Database Specification Document: Hotel Reservation System

1. Introduction

This document specifies the complete database design for a Hotel Reservation System implemented in Microsoft SQL Server. The system manages hotel operations including customer information, room inventory, reservations, payments, and staff management.

2. Database Overview

- Database Name: Hotel

- Target DBMS: Microsoft SQL Server

- Design Focus: Operational efficiency with constraints, triggers, and stored procedures

3. Functional Requirements

3.1 Core Functions

- Store and manage customer information with contact details

- Maintain room inventory with pricing and status tracking

- Process reservations with check-in/check-out functionality

- Record and track payments linked to reservations

- Manage staff information and assignments

- Prevent double-booking of rooms

- Track reservation status throughout customer journey

3.2 Business Rules

- Prevent overlapping reservations for the same room

- Ensure staff salaries are always positive values

- Maintain referential integrity through cascade operations

- Rooms can have three statuses: Available, Occupied, or Under Maintenance

- Reservations can have five statuses: pending, Confirmed, Cancelled, Checked-in, Checked-out

4. Database Schema

4.1 Tables

Customer Table

- Purpose: Stores customer personal information

- Primary Key: Customer_ID (auto-incrementing identity)

- Constraints:

- Cus_Name: NOT NULL

- Cus_Email: NOT NULL

- Default values for country ('Egypt') and city ('cairo')

Cus_Phones Table

- Purpose: Stores multiple phone numbers for each customer

- Primary Key: Composite (customer_id, phone)

- Foreign Key: customer_id references customer(Customer_ID) with CASCADE operations

Room Table

- Purpose: Defines room properties and status

- Primary Key: Room_ID (auto-incrementing identity)

- Constraints:

- status: CHECK constraint ('Available', 'Occupied', 'Under Maintenance')

- RoomType: CHECK constraint ('Single', 'Double', 'Triple')

- capacity: TINYINT NOT NULL

- price: INT NOT NULL

Staff Table

- Purpose: Stores Staff information

- Primary Key: staff_ID (auto-incrementing identity)

- Constraints:

- salary: DECIMAL(8,2) NOT NULL with CHECK constraint (salary > 0)

- All fields: NOT NULL

Reservation Table

- Purpose: Core reservation tracking

- Primary Key: reservation_id (auto-incrementing identity)

- Calculated Field: num_of_days (computed from check_in_date and check_out_date)

- Constraints:

- status: CHECK constraint ('pending', 'Confirmed', 'Cancelled', 'Checked-in', 'Checked-out')

- Foreign keys to customer, room, and staff tables with CASCADE operations

Payment Table

- Purpose: Records financial transactions

- Primary Key: Payment_ID (auto-incrementing identity)

- Foreign Key: reservation_Id references reservation(reservation_id) with CASCADE operations

t_changespayment Table (Audit Table)

- Purpose: Tracks all changes to payment records

- Primary Key: id (auto-incrementing identity)

- Audit Fields: ActionType, ActionDate (default GETDATE()), ActionUser (default SUSER_SNAME())

4.2 Relationships

- customer (1) to (many) Cus_Phones

- customer (1) to (many) reservation

- room (1) to (many) reservation

- staff (1) to (many) reservation

- reservation (1) to (many) payment

5. Advanced Database Objects

5.1 Views

Current_Occupied View

- Purpose: Shows currently occupied rooms with guest names

- Logic: Joins room, reservation, and customer tables

- Filter: Rooms where current date is between check-in and check-out dates

Customer_History View

- Purpose: Shows past stays and payments for customers

- Logic: Joins reservation, customer, and payment tables

- Filter: Reservations with check-out date before current date

5.2 Stored Procedures

sp_MakeReservation

- Parameters: @CustomerID INT, @RoomID INT, @CheckIn DATETIME, @CheckOut DATETIME

- Functionality: Checks room availability and creates reservation

- Validation: Prevents overlapping reservations

sp_Check_In

- Parameters: @ReservationID INT, @StaffID INT

- Functionality: Processes check-in operation

- Validation: Checks if reservation exists and isn't already checked in

sp_CheckOut

- Parameters: @ReservationID INT

- Functionality: Processes check-out operation

- Validation: Checks if reservation exists and isn't already checked out

5.3 Functions

fn_CalculateStayDays

- Type: Scalar function

- Parameters: @CheckIn DATE, @CheckOut DATE

- Returns: Number of nights (INT)

- Logic: Calculates date difference in days

fn_RoomAvailability

- Type: Scalar function

- Parameters: @RoomID INT, @Date DATE

- Returns: BIT (1 if available, 0 if occupied)

- Logic: Checks if room exists and has no reservations for given date

5.4 Triggers

ovlap_tr Trigger

- Table: reservation

- Type: INSTEAD OF INSERT

- Purpose: Prevents overlapping reservations for the same room

- Logic: Only allows insertion if no date conflict exists

changes_payment Trigger

- Table: payment

- Type: AFTER INSERT, UPDATE, DELETE

- Purpose: Audits all changes to payment records

- Logic: Logs changes to t_changespayment table with action type and user

5.5 Indexes

- Non-clustered index on reservation(check_in_date, check_out_date)

- Non-clustered index on payment(reservation_id)

6. Constraints

6.1 Primary Key Constraints

- All tables have properly defined primary keys

- Cus_Phones uses composite primary key

6.2 Foreign Key Constraints

- All relationships enforced with foreign keys

- Cascade delete and update operations implemented

6.3 Check Constraints

- room.status: ('Available', 'Occupied', 'Under Maintenance')

- room.RoomType: ('Single', 'Double', 'Triple')

- reservation.status: ('pending', 'Confirmed', 'Cancelled', 'Checked-in', 'Checked-out')

- staff.salary: Must be greater than 0

6.4 Default Constraints

- customer.country: Default 'Egypt'

- customer.city: Default 'cairo'

- t_changespayment.ActionDate: Default GETDATE()

- t_changespayment.ActionUser: Default SUSER_SNAME()

7. Data Integrity Measures

7.1 Referential Integrity

- Foreign key constraints with cascade operations

- Proper table creation order to respect dependencies

7.2 Domain Integrity

- Check constraints enforce valid values

- Data types appropriate for each field

7.3 Business Rule Enforcement

- Overlapping reservation prevention

- Positive salary requirement

- Room availability checking

8. Performance Considerations

8.1 Index Strategy

- Index on reservation dates for availability queries

- Index on payment reservation_id for join operations

8.2 Calculated Columns

- reservation.num_of_days: Computed from check-in/check-out dates

9. Audit and History Tracking

9.1 Payment Audit

- t_changespayment table tracks all payment modifications

- Records old and new values for updates

- Captures user and timestamp information

10. Implementation Notes

10.1 SQL Server Specific Features

- Uses T-SQL specific functions (GETDATE(), SUSER_SNAME())

- Implements identity columns for surrogate keys

- Utilizes INSTEAD OF trigger for business logic

10.2 Error Handling

- Stored procedures include basic validation and error messages