Guide to Designing and Implementing a Resort Management Database System

Introduction

A well-designed database system is crucial for efficient resort management, enabling seamless operations, data analysis, and customer service. This guide provides step-by-step instructions for designing and implementing a Resort Management Database System using Microsoft SQL Server Management Studio (SSMS), along with best practices for data modeling, security, and visualization.

Prerequisites

Before starting, ensure you have:

- Microsoft SQL Server Management Studio installed
- Basic understanding of database design principles and SQL
- Access to PowerBI for data visualization

Step 1: Database Design

1. Create an Entity-Relationship Diagram (ERD)

- Identify key entities such as Resort, Employee, Customer, Room and Booking
- Define relationships between entities (e.g., one-to-many between Resort and Employee)
- Normalize the database structure to minimize redundancy

2. Implement the Database Schema

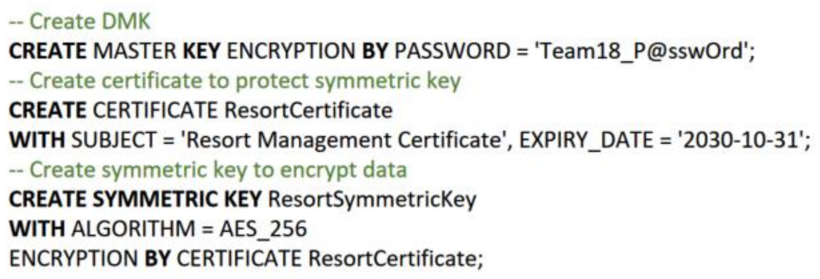
- Use SQL Data Definition Language (DDL) to create tables
- Define primary keys, foreign keys, and constraints

Step 2: Data Security Implementation

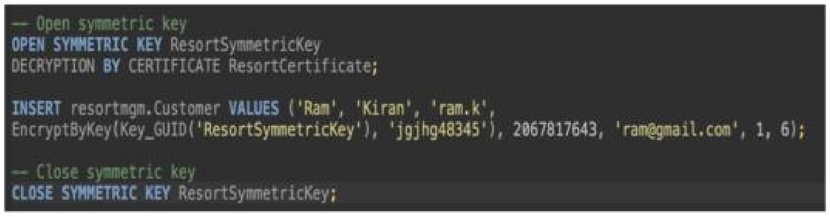
1. Set Up Encrypted Columns

- Identify sensitive data (e.g., credit card information)
- Use SQL Server's built-in encryption functions to secure this data

E.g: Sample Query to encrypt *Password* column



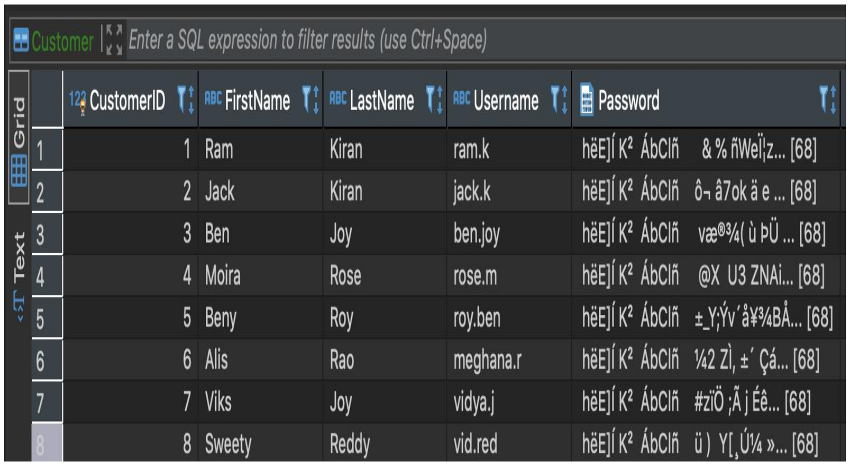
```sql
-- Create DMK
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Team18_P@sswOrd';
-- Create certificate to protect symmetric key
CREATE CERTIFICATE ResortCertificate
WITH SUBJECT = 'Resort Management Certificate', EXPIRY_DATE = '2030-10-31';
-- Create symmetric key to encrypt data
CREATE SYMMETRIC KEY ResortSymmetricKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE ResortCertificate;
```



```
-- Open symmetric key
OPEN SYMMETRIC KEY ResortSymmetricKey
DECRYPTION BY CERTIFICATE ResortCertificate;

INSERT resortmgm.Customer VALUES ('Ram', 'Kiran', 'ram.k',
EncryptByKey(Key_GUID('ResortSymmetricKey'), 'jgjhg48345'), 2067817643, 'ram@gmail.com', 1, 6);

-- Close symmetric key
CLOSE SYMMETRIC KEY ResortSymmetricKey;
```



