KloudOne Assignment – 2 21-08-2020

Various Datatypes :

Comparison between sql server and postgresql

|  |  |  |
| --- | --- | --- |
| **Data type** | **PostgreSQL** | **SQL Server** |
| **64-bit integer** | **BIGINT** | **BIGINT** |
| **Fixed length byte string** | **BYTEA** | **BINARY(n)** |
| **1, 0 or NULL** | **BOOLEAN** | **BIT** |
| **Fixed length char string, 1 <= n <= 8000** | **CHAR(n)** | **CHAR(n)** |
| **Variable length char string, 1 <= n <= 8000** | **VARCHAR(n)** | **VARCHAR(n)** |
| **Variable length char string, <= 2GB** | **TEXT** | **VARCHAR(max)** |
| **Variable length byte string , 1 <= n <= 8000** | **BYTEA** | **VARBINARY(n)** |
| **Variable length byte string , <= 2GB** | **BYTEA** | **VARBINARY(max)** |
| **Variable length Unicode UCS-2 string** | **VARCHAR(n)** | **NVARCHAR(n)** |
| **Variable length Unicode UCS-2 data, <= 2GB** | **TEXT** | **NVARCHAR(max)** |
| **Variable length character data, <= 2GB** | **TEXT** | **TEXT** |
| **Variable length Unicode UCS-2 data, <= 2GB** | **TEXT** | **NTEXT** |
| **Double precision floating point number** | **DOUBLE PRECISION** | **DOUBLE PRECISION** |
| **Floating point number** | **DOUBLE