To create a computerized transaction system for Shop Here by using SQL Server 2005, the project team needs to perform the following tasks:

1. Create a database, **ShopHere**.
2. Create the tables as per the relationship diagram, ensuring minimum disk space utilization.
3. Perform validations on the tables as per the following guidelines:

Table: Items.ItemDetails

- ItemID must be auto generated.
- ItemName should not be left blank.
- ItemDescription should not be left blank.
- QuantityInHand should be greater than 0. The record should not be inserted or modified manually if QuantityInHand is 0.
- UnitPrice should be greater than 0.
- ReorderQuantity should be greater than 0.
- ReorderLevel should be greater than 0.
- CategoryID should be the foreign key from the ProductCategory table.
- SupplierID should be the foreign key from the SupplierDetails table.

Table: Items.ProductCategory

- CategoryID must be auto generated.
- CategoryName and CategoryDescription should not be left blank.
- CategoryName should be Household, Sports, Accessories, or Clothing.

Table: Supplier.SupplierDetails

- SupplierID must be auto generated.
- FirstName, LastName, Address, Phone, and Country should not be left blank.
- Phone number must be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: HumanResources.Employee

- EmployeeID must be auto generated.
- Employee FirstName, LastName, City, and Phone should not be left blank.

- Phone must be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'

  Example: 11-111-1111-111-111

Table: Transactions.OrderDetails

- PurchaseOrderID must be auto generated.
- OrderDate should not be greater than the current date.
- If the order date is not entered, the current date should be taken as the default date.
- QuantityOrdered, QuantityReceived, and UnitPrice should be greater than 0.
- QuantityReceived should allow NULL.
- QuantityReceived cannot be greater than QuantityOrdered.
- QuantityReceived should be added to QuantityInHand in the Items table.
- When a record is inserted into the table, QuantityInHand in the Items table should be updated automatically.
- OrderStatus must be any of the following values: 'InTransit', 'Received', or 'Cancelled'.
- RecievingDate should allow NULL and should be greater than OrderDate.

4. Create appropriate relationships between the tables.
5. Store the order details in a text file on a day-to-day basis. Make use of the required tools to perform the data transfer.
6. Create the appropriate indexes to speed up the execution of the following tasks:
   - Extract the order details for all the purchase orders in the current month.
   - Extract the details of all the orders placed more than two years back.
   - Extract the details of the top five suppliers to whom the maximum number of orders have been placed in the current month.
7. Simplify the following tasks:
   - Calculation of the total cost for a particular order
   - Calculation of the total of all the orders placed by a particular employee in a particular month
8. Implement an appropriate security policy on the database. For this, create logins named George, John, and Sara. George is the database administrator and John and Sara are database developers.
9. Back up the database daily and store it in the C drive.
10. Store crucial data in encrypted format.
11. Ensure that an alert is sent to George whenever the size of the temporary space in the database server falls below 20 MB.

Case Study 2: Showman House

Showman House is a very large event management company in South America. It has various offices in Peru, Chile, and Argentina and the head office is in Brazil.

Showman House is well-known for its efficiency, good-quality staff, and affordable charges.

Current System

The company manages various types of events throughout the year. These events include fashion shows, celebrity shows, chat shows, musical extravaganzas, exhibitions, fairs, and charity shows. The details related to the type of events are stored in the EventType file. This file contains the event type id, description of the event type, and charge per person for a particular event. Any organization that needs to organize an event needs to specify the type of the event and make the payment. The details of the organization, such as the ID, name, address, city, state, and phone numbers are stored in the Customers file.

The payment details, which include the payment id, event id, payment amount, payment date, credit card number, card holder's name, credit card expiry date, payment method id, and cheque number, are stored in the Payments file.

The details about the type of payments are stored in the PaymentMethods file. The payment method details, such as the payment method id and the description of the payment are stored in the PaymentMethods file. The payment methods can be cash, cheque, or credit card.

All details related to an event, such as the event id, event name, event type id, location of the event, start date, end date, staff required for the event, employee id of the employee managing the event, customer id of the organization, and the number of people attending the event, are stored in the Events file.

All Showman House events are managed by its efficient employees. The employee details such as the employee id, first name, last name, address, title, and phone numbers are stored in the Employee file.

Because the business is growing, the Showman House is unable to maintain the details manually. At any given point of time, Showman House manages at least 20 events. Maintaining all the information about these events manually is difficult and time-consuming.

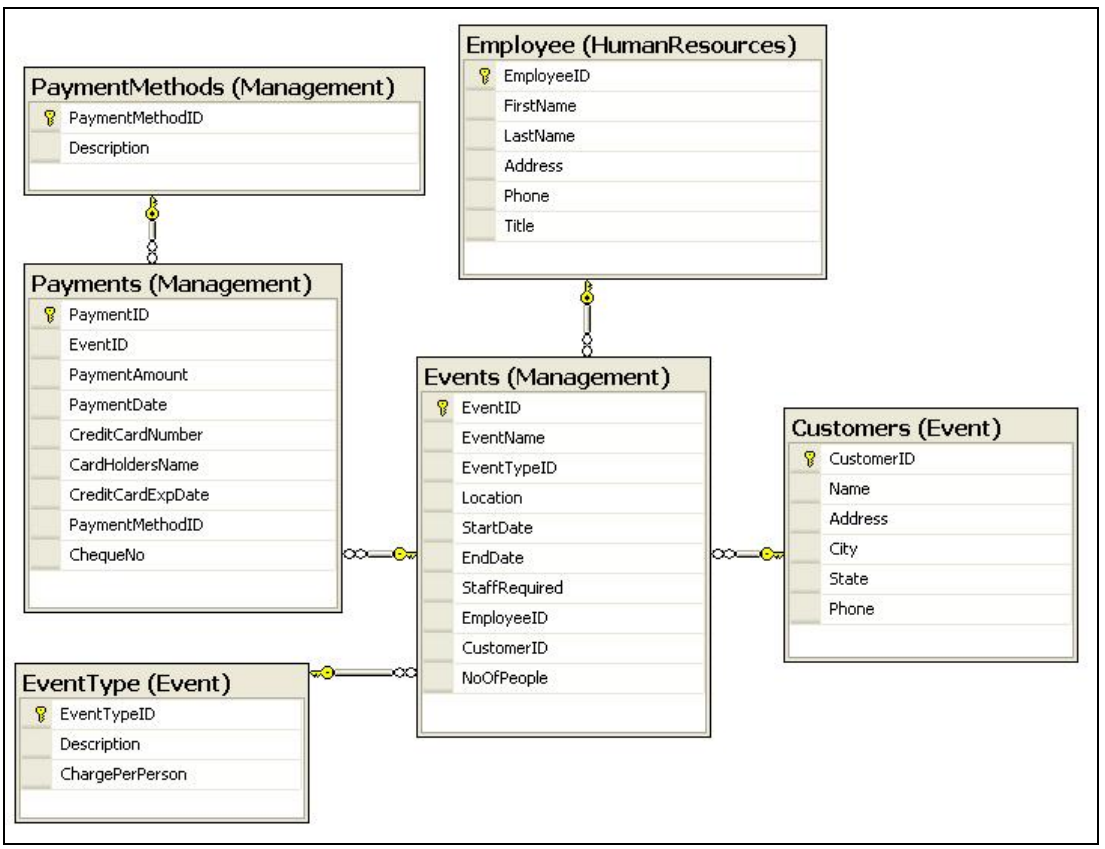
Envisioned System

To overcome these problems, Showman House has decided to computerize its entire system. It has awarded the project to CreateMyDb, Inc.

CreateMyDb Inc. has placed a team in Showman House to analyze the business requirements. After a month, the team has:

1. Identified the various entities involved.
2. Identified the attributes of the various entities. These attributes completely define the entities.
3. Drawn an E/R diagram to demonstrate the relationships between the various entities.
4. Identified the schemas involved. The schemas required are Management, Events, and HumanResources.
5. Mapped the E/R diagram to the tables.
6. Normalized the tables to 3 NF. The normalized tables should have:
   - Table names.
   - Field names, data types, and size of fields.
   - Primary and foreign key identification and integrity.
7. Drawn a diagram to show the relationships between the various tables.

The final relationship diagram created by the project team is shown in the following figure.



Relationship Diagram for Showman House

To create the computerized event management system for Showman House, the project team of CreateMyDb Inc. needs to perform the following tasks:

1. Create a database called **ShowmanHouse**.
2. Create the table designs as per the relationship diagram ensuring minimum disk space utilization.
3. Perform validations on the tables as per the following guidelines:

Table: HumanResources.Employees

- EmployeeID should be auto generated.
- FirstName, LastName, Address, and Phone should not be left blank.
- Title should have one of the following values: Executive, Senior Executive, Management Trainee, Event Manager, or Senior Event Manager.

- Phone should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: Management.Events

- EventID should be auto generated.
- EventName, StartDate, EndDate, Location, NoOfPeople, and StaffRequired should not be left blank.
- StaffRequired should be greater than 0.
- StartDate should be less than the End Date.
- StartDate and EndDate should be greater than the current date.
- NoOfPeople should be greater than or equal to 50.
- EventTypeID is a foreign key from the EventType table.
- CustomerID is a foreign key from the Customers table.
- EmployeeID is a foreign key from the Employee table.

