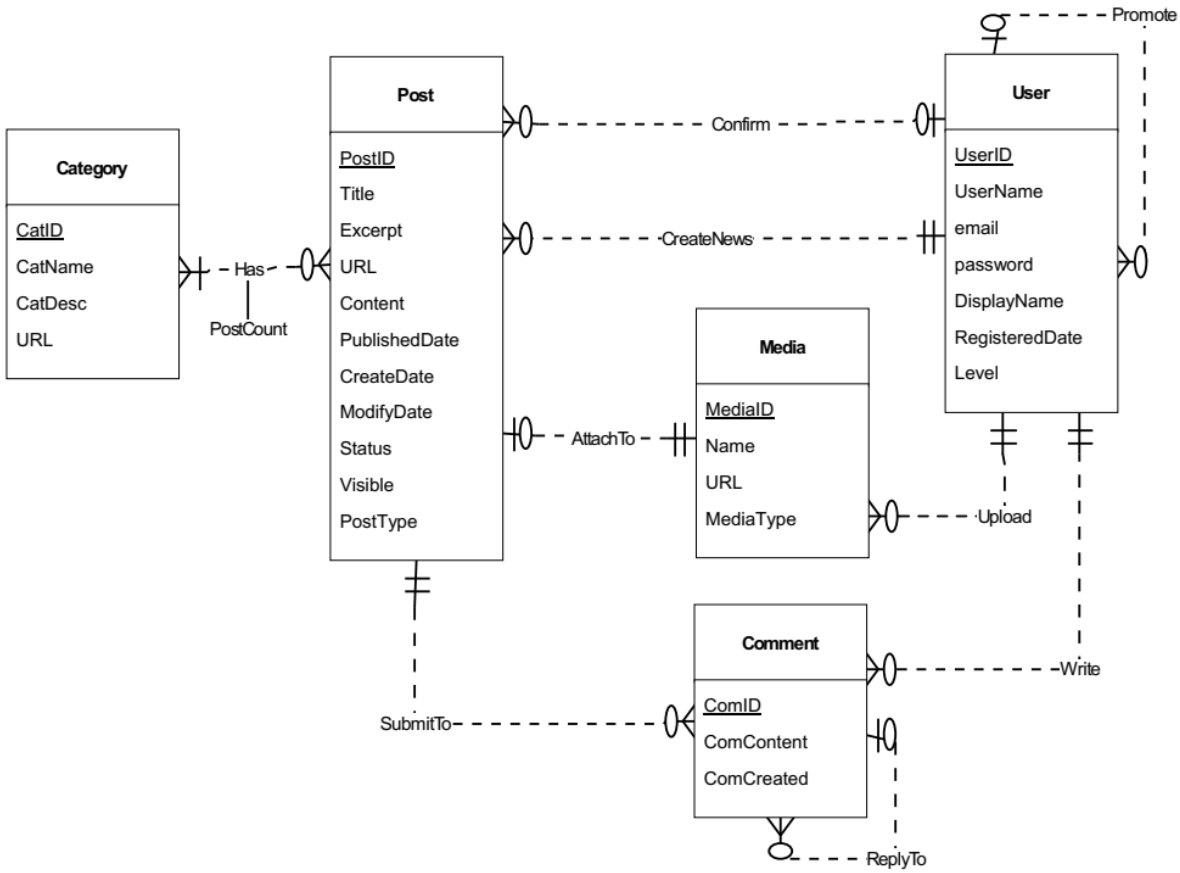
Xây dựng CSDL Website Tin Tức bằng T-SQL

Người thực hiện: Nguyễn Trần Phối
Mã Học Viên: FX10207
Lớp CCDN

Mô hình ERD đã tạo từ Assignment 1



Lược đồ CSDL

CREATE TABLE User (

```
        PRIMARY KEY (UserID),
        FOREIGN KEY (PromoID) REFERENCES (User.UserID), …
        )

CREATE TABLE Category(

PRIMARY KEY (CatID), …
)

CREATE TABLE Media(

        PRIMARY KEY (MediaID),
        FOREIGN KEY (Author) REFERENCES (User.UserID) NOT NULL,…

)

CREATE TABLE Post (

        PRIMARY KEY (PostID),
        FOREIGN KEY (FeatureImage) REFERENCES (Media.MediaID) NOT
        NULL,
        FOREIGN KEY (Author) REFERENCES (User.UserID) NOT NULL,
        FOREIGN KEY (Editor) REFERENCES (User.UserID), …
        )

CREATE TABLE Comment(

        PRIMARY KEY (ComID),
        FOREIGN KEY (PostID) REFERENCES (Post.PostID) NOT NULL,
        FOREIGN KEY (ComAuthor) REFERENCES (User.UserID) NOT NULL,
        FOREIGN KEY (ComParent) REFERENCES (Comment.ComID), …
        )

CREATE TABLE PostInCat (

        PRIMARY KEY (PostID, CatID),
        FOREIGN KEY (PostID) REFERENCES (Post.PostID),
        FOREIGN KEY (CatID) REFERENCES (Category.CatID), …
        )
```

