Project

Instructor Inputs

Case Study 1: Shop Here

The entities for the Shop Here database management system are:

- Employee
- ItemDetails
- OrderDetails
- SupplierDetails
- ProductCategory

The attributes of the entities listed above are as follows:

- **Employees**: Employee ID, First Name, Last Name, City, Phone
- **ItemDetails**: Item ID, Item Name, Item Description, Unit Price, Quantity In Hand, Reorder Level, Reorder Quantity, Category ID, Supplier ID
- **OrderDetails**: Purchase Order ID, Employee ID, Order Date, Receiving Date, Item ID, Quantity Ordered, Quantity Received, Unit Price, Ship Method, Order Status
- **SupplierDetails**: Supplier ID, First Name, Last Name, Address, Phone, Country
- **ProductCategory**: Category ID, Category Name, Category Description

To create a computerized transaction system for Shop Here using SQL Server 2005, the project team will have 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 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'.
- ReceivingDate 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

The entities for the Showman House database management system are:

- Customers
- Events
- Employee
- EventType
- Payments
- PaymentMethod

The attributes of the entities listed above are:

- **Customers**: Customer ID, Name, Address, City, State, Phone
- **Events**: Event ID, Event Name, Event Type ID, Location, Start Date, End Date, Staff Required, Employee ID, Customer ID, No Of People
- **Payments**: Payment ID, Event ID, Payment Amount, Payment Date, Credit Card Number, Card Holders Name, Credit Card Expiry Date, Payment Method ID, Cheque No.
- **PaymentMethods**: Payment Method ID, Description
- **Employee**: Employee ID, First Name, Last Name, Address, Phone, Title
- **EventType**: Event Type ID, Description, Charge Per Person

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, 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

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.

The entities for the New Project database management system are:

- Employees
- Projects
- Clients
- Time Cards
- Work Codes
- Time Card Expenses
- Time Card Hours
- Expense Codes
- Payments
- Payment Methods

The attributes of the entities listed above are:

- **Employees**: Employee ID, First Name, Last Name, Title, Phone, Billing Rate
- **Projects**: Project ID, Project Name, Project Description, Client ID, Billing Estimate, Employee ID, Start Date, End Date
- **Clients**: Client ID, Company Name, Address, City, State, Zip, Country, Contact Person, Phone
- **Time Cards**: Time Card ID, Employee ID, Date Issued
- **Time Card Hours**: Time Card Detail ID, Time Card ID, Date Worked, Project ID, Work Description, Billable Hours, Total Cost, Work Code ID
- **Work Codes**: Work Code ID, Description
- **Time Card Expenses**: Time Card Expense ID, Time Card ID, Expense Date, Project ID, Expense Description, Expense Amount, Expense Code ID
- **Expense Codes**: Expense Code ID, Description
- **Payments**: Payment ID, Project ID, Payment Amount, Payment Date, Credit Card Number, Card Holders Name, Credit Card Expiry Date, Payment Method ID
- **Payment Methods**: Payment Method ID, Description

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

```
Project ID: ...........
Project Name: ...........
Employee Name: ...........
Employee Title: ...........
```

Format for Project and Employee Details Report

9. 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.
10. Store all confidential data in encrypted format.
11. Back up the database and store the backup in the C drive.

Case Study 4: Fit-n-Fine Health Club

The entities for the Fit-n-Fine health club database management system are:

- StaffDetails
- MemberDetails
- Booking
- Revenue
- PlanDetails
- PlanMaster
- Feedback
- FollowUp

The attributes of the entities listed above are:

- **StaffDetails**: StaffID, Branch_ID, FirstName, LastName, Designation, Address, Phone_Num
- **MemberDetails**: MemberID, FirstName, LastName, Gender, Address, Phone_Num, PlanID
- **Booking**: StaffID, MemberID, PlanID, FacilityID, Desired_Day, Desired_Time, Max_Num, Actual_Num, Booking_Status
- **Revenue**: PaymentID, MemberID, PaymentDate, PaymentMethod, CC_NUM, CC_Name, Check_Num, PaymentStatus
- **PlanDetails**: FacilityID, PlanID, Plan_Details
- **PlanMaster**: PlanID, PlanType, Fee
- **Feedback**: RefID, StaffID, MemberID, Feedback_Det, Feedback_Type, Action_Taken
- **FollowUp**: Pros_MemberID, StaffID, Branch_ID, Pros_Fname, Pros_Lname, Phone_Num, Visit_Date

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 three 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 three 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