Table: Management.Payments

- PaymentID must be auto generated.
- PaymentDate should be less than or equal to the start date of the event.
- PaymentDate cannot be less than the current date.
- PaymentMethodID is a foreign key from the PaymentMethods table.
- If the client wants to pay by using a credit card, its details need to be entered in the record. If a credit card is not being used it must be ensured that the credit card details are blank.
- ExpiryDate of the credit card should be greater than the current date.
- PaymentAmount should be calculated depending on the event type id and the charge per person.
- ChequeNo should be entered if the payment is done through cheque, else it should be left blank.
- PaymentAmount must be calculated by using the following formula:
  *PaymentAmount = ChargePerPerson * NoOfPeople*

Table: Events.Customers

- CustomerID must be auto generated.
- Name, Address, City, State, Phone should not be left blank.

- Phone should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: Event.EventTypes

- EventTypeID must be auto generated.
- Description should not be left blank.
- ChargePerPerson should be greater than 0.

Table: Management.PaymentMethods

- PaymentMethodID must be auto generated.
- Description must contain any of the three values: cash, cheque, or credit card.

4. Create appropriate relationships between the tables.
5. Store the details of all the employees who have managed an event in the current month in a text file. This information will be displayed on the organization's website. Make use of the required tools to perform the data transfer.
6. Create appropriate indexes to speed up the execution of the following tasks:
   - Extracting the customer details for an event organized on a particular date.
   - Extracting event details for all the events where the payment is pending.
   - Displaying the details of all the events where the staff required is greater than 25.
7. Implement a proper security policy in the database. For this, create logins named William, Sam, Chris, and Sara. Chris is the database administrator and Williams, Sam, and Sara are database developers.
8. Back up the database daily and store the backup in the C drive.
9. Store crucial data in encrypted format.
10. Ensure that an alert is sent to Chris whenever the size of the temporary space in the database falls below 20 MB.

Case Study 3: New Project Ltd.

New Project Ltd. is a leading company in the United States that provides staff on contract for various software project assignments. The company has offices in Texas, Washington, and New York.

The company is known for providing qualified and skilled staff that can fulfill the project requirements efficiently. New Project employs professionals from various fields to match the requirement of its clients.

Current System

New Project has a system of time cards. Time cards are issued to its employees for working at the client site until the completion of the project. The employees use their time cards to record all work processes and expenses during the contract period. The time card data enables New Project to prepare the final billing that needs to be submitted to the client.

When a work agreement is signed, a project form needs to be filled out. The format of this form is shown in the following figure.

PROJECT DETAIL FORM

CLIENT SECTION

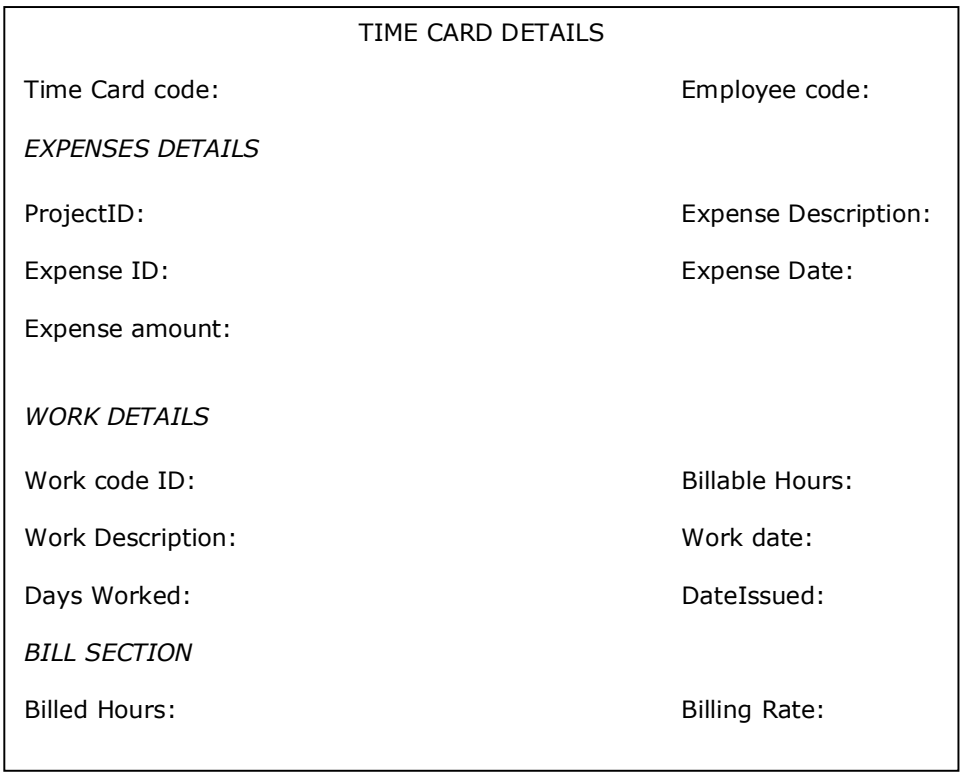
Client code:	Company Name:
Address:	Country:
Contact Person:	Phone:

PROJECT SECTION

Project code:	Project Name:
Project Description:	Estimate Bill:
Start Date:	End Date:

Project Detail Form

The client details are stored in the Clients file, the project details are stored in the Projects file, and the employee details are stored in the Employees file. Next, the details of time spent by each employee on a project needs to be filled in the Time Card Details form. The format of this form is shown in the following figure.



Time Card Details Form

All employees fill the time card details on a daily basis. These details include TimeCardID, EmployeeID, DateIssued, DaysWorked, ProjectID, WorkDescription, BillableHours, and WorkCodeID. The staffing-in-charge of New Project stores these details in the TimeCards file. The WorkCodeID defines the category of the task performed by the employee in the project. The details related to the work codes are stored in the WorkCodes file which includes details such as work code id and work code description.

During the project life cycle, employees might also incur expenses on items such as conveyance and food. The details of the expenses incurred by an employee in a project are stored in the TimeCardExpenses file. These details include TimeCardExpenseID, TimeCardID, ExpenseDate, ExpenseAmount, ExpenseDescription, ProjectID, and ExpenseID. After every project, the company calculates its payment due from the client and raises an invoice. On realization of the payment, all related data is stored in the Payments file. This file contains details such as PaymentCode, PaymentModeID,

ProjectID, PaymentAmount, PaymentDate, PaymentDue. If payment is made through a credit card, details such as CreditCardNumber, CardHoldersName, and CreditCardExpDate are recorded in the file.

Because New Project Ltd. works on many projects simultaneously, it is difficult and time-consuming for them to manually manage the details of all the projects.

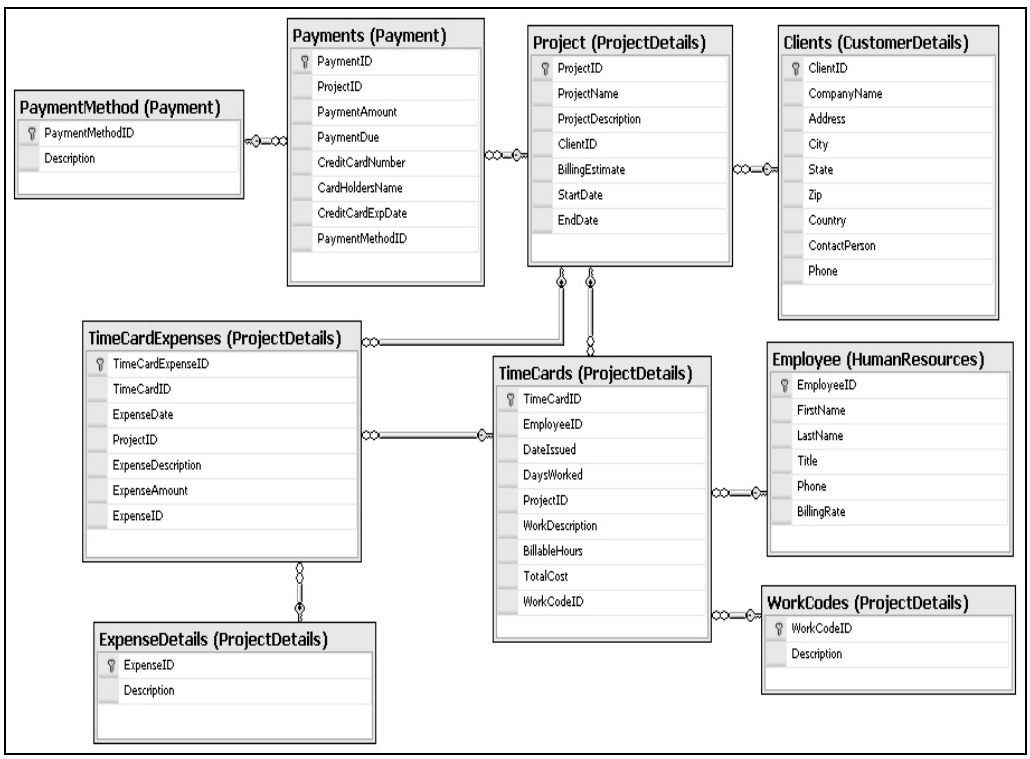
Envisioned System

The management of New Project Ltd. has decided to computerize the whole process.

An in-house development team has:

1. Identified the various entities involved.
2. Identified the attributes of the various entities. The attributes completely define the entities.
3. Drawn an E/R diagram to demonstrate the relationship between the various entities.
4. Mapped the E/R diagram to the tables.
5. Identified the various schemas involved in the design.
6. Normalized the tables to 3 NF. The normalized tables have:
   - Table names.
   - Field names, data types, and size of fields.
   - Primary and foreign key identification and integrity.
7. Drawn a diagram to show the relationships between the various tables.

The final relationship diagram created by the project team is shown in the following figure.



