Database Workshop1

Workshop

Data Definition Language

Data Manipulation Language

Data Control Language

Aitrich Technologies Pvt. Ltd.

Database Workshop1

Introduction

This document explains the workshop module for the topic, DDL,DCL,DML statements. It illustrates the examples of DDL,DCL.DML statements.

Objectives

The objectives of this workshop are to allow the student to explore and try the theory he learned during the lecture, and to make him well versed in the topic, so that he can use it in future.

How is it organized?

This workshop contains the SQL statements for creating a database and its associated tables related Pets App

1. Create a database for Pet Management System

```
Create database PetsAppDB
```

2. List all databases in sql server

```
SELECT name FROM master.sys.databases ORDER BY name;
```

3. Create a login

```
CREATE LOGIN PetUserLogin WITH PASSWORD =
'StrongPassword123!';
```

4. Create a database user for PetsAppDB database

```
USE PetsAppDB;
CREATE USER PetUser FOR LOGIN PetUserLogin;
```

5. For Select Database

```
USE PetsAppDB;
```

Database Workshop1

Create tables for the following:

1. Create table PetsCategory with Id as Primary Key

create table PetsCategory (Id uniqueidentifier not null primary key ,Name nvarchar(50),Image varbinary(max) not null)

PetsCategory		
Field Name	**DataType**	**Constraints**
Id	uniqueidentifier	Primary Key, Not Null
Name	nvarchar(50)	
Image	varbinary(max)	

INSERT INTO PetsCategory (Id, Name, Image)
VALUES (NEWID(), 'Cow', CAST('image1.jpg' AS varbinary(max)));

NEWID() generates a new unique identifier.
'Dog' is the name of the pet category.
CAST('image.jpg' AS varbinary(max)) simulates storing the image as binary data. In a real scenario, you would insert actual binary data for the image.

- To describe the structure of a table in SQL Server

EXEC sp_help PetsCategory;

- **Using INFORMATION_SCHEMA:**

SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, IS_NULLABLE FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = PetsCategory;

- ALTER TABLE PetsCategory

- ALTER COLUMN Image NVARCHAR(200)

2. Create table Breed with Id as Primary Key

create table Breed (Id uniqueidentifier not null primary key,Name nvarchar(50),PetsCategory uniqueidentifier Foreign key references PetsCategory(Id))

Breed		
Field Name	**DataType**	**Constraints**
Id	uniqueidentifier	Primary Key, Not Null
Name	nvarchar(50),	
PetsCategory	uniqueidentifier	Foreign key

3. Create table PublicUser with Id as Primary Key

```
CREATE TABLE PublicUser(
   Id UNIQUEIDENTIFIER NOT NULL primary key,
   FirstName NVARCHAR(50),
   LastName NVARCHAR(50),
   Email NVARCHAR(50),
   Phone NVARCHAR(10),
Role VARCHAR(50),
CONSTRAINT CK_Phone_Length CHECK (LEN(Phone) = 10
AND Phone NOT LIKE '%[^0-9]%')
);
```

- `INSERT INTO PublicUser VALUES(NEWID(),'SOUDHA','AM','Soudhasoudhu99@gmail.com','9956858989','Admin')`
- `INSERT INTO PublicUser VALUES(NEWID(),'Sana','AM','Sana99@gmail.com','9956858989','User')`

Database Workshop1

- INSERT INTO VALUES(NEWID(),'SOUDHA','AM','Soudhasoudhu99@gmail.com','9956858989','Admin')

6. Retrieves all columns and records from the PublicUser table

Select * From PublicUser

7. Write an SQL query to retrieve distinct names from the PublicUser table, ensuring that any duplicate names are removed from the result set.