The entities for the NewHope hospital database management system are:

- DoctorDetails
- PatientDetails
- WardDetails
- MedicalHistory
- Payments

The attributes of the entities listed above are:

- **DoctorDetails**: DoctorID, FirstName, LastName, Address, Phone_Num, Employment_Type, Ward_ID, Specialization
- **PatientDetails**: PatientID, FirstName, LastName, Address, Age, Height, Weight, Blood_Grp, Admit_Date, Discharge_Date, Treatment_Type, DoctorID, Ward_ID, Phone_Num
- **WardDetails**: WardID, WardName, Total_Beds, Ward_Charge, Avail_Beds
- **MedicalHistory**: RecordID, PatientID, DoctorID, Disease, OriginalWard, DischargeWard
- **Payments**: PaymentID, PatientID, PaymentDate, PaymentMethod, CC_Num, CardHoldersName, Check_Num, AdvancePayment, FinalPayment, PaymentStatus

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

The suggestions for the faculty for project allocation and evaluation are:

- The project should be allocated to individual students by the end of the first cycle.
- The student can start the project earliest after the Machine Room session of the second cycle.
- During allocation, explain to the students the scope of the project by referring to the topic, Project Activities and Project Timelines given in Project SG.
- Ask the students to refer to the sample case study given in Project SG.
- Provide the students with the solution and sample project documentation of the sample case study.
- Ask the students to refer to the topic, Project Standards and Guidelines given in Project SG before starting the project documentation.
- Before project evaluation day, ask the students to verify their projects according to the standards and guidelines given in the topic, Project Standards and Guidelines given in Project SG.
- Evaluate the students according to the guidelines given in the topic, Project Evaluation Guidelines given in Project SG.

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 provided in the **Project Solution/ProcessIT** folder of the TIRM CD:

- **CreateProcessITDatabase.sql**: This script contains the code for creating a database.
- **CreateProcessITDatabaseObjects.sql**: This script contains the statements required to create the tables of the database.
- **CreateProcessITRelationships.sql**: This script contains the statements required to create the relationships between the tables.
- **CreateProcessITConstraints.sql**: This script contains the SQL statements used for creating constraints, rules, and defaults as per the validations required.
- **CreateProcessITProcedures.sql**: This script contains the batch statements that create the generic procedures and triggers in the database.
- **CreateProcessITIndexes.sql**: This script contains the SQL statements used to create indexes to speed up the execution of the frequently used queries.
- **CreateProcessITUsers.sql**: This script contains the SQL queries used to create the users and logins in the database server.
- **ProcessITEncryption.sql**: This script contains statements that encrypt the values of the CreditCardNumber column.

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 SQL Server.
- Ensure that you have enabled the FILESTREAM data access on the SQL Server instance and configured the FILESTREAM access level as 2 by using the following statements:

  ```
  USE master
  EXEC sp_configure filestream_access_level, 2
  ```

  Then, execute the following statement to apply the configuration changes:

  ```
  RECONFIGURE
  ```

- Ensure that the queries are executed in the same order as listed above.

Executing the Solution

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

1. Create a folder named **DataFiles** in the **C** drive of the computer where the instance of SQL Server is installed and copy all the solution scripts into it from the **Project Solution/ProcessIT** folder of the TIRM CD.
2. Create a folder named **Data** in the **C** drive of the computer where the instance of SQL Server is installed.
3. 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

PROJECT ON

Process IT Database Management System

Developed by

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

NIIT

Process IT Database Management System

(Project Title)

Batch Code	: B010101
Start Date	: June 1, 2010
End Date	: June 10, 2010
Name of the Coordinator	: Alex Norton
Name of Developer	: Debbie Howe
Date of Submission	: June 11, 2010

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:

a. *CustomerRelationship*
b. *HumanResources*
c. *Payment*
d. *Shipment*
e. *OrderProcessing*