Relationship Diagram for New Project

To create the computerized time card management system for New Project Ltd., the development team needs to perform the following tasks:

1. Create a database called **TimeCard**.
2. Create the schemas called Payment, ProjectDetails, CustomerDetails, and HumanResources.
3. Create the table designs as per the relationship diagram ensuring minimum disk space utilization.
4. Validate the tables as per the following guidelines:

Table:CustomerDetails.Clients

- ClientID should be auto generated.
- CompanyName, ContactPerson, Address, City, State, Zip, Country and Phone should be mandatory.

- Phone must be in the format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: HumanResources.Employees

- EmployeeID should be auto generated.
- Title must have any one of the following values:
- Trainee, Team Member, Team Leader, Project Manager, or Senior Project Manager.
- BillingRate should be greater than 0.
- FirstName and Phone should not be left blank.
- Phone must be in the format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: ProjectDetails.Projects

- ProjectID should be auto generated.
- ProjectName, StartDate, and EndDate should not be left blank.
- BillingEstimate should be greater than 1000 USD.
- EndDate should be greater than the StartDate.
- ClientID is a foreign key from the Clients table.

Table: Payment.PaymentMethod

- PaymentMethodID should be auto generated.
- Description should not be blank.

Table: Payment.Payments

- PaymentID should be auto generated.
- PaymentAmount should be greater than 0.
- PaymentDate should be greater than the end date of the project.
- If a credit card is not used for payments, CreditCardNumber, CardHoldersName, and CreditCardExpDate should be made NULL.
- If a credit card is used, the CreditCardExpDate value should be greater than the payment date.

- ProjectID is a foreign key from the Project table.
- PaymentDue can allow NULL but should not be greater than PaymentAmount.

Table: ProjectDetails.WorkCodes

- WorkCodeID should be auto generated.
- Description should not allow NULL.

Table: ProjectDetails.ExpenseDetails

- ExpenseID should be auto generated.
- Description should not allow NULL.

Table: ProjectDetails.TimeCards

- TimeCardID should be auto generated.
- EmployeeID is a foreign key from the Employee table.
- DateIssued should be greater than the current date and the project start date.
- DaysWorked should be greater than 0.
- ProjectID is a foreign key from the Projects table.
- Billable hours should be greater than 0.
- TotalCost should be automatically calculated by using the following formula: TotalCost=Billable Hours * BillingRate.
- WorkCodeID is a foreign key from the WorkCodes table.

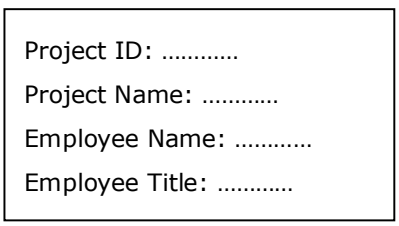
Table: ProjectDetails.TimeCardExpenses

- TimeCardExpenseID should be auto generated.
- TimeCardID is a foreign key from the TimeCards table.
- ExpenseDate should be less than the project end date.
- ExpenseAmount should be greater than 0.
- ExpenseDate should not allow NULL.
- ProjectID is a foreign key from the Projects table.
- ExpenseID is a foreign key from the ExpenseDetails table.

5. Create appropriate relationships between the tables.
6. Document the payment details of each client in a text file.
7. Create appropriate indexes to speed up the executions of the following tasks:
   - Extracting the time card details for an employee based on employee names.
   - Extracting the expense details for all employees with time card IDs in a given range.

- Displaying the name of an employee with the list of all the projects he has worked on.

8. Extracting a list of all projects that need to be completed in the current month.
9. Create a report to display the projects and the details of the employee assigned to a project in the following format.



Format for Project and Employee Details Report

10. Implement a proper security policy in the database. For this, create logins for Sam, John, and Samantha. Sam is the database administrator of the database server. John and Samantha have read/write permissions on the database server.
11. Store all confidential data in encrypted format.
12. Back up the database and store the backup in the C drive.

Case Study 4: Fit-n-Fine Health Club

Fit-n-Fine Health Club is a recreational club that provides various fitness- and health-related services to its members. The club has many branches in the city of Boston. All Fit-n-Fine branches have top-quality gymnasium and health facilities. In addition, the club provides premium services, such as sauna/steam baths, yoga classes, aerobic classes, and an Olympic size swimming pool facility. All the branches are managed by well-qualified instructors/coaches. All these factors have helped the club increase the number of registered members.

Current System

The club's Customer Relations (CR) department serves as a liaison between the club and its members. The main responsibilities of this department are to provide various details about the club to existing and prospective members, perform the registration process for the new members, and monitor the feedback given by the members about the facilities and services of the club. The details about the employees of the CR staff, such as the employee name, designation, address, and phone number are maintained in the Staff Details register.

The CR staff maintains various registers. The Plans Master register contains information related to the various plans offered (Premium/Standard/Guest), the corresponding plan IDs, and the fee for each plan. The Plan Details register contains a list of all the facilities included in each plan. Whenever someone visits the front office of the health club and makes an enquiry about the plans offered, the CR staff refers to the Plan Details register to inform the visitor about the club's services.

The details of the registered members are recorded by the staff in the Member Details register. These details include the member's ID, the name, the subscribed plan, the contact number, and the contact address.

When a member is admitted to the club, he or she needs to pay the joining fee for the plan opted for. The payment details, which include the payment ID, member ID, plan ID, payment amount, payment date, payment method, credit card number, card holder's name, and check number, are stored in the Revenue register. The payment can be made by using cash, check or credit card.

Whenever a member wishes to visit the club to use a facility, he or she needs to book the facility in advance by providing information such as the desired day and time. On receiving such a request, the staff refers to the Booking Details register to check the booking status of the desired facility. If the facility is already booked to capacity for that day and time slot, the new request is rejected. On the other hand, if the number of persons requesting the use of the facility on the desired day and time has not reached the specified

capacity, the new request is accepted and the booking status is updated. The status is marked as booked after the maximum number allowed is reached.

The club members provide regular feedback about the services provided by the club. These comments are accepted by the CR personnel and recorded in the Feedback Register. The entries are made depending on the type of feedback. The entries include information such as the reference number, the member name, the member ID, the feedback details, and the type of feedback. The type of feedback can be:

- Complaints - C
- Suggestions - S
- Appreciations – A

Envisioned System

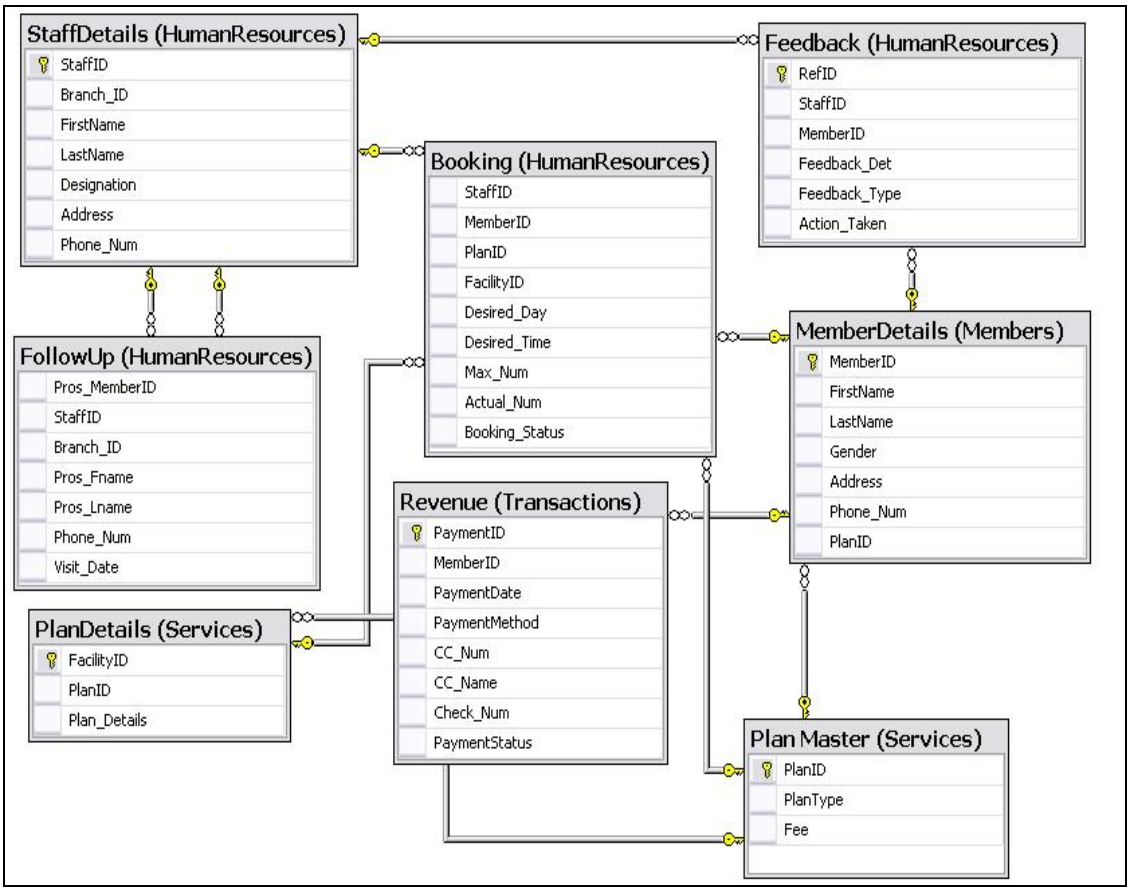
Fit-n-Fine Health Club wants to use a system that can store and manage all information quickly and easily. The manager of the CR department, Jim Morris, realizes that it is becoming increasingly difficult to manually maintain the various registers. Fit-n-Fine also wants to include the information about employees who interact with the members while recording their feedback.

