Silver oaks hotel system documentation

All SQL Used

Group 03

Hein du Plessis – 41125614

Lohard j.v. Rensburg – 37248529

Salim Sofinia – 37284177

Coenri van Rooyen – 37330527

Table of Contents

[Creating the Database (Data Definition Language) 3](#_Toc144780397)

[Maintaining All Tables in the Database (read, insert, update and delete) 5](#_Toc144780398)

[Query the Database for Reports (Data Manipulation Language) 9](#_Toc144780399)

# Creating the Database (Data Definition Language)

**Booking Table:**

CREATE TABLE [dbo].[Booking] (

[Booking\_ID] INT IDENTITY (1, 1) NOT NULL,

[Number\_Adults] INT NULL,

[Number\_Child] INT NULL,

[Booking\_Total] MONEY NULL,

[Check\_InDate] DATETIME NULL,

[Check\_OutDate] DATETIME NULL,

[Booking\_Active] BIT NULL,

[Booking\_Paid] BIT NULL,

[Room\_ID] INT NULL,

[Client\_ID] INT NULL,

[Package\_ID] INT NULL,

[Employee\_ID] INT NULL,

PRIMARY KEY CLUSTERED ([Booking\_ID] ASC),

CONSTRAINT [FK\_Client\_ID] FOREIGN KEY ([Client\_ID]) REFERENCES [dbo].[Client] ([Client\_ID]),

CONSTRAINT [FK\_Employee\_ID] FOREIGN KEY ([Employee\_ID]) REFERENCES [dbo].[Employee] ([Employee\_ID]),

CONSTRAINT [FK\_Room\_ID] FOREIGN KEY ([Room\_ID]) REFERENCES [dbo].[Room] ([Room\_ID]),

CONSTRAINT [FK\_Package\_ID] FOREIGN KEY ([Package\_ID]) REFERENCES [dbo].[Package] ([Package\_ID])

);

**Client Table:**

CREATE TABLE [dbo].[Client] (

[Client\_ID] INT IDENTITY (1, 1) NOT NULL,

[Client\_Last\_Name] NVARCHAR (50) NULL,

[Client\_First\_Name] NVARCHAR (50) NULL,

[Client\_Email\_Address] NVARCHAR (50) NULL,

[Client\_Phone\_Number] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Client\_ID] ASC)

);

**Employee Table:**

CREATE TABLE [dbo].[Employee] (

[Employee\_ID] INT IDENTITY (1, 1) NOT NULL,

[Employee\_Last\_Name] NVARCHAR (50) NULL,

[Employee\_First\_Name] NVARCHAR (50) NULL,

[Employee\_ID\_Number] NVARCHAR (50) NULL,

[Employee\_Phone\_Number] NVARCHAR (50) NULL,

[Employee\_Email\_Address] NVARCHAR (50) NULL,

[Employee\_Authorisation\_Role] NVARCHAR (50) NULL,

[Username] NVARCHAR (50) NULL,

[Password] NVARCHAR (50) NULL,

CONSTRAINT [PK\_Employee] PRIMARY KEY CLUSTERED ([Employee\_ID] ASC)

);

**Package Table:**

CREATE TABLE [dbo].[Package] (

[Package\_ID] INT IDENTITY (1, 1) NOT NULL,

[Package\_Name] NVARCHAR (50) NULL,

[Package\_Price] MONEY NULL,

PRIMARY KEY CLUSTERED ([Package\_ID] ASC)

);

**Room Table:**

CREATE TABLE [dbo].[Room] (

[Room\_ID] INT IDENTITY (1, 1) NOT NULL,

[Room\_Number] INT NULL,

[Room\_Available] BIT NULL,

[Size\_ID] INT NULL,

PRIMARY KEY CLUSTERED ([Room\_ID] ASC),

CONSTRAINT [FK\_Size\_ID] FOREIGN KEY ([Size\_ID]) REFERENCES [dbo].[Room\_Size] ([Size\_ID])

);

**Room\_Size Table:**

CREATE TABLE [dbo].[Room\_Size] (

[Size\_ID] INT IDENTITY (1, 1) NOT NULL,

[Size\_Description] NVARCHAR (100) NULL,

[Size\_Price] MONEY NULL,

PRIMARY KEY CLUSTERED ([Size\_ID] ASC)

);

# Maintaining All Tables in the Database (read, insert, update and delete)

**Method used to maintain all tables:**

public static void CrudOperations(char crud, string sql, DataGridView dgv)

{

try

{

if (con.State == ConnectionState.Closed)

{

con.Open();

}

ds = new DataSet();

command = new SqlCommand(sql, con);

dataAdapter = new SqlDataAdapter();

switch (crud)

{

case 'C':

{

dataAdapter.InsertCommand = command;

dataAdapter.InsertCommand.ExecuteNonQuery();

break;

}

case 'R':

{

dataAdapter.SelectCommand = command;

dataAdapter.Fill(ds, "SourceTable");

dgv.DataSource = ds;

dgv.DataMember = "SourceTable";

break;

}

case 'U':

{

command = new SqlCommand(sql, con);

dataAdapter.UpdateCommand = command;

dataAdapter.UpdateCommand.ExecuteNonQuery();

break;

}

case 'D':

{

dataAdapter.DeleteCommand = command;

dataAdapter.DeleteCommand.ExecuteNonQuery();

break;

}

default:

{

MessageBox.Show("No CRUD character identified! Please contact your software solutions provider.", "Developer Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

}

}

command.Dispose();

con.Close();

}

catch (Exception ex)

{

MessageBox.Show("Please contact your software solutions provider.\n" + ex.Message, "Developer Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

**Maintaining of Booking Table:**

AllMethods.CrudOperations('C', $"INSERT INTO Client (Client\_Last\_Name, Client\_First\_Name, Client\_Email\_Address, Client\_Phone\_Number) VALUES ('{txtCSname.Text}','{txtCName.Text}','{txtCEmail.Text}','{txtCNumber.Text}')", null);

AllMethods.CrudOperations('R',"SELECT \* FROM Booking",dgvActiveBookings);

AllMethods.CrudOperations('U',$"UPDATE Booking SET Package\_ID = {Convert.ToInt32(packageID)}, Booking\_Total = {AllMethods.GetBookingPrice(cbxUpdatePackage.Text, Convert.ToInt32(txtUBookingID.Text))} WHERE Booking\_ID = {Convert.ToInt32(txtUBookingID.Text)}",null);

AllMethods.CrudOperations('D',$"DELETE FROM Booking WHERE Booking\_ID = {Convert.ToInt32(txtDBookingID.Text)}",dgvDeleteBooking);

**Maintaining of Client Table:**

AllMethods.CrudOperations('C', $"INSERT INTO Client (Client\_Last\_Name, Client\_First\_Name, Client\_Email\_Address, Client\_Phone\_Number) VALUES ('{txtCSname.Text}','{txtCName.Text}','{txtCEmail.Text}','{txtCNumber.Text}')", null);

AllMethods.CrudOperations('R', "SELECT \* FROM Client", dgvViewClients);

AllMethods.CrudOperations('U', $"UPDATE Client SET Client\_Last\_Name = '{txtUpdateClientSName.Text}', Client\_First\_Name = '{txtUpdateClientName.Text}', Client\_Email\_Address = '{txtUpdateClientEmail.Text}', Client\_Phone\_Number = '{txtUpdateClientNumber.Text}' WHERE Client\_ID = {Convert.ToInt32(txtSearchCName.Text)}", dgvUpdateClient);

AllMethods.CrudOperations('D',$"DELETE FROM Client WHERE Client\_ID = {txtSearchDeleteClientName.Text}",dgvDeleteClient);

**Maintaining of Employee Table:**

AllMethods.CrudOperations('C', $"INSERT INTO Employee (Employee\_Last\_Name,Employee\_First\_Name,Employee\_ID\_Number,Employee\_Phone\_Number,Employee\_Email\_Address,Employee\_Authorisation\_Role,Username,Password) VALUES ('{EmployeeLastName}','{EmployeeFirstName}','{EmployeeIDNumber}','{EmployeePhoneNumber}','{employeeEmailAddress}','{employeeAuthorisationRole}','{EmployeeUsername}','{EmployeePassword}')", dgvAllEmployee);

AllMethods.CrudOperations('R', "SELECT \* FROM Employee", dgvAllEmployee);

AllMethods.CrudOperations('U', $"UPDATE Employee SET Employee\_Last\_Name = '{EmployeeLastName}',Employee\_First\_Name = '{EmployeeFirstName}',Employee\_ID\_Number = '{EmployeeIDNumber}',Employee\_Phone\_Number = '{EmployeePhoneNumber}',Employee\_Email\_Address = '{employeeEmailAddress}',Employee\_Authorisation\_Role = '{employeeAuthorisationRole}',Username = '{EmployeeUsername}',Password = '{EmployeePassword}' WHERE Employee\_ID = {Convert.ToInt32(txtUpdatesearchEmployeeID.Text)}", dgvAllEmployee);

AllMethods.CrudOperations('D',$"DELETE FROM Employee WHERE Employee\_ID = {Convert.ToInt32(txtDeleteEID.Text)}",dgvDeleteEmployee);

**Maintaining of Package Table:**

AllMethods.CrudOperations('C', $"INSERT INTO Package (Package\_Name, Package\_Price) VALUES ('{PackageName}',{PackagePrice})", dgvAllPackages);

AllMethods.CrudOperations('R',"SELECT \* FROM Package",dgvUpdatePackage);

AllMethods.CrudOperations('U',$"UPDATE Package SET Package\_Name = '{txtUpdatePackName.Text}', Package\_Price = {Convert.ToDouble(txtUpdatePackPrice.Text)} WHERE Package\_Name = '{txtUpdatePackSearch.Text}'",dgvUpdatePackage);

AllMethods.CrudOperations('D', $"DELETE FROM Package WHERE Package\_Name = '{txtDeletePackName.Text}'", dgvDeletePackage);

**Maintaining of Room Table:**

AllMethods.CrudOperations('R', $"SELECT \* FROM Room WHERE CAST(Room\_Number AS VARCHAR) LIKE '%{txtSearchRoomNr.Text}%'", dgvViewAllRooms);

AllMethods.CrudOperations('U', $"UPDATE Room SET Room\_Number = {RoomNumber}, Room\_Available = '{RoomAvailable.ToString()}' WHERE Room\_Number = {Convert.ToInt32(txtSearchUpdateRoomNr.Text)}", dgvUpdateRoom);

**Maintaining of Room\_Size Table:**

AllMethods.CrudOperations('R', $"SELECT \* FROM Room\_Size WHERE CAST(Size\_ID AS VARCHAR) LIKE '%{txtRoomSizeIdSearch.Text}%'", dgvRoomSizeView);

AllMethods.CrudOperations('U', $"UPDATE Room\_Size SET Size\_Description = '{RoomDescription}', Size\_Price = {RoomPrice} WHERE Size\_ID = {Convert.ToInt32(txtSearchUpdateRoomId.Text)}", dgvUpdateRoomSize);

# Query the Database for Reports (Data Manipulation Language)

**Booking Report Per Time Period:**

public static List<String> BookingReportTimePeriod(DateTime start, DateTime end)

{

List<String> lines = new List<string>();

int count = 0;

try

{

con.Open();

string sql = $"SELECT \* FROM Booking WHERE Check\_InDate >= '{start}' AND Check\_OutDate <= '{end}'";

command = new SqlCommand(sql, con);

dataReader = command.ExecuteReader();

lines.Add("Bookings from " + start.ToShortDateString() + " to " + end.ToShortDateString());

lines.Add("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

while (dataReader.Read())

{

count++;

}

if(count == 0)

{

return lines;

}

lines.Add("Amount of Bookings: \t" + count.ToString());

con.Close();

con.Open();

int totalAdults = 0;

string sqlAdults = $"SELECT Number\_Adults FROM Booking WHERE Check\_InDate >= '{start}' AND Check\_OutDate <= '{end}'";

command = new SqlCommand(sqlAdults, con);

dataReader = command.ExecuteReader();

// report += "\nTotal number of Adults: ";

while (dataReader.Read())

{

totalAdults += dataReader.GetInt32(0);

}

lines.Add("Total number of Adults: \t" + totalAdults);

lines.Add("Average number of Adults: \t" + Decimal.Truncate(totalAdults / count).ToString());

con.Close();

con.Open();

int totalChildren = 0;

string sqlChildren = $"SELECT Number\_Child FROM Booking WHERE Check\_InDate >= '{start}' AND Check\_OutDate <= '{end}'";

command = new SqlCommand(sqlChildren, con);

dataReader = command.ExecuteReader();

while (dataReader.Read())

{

totalChildren += dataReader.GetInt32(0);

}

lines.Add("Total number of Children: \t"+totalChildren);

lines.Add("Average number of Children: \t" + Decimal.Truncate(totalChildren / count).ToString());

con.Close();

return lines;

}

catch (Exception ex)

{

List<String> error = new List<String>();

MessageBox.Show("Please contact your software solutions provider \n" + ex.Message, "Developer Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return error;

}

}

**Package Report Per Time Period:**

public static List<String> PackageReportTimePeriod(DateTime start, DateTime end)

{

try

{

List<String> lines = new List<string>();

con.Open();

string sql = $"SELECT p.Package\_ID, p.Package\_Name, p.Package\_Price, COUNT(b.Package\_ID) AS Package\_Count FROM Package AS p LEFT JOIN Booking AS b ON p.Package\_ID = b.Package\_ID WHERE b.Check\_InDate >= '{start}' AND b.Check\_OutDate <= '{end}' GROUP BY p.Package\_ID, p.Package\_Name, p.Package\_Price ORDER BY Package\_Count DESC";

command = new SqlCommand(sql, con);

dataReader = command.ExecuteReader();

lines.Add("Packages Sorted by Popularity from " + start.ToShortDateString() + " to " + end.ToShortDateString());

lines.Add("");

lines.Add("Package\_ID \t Package Name \t Package Price \t Frequency");

lines.Add("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

while (dataReader.Read())

{

lines.Add(dataReader.GetValue(0).ToString() + "\t\t" + dataReader.GetValue(1).ToString() + "\t\t" + dataReader.GetValue(2).ToString() + "\t\t" + dataReader.GetValue(3).ToString());

}

con.Close();

return lines;

}

catch (Exception ex)

{

List<String> error = new List<string>();

MessageBox.Show("Please contact your software solutions provider \n" + ex.Message, "Developer Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

error.Add("An error has occured");

return error;

}

}