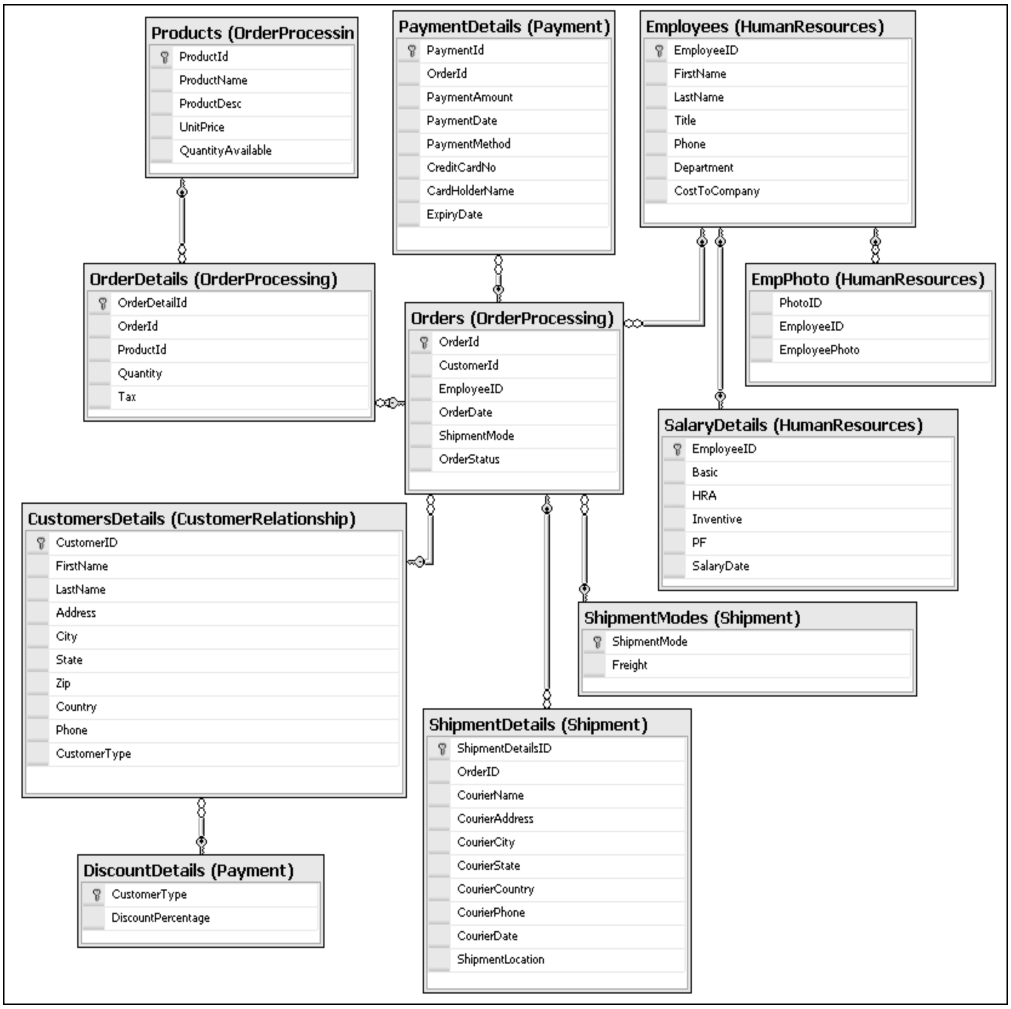
Number of tables: *11*

Table Names:

a. *CustomerRelationship.CustomersDetails*
b. *HumanResources.Employees*
c. *HumanResources.EmpPhoto*
d. *HumanResources.SalaryDetails*
e. *Payment.PaymentDetails*
f. *Payment.DiscountDetails*
g. *Shipment.ShipmentDetails*
h. *Shipment.ShipmentModes*
i. *OrderProcessing.OrderDetails*
j. *OrderProcessing.Orders*
k. *OrderProcessing.Products*

SCHEMATIC DIAGRAM OF THE DATABASE

Database Name: *ProcessIT*



Schematic Diagram of ProcessIT Database

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*

The 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*

The HumanResources.Employees Table

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*

The HumanResources.SalaryDetails Table

Field Name	***Data Type***	***Width***	***Description***
PhotoID	*UNIQUEIDENTIFIER ROWGUIDCOL*		*Photo number*
EmployeeID	*int*		*Employee number*
EmployeePhoto	*VARBINARY(MAX) FILESTREAM*		*Photo of the employees*

The HumanResources.EmpPhoto Table

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

The Shipment.ShipmentModes Table

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*

The OrderProcessing.OrderDetails Table

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*

The OrderProcessing.Orders Table

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*

The OrderProcessing.Products Table

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*

The Payment.PaymentDetails Table

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

The Payment.DiscountDetails Table

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*
ShipmentLocation	*geography*		*Location of shipment*

The Shipment.ShipmentDetails Table

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*

The Shipment.ShipmentDetails Table

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*

The OrderProcessing.OrderDetails Table

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]'.	*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*

The Payment.PaymentDetails Table

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*