The club management has also decided to maintain details of prospective members who visit a branch and inquire about the services or other details of the club. This will enable the CR personnel to follow up and convert them into members.

Therefore, the management of the club decides to computerize all the processes. It asks IT4Enterprises, Inc. to develop the software. Tony Cage, a project manager working with IT4Enterprises, is assigned this project. After analyzing the requirements, Tony's team has:

1. Identified the various entities involved.
2. Identified the attributes of the various entities. These attributes completely define the entities.
3. Drawn an E/R diagram to demonstrate the relationships between the various entities.
4. Identified the schemas involved. These are HumanResources, Transactions, Services, and Members.
5. Mapped the E/R diagram to the tables.
6. Normalized the tables to 3 NF. The normalized tables have:
    - Table names.
    - Field names, data types, and size of fields.
    - Primary and foreign key identification and integrity.
7. Drawn a diagram to show the relationships between the various tables.

The final relationship diagram created by RedSky Systems is shown in the following figure.



Relationship Diagram for FitnFine

To create the software system for Fit-n-Fine, IT4Enterprises, Inc. needs to perform the following tasks:

1. Create a database called **FitnFine**.
2. Create the table designs as per the relationship diagrams, ensuring minimum disk space utilization.
3. Perform validations on the tables as per the following guidelines:

Table: HumanResources.StaffDetails

- StaffID should be auto generated.
- Branch_ID should be auto generated.
- FirstName, LastName, Designation, Address, and Phone_Num should not be left blank.
- Phone_Num should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: Services.MembershipDetails

- PlanID should be auto generated.
- PlanType should have any of the following values: Premium, Standard, or Guest.
- Fee should store the fee for each plan that is offered by the club.

Table: Services.PlanDetails

- FacilityID should be auto generated.
- PlanID is a foreign key from the MembershipDetails table.
- Plan_Details should store the facilities and services offered under each plan.

Table: Members.MemberDetails

- MemberID should be auto generated.
- FirstName, LastName, Gender, Address, and Phone_Num cannot be left blank.
- PlanID is a foreign key from the MembershipDetails table.
- Gender should have the Male or Female value.
- Phone_Num should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: Transaction.Revenue

- PaymentID must be auto generated.
- MemberID is a foreign key from the MemberDetails table.
- PaymentDate cannot be less than the current date.
- PaymentMethod should have any of the following values:
- Cash, Check, or Credit_Card.
- CC_Num and CC_Name should record the credit card number and card holder's name, respectively. These details should be entered if the payment is made by using a credit card. If a credit card is not being used to make the payment, these fields should be left blank.
- Check_Num will record the check number and should be entered if the payment is made through check. If a check is not being used to make the payment, this field should be left blank.
- PaymentStatus can have either 'Paid' or 'Pending' because its value depends on whether the member has made the payment for the plan opted for or not.

Table: HumanResources.Booking

- StaffID is a foreign key from the StaffDetails table.
- MemberID is a foreign key from the MemberDetails table.
- PlanID is a foreign key from the MembershipDetails table.
- FacilityID is a foreign key from the PlanDetails table.
- Max_Num should store the maximum number of members allowed to use a facility at a given time.
- Actual_Num should store the number of bookings already made by the members for a facility. Its value cannot exceed the value of Max_Num.
- Booking_Status can have either Booked or Available as its value. By default, the status should be marked as Available. The status should be changed to Booked after the value of Actual_Num becomes equal to the value of Max_Num.

Table: HumanResources.Feedback

- RefID should be autogenerated.
- StaffID is a foreign key from the StaffDetails table.
- MemberID is a foreign key from the MemberDetails table.
- Feedback_Type should have any of the following values:

  Complaint, Suggestion, or Appreciation.

Table: HumanResources.FollowUp

- Pros_MemberID should be autogenerated.
- StaffID is a foreign key from the StaffDetails table.
- Branch_ID is a foreign key from the StaffDetails table.
- Pros_Fname, Pros_Lname, and Phone_Num should not be left blank.
- Phone_Num should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

4. Create appropriate relationships between the tables.
5. Store the revenue details for each month in a text file. Make use of the required tools to perform the data transfer.
6. Create appropriate indexes to speed up the execution of the following tasks:
   - Extracting the member details admitted to the club in a particular month.
   - Extracting the member details for each member whose payment is pending.
   - Extracting the plan details, such as the plan name, the facilities offered, and the fee structure for each plan. These reports are required to be distributed to the prospective members who visit a branch and inquire about such details.
   - Displaying a list of all the prospective members who visited a branch in a particular month. The list should also display the name of the branch visited by the person.
   - Displaying a list of feedbacks received from the members in a month and the action taken against a complaint or suggestion.
7. Implement a proper security policy for the database. For this, create logins named Tony, Andrew, Smith, and Bourne. Tony is the database administrator. Andrew, Smith, and Bourne are database developers.
8. Back up the database daily and store the backup in drive C.
9. Store crucial data in encrypted format.
10. Ensure that an alert is sent to Tony whenever the size of the temporary space in the database is less than 20 MB.

Case Study 5: NewHope Hospital

NewHope Hospital is a famous name in the United States in the field of medical sciences. The hospital has a dedicated team of doctors and researchers who are continuously working to build a better and healthy society. Today, NewHope has a capacity of 1,500 beds, and its Out Patient Department (OPD) can handle nearly 100 patients in a day.

Current System

Every year, several thousands of patients are treated at NewHope. The hospital maintains a record of each patient in the Patient Details register. The record of a patient includes the name and address of the patient, nature of treatment being done, and ID of the doctor who is treating the patient. The medical history of each patient is maintained in Medical History register by the NewHope staff. The medical history is maintained for future reference for similar cases or if the same patient is readmitted. The medical history record of a patient is filed according to the ward where the patient was admitted last. Information related to the various wards is recorded in the Ward Details register. The various wards at NewHope hospital are:

- Out Patient Department (OPD)
- Intensive Care Unit (ICU)
- Cardiac Care Unit (CCU)
- Special Wards
- General Wards
- Emergency Unit

The details recorded in the Ward Details register also include the number of beds in each ward and their availability status.

When a patient is admitted to the hospital, his or her representative needs to make an initial payment in the front office. The payment details, which include the payment ID, patient ID, payment amount, payment date, payment method, credit card number, card holder's name, and check number, are recorded in the Payments register. The final payment is to be made at the time of discharge of the patient. The details of the same are also recorded in the Payments register. The payment can be made by using cash, check, or credit card.

The Hospital also maintains a record of all the doctors (resident/visiting) in the Doctor Details register. This register contains details such as the doctor's name, the ID number, the contact number, the employment type (resident/visiting), the area of specialization, and the ward assigned.

The medical history records also show the movement of a patient from one ward to another as recommended by a specified doctor. In addition, the medical history record contains information related to the medical prescription sheet and other tests conducted on the patient.

The format of the medical history record is shown in the following figure.

NewHope Hospital

Medical History Record

Patient Name & Address:

Relative Name & Address:

(Name of Father/Mother in case of Minor)

Age: Blood Group: Height: Weight:

Date When Patient was Admitted:

Nature of Disease:

Current Ward:

Ward to Which Patient was Admitted:

Doctor's Name:

Document's Enclosed:

Doctor's Signature: Patient/Relative Signature:

Medical History Record

Before a patient is shifted to a ward, the staff members check the availability of beds in the given ward by referring to the Ward Details register.

