Modello Dati PostgreSQL per WebUI Multitenant SciPhi AI R2R

Panoramica

Questo documento descrive il modello dati PostgreSQL per il sistema WebUI multitenant di SciPhi AI R2R. Il modello è progettato con Entity Framework Core utilizzando un approccio "Shared Database, Shared Schema" con isolamento a livello di CompanyId per garantire un efficace isolamento dei dati tra tenant mantenendo l'efficienza delle risorse.

Configurazione Database

Credenziali di Accesso (VERIFIED):

- Host: 192.168.1.4
- Port: 5433
- Database: cleverdocs
- Username: admin
- Password: MiaPassword123

Test Users (CREATED & VERIFIED):

- **System Admin**: info@hybrid.it / Florealia2025! (Role=1, Company: Hybrid IT)
- **Company Admin**: info@micrOsis.it / Maremmabona1! (Role=2, Company: Microsis srl)
- **User 1**: r.antoniucci@microsis.it / Maremmabona1! (Role=3, Company: Microsis srl)
- **User 2**: m.bevilacqua@microsis.it / Maremmabona1! (Role=3, Company: Microsis srl)

Struttura degli Schemi

Il database è organizzato in schemi distinti:

1. **dbo**: Schema di sistema contenente tabelle condivise e di configurazione globale
2. **company_{companyId}**: Schema specifico per ogni azienda/tenant, creato dinamicamente

Tabelle di Sistema (Schema dbo)

Companies

```
CREATE TABLE dbo.Companies (
    CompanyId INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    LogoUrl NVARCHAR(255) NULL,
    SchemaName NVARCHAR(50) NOT NULL,
    IsActive BIT NOT NULL DEFAULT 1,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE()
);
```

GlobalSettings

```
CREATE TABLE dbo.GlobalSettings (
    SettingId INT IDENTITY(1,1) PRIMARY KEY,
    SettingKey NVARCHAR(50) NOT NULL,
    SettingValue NVARCHAR(MAX) NULL,
    Description NVARCHAR(255) NULL,
    IsEncrypted BIT NOT NULL DEFAULT 0,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT UQ_GlobalSettings_Key UNIQUE (SettingKey)
);
```

R2RConfiguration

```
CREATE TABLE dbo.R2RConfiguration (
    ConfigId INT IDENTITY(1,1) PRIMARY KEY,
    ApiEndpoint NVARCHAR(255) NOT NULL,
    DefaultApiKey NVARCHAR(255) NULL,
    IsEncrypted BIT NOT NULL DEFAULT 1,
    MaxConcurrentRequests INT NOT NULL DEFAULT 10,
    RequestTimeout INT NOT NULL DEFAULT 30,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE()
);
```

AdminUsers

```
CREATE TABLE dbo.AdminUsers (
    AdminId INT IDENTITY(1,1) PRIMARY KEY,
    Username NVARCHAR(50) NOT NULL,
    Email NVARCHAR(100) NOT NULL,
    PasswordHash NVARCHAR(255) NOT NULL,
    FirstName NVARCHAR(50) NULL,
    LastName NVARCHAR(50) NULL,
    IsActive BIT NOT NULL DEFAULT 1,
    LastLoginAt DATETIME2 NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT UQ_AdminUsers_Username UNIQUE (Username),
    CONSTRAINT UQ_AdminUsers_Email UNIQUE (Email)
);
```

SystemLogs

```
CREATE TABLE dbo.SystemLogs (
    LogId BIGINT IDENTITY(1,1) PRIMARY KEY,
    LogLevel NVARCHAR(20) NOT NULL,
    Message NVARCHAR(MAX) NOT NULL,
    Exception NVARCHAR(MAX) NULL,
    Source NVARCHAR(255) NULL,
    CompanyId INT NULL,
    UserId INT NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    INDEX IX_SystemLogs_CompanyId (CompanyId),
    INDEX IX_SystemLogs_CreatedAt (CreatedAt)
);
```