Các bước tạo hệ CSDL cho một website Tin tức

Khởi tạo database mới (NEWS_WEBSITE)

Đơn giản nhất, ta có thể khởi tạo một hệ CSDL mới thông qua giao diện của SSMS. Trong cửa sổ *Object Explorer*, chuột phải vào thư mục *Databases* và chọn *New Database.* Đặt tên cho database (NEWS_WEBSITE) rồi nhấp *OK*.

Để lưu lại truy vấn tạo CSDL trên, chuột phải vào database vừa tạo trong thư mục *Databases* (NEWS_WEBSITE) -> Script database as -> CREATE To -> New query editor window. Lúc này, một query window được mở ra và chứa tất cả các statement cần thiết để tạo một CSDL.

Tạo các table trong CSDL

Trong query window đã tạo CSDL ở trên, trước khi muốn tạo các table trong CSDL này, ta cần chuyển không gian làm việc hiện tại (master) vào CSDL (NEWS_WEBSITE) bằng câu lệnh USE [NEWS_WEBSITE].

Tiếp đó, dựa vào mô hình ERD và lược đồ CSDL ta tạo các table cho CSDL. Xác định các mối quan hệ parent – child hoặc thực tế thực thể nào có trước cần tạo trước. Theo thứ tự:

1. tblUser
2. tblCategory
3. tblMedia
4. tblPost
5. tblComment
6. tblPostInCat

Sử dụng câu lệnh CREATE TABLE tblName (Attribute 1, Attribute 2, …) để tạo table. Đồng thời xác định khóa chính, khóa ngoại và các ràng buộc trong lúc tạo bảng.

Chi tiết trong source code: CreateTable.sql

Insert dữ liệu vào bảng

Cũng theo thứ tự tạo bảng, insert dữ liệu bằng câu lệnh:

INSERT INTO tblName (Attribute1, Attribute2,…)
VALUES (ValAttr1.1, ValAttr1.2,…), (ValAttr2.1, ValAttr2.2,…),…

Chi tiết source code: InsertTable.sql

Update & Delete dữ liệu

Nếu cần thiết, sử dụng câu lệnh dưới đây để cập nhật dữ liệu:

UPDATE tblName SET Attribute = Value WHERE condition

Hoặc xóa dữ liệu:

DELETE FROM tblName WHERE condition

TRIGGER

(chi tiết trong BasicFunction.sql)

```
CREATE TRIGGER trg_checkUserLevel ON [dbo].[tblPOST]
AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON
	DECLARE @AUTHOR INT
	SELECT @AUTHOR = PostAuthor from inserted
	IF NOT EXISTS (SELECT * FROM tblUSER WHERE UserID =
@AUTHOR AND UserRole < 3)
	BEGIN
		RAISERROR('User do not have permission!',15,0)
		ROLLBACK
	END
END
GO
```

Trigger được tạo trên bảng tblPOST nhằm kiểm tra dữ liệu INSERT vào bảng có PostAuthor (khóa ngoại từ tblUser.UserID) đã được khởi tạo trong bảng tblUSER hay chưa, và đồng thời USER này có được phép viết bài trên trang tin tức này hay không.

Trong tblUSER có thuộc tính UserRole quy định quyền của USER trên Website với các giá trị:

- 1: quyền cao nhất, quyền biên tập viên (Editor)

- 2: quyền viết bài, upload hình ảnh (nhưng không được đăng bài mà phải chờ duyệt)
- 3: quyền thông thường, xem và comment vào bài viết.

Trigger sẽ kiểm tra User có thỏa điều kiện đã được khởi tạo trong tblUSER và có UserRole < 3 (quyền cấp 1 hoặc cấp 2). Nếu thỏa, không cần làm gì, ngược lại, thông báo lỗi và rollback câu lệnh INSERT.

STORED PROCEDURE

(chi tiết trong BasicFunction.sql)

```
CREATE PROC proc_showPostInCat (@CatID int)
AS
BEGIN
	SELECT * FROM tblPOST
	WHERE
		PostID IN
		(SELECT PostID FROM tblCATPOST WHERE CatID = @CatID)
		AND
		PostStatus = 'Published’
	ORDER BY PublishedDate DESC
END

GO
```

Store Procedure nhận một parameter là CatID (khóa chính của tblCAT) và sẽ truy vấn dữ liệu từ bảng tblPOST những dữ liệu được nhóm bởi CatID này thông qua bảng liên kết tblCATPOST (được tạo từ kết nối M-N giữa POST entity và CATEGORY entity).