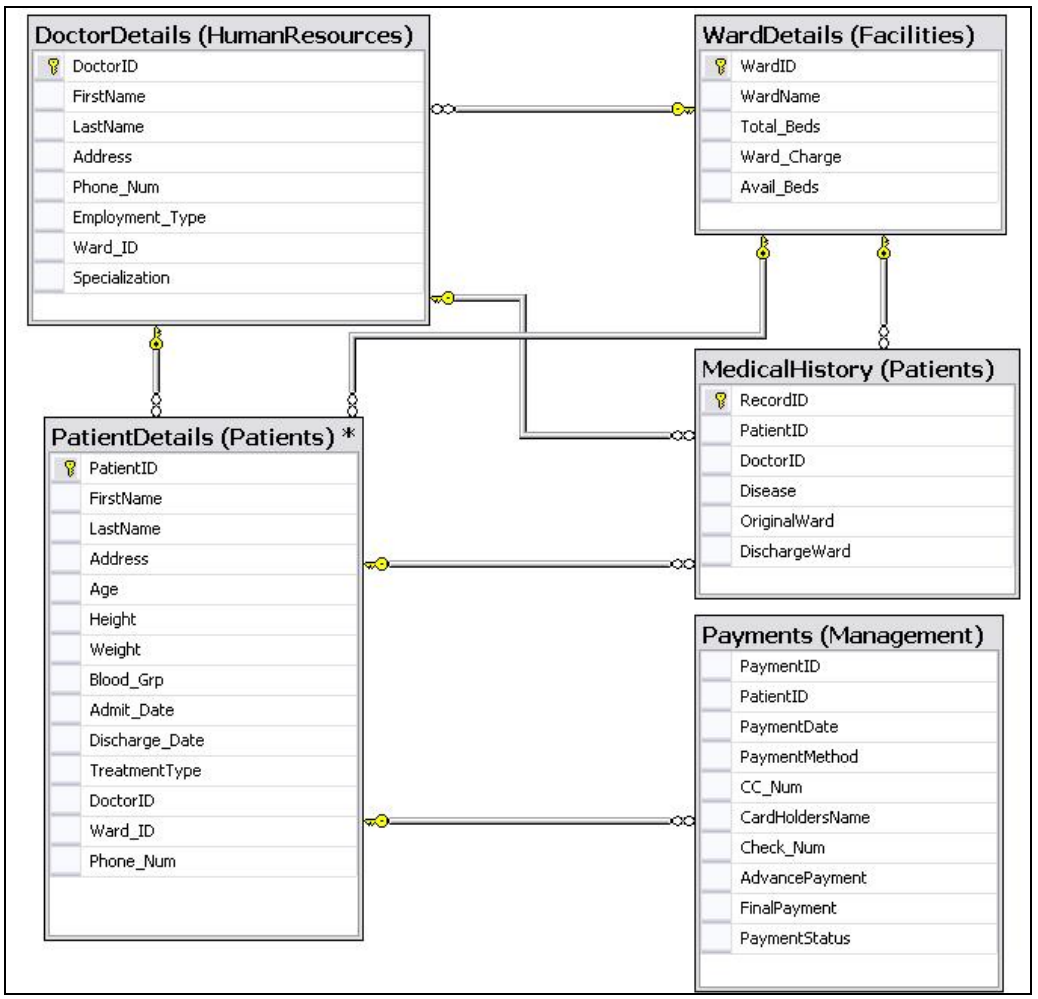
Envisioned System

With the increase in the number of records of patients, it has become difficult for the staff at NewHope to maintain the details of all the patients. Processing the large volume of patient information is becoming difficult and time-consuming. Therefore, the management of NewHope has decided to computerize the process of maintaining the patient records.

The NewHope management has contacted RedSky Systems to develop a software system. RedSky Systems has:

1. Identified the various entities involved.
2. Identified the attributes of the various entities. These attributes completely define the entities.
3. Drawn an E/R diagram to demonstrate the relationships between the various entities.
4. Identified the schemas involved. These are Management, HumanResources, Facilities, and Patients.
5. Mapped the E/R diagram to the tables.
6. Normalized the tables to 3 NF. The normalized tables should have:
   - Table names.
   - Field names, data types, and size of fields.
   - Primary and foreign key identification and integrity.
7. Drawn a diagram to show the relationships between the various tables.

The final relationship diagram created by RedSky Systems is shown in the following figure.



Relationship Diagram for NewHope

To create the software system for NewHope Hospital, RedSky System needs to perform the following tasks:

1. Create a database called **NewHope**.
2. Create the table designs as per the relationship diagram, ensuring minimum disk space utilization.
3. Perform validations on the tables as per the following guidelines:

Table: Patients.PatientDetails

- PatientID should be auto generated.
- FirstName, LastName, Address, and Phone should not be left blank.
- Age, Height, and Weight cannot be less than 0.
- Blood_Grp should have any of the following values: A, B, AB, or O.
- Admit_Date and Discharge_Date cannot be less than the current date.
- DoctorID is a foreign key from the DoctorDetails table.
- Ward_ID is a foreign key from the WardDetails table.
- Phone_Num should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: HumanResources.DoctorDetails

- DoctorID should be auto generated.
- FirstName, LastName, Address, and Phone_Num should not be left blank.
- Employment_Type of each doctor should have one of the following values: Resident or Visiting.
- Ward_ID is a foreign key from the WardDetails table.
- Phone_Num should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111

Table: Facilities.WardDetails

- WardID should be auto generated.
- WardName should have any of the following values:
  OPD, ICU, CCU,Spl_Ward, General_Ward, or Emergency.

Table: Patients.MedicalHistory

- RecordID should be auto generated.
- PatientID is a foreign key from the PatientsDetails table.
- DoctorID is a foreign key from the DoctorDetails table.
- Disease cannot be left blank.
- OriginalWard is a foreign key from the WardDetails table.
- DischargeWard is a foreign key from the WardDetails table.

Table: Management.Payments

- PaymentID must be auto generated.
- PatientID is a foreign key from the PatientDetails table.
- PaymentDate cannot be less than the current date.
- PaymentMethod should have any of the following values:

  Cash, Check, or Credit_Card.
- CC_Num and CC_Name will record the credit card number and card holder's name, respectively. These details should be entered if the payment is made by using a credit card. If a credit card is not used to make the payment, these fields should be left blank.
- Check_Num will record the check number and should be entered if the payment is made through check. If a check is not used to make the payment, this field should be left blank.
- AdvancePayment should store the initial payment received on behalf of the patient, at the time of his or her admission to the hospital.
- FinalPayment should store the final amount chargeable from the patient and is calculated at the time of discharge of the patient.
- The FinalPayment amount must be calculated by using the following formula:

  FinalPayment = Total Bill – AdvancePayment.

  The total bill will be calculated at the time of discharge by adding the ward charges for the number of days the patient was admitted to the ward, the doctor's fee, and other charges as applicable.
- PaymentStatus can have either 'Paid' or 'Pending'. Its value depends on whether the patient has made the initial payment or not.

4. Create appropriate relationships between the tables.
5. In a text file, store the details of all the doctors who have managed a particular ward in the current month. Make use of the required tools to perform the data transfer.
6. Store the payment details for each month in a text file. Make use of the required tools to perform the data transfer.

7. Create appropriate indexes to speed up the execution of the following tasks:
   - Extracting the patient details admitted or discharged on a particular date.
   - Extracting the patient details for each patient whose initial payment is pending.
   - Displaying the number of beds available in each ward along with the doctors assigned to each ward.
8. Implement a proper security policy for the database. For this, create logins named James, Bryan, John, and Maria. James is the database administrator. Bryan, John, and Maria are database developers.
9. Back up the database daily and store the backup in drive C.
10. Store crucial data in encrypted format.
11. Ensure that an alert is sent to James whenever the size of the temporary space in the database is less than 20 MB.

Project Execution

This book contains three case studies. These case studies will be allocated to individual students.

Phases in Project Execution

The project will be carried out in the following phases:

- **System Analysis**: System analysis refers to an in-depth study of the existing system to depict the functionality of the system. The analysis phase is the most crucial phase in a project because it helps developers to identify the processes in the system and functioning of each process. Each student will analyze their respective case studies before moving on to the development phase.
- **Development**: This phase involves developing the project based on the specifications.
- **Testing and debugging**: This phase involves testing the project before submitting it to the coordinator.
- **Documentation**: Documentation is one of the most important aspects of computer programming. The project documentation should be submitted to the coordinator in the formats given in this book before the project walkthrough. The blank report following the case studies is to be filled up, detached from the book, and submitted on the given date.

Project Evaluation Guidelines

The project is to be evaluated based on the following parameters:

- **Quality**: Refers to the following requirements: – 20 Marks
  - The solution maps to the requirements specified along with the case study.
- **Timeliness**: Refers to timely implementation of the project. – 20 Marks
- **Quality of documentation**: Refers to the following requirements: – 40 Marks
  - Completion of all the formats
  - Accuracy of design
  - Adherence to standards and processes
- **Query handling:** Refers to the handling of queries during project walkthrough. – 20 Marks

Project Standards and Guidelines

The following standards and guidelines should be followed while creating the project. Following these standards and guidelines is one of the evaluation criteria for the project:

- A consistent naming convention should be used for all database objects.
- Fields in a table should not have a "." character.

Project Activities

The students will get 16 hours to complete the project. The activities to be performed during this period are:

1. Analyze the case study to identify the system processes.
2. Design the database objects.
3. Test the tables with three or four records.
4. Debug the tables, stored procedures, triggers, and transactions in case of any errors.
5. Create the users and assign each user the required permission on the database and database objects.
6. Create alert, backup, recovery, and mirroring for the projects depending on the requirements.
7. Document the project by using the formats given in the later section.
8. Submit the documentation to the faculty.
9. After submitting the documentation, the student will present the project to the faculty. The faculty will assign marks to the student based on the evaluation criteria specified in this section.

Project Timelines

The tasks listed in the following table should be completed in the specified sessions.

Session #	***Tasks to be Performed***
1	*Analyze the case study and create the table design with appropriate fields and their data types.*
2	*Create and execute the database creation script. Ask your faculty to provide you with a database creation script of the sample case study, Process IT, for reference.* *Create the tables by applying the appropriate constraints and add few records in the tables.*

Session #	***Tasks to be Performed***
3	*According to the validations, create the appropriate stored procedures, triggers, and transactions.*
4	*Test the stored procedures, triggers, and transactions as per the requirements of the project.*
5	*Create the appropriate indexes for the tables as per the queries defined in the requirements.*
6	*Create users and assign permissions to the users.*
7	*Create alerts, backup, recovery and mirroring depending on the requirement.*
8	*Perform final validations and document the project.*