	CustomerID	FirstName	LastName	Username	Password
1	1	Ram	Kiran	ram.k	hëE]Í K² ÁbClñ &% ñWeÏ¦z... [68]
2	2	Jack	Kiran	jack.k	hëE]Í K² ÁbClñ ô¬ â7ok ä e ... [68]
3	3	Ben	Joy	ben.joy	hëE]Í K² ÁbClñ væ®¾(ù ÞÜ ... [68]
4	4	Moira	Rose	rose.m	hëE]Í K² ÁbClñ @X U3 ZNAi... [68]
5	5	Beny	Roy	roy.ben	hëE]Í K² ÁbClñ ±_Y;Ýv´å¥¾BÅ... [68]
6	6	Alis	Rao	meghana.r	hëE]Í K² ÁbClñ ¼2 ZÌ, ±´ Çá... [68]
7	7	Viks	Joy	vidya.j	hëE]Í K² ÁbClñ #zïÖ ;Ã j Éê... [68]
8	8	Sweety	Reddy	vid.red	hëE]Í K² ÁbClñ ü) Y[¸Ú¼ »... [68]

2. Implement Access Controls

- Create user roles with appropriate permissions
- Use SQL Server's security features to restrict access to sensitive data

Step 3: Database Optimization

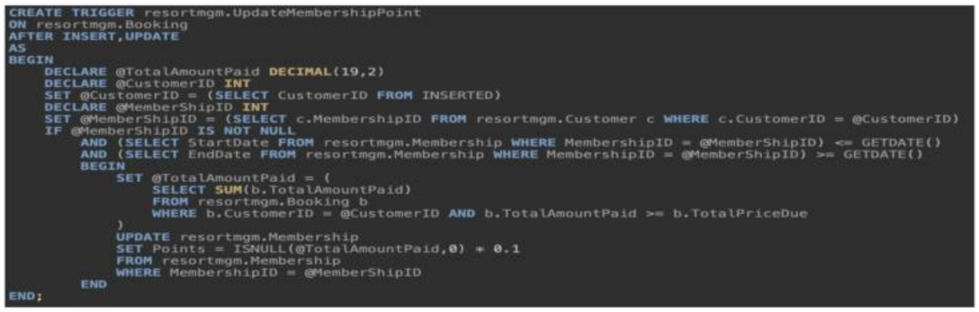
1. Create Indexes

- Identify frequently queried columns
- Create appropriate indexes to improve query performance

2. Implement Triggers

- Design triggers for data integrity (e.g., updating membership points)
- Use triggers for auditing purposes (e.g., logging changes to customer information)

E.g: Trigger to update membership points



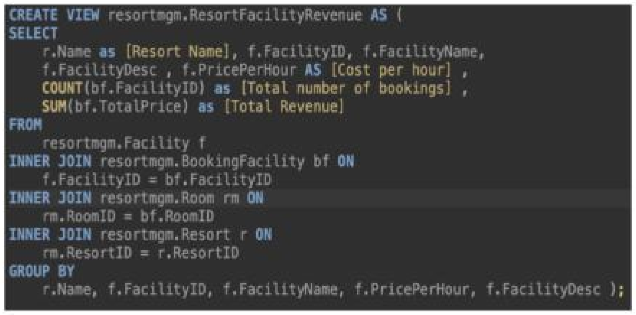
```
CREATE TRIGGER resortmgm.UpdateMembershipPoint
ON resortmgm.Booking
AFTER INSERT,UPDATE
AS
BEGIN
    DECLARE @TotalAmountPaid DECIMAL(19,2)
    DECLARE @CustomerID INT
    SET @CustomerID = (SELECT CustomerID FROM INSERTED)
    DECLARE @MemberShipID INT
    SET @MemberShipID = (SELECT c.MembershipID FROM resortmgm.Customer c WHERE c.CustomerID = @CustomerID)
    IF @MemberShipID IS NOT NULL
        AND (SELECT StartDate FROM resortmgm.Membership WHERE MembershipID = @MemberShipID) <= GETDATE()
        AND (SELECT EndDate FROM resortmgm.Membership WHERE MembershipID = @MemberShipID) >= GETDATE()
        BEGIN
            SET @TotalAmountPaid = (
                SELECT SUM(b.TotalAmountPaid)
                FROM resortmgm.Booking b
                WHERE b.CustomerID = @CustomerID AND b.TotalAmountPaid >= b.TotalPriceDue
            )
            UPDATE resortmgm.Membership
            SET Points = ISNULL(@TotalAmountPaid,0) * 0.1
            FROM resortmgm.Membership
            WHERE MembershipID = @MemberShipID
        END
END;
```

Step 4: Data Analysis and Visualization

1. Develop SQL Queries

- Write optimized SQL queries for common reporting needs
- Utilize joins, subqueries, and aggregate functions for complex analyses