QueueItems

```
CREATE TABLE dbo.QueueItems (
    QueueItemId BIGINT IDENTITY(1,1) PRIMARY KEY,
    QueueName NVARCHAR(50) NOT NULL,
    ItemType NVARCHAR(50) NOT NULL,
    ItemData NVARCHAR(MAX) NOT NULL,
    Status NVARCHAR(20) NOT NULL DEFAULT 'Pending',
    Priority INT NOT NULL DEFAULT 0,
    CompanyId INT NULL,
    UserId INT NULL,
    RetryCount INT NOT NULL DEFAULT 0,
    MaxRetries INT NOT NULL DEFAULT 3,
    NextRetryAt DATETIME2 NULL,
    ProcessingStartedAt DATETIME2 NULL,
    CompletedAt DATETIME2 NULL,
    ErrorMessage NVARCHAR(MAX) NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    INDEX IX_QueueItems_Status (Status),
    INDEX IX_QueueItems_CompanyId (CompanyId),
    INDEX IX_QueueItems_NextRetryAt (NextRetryAt)
);
```

Tabelle Specifiche per Tenant (Schema company_{companyId})

CompanySettings

```
CREATE TABLE company_{companyId}.CompanySettings (
    SettingId INT IDENTITY(1,1) PRIMARY KEY,
    SettingKey NVARCHAR(50) NOT NULL,
```

```
    SettingValue NVARCHAR(MAX) NULL,
    Description NVARCHAR(255) NULL,
    IsEncrypted BIT NOT NULL DEFAULT 0,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT UQ_CompanySettings_Key UNIQUE (SettingKey)
);
```

R2RCompanyConfiguration

```
CREATE TABLE company_{companyId}.R2RCompanyConfiguration (
    ConfigId INT IDENTITY(1,1) PRIMARY KEY,
    LlmModel NVARCHAR(50) NOT NULL,
    ApiKey NVARCHAR(255) NULL,
    IsEncrypted BIT NOT NULL DEFAULT 1,
    MaxDocumentsPerUser INT NOT NULL DEFAULT 100,
    MaxQueriesPerHour INT NOT NULL DEFAULT 50,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE()
);
```

Users

```
CREATE TABLE company_{companyId}.Users (
    UserId INT IDENTITY(1,1) PRIMARY KEY,
    Username NVARCHAR(50) NOT NULL,
    Email NVARCHAR(100) NOT NULL,
    PasswordHash NVARCHAR(255) NOT NULL,
    FirstName NVARCHAR(50) NULL,
    LastName NVARCHAR(50) NULL,
    Role NVARCHAR(20) NOT NULL DEFAULT 'User',
    IsActive BIT NOT NULL DEFAULT 1,
    LastLoginAt DATETIME2 NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT UQ_Users_Username UNIQUE (Username),
    CONSTRAINT UQ_Users_Email UNIQUE (Email)
);
```

Collections

```
CREATE TABLE company_{companyId}.Collections (
    CollectionId INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100) NOT NULL,
    Description NVARCHAR(500) NULL,
```

```
    R2RCollectionId NVARCHAR(100) NOT NULL,
    UserId INT NOT NULL,
    IsShared BIT NOT NULL DEFAULT 0,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_Collections_Users FOREIGN KEY (UserId) REFERENCES
company_{companyId}.Users(UserId),
    INDEX IX_Collections_UserId (UserId)
);
```

Documents

```
CREATE TABLE company_{companyId}.Documents (
    DocumentId INT IDENTITY(1,1) PRIMARY KEY,
    FileName NVARCHAR(255) NOT NULL,
    FileSize BIGINT NOT NULL,
    FileType NVARCHAR(50) NOT NULL,
    R2RDocumentId NVARCHAR(100) NOT NULL,
    CollectionId INT NOT NULL,
    UserId INT NOT NULL,
    Status NVARCHAR(20) NOT NULL DEFAULT 'Pending',
    ProcessingStartedAt DATETIME2 NULL,
    ProcessingCompletedAt DATETIME2 NULL,
    ErrorMessage NVARCHAR(MAX) NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_Documents_Collections FOREIGN KEY (CollectionId) REFERENCES
company_{companyId}.Collections(CollectionId),
    CONSTRAINT FK_Documents_Users FOREIGN KEY (UserId) REFERENCES
company_{companyId}.Users(UserId),
    INDEX IX_Documents_CollectionId (CollectionId),
    INDEX IX_Documents_UserId (UserId),
    INDEX IX_Documents_Status (Status)
);
```

Conversations

```
CREATE TABLE company_{companyId}.Conversations (
    ConversationId INT IDENTITY(1,1) PRIMARY KEY,
    Title NVARCHAR(255) NULL,
    UserId INT NOT NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_Conversations_Users FOREIGN KEY (UserId) REFERENCES
company_{companyId}.Users(UserId),
    INDEX IX_Conversations_UserId (UserId)
);
```