Tasks to be Performed

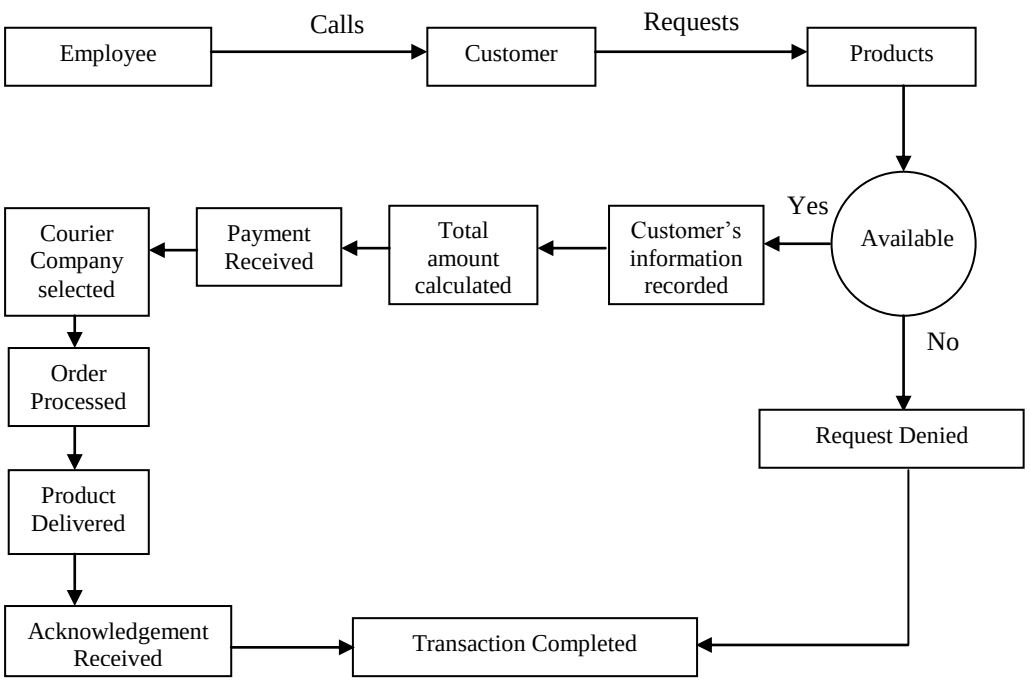
Sample Case Study: Process IT

Process IT is one of the largest order processing companies in Canada. Process IT has various offices in Texas, Los Angeles, and New York. Its head office is in Ottawa.

Current System

To simplify and organize the work, the order processing business of Process IT is divided into various departments. These departments are Shipment, Human Resources, Payment, Order Processing, and Customer Relationships. These departments have a combined strength of more than 1500 employees. Currently, all the work is carried out manually. All departments maintain separate files for storing the details related to employees, orders, shipment, payment, and customers.

The following process diagram explains the working of the Order Processing division.



Process Flow Diagram of Process IT

To make the sales, the employees of Process IT call the people at different locations and explain the features of the products available with them. All employees have a monthly target that they need to attain to get the regular incentives. The following table shows the incentive rates.

Sales	***Percentage***
=<1000$	*5%*
>=1000$ and <10000$	*7%*
>=10000$	*10%*

Incentives Percentage Table

When an order is placed with Process IT, employees are given a form to fill the required details. In addition to all the details required in the form, employees also need to select the required shipping mode. Shipping modes can be Slow, Normal, or Fast. The freight charges are based on the shipping mode.

After the request is received, the availability of the product is checked from the Product file available with the Order Processing department. Further, this request is shared with the Order Processing and Customer Relationship departments. All the details related to the orders are stored in the Orders and OrderDetails files of the Order Processing department. All the information related to the customer is stored in the CustomerDetails file of the Customer Relationship department.

The payments for orders are accepted through cash or credit card. To calculate the total cost for a particular product, the unit price is multiplied by the quantity ordered. Further, taxes are added to the amount. On the basis of the frequency of the orders placed, a status is provided to the customers. The status can be Regular, Occasional, or First Timer. Customers are offered discounts based on the total amount and their status.

When the customer makes the payment, the Payments file is updated with the data, such as payment code, order code, payment amount, payment date, and payment mode. If a credit card is used, details such as card number, cardholder's name, and card expiry date are required.

After the payment is received, the Payment file of the Payment department is updated. Further, an entry is made in the ShipmentDetails files available with the Shipment department. On the basis of the shipment mode selected by the customer, the courier company is selected to dispatch the product. After the customer receives the product, the Shipment file is updated with the acknowledgement of the receipt and the order is consider to be dispatched.

Next, the status of the order is updated to "Delivered" in the Orders file once the customer accepts the product and signs an acknowledgement receipt for the same.

This process of the order processing department is performs as desired but its execution takes a lot of time to complete. For example, if a customer calls to inquire the status of the order placed by him or her, all the files related to customer, orders, and shipment are searched through to inform the customer about the current status.

Now, with an exponential growth in business, it is becoming difficult for the company to follow the above-mentioned process and to record all the required information manually. In addition, the concerned employees are facing problems in maintaining and retrieving the recorded data. Therefore, the management wants that the data should always be available with the branch offices and should be easily accessible.

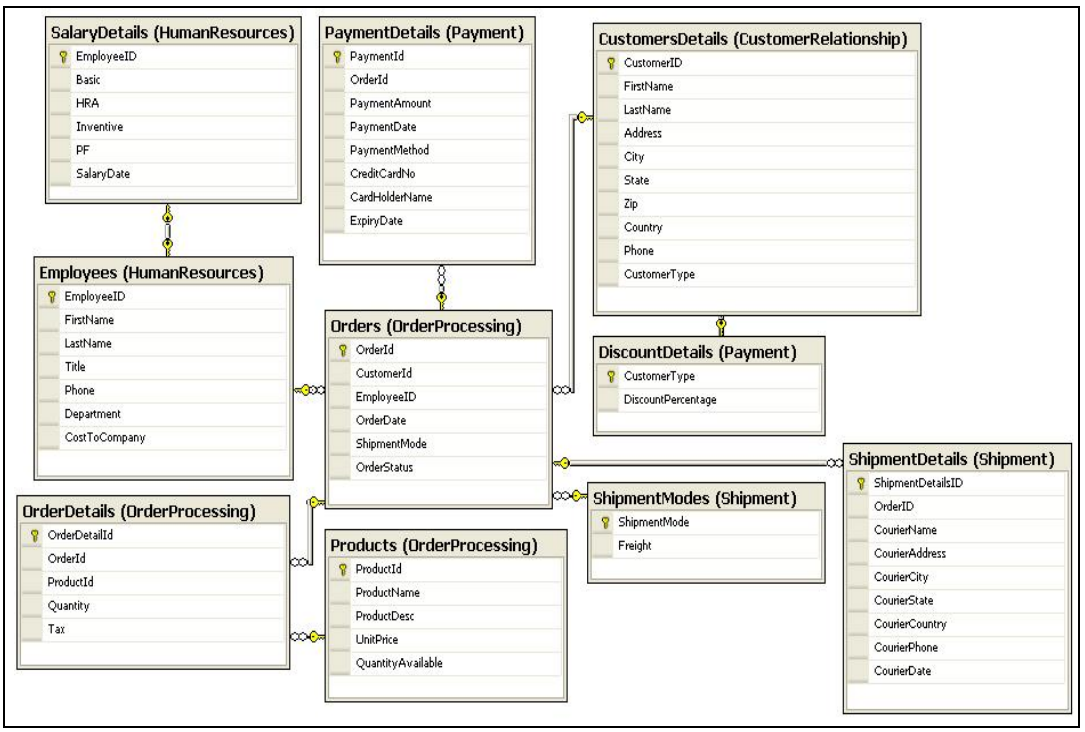
Envisioned System

The management of Process IT has decided to implement a computerized transactional management system in the organization. This will provide the company with a reliable transaction storing and processing mechanism. In addition, it will enable the head office to communicate quickly with the branch offices and its clients.

Process IT has asked DbDesign, Inc. to create a database design for the company. A team from DbDesign, Inc. has:

1. Identified the various entities involved.
2. Identified the attributes of the various entities. The attributes completely define the entities.
3. Drawn an E/R diagram to demonstrate the relationships between the various entities.
4. Mapped the E/R diagram to the tables.
5. Normalized the tables to 3 NF. The normalized tables have:
   - Table names.
   - Field names, data types, and size of fields.
   - Primary and foreign key identification and integrity.
6. Drawn a diagram to show the relationships between the various tables.
7. Identified the schemas as HumanResources, Payment, CustomerRelationship, OrdeProcessing, and Shipment.

The final relationship diagram created by the project team is shown in the following figure.



Relationship Diagram of Process IT

To implement and maintain the database design, Process IT has recruited a database developer's team. To implement the database design, the database team needs to perform the following tasks:

1. Create a database called **ProcessIT**.
2. Create the schemas called Shipment, HumanResources, Payment, OrderProcessing, and CustomerRelationship.
3. Create the table designs as per the relationship diagram, ensuring appropriate disk space utilization.
4. Validate the tables as per the following guidelines:

Table: CustomerRelationship.CustomersDetails

- CustomerID must be auto generated.
- FirstName, LastName, Address, City, State, and Phone should not be left blank.

- Phone should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111
  CustomerType should be Regular, Occasional, or First Timer.

Table: HumanResources.Employees

- EmployeeID should be automatically generated.
- Phone should be in the following format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111
- Title should be Executive, Senior Executive, Manager, or Chairman.
- FirstName and LastName should not be left blank.
- Department should be Shipment, Human Resources, Payment, Order Processing, or Customer Relationship.

Table: OrderProcessing.Products

- ProductID must be auto generated.
- UnitPrice should be greater than 0.
- QuantityAvailable should not be zero or negative.
- ProductName, ProductDesc and UnitPrice should not be left blank.

Table: Shipment.ShipmentModes

- ShipmentMode can have any of the following values:
  Fast, Slow, or Normal.
- Freight should not be less than 0.

Table: OrderProcessing.Orders

- OrderID must be auto generated.
- OrderDate should not be greater than the current date.
- If the OrderDate is not entered, the current date should be taken as the default order date.
- When an entry is made in the Orders table, the default value for the OrderStatus field should be made as "In-Transit".