Select Distinct FirstName from PublicUser

PublicUser		
Field Name	**DataType**	**Constraints**
Id	uniqueidentifier	Primary Key, Not Null
FirstName	nvarchar(50)	
LastName	nvarchar(50)	
Email	nvarchar(50)	
Phone	nvarchar(50)	CHECK
Role	varchar(50)	

4. Create table Pet with Id as Primary Key

create table Pet (Id uniqueidentifier not null,Name nvarchar(50),BreedId uniqueidentifier not null
,Age int not null, Gender nvarchar(50),Discription nvarchar(50) not null,Image varbinary(max),CategoryId uniqueidentifier not null,User_id uniqueidentifier not null,

Location varchar(50), Primary key(Id),foreign key(BreedId) references Breed(Id),foreign key(User_id) references PublicUser(Id));

Pet		
Field Name	**DataType**	**Constraints**
Id	uniqueidentifier	Primary Key, Not Null
Name	nvarchar(50)	
BreedId	uniqueidentifier	
Age	int	
Gender	nvarchar(50)	CHECK
Discription	varchar(50)	
Image	varbinary(max)	
CategoryId	uniqueidentifier	
User_id	uniqueidentifier	Foreign key
Location	string	

1. Add a column rate to the table Pet

```
ALTER TABLE Pet ADD Rate int
```

2. How To Drop a column

```
ALTER TABLE Pet DROP COLUMN Rate;
```

3. Display all tables in the database PetsAppDB

```
SELECT * FROM information_schema.tables WHERE
```

TABLE_TYPE='BASE TABLE'

4. Change the name of the table PetsCategory to Category

```
EXEC sp_rename  ' PetsCategory ',' Category'
```

5. Create a view from table Pets with fields Id, Name and Age

```
CREATE VIEW ViewOfPets  AS  SELECT Id, Name, Age
FROM Pet
```

6. Display view table

```
Select * from ViewOfPets
```

7. Remove the view ViewOfEmployee

```
Drop view ViewOfPets
```

8. Grand permission for the user 'PetUser' to Select and Update the table Pet

```
GRANT CREATE TABLE TO PetUser;
```

9. Write an SQL UPDATE statement to change the Phone number from '9956858989' to '8592093362' for the user with the email 'Soudhasoudhu99@gmail.com' in the PublicUser table.

```
UPDATE PublicUser
SET Phone = '8592093362'
WHERE Email = 'Soudhasoudhu99@gmail.com';
```

UPDATE PublicUser: Specifies the table you want to update.

SET Phone = '8592093362': Defines the new value for the Phone column.
WHERE Email = 'Soudhasoudhu99@gmail.com': Identifies the specific row to update based on the Email column.

10. How can you delete a user from the PublicUser table based on their email address 'sana99@gmail.com'?

Delete from PublicUser where Email ='sana99@gmail.com';

11. List all owners whose name starts with A

SELECT FirstName FROM PublicUser

WHERE FirstName LIKE ‘A%’

SQL Join

Certainly! Here are various questions based on the provided `PublicUser` and `Pet` tables, including different types of joins:

1. Basic Join:
   - Write a SQL query to list all pets along with their owner's first and last name.
   - Answer:

```
SELECT p.Name AS PetName, pu.FirstName, pu.LastName
FROM Pet p
JOIN PublicUser pu ON p.User_id = pu.Id;
```

2. Left Join:
   - Write a SQL query to list all users and their pets, including users who do not own any pets.
   - Answer:

```
SELECT pu.FirstName, pu.LastName, p.Name AS PetName
```

```
FROM PublicUser pu
LEFT JOIN Pet p ON pu.Id = p.User_id;
```

3. Right Join:

Write a SQL query to list all pets and their owners, including pets that do not have an owner (assuming that the `User_id` could be null in such cases).

Answer :

```
SELECT p.Name AS PetName, pu.FirstName, pu.LastName
FROM Pet p
RIGHT JOIN PublicUser pu ON p.User_id = pu.Id;
```

4. Full Outer Join :

- Write a SQL query to list all users and all pets, showing users even if they don't own any pets and pets even if they don't have an owner.

Answer

```
SELECT pu.FirstName, pu.LastName, p.Name AS PetName
```

FROM PublicUser pu FULL OUTER JOIN Pet p ON pu.Id = p.User_id;

5. Self Join:

-Assume that a user can refer other users to the system, and you need to find pairs of users where one user referred another. Modify the `PublicUser` table to include a `ReferredBy` column, which is a foreign key referencing the `Id` of another `PublicUser`. Then, write a query to list each user along with the user who referred them.

```
ALTER TABLE PublicUser
ADD ReferredBy UNIQUEIDENTIFIER;

ALTER TABLE PublicUser
ADD CONSTRAINT FK_ReferredBy FOREIGN KEY (ReferredBy) REFERENCES PublicUser(Id);

SELECT pu1.FirstName AS UserFirstName, pu1.LastName AS UserLastName,
       pu2.FirstName AS ReferredByFirstName, pu2.LastName AS ReferredByLastName
FROM PublicUser pu1
LEFT JOIN PublicUser pu2 ON pu1.ReferredBy = pu2.Id;
```

6. Cross Join

Write a SQL query to create a combination of all users with all pets.

Answer:

```
SELECT pu.FirstName, pu.LastName, p.Name AS PetName
FROM PublicUser pu
CROSS JOIN Pet p;
```

7. Aggregation with Join:

Write a SQL query to find the number of pets each user owns.

- Answer:

```
SELECT pu.FirstName, pu.LastName, COUNT(p.Id) AS PetCount
FROM PublicUser pu
LEFT JOIN Pet p ON pu.Id = p.User_id
GROUP BY pu.FirstName, pu.LastName;
```

8. Join with Filtering:

Write a SQL query to list all male pets and their owners' names.

- Answer:

```
SELECT p.Name AS PetName, pu.FirstName,
pu.LastName
FROM Pet p
JOIN PublicUser pu ON p.User_id = pu.Id
WHERE p.Gender = 'Male';
```

Stored Procedure

To create a stored procedure for adding a new record to the PublicUser table, you can use the following SQL script. This procedure will accept parameters for each of the columns in the table, ensure the Id is a unique identifier, and insert the new record.

```
CREATE PROCEDURE InsertPublicUser
    @Id UNIQUEIDENTIFIER,
    @FirstName NVARCHAR(50),
    @LastName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Phone NVARCHAR(10),
    @Role VARCHAR(50)
AS
BEGIN
```

```
    SET NOCOUNT ON;

    -- Validate the phone number length and format
    IF LEN(@Phone) != 10 OR @Phone LIKE '%[^0-9]%'
    BEGIN
        RAISERROR('Phone number must be exactly 10 digits and contain only numeric characters.', 16, 1);
        RETURN;
    END

    -- Insert the record into PublicUser table
    BEGIN TRY
        INSERT INTO PublicUser (Id, FirstName, LastName, Email, Phone, Role)
        VALUES (@Id, @FirstName, @LastName, @Email, @Phone, @Role);
    END TRY
    BEGIN CATCH
        -- Handle any errors that occur during the insert operation
        DECLARE @ErrorMessage NVARCHAR(4000);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;
```

```
        SELECT
            @ErrorMessage = ERROR_MESSAGE(),
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();

        RAISERROR(@ErrorMessage, @ErrorSeverity,
@ErrorState);
    END CATCH
END;
GO
```

Database Workshop1

Explanation:

Parameter Declaration:

The procedure accepts parameters corresponding to each column in the PublicUser table.

Validation:

Before inserting the data, it checks if the @Phone parameter is exactly 10 digits long and contains only numeric characters.

If the validation fails, it raises an error and stops the execution.

Insert Operation:

It attempts to insert the provided values into the PublicUser table within a BEGIN TRY block.

If an error occurs during the insert operation, it is caught in the BEGIN CATCH block, which retrieves the error message, severity, and state, and re-raises the error.

Database Workshop1