ConversationMessages

```
CREATE TABLE company_{companyId}.ConversationMessages (
    MessageId INT IDENTITY(1,1) PRIMARY KEY,
    ConversationId INT NOT NULL,
    Role NVARCHAR(20) NOT NULL,
    Content NVARCHAR(MAX) NOT NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_ConversationMessages_Conversations FOREIGN KEY
(ConversationId) REFERENCES company_{companyId}.Conversations(ConversationId),
    INDEX IX_ConversationMessages_ConversationId (ConversationId)
);
```

ConversationCollections

```
CREATE TABLE company_{companyId}.ConversationCollections (
    ConversationCollectionId INT IDENTITY(1,1) PRIMARY KEY,
    ConversationId INT NOT NULL,
    CollectionId INT NOT NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_ConversationCollections_Conversations FOREIGN KEY
(ConversationId) REFERENCES company_{companyId}.Conversations(ConversationId),
    CONSTRAINT FK_ConversationCollections_Collections FOREIGN KEY
(CollectionId) REFERENCES company_{companyId}.Collections(CollectionId),
    CONSTRAINT UQ_ConversationCollections UNIQUE (ConversationId, CollectionId)
);
```

UserLogs

```
CREATE TABLE company_{companyId}.UserLogs (
    LogId BIGINT IDENTITY(1,1) PRIMARY KEY,
    UserId INT NOT NULL,
    ActionType NVARCHAR(50) NOT NULL,
    ActionDetails NVARCHAR(MAX) NULL,
    IpAddress NVARCHAR(50) NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_UserLogs_Users FOREIGN KEY (UserId) REFERENCES
company_{companyId}.Users(UserId),
    INDEX IX_UserLogs_UserId (UserId),
    INDEX IX_UserLogs_CreatedAt (CreatedAt)
);
```

UserQuotaUsage

```
CREATE TABLE company_{companyId}.UserQuotaUsage (
    QuotaUsageId INT IDENTITY(1,1) PRIMARY KEY,
    UserId INT NOT NULL,
    QuotaType NVARCHAR(50) NOT NULL,
    UsageCount INT NOT NULL DEFAULT 0,
    ResetAt DATETIME2 NOT NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
    CONSTRAINT FK_UserQuotaUsage_Users FOREIGN KEY (UserId) REFERENCES
company_{companyId}.Users(UserId),
    CONSTRAINT UQ_UserQuotaUsage UNIQUE (UserId, QuotaType, ResetAt),
    INDEX IX_UserQuotaUsage_UserId (UserId),
    INDEX IX_UserQuotaUsage_ResetAt (ResetAt)
);
```

Stored Procedures

Creazione Schema per Nuovo Tenant

```
CREATE PROCEDURE dbo.CreateTenantSchema
    @CompanyId INT,
    @SchemaName NVARCHAR(50)
AS
BEGIN
    DECLARE @SQL NVARCHAR(MAX);

    -- Crea lo schema
    SET @SQL = N'CREATE SCHEMA ' + QUOTENAME(@SchemaName);
    EXEC sp_executesql @SQL;

    -- Crea le tabelle per il tenant
    -- CompanySettings
    SET @SQL = REPLACE('
    CREATE TABLE [schema_name].CompanySettings (
        SettingId INT IDENTITY(1,1) PRIMARY KEY,
        SettingKey NVARCHAR(50) NOT NULL,
        SettingValue NVARCHAR(MAX) NULL,
        Description NVARCHAR(255) NULL,
        IsEncrypted BIT NOT NULL DEFAULT 0,
        CreatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
        UpdatedAt DATETIME2 NOT NULL DEFAULT GETUTCDATE(),
        CONSTRAINT UQ_CompanySettings_Key UNIQUE (SettingKey)
    );', '[schema_name]', QUOTENAME(@SchemaName));
    EXEC sp_executesql @SQL;

    -- Continua con la creazione delle altre tabelle...
END
```

Gestione Quota Utente

```
CREATE PROCEDURE company_{companyId}.CheckAndUpdateUserQuota
    @UserId INT,
    @QuotaType NVARCHAR(50),
    @IncrementBy INT = 1,
    @IsAllowed BIT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @MaxAllowed INT;
    DECLARE @CurrentUsage INT;
    DECLARE @ResetAt DATETIME2;

    -- Ottieni il limite massimo dalla configurazione
    IF @QuotaType = 'DocumentUpload'
        SELECT @MaxAllowed = MaxDocumentsPerUser FROM
company_{companyId}.R2RCompanyConfiguration;
    ELSE IF @QuotaType = 'QueryPerHour'
        SELECT @MaxAllowed = MaxQueriesPerHour FROM
company_{companyId}.R2RCompanyConfiguration;
    ELSE
        SET @MaxAllowed = 100; -- Default

    -- Determina il periodo di reset
    IF @QuotaType = 'QueryPerHour'
        SET @ResetAt = DATEADD(HOUR, 1, DATEADD(HOUR, DATEDIFF(HOUR, 0,
GETUTCDATE()), 0));
    ELSE
        SET @ResetAt = DATEADD(DAY, 1, DATEADD(DAY, DATEDIFF(DAY, 0,
GETUTCDATE()), 0));

    -- Verifica se esiste già un record per questo periodo
    SELECT @CurrentUsage = UsageCount
    FROM company_{companyId}.UserQuotaUsage
    WHERE UserId = @UserId AND QuotaType = @QuotaType AND ResetAt = @ResetAt;

    IF @CurrentUsage IS NULL
    BEGIN
        -- Crea un nuovo record
        INSERT INTO company_{companyId}.UserQuotaUsage (UserId, QuotaType,
UsageCount, ResetAt)
        VALUES (@UserId, @QuotaType, @IncrementBy, @ResetAt);

        SET @CurrentUsage = @IncrementBy;
    END
    ELSE
    BEGIN
        -- Aggiorna il record esistente
        UPDATE company_{companyId}.UserQuotaUsage
```

```
        SET UsageCount = UsageCount + @IncrementBy,
            UpdatedAt = GETUTCDATE()
        WHERE UserId = @UserId AND QuotaType = @QuotaType AND ResetAt =
@ResetAt;

        SET @CurrentUsage = @CurrentUsage + @IncrementBy;
    END

    -- Verifica se la quota è stata superata
    IF @CurrentUsage <= @MaxAllowed
        SET @IsAllowed = 1;
    ELSE
        SET @IsAllowed = 0;
END
```

Indici e Ottimizzazioni

Oltre agli indici già definiti nelle tabelle, si consiglia di creare i seguenti indici aggiuntivi per ottimizzare le query più frequenti:

```
-- Indici per migliorare le prestazioni delle query di ricerca documenti
CREATE INDEX IX_Documents_CreatedAt ON company_{companyId}.Documents
(CreatedAt);
CREATE INDEX IX_Documents_FileType ON company_{companyId}.Documents (FileType);

-- Indici per migliorare le prestazioni delle query di ricerca conversazioni
CREATE INDEX IX_Conversations_CreatedAt ON company_{companyId}.Conversations
(CreatedAt);
CREATE INDEX IX_ConversationMessages_CreatedAt ON
company_{companyId}.ConversationMessages (CreatedAt);

-- Indici per migliorare le prestazioni delle query di monitoraggio
CREATE INDEX IX_UserLogs_ActionType ON company_{companyId}.UserLogs
(ActionType);
CREATE INDEX IX_UserQuotaUsage_QuotaType ON company_{companyId}.UserQuotaUsage
(QuotaType);
```

Considerazioni sulla Sicurezza

1. **Crittografia dei Dati Sensibili**: Le password sono memorizzate come hash, mentre le API key e altri dati sensibili sono crittografati.
2. **Isolamento dei Dati**: Ogni tenant ha il proprio schema, garantendo l'isolamento dei dati.
3. **Controllo degli Accessi**: Le stored procedure e le viste possono essere utilizzate per implementare un controllo degli accessi granulare.
4. **Audit Trail**: Le tabelle di log registrano tutte le azioni degli utenti per scopi di audit.

Considerazioni sulla Scalabilità

1. **Partizionamento**: Per tabelle che crescono rapidamente (come SystemLogs, UserLogs, ConversationMessages), considerare l'implementazione del partizionamento per data.

2. **Indici Columnstore**: Per tabelle di grandi dimensioni utilizzate principalmente per analisi, considerare l'uso di indici columnstore.

3. **Archivio Dati**: Implementare una strategia di archiviazione per i dati storici, spostando i dati meno recenti in tabelle di archivio.

Conclusioni

Il modello dati proposto è progettato per supportare un'applicazione multitenant enterprise-grade con isolamento dei dati, scalabilità e sicurezza. La struttura a schema separato per tenant offre un buon equilibrio tra isolamento e efficienza delle risorse, mentre le stored procedure e gli indici ottimizzano le operazioni comuni.