E.g: SQL View to find out resort's revenue from recreational activities

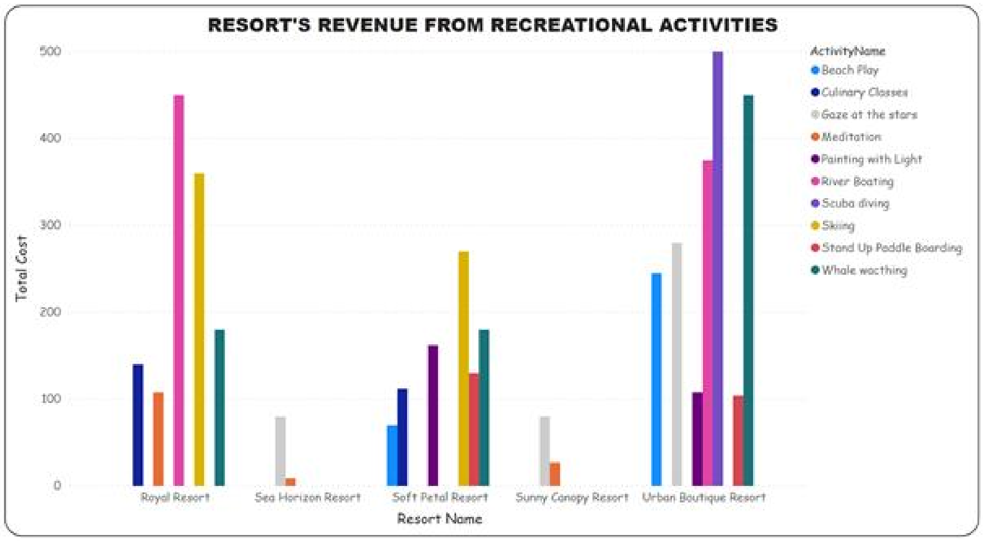


```sql
CREATE VIEW resortmgm.ResortFacilityRevenue AS (
SELECT
    r.Name as [Resort Name], f.FacilityID, f.FacilityName,
    f.FacilityDesc , f.PricePerHour AS [Cost per hour] ,
    COUNT(bf.FacilityID) as [Total number of bookings] ,
    SUM(bf.TotalPrice) as [Total Revenue]
FROM
    resortmgm.Facility f
INNER JOIN resortmgm.BookingFacility bf ON
    f.FacilityID = bf.FacilityID
INNER JOIN resortmgm.Room rm ON
    rm.RoomID = bf.RoomID
INNER JOIN resortmgm.Resort r ON
    rm.ResortID = r.ResortID
GROUP BY
    r.Name, f.FacilityID, f.FacilityName, f.PricePerHour, f.FacilityDesc );
```

2. Connect to PowerBI

- Establish a connection between SQL Server and PowerBI
- Create interactive dashboards for key metrics (e.g., occupancy rates, revenue)

E.g: Dashboard Representation of Resort's Revenue from Recreational Activities

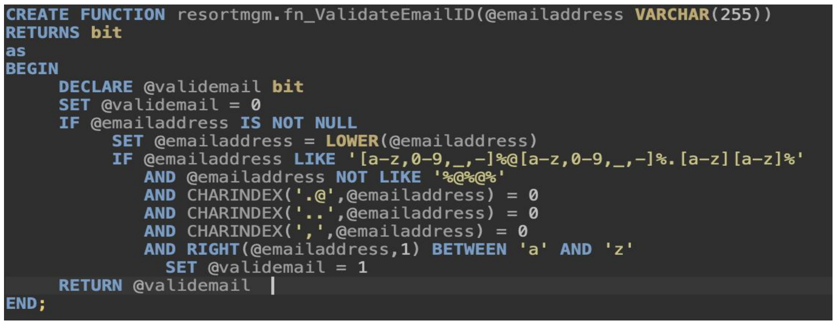


Best Practices for Resort Management Database

Data Integrity

- Implement check constraints to ensure data validity
- Use functions / stored procedures for complex operations to maintain consistency

Ex: Function to validate the Email address of *Employee* and *Customer*



```sql
CREATE FUNCTION resortmgm.fn_ValidateEmailID(@emailaddress VARCHAR(255))
RETURNS bit
as
BEGIN
     DECLARE @validemail bit
     SET @validemail = 0
     IF @emailaddress IS NOT NULL
          SET @emailaddress = LOWER(@emailaddress)
          IF @emailaddress LIKE '[a-z,0-9,_,-]%@[a-z,0-9,_,-]%.[a-z][a-z]%'
             AND @emailaddress NOT LIKE '%@%@%'
             AND CHARINDEX('.@',@emailaddress) = 0
             AND CHARINDEX('..',@emailaddress) = 0
             AND CHARINDEX(',',@emailaddress) = 0
             AND RIGHT(@emailaddress,1) BETWEEN 'a' AND 'z'
               SET @validemail = 1
     RETURN @validemail
END;
```

Performance Optimization

- Regularly update statistics and rebuild indexes
- Use query execution plans to identify and resolve performance bottlenecks

Backup and Recovery

- Implement a regular backup schedule
- Test recovery procedures to ensure data can be restored in case of failure

Documentation

- Maintain up-to-date documentation of database schema and relationships
- Document stored procedures, triggers, and other database objects

Conclusion

By following these steps and best practices, you can create a robust Resort Management Database System that ensures data integrity, security, and efficient operations. This approach enables data-driven decision-making and enhances overall resort management capabilities.