Mục đích: Đáp ứng chức năng cho phép người dùng web có thể lọc các bài viết (post) trong một danh mục (Category) cụ thể.

Thực hiện:

- SELECT * FROM tblPOST, truy vấn các dữ liệu trong bảng tblPOST.
- WHERE PostID IN, có điều kiện PostID cần thỏa.
- (SELECT PostID FROM tblCATPOST WHERE CatID = @CatID), subquery truy vấn các PostID xuất hiện trong bảng tblCATPOST có CatID là parameter nhập vào ban đầu.
- AND PostStatus = 'Published’, là các post đã được đăng lên website.

FUNCTION (UDF)

(chi tiết trong BasicFunction.sql)

```
CREATE FUNCTION fc_totalPostInCat (@CatID int)
RETURNS INT
AS
BEGIN
     DECLARE @TOTAL INT
     SELECT @TOTAL = COUNT(*) FROM tblCATPOST WHERE CatID =
@CatID
    RETURN @TOTAL
END
GO
```

Một scalar function nhận một parameter là CatID và trả về một giá trị kiểu INT là tổng số Post được nhóm bởi Category có CatID đó.

Mục đích: Thể hiện số lượng bài viết trong mỗi danh mục tại giao diện quản trị để quản lý và phân bộ nội dung cần thiết cho website.

Thực hiện:

- DECLARE @TOTAL INT, khởi tạo một biến kiểu INT để lưu giá trị trả về
- SELECT @TOTAL = COUNT(*) FROM tblCATPOST WHERE CatID = @CatID, gán giá trị vào biến thông qua hàm đếm số lượng các đối tượng xuất hiện trong bảng tblCATPOST có CatID thỏa điều kiện tham số nhập vào ban đầu.

INDEX

(chi tiết trong BasicFunction.sql)

```
CREATE NONCLUSTERED INDEX IX_tblPOST_PublishedDate
    ON [dbo].[tblPOST]([PublishedDate])
     INCLUDE (PostTitle, PostUrl, PostExcerpt)
     WHERE PostStatus = 'Published'
GO
```

Thông thường các bài viết trên website, nhất là website tin tức phải hiển thị theo thứ tự mới nhất đến cũ nhất và người dùng cũng thường tìm kiếm bài

viết theo khoảng ngày đăng. Do đó, việc tạo một chỉ mục cho tblPOST dựa trên ngày đăng là rất cần thiết.

Do tblPOST có khóa chính là PostID, cũng chính là Clustered Index của bảng, vì thế ta không thể tạo thêm 1 clustered Index nữa, mà thay vào đó là một Nonclustered Index. Ở đây, chỉ những bài viết đã được đăng và hiển thị trên website mới cần thiết cho người xem, vì thế INDEX này cũng được FILTER để tối ưu hơn.

TRANSACTION

(chi tiết trong BasicFunction.sql)

```
BEGIN TRANSACTION MyTransaction WITH MARK N'Add New Attribute
To tblPOST'
     ALTER TABLE [dbo].[tblPOST]
     ADD ComAllowed bit NOT NULL CONSTRAINT
default_allowComment default 0
     GO

     UPDATE [dbo].[tblPOST] SET ComAllowed = 1 WHERE PostID IN
(SELECT PostID from tblCOMMENT)
COMMIT TRAN
GO
```

Transaction bao gồm các câu lệnh thêm cột và update dữ liệu của cột vừa thêm.

Mục đích: Thêm một cột điều kiện cho phép comment vào một post hay không. Sau khi thêm cột và các ràng buộc vào bảng, ta cần cập nhật lại dữ liệu để thỏa điều kiện cho các dữ liệu có sẵn.

Ngoài ra, vì là cột điều kiện nên có thể cần một trigger để ràng buộc dữ liệu nhập vào hay chỉnh sửa.

```
CREATE TRIGGER tg_ComInPost ON [dbo].[tblCOMMENT]
     AFTER INSERT, UPDATE
     AS
     BEGIN
          DECLARE @POST INT, @ALLOWED bit
          SELECT @POST = PostID from inserted
          SELECT @ALLOWED = ComAllowed FROM tblPOST WHERE
PostID = @POST
          IF @ALLOWED = 0
```

```
          BEGIN
               RAISERROR('Post nay khong cho phep
comment',15,0)
               ROLLBACK
          END
     END
GO
```

Truy vấn dữ liệu

Chi tiết trong source code: BasicQuery.sql