Table: Shipment.ShipmentDetails

- CourierName, CourierAddress, CourierCity, CourierState, CourierPhone, and CourierCountry fields should not be left blank.
- CourierPhone should be in the format:
  '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'
  Example: 11-111-1111-111-111
- CourierDate should be greater than the OrderDate.

Table: OrderProcessing.OrderDetails

- OrderDetailId must be auto generated.
- Quantity should be greater than 0.
- Tax should be calculated by using the following formula:
  *Total Amount= OrderDetails.Quantity * Products.UnitPrice*

Table: Payment,PaymentDetails

- PaymentId should be auto generated.
- PaymentAmount should be greater than 0.
- PaymentAmount should be equal to the sum of the total cost for a given order id in the OrderDetails table, including the tax and freight charges. If the amounts do not match, the transaction should be rejected and a user-friendly message should be displayed to the user.
- PaymentDate should not be less than the OrderDate.
- CreditCardNo should be in the format [0-9][0-9] [0-9] [0-9]-[0-9][0-9][0-9][0-9]- [0-9] [0-9][0-9][0-9] and should allow null.
- PaymentAmount and PaymentDate fields should not be left blank.
- If a credit card is used, its details should be filled in the appropriate fields.
- If a credit card is not used, its details should be left blank.
- ExpiryDate for the credit card should be greater than the current date.

Table: Payment.DiscountDetails

- CustomerType should be Regular, Occasional, or First Timer.
- DiscountPercentage should be 10 for regular, 7 for occasional, and 5 for first-time customers.

Table: HumanResources.SalaryDetails

- Basic, HRA, Incentives, and PF should not be left blank.
- Basic, HRA, and PF should be more than 0.

5. Create appropriate relationships between the tables.
6. Create appropriate indexes to speed up the execution of the following tasks:
   - Extract order details of the orders placed on a particular date along with the product details.
   - Extract details of all orders whose payments are pending.
   - Display the customer details of all customers living in a particular city.
   - Display the customer details of all customers with a given first name and last name.
7. Generate the yearly sales report. This report needs to be sent to each customer.
8. Simplify the following tasks:
   - Calculation of the total cost against a particular order.
   - Calculation of incentives for a particular employee.
9. Ensure that the database is available 24x7 with the least possible down time. Process IT is willing to use an additional server for this purpose.
10. Ensure that a proper security policy is implemented on the database server. The management wants that only five users, Sam, Billy, Joe, Clients, and Branch, should be able to access the database. Joe is the administrator of the database and the others have only read/write permissions.
11. Store all critical data in encrypted format.
12. Ensure that an alert message is sent to Joe if the database server fails.

Solution to Sample Case Study: Process IT

To create the **ProcessIT** database and its objects, you need to make use of the following SQL Scripts.

1. **CreateProcessITDatabase.sql**: This script contains the code for creating a database.
2. **CreateProcessITDatabaseObjects.sql**: This script contains the statements required to create the tables of the database.
3. **CreateProcessITRelationships.sql**: This script contains the statements required to create the relationships between the tables.
4. **CreateProcessITConstraints.sql**: This script contains the SQL statements used for creating constraints, rules, and defaults as per the validations required.
5. **CreateProcessITProcedures.sql**: This script contains the batch statements that create the generic procedures and triggers in the database.
6. **CreateProcessITIndexes.sql**: This script contains the SQL statements used to create indexes to speed up the execution of the frequently used queries.
7. **CreateProcessITUsers.sql**: This script contains the SQL queries used to create the users and logins in the database server.
8. **ProcessITEncryption.sql**: This script contains statements that encrypt the values of the CreditCardNumber column.
9. **ProcessITAlert.sql**: This script contains statements required to set-up the required alerts.
10. **ProcessITMirroringPart1.sql**: This script contains the statements to set up the database mirroring on the principal instance.
11. **ProcessITMirroringPart2.sql**: This script contains the statements to set up the database mirroring on the mirror instance.

Guidelines for Executing the Solution

Before executing the solution, you need to keep the following points in mind:

- All the queries should be executed with a login with administrative privileges on the SQL Server.
- Ensure that the queries are executed in the same order as listed above.
- To implement database mirroring, you need to install another instance of SQL Server 2005, which will act as the mirror server. ProcessITPart1.sql will be executed on the principal server, and ProcessITPart2.sql will be executed on the mirror server. Ensure that you do not have any mirroring endpoint specified on any of the servers.

Executing the Solution

To execute the solution, you need to perform the following steps:

1. Create a folder named **DataFiles** in the drive **C** of your computer and copy all the solution scripts into it from the **TIRM/Project Solution/Process IT** folder.
2. Execute the following SQL scripts in sequence:
   a. CreateProcessITDatabase.sql
   b. CreateProcessITDatabaseObjects.sql
   c. CreateProcessITRelationships.sql
   d. CreateProcessITConstraints.sql
   e. CreateProcessITProcedures.sql
   f. CreateProcessITIndexes.sql
   g. CreateProcessITUsers.sql
   h. ProcessITEncryption.sql
3. Set the recovery model of the database to full, take the full backup of the **ProcessIT** database, and restore it with the **NO-RECOVERY** option to the mirror server. Performing these steps allow you to implement database mirroring. Further, you need to execute the **ProcessITMirroringPart1.sql** file on the principal instance. Next, execute the **ProcessITMirroringPart2.sql** file on the mirror instance. Finally, execute the **ProcessITMirroringPart3.sql** file on the mirror instance and execute the **ProcessITMirroringPart4.sql** file on the principal instance.

Sample Project Documentation: Process IT

PROJECT ON

Process IT Database Management System

Developed by

Name: Debbie Howe Reg. No.: 4701-10-258

Process IT Database Management System

(Project Title)

Batch Code	: B010101
Start Date	: June 1, 2007
End Date	: June 10, 2007
Name of the Coordinator	: Alex Norton
Name of Developer	: Debbie Howe
Date of Submission	: June 11, 2007

NIIT

CERTIFICATE

This is to certify that this report titled *Process IT Database Management System* embodies the original work done by *Debbie Howe* in partial fulfillment of their course requirement at NIIT.

Coordinator:

Alex Norton

ACKNOWLEDGEMENT

We have benefited a lot from the feedback and suggestions given to us by Mr. Alex Norton and other faculty members.

SYSTEM ANALYSIS

System Summary: *Process IT is a consultancy firm that manages order processing for its customers.*

The company has decided to stop manual data entry. Instead, it has decided to make use of a computerized database management for processing orders and payment realizations. This will enable the company to improve communication with its clients.

DATABASE DESIGN

Database Name: *ProcessIT*

Number of Schemas: *5*

Schema Names:

1. *CustomerRelationship*
2. *HumanResources*
3. *Payment*
4. *Shipment*
5. *OrderProcessing*

Number of tables: *10*

Table Names:

1. *CustomerRelationship.CustomersDetails*
2. *HumanResources.Employees*
3. *HumanResources.SalaryDetails*
4. *Payment.PaymentDetails*
5. *Payment.DiscountDetails*
6. *Shipment.ShipmentDetails*
7. *Shipment.ShipmentModes*
8. *OrderProcessing.OrderDetails*
9. *OrderProcessing.Orders*
10. *OrderProcessing.Products*

SCHEMATIC DIAGRAM OF THE DATABASE

Database Name: *ProcessIT*

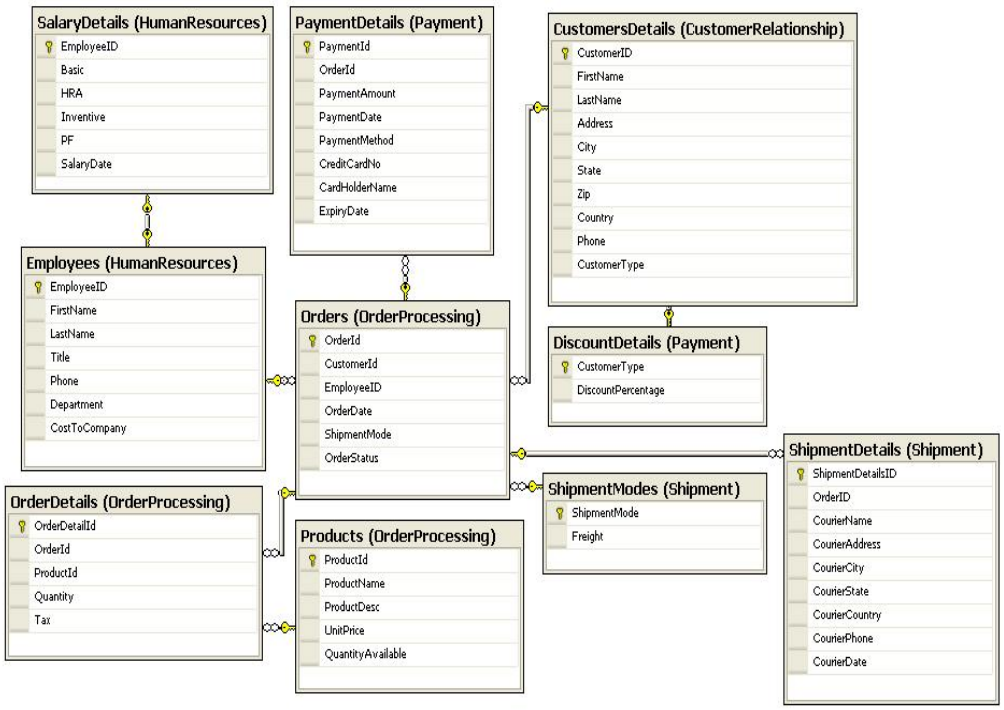


TABLE DESIGN

Database Name: *ProcessIT*

Field Name	*Data type*	*Width*	*Description*
CustomerID	*int*		*Customer number*
FirstName	*varchar*	*30*	*Customer first name*
LastName	*varchar*	*30*	*Customer last name*
Address	*varchar*	*50*	*Customer address*
City	*varchar*	*30*	*Customer city*
State	*varchar*	*30*	*Customer state*
Zip	*char*	*10*	*Zip code*
Country	*varchar*	*20*	*Customer country*
Phone	*Char*	*15*	*Phone number*
CustomerType	*varchar*	*20*	*Type of the customer*

CustomerRelationship.CustomersDetails Table

Field Name	*Data type*	*Width*	*Description*
EmployeeID	*int*		*Employee number*
FirstName	*varchar*	*30*	*Employee first name*
LastName	*varchar*	*30*	*Employee last name*
Title	*varchar*	*30*	*Employee designation*
Phone	*char*	*15*	*Phone number*
Department	*varchar*	*20*	*Name of the department of the employee*
CostToCompany	*money*		*The cost to the company*

HumanResources.EmployeesTable

Field Name	*Data type*	*Width*	*Description*
EmployeeID	*int*		*ID of the employee*
Basic	*int*		*Basic pay of the employee*
HRA	*int*		*HRA of the employee*
Incentive	*int*		*Incentives received on the base of the sales*
PF	*int*		*PF deduction to be made from the salary*
SalaryDate	*datetime*		*Date of the payment*

HumanResources.SalaryDetails

Field Name	*Data type*	*Width*	*Description*
ShipmentMode	*varchar*	*5*	*Mode of shipment*
Freight	*int*		*Freight charges*

Shipment.ShipmentModes

Field Name	*Data type*	*Width*	*Description*
OrderDetailID	*int*		*Identification number of an order detail*
OrderID	*int*		*Identification number of an order*
ProductID	*int*		*Identification number of the product sold*
Quantity	*int*		*Quantity of the product sold*
Tax	*int*		*Tax on the cost of the product*

OrderProcessing.OrderDetails

Field Name	*Data type*	*Width*	*Description*
OrderID	*int*		*Identification number of an Order*
CustomerID	*int*		*Identification number of the customer*
EmployeeID	*int*		*Identification number of the employee*
OrderDate	*datetime*		*Date of the order*
ShipmentMode	*varchar*	*5*	*Mode of shipment*
OrderStatus	*varchar*	*20*	*Status of the Order*

OrderProcessing.Orders

Field Name	*Data type*	*Width*	*Description*
ProductID	*int*		*Identification number of a product*
ProductName	*varchar*	*30*	*Name of the product*
ProductDesc	*varchar*	*50*	*Description of the product*
UnitPrice	*money*		*Unit price of the product*
QuantityAvailable	*int*		*Quantity in hand*

OrderProcessing.Products

Field Name	*Data type*	*Width*	*Description*
PaymentId	*int*		*Payment number*
OrderId	*char*	*4*	*Order number*
PaymentAmount	*money*		*Amount paid for the order*
PaymentDate	*datetime*		*Date of payment*
PaymentMethod	*char*	*5*	*Payment method number*
CreditCardNo	*char*	*20*	*Credit card number*
CardHolderName	*varchar*	*30*	*Credit card holder's name*
ExpiryDate	*datetime*		*Expiry date of the credit card*

Payment.PaymentDetails

Field Name	Data type	Width	Description
CustomerType	varchar	20	Type of the customer
DiscountPercentage	int		Percentage of discount

Payment.DiscountDetails

Field Name	Data type	Width	Description
ShipmentDetailsID	int		Shipment details number
OrderID	int		Order number
CourierName	varchar	30	Name of the parcel company
CourierAddress	varchar	30	Place for shipment
CourierCity	varchar	30	Shipment city
CourierState	varchar	30	Shipment state
CourierCountry	varchar	20	Shipment country
CourierPhone	char	15	Shipment phone
CourierDate	datetime		Date of the shipment

Shipment.ShipmentDetails

VALIDATIONS PERFORMED

Validation Required	*Method(s) Used For Validation*
CustomerID must be unique.	*Primary key constraint*
CustomerID must be auto generated.	*Identity specification*
FirstName, LastName, Address, City, State, and Phone should not be left blank.	*NOT NULL clause in the CREATE TABLE statement*
Phone should be entered in the format '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9] [0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'	*CHECK constraint*
CustomerType should be Regular, Occasional, or First Timer.	*CHECK constraint*

HumanResources.Employees

Validation Required	*Method(s) Used For Validation*
EmployeeID should be automatically generated.	*Identity Specification*
Phone should be entered in the format '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9] [0-9] [0-9]'	*CHECK constraint*
Title should be Executive, Senior Executive, Manager, or Chairman.	*CHECK constraint*
Department should be Shipment, Human Resources, Payment, Order Processing, or Customer Relationship.	*CHECK constraint*
FirstName, LastName, Title, and Phone should not be left blank.	*NOT NULL clause in the CREATE TABLE statement*

CustomerRelationship.CustomersDetails

Validation Required	*Method(s) Used For Validation*
ProductId must be auto generated.	*Identity Specification*
UnitPrice should be greater than 0.	*CHECK constraint*
QuantityAvailable should not be in negative.	*CHECK constraint*
ProductName, description, and UnitPrice should not be left blank.	*NOT NULL clause in the CREATE TABLE statement*

OrderProcessing.Products

Validation Required	*Method(s) Used For Validation*
Freight should not be less than 0.	*CHECK constraint*

Shipment.ShipmentModes

Validation Required	*Method(s) Used For Validation*
OrderID must be auto generated.	*Identity specification*
OrderDate should not greater than the current date.	*CHECK constraint*
If the OrderDate is not entered, the current date should be taken as the default order date.	*DEFAULT constraint*
When an entry is made in the Orders table, the default value for the status should be In-Transit.	*DEFAULT constraint*

OrderProcessing.Orders

Validation Required	*Method(s) Used For Validation*
CourierName, CourierAddress, CourierCity, CourieState,CourierPhone, and CourierCountry should not be left blank.	*NOT NULL keyword with the CREATE TABLE statement*
CourierPhone should be entered in the format '[0-9][0-9]-[0-9] [0-9][0-9]-[0-9] [0-9][0-9][0-9]-[0-9][0-9] [0-9]-[0-9] [0-9] [0-9]'	*Rule*
The CourierDate should be greater than the order date.	*Trigger*

Shipment.ShipmentDetails

Validation Required	*Method(s) Used For Validation*
OrderDetailID must be unique.	*Primary Key constraint*
OrderDetailId must be auto generated.	*Stored Procedure and batch programming*
Quantity should not be left blank.	*NOT NULL clause in the CREATE TABLE statement*
Total cost should be calculated automatically and added to the TotalCost field.	*Trigger*
Quantity should be greater than 0.	*Rule*

OrderProcessing.OrderDetails

Validation Required	*Method Used For Validation*
PaymentId must be unique.	*Primary Key constraint*
PaymentId should be auto generated.	*Stored Procedure and batch programming*
PaymentAmount should be greater than 0.	*CHECK constraint*
PaymentAmount should be equal to the sum of the total costs for a given OrdeId in the OrderDetails table. If the amounts do not match, the transaction should be rejected and a message should be displayed to the user.	*Triggers, Aggregate Functions, and Joins*
ShipmentDate should be updated to two days ahead of the PaymentDate if the transaction is successful.	*Trigger and Update statement*
The PaymentDate should be greater than the OrderDate.	*Trigger and batch programming*
CreditCardNo must be in the following format, '[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'.	*CHECK constraint*
PaymentAmount and PaymentDate fields should not be left blank.	*NOT NULL clause in the CREATE TABLE statement*
If PaymentMethod is 'Credit Card', ensure that the credit card details are not blank.	*Trigger*
If PaymentMethod is not 'Credit Card', ensure that the credit card details are blank.	*Trigger*
ExpiryDate for the credit card should be greater than the current date.	*Trigger*

Payment.PaymentDetails

REPORTS OUTLINE

Report Name	*Report Type*	*Description*	*Tables/Queries Used*
Customer Report	*Query Output*	*This report displays the order details such as the Order Detail Id, Order Id, Product Name, Unit Price, Freight Charge, Tax, Discount, Total Cost, and Quantity for a given customer id.*	*OrderDetails, Orders, ShipmentModes, CustomerType, and CustomersDetails*

Details of the Generated Report(s)

CONFIGURATION

Hardware: *PC compatible with an Intel Pentium-IV processor, 512-MB RAM, and 40GB of Hard disk*

Operating System: *Microsoft Windows XP with Service Pack 2*

Software: *Microsoft SQL Server 2005 Standard Edition*

PROJECT FILE DETAILS

S.No	File Name	Remarks
1	*ProcessIT.mdf* *ProcessIT_Log.ldf*	*SQL Server database that contains tables, procedures, triggers, constraints, and queries.*

PROJECT ON

Developed by

Name: Reg. No.:

NIIT

(Project Title)

Batch Code :

Start Date :

End Date :

Name of the Coordinator :

Names of the Developer :

Date of Submission :

NIIT

CERTIFICATE

This is to certify that this report, titled _____ embodies the original work done by _____, in partial fulfillment of his course requirement at NIIT.

Coordinator:

ACKNOWLEDGEMENT

SYSTEM ANALYSIS

System Summary:

DETAILED DESIGN DOCUMENT FOR <PROJECT TITLE>

VALIDATION PERFORMED

REPORTS OUTLINE

CONFIGURATION

Hardware:

Operating system:

Software: