**Projet - Gestion des Réservations de Salles SQL Server**

**1. Modélisation de la Base de Données**

**Modèle Conceptuel (Entités principales)**

**Entité UTILISATEURS**

* **UserID** (PK) : INT IDENTITY
* **Nom** : NVARCHAR(100)
* **Prenom** : NVARCHAR(100)
* **Email** : NVARCHAR(255) UNIQUE
* **MotDePasse** : NVARCHAR(255)
* **RoleID** : INT (FK vers ROLES)
* **DateCreation** : DATETIME
* **Actif** : BIT

**Entité ROLES**

* **RoleID** (PK) : INT IDENTITY
* **NomRole** : NVARCHAR(50) (Employé, Manager, Admin)
* **Description** : NVARCHAR(255)

**Entité SALLES**

* **SalleID** (PK) : INT IDENTITY
* **NomSalle** : NVARCHAR(100)
* **Capacite** : INT
* **Localisation** : NVARCHAR(255)
* **Equipements** : NVARCHAR(500)
* **Active** : BIT

**Entité TYPES\_EVENEMENTS**

* **TypeEventID** (PK) : INT IDENTITY
* **NomType** : NVARCHAR(100) (Réunion, Formation, Conférence, etc.)
* **Description** : NVARCHAR(255)
* **DureeDefaut** : INT (en minutes)

**Entité RESERVATIONS**

* **ReservationID** (PK) : INT IDENTITY
* **UserID** : INT (FK vers UTILISATEURS)
* **SalleID** : INT (FK vers SALLES)
* **TypeEventID** : INT (FK vers TYPES\_EVENEMENTS)
* **DateReservation** : DATE
* **HeureDebut** : TIME
* **HeureFin** : TIME
* **ObjetReunion** : NVARCHAR(255)
* **NombreParticipants** : INT
* **Statut** : NVARCHAR(20) (En attente, Validée, Annulée, Refusée)
* **DateCreation** : DATETIME
* **DateValidation** : DATETIME
* **ValidePar** : INT (FK vers UTILISATEURS - Manager)
* **Commentaires** : NVARCHAR(500)

**Entité LOGS\_RESERVATIONS**

* **LogID** (PK) : INT IDENTITY
* **ReservationID** : INT (FK vers RESERVATIONS)
* **UserID** : INT (FK vers UTILISATEURS)
* **Action** : NVARCHAR(50) (Création, Validation, Annulation, Modification)
* **DateAction** : DATETIME
* **Details** : NVARCHAR(500)

**2. Script SQL - Création des Tables**

-- Base de données

CREATE DATABASE GestionReservationsSalles;

GO

USE GestionReservationsSalles;

GO

-- Table ROLES

CREATE TABLE ROLES (

RoleID INT IDENTITY(1,1) PRIMARY KEY,

NomRole NVARCHAR(50) NOT NULL UNIQUE,

Description NVARCHAR(255)

);

-- Table UTILISATEURS

CREATE TABLE UTILISATEURS (

UserID INT IDENTITY(1,1) PRIMARY KEY,

Nom NVARCHAR(100) NOT NULL,

Prenom NVARCHAR(100) NOT NULL,

Email NVARCHAR(255) NOT NULL UNIQUE,

MotDePasse NVARCHAR(255) NOT NULL,

RoleID INT NOT NULL,

DateCreation DATETIME DEFAULT GETDATE(),

Actif BIT DEFAULT 1,

FOREIGN KEY (RoleID) REFERENCES ROLES(RoleID)

);

-- Table SALLES

CREATE TABLE SALLES (

SalleID INT IDENTITY(1,1) PRIMARY KEY,

NomSalle NVARCHAR(100) NOT NULL UNIQUE,

Capacite INT NOT NULL CHECK (Capacite > 0),

Localisation NVARCHAR(255),

Equipements NVARCHAR(500),

Active BIT DEFAULT 1

);

-- Table TYPES\_EVENEMENTS

CREATE TABLE TYPES\_EVENEMENTS (

TypeEventID INT IDENTITY(1,1) PRIMARY KEY,

NomType NVARCHAR(100) NOT NULL UNIQUE,

Description NVARCHAR(255),

DureeDefaut INT DEFAULT 60 -- en minutes

);

-- Table RESERVATIONS

CREATE TABLE RESERVATIONS (

ReservationID INT IDENTITY(1,1) PRIMARY KEY,

UserID INT NOT NULL,

SalleID INT NOT NULL,

TypeEventID INT NOT NULL,

DateReservation DATE NOT NULL,

HeureDebut TIME NOT NULL,

HeureFin TIME NOT NULL,

ObjetReunion NVARCHAR(255) NOT NULL,

NombreParticipants INT DEFAULT 1,

Statut NVARCHAR(20) DEFAULT 'En attente' CHECK (Statut IN ('En attente', 'Validée', 'Annulée', 'Refusée')),

DateCreation DATETIME DEFAULT GETDATE(),

DateValidation DATETIME,

ValidePar INT,

Commentaires NVARCHAR(500),

FOREIGN KEY (UserID) REFERENCES UTILISATEURS(UserID),

FOREIGN KEY (SalleID) REFERENCES SALLES(SalleID),

FOREIGN KEY (TypeEventID) REFERENCES TYPES\_EVENEMENTS(TypeEventID),

FOREIGN KEY (ValidePar) REFERENCES UTILISATEURS(UserID),

CONSTRAINT CK\_HeureReservation CHECK (HeureFin > HeureDebut)

);

-- Table LOGS\_RESERVATIONS

CREATE TABLE LOGS\_RESERVATIONS (

LogID INT IDENTITY(1,1) PRIMARY KEY,

ReservationID INT,

UserID INT NOT NULL,

Action NVARCHAR(50) NOT NULL,

DateAction DATETIME DEFAULT GETDATE(),

Details NVARCHAR(500),

FOREIGN KEY (ReservationID) REFERENCES RESERVATIONS(ReservationID),

FOREIGN KEY (UserID) REFERENCES UTILISATEURS(UserID)

);

**3. Données de Test**

-- Insertion des rôles

INSERT INTO ROLES (NomRole, Description) VALUES

('Employé', 'Utilisateur standard pouvant réserver des salles'),

('Manager', 'Peut valider/refuser les réservations'),

('Admin', 'Administrateur système avec tous les droits');

-- Insertion des types d'événements

INSERT INTO TYPES\_EVENEMENTS (NomType, Description, DureeDefaut) VALUES

('Réunion', 'Réunion de travail standard', 60),

('Formation', 'Session de formation', 120),

('Conférence', 'Présentation ou conférence', 90),

('Entretien', 'Entretien de recrutement', 45),

('Présentation', 'Présentation client', 90);

-- Insertion des salles

INSERT INTO SALLES (NomSalle, Capacite, Localisation, Equipements) VALUES

('Salle Alpha', 12, 'Étage 1', 'Projecteur, Tableau blanc, Visioconférence'),

('Salle Beta', 8, 'Étage 1', 'Écran TV, Tableau blanc'),

('Salle Gamma', 20, 'Étage 2', 'Projecteur, Sonorisation, Visioconférence'),

('Salle Delta', 6, 'Étage 2', 'Écran TV'),

('Salle Epsilon', 15, 'Étage 3', 'Projecteur, Tableau interactif'),

('Auditorium', 50, 'Rez-de-chaussée', 'Sonorisation, Éclairage, Projection');

-- Insertion des utilisateurs

INSERT INTO UTILISATEURS (Nom, Prenom, Email, MotDePasse, RoleID) VALUES

('Dupont', 'Jean', 'jean.dupont@entreprise.com', 'password123', 1),

('Martin', 'Marie', 'marie.martin@entreprise.com', 'password123', 2),

('Durand', 'Pierre', 'pierre.durand@entreprise.com', 'password123', 1),

('Bernard', 'Sophie', 'sophie.bernard@entreprise.com', 'password123', 2),

('Petit', 'Luc', 'luc.petit@entreprise.com', 'password123', 1),

('Moreau', 'Anne', 'anne.moreau@entreprise.com', 'password123', 3);

**4. Fonctions SQL Server**

-- Fonction pour vérifier la disponibilité d'une salle

CREATE FUNCTION fn\_SalleDisponible(@SalleID INT, @Date DATE, @HeureDebut TIME, @HeureFin TIME)

RETURNS BIT

AS

BEGIN

DECLARE @Disponible BIT = 1;

IF EXISTS (

SELECT 1 FROM RESERVATIONS

WHERE SalleID = @SalleID

AND DateReservation = @Date

AND Statut IN ('En attente', 'Validée')

AND NOT (@HeureFin <= HeureDebut OR @HeureDebut >= HeureFin)

)

BEGIN

SET @Disponible = 0;

END

RETURN @Disponible;

END;

GO

-- Fonction pour vérifier la capacité disponible

CREATE FUNCTION fn\_CapaciteDisponible(@SalleID INT, @Date DATE)

RETURNS INT

AS

BEGIN

DECLARE @Capacite INT;

SELECT @Capacite = Capacite

FROM SALLES

WHERE SalleID = @SalleID AND Active = 1;

RETURN ISNULL(@Capacite, 0);

END;

GO

-- Fonction pour obtenir le statut d'un utilisateur

CREATE FUNCTION fn\_StatutUtilisateur(@UserID INT)

RETURNS NVARCHAR(50)

AS

BEGIN

DECLARE @Statut NVARCHAR(50);

SELECT @Statut = R.NomRole

FROM UTILISATEURS U

JOIN ROLES R ON U.RoleID = R.RoleID

WHERE U.UserID = @UserID AND U.Actif = 1;

RETURN ISNULL(@Statut, 'Inactif');

END;

GO

-- Fonction pour calculer la durée d'une réservation

CREATE FUNCTION fn\_DureeReservation(@ResID INT)

RETURNS INT

AS

BEGIN

DECLARE @Duree INT;

SELECT @Duree = DATEDIFF(MINUTE, HeureDebut, HeureFin)

FROM RESERVATIONS

WHERE ReservationID = @ResID;

RETURN ISNULL(@Duree, 0);

END;

GO

**5. Procédures Stockées**

-- Procédure pour réserver une salle

CREATE PROCEDURE sp\_ReserverSalle

@UserID INT,

@SalleID INT,

@TypeEventID INT,

@DateReservation DATE,

@HeureDebut TIME,

@HeureFin TIME,

@ObjetReunion NVARCHAR(255),

@NombreParticipants INT = 1,

@ReservationID INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

-- Vérifications

IF dbo.fn\_SalleDisponible(@SalleID, @DateReservation, @HeureDebut, @HeureFin) = 0

BEGIN

RAISERROR('Salle non disponible pour ce créneau', 16, 1);

RETURN;

END

IF @NombreParticipants > dbo.fn\_CapaciteDisponible(@SalleID, @DateReservation)

BEGIN

RAISERROR('Nombre de participants supérieur à la capacité de la salle', 16, 1);

RETURN;

END

-- Insertion de la réservation

INSERT INTO RESERVATIONS (UserID, SalleID, TypeEventID, DateReservation, HeureDebut, HeureFin, ObjetReunion, NombreParticipants)

VALUES (@UserID, @SalleID, @TypeEventID, @DateReservation, @HeureDebut, @HeureFin, @ObjetReunion, @NombreParticipants);

SET @ReservationID = SCOPE\_IDENTITY();

-- Log de l'action

INSERT INTO LOGS\_RESERVATIONS (ReservationID, UserID, Action, Details)

VALUES (@ReservationID, @UserID, 'Création', 'Nouvelle réservation créée');

END;

GO

-- Procédure pour valider une réservation

CREATE PROCEDURE sp\_ValiderReservation

@ManagerID INT,

@ReservationID INT

AS

BEGIN

SET NOCOUNT ON;

-- Vérifier que l'utilisateur est manager

IF dbo.fn\_StatutUtilisateur(@ManagerID) NOT IN ('Manager', 'Admin')

BEGIN

RAISERROR('Seuls les managers peuvent valider les réservations', 16, 1);

RETURN;

END

-- Mettre à jour la réservation

UPDATE RESERVATIONS

SET Statut = 'Validée',

DateValidation = GETDATE(),

ValidePar = @ManagerID

WHERE ReservationID = @ReservationID

AND Statut = 'En attente';

IF @@ROWCOUNT = 0

BEGIN

RAISERROR('Réservation introuvable ou déjà traitée', 16, 1);

RETURN;

END

-- Log de l'action

INSERT INTO LOGS\_RESERVATIONS (ReservationID, UserID, Action, Details)

VALUES (@ReservationID, @ManagerID, 'Validation', 'Réservation validée par le manager');

END;

GO

**6. Triggers**

-- Trigger pour vérifier les conflits de réservation

CREATE TRIGGER tr\_VerifierConflitReservation

ON RESERVATIONS

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (

SELECT 1

FROM inserted i

JOIN RESERVATIONS r ON i.SalleID = r.SalleID

AND i.DateReservation = r.DateReservation

AND i.ReservationID != r.ReservationID

AND r.Statut IN ('En attente', 'Validée')

AND NOT (i.HeureFin <= r.HeureDebut OR i.HeureDebut >= r.HeureFin)

)

BEGIN

ROLLBACK TRANSACTION;

RAISERROR('Conflit de réservation détecté', 16, 1);

END

END;

GO

-- Trigger pour supprimer les réservations en cascade

CREATE TRIGGER tr\_SupprimerReservationsUtilisateur

ON UTILISATEURS

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

-- Supprimer les réservations non validées

DELETE FROM RESERVATIONS

WHERE UserID IN (SELECT UserID FROM deleted)

AND Statut = 'En attente';

-- Marquer les autres comme annulées

UPDATE RESERVATIONS

SET Statut = 'Annulée',

Commentaires = 'Utilisateur supprimé'

WHERE UserID IN (SELECT UserID FROM deleted)

AND Statut IN ('Validée');

END;

GO

**7. Requêtes Avancées**

-- Rapport journalier des réservations validées

SELECT

S.NomSalle,

R.DateReservation,

R.HeureDebut,

R.HeureFin,

R.ObjetReunion,

U.Nom + ' ' + U.Prenom AS Organisateur,

TE.NomType

FROM RESERVATIONS R

JOIN SALLES S ON R.SalleID = S.SalleID

JOIN UTILISATEURS U ON R.UserID = U.UserID

JOIN TYPES\_EVENEMENTS TE ON R.TypeEventID = TE.TypeEventID

WHERE R.Statut = 'Validée'

AND R.DateReservation = CAST(GETDATE() AS DATE)

ORDER BY S.NomSalle, R.HeureDebut;

-- Classement des salles les plus utilisées

SELECT

S.NomSalle,

COUNT(R.ReservationID) AS NombreReservations,

RANK() OVER (ORDER BY COUNT(R.ReservationID) DESC) AS Rang

FROM SALLES S

LEFT JOIN RESERVATIONS R ON S.SalleID = R.SalleID

AND R.Statut = 'Validée'

AND R.DateReservation >= DATEADD(MONTH, -1, GETDATE())

GROUP BY S.SalleID, S.NomSalle

ORDER BY Rang;

**8. Plan de Développement**

**Phase 1 : Base de données (Semaine 1)**

* Modélisation conceptuelle
* Création des tables et contraintes
* Insertion des données de test
* Tests de base

**Phase 2 : Logique métier (Semaine 2)**

* Développement des fonctions
* Création des procédures stockées
* Implémentation des triggers
* Tests des fonctionnalités

**Phase 3 : Requêtes avancées (Semaine 3)**

* Développement des rapports
* Optimisation des performances
* Tests de charge

**Phase 4 : Interface (Optionnel)**

* Choix de la technologie (ASP.NET, PHP, etc.)
* Développement des écrans
* Intégration avec la base de données

**9. Livrables**

1. **Schémas de base de données** (MCD/MLD)
2. **Script SQL complet** avec toutes les fonctionnalités
3. **Documentation technique** et guide d'utilisation
4. **Rapport final** avec analyse et recommandations

**10. Conseils pour la Réussite**

* **Commencez par la modélisation** : Une bonne conception évite les problèmes futurs
* **Testez régulièrement** : Vérifiez chaque fonctionnalité au fur et à mesure
* **Documentez votre code** : Commentez les procédures et fonctions complexes
* **Gérez les erreurs** : Prévoyez la gestion des cas d'erreur
* **Optimisez les performances** : Créez les index nécessaires sur les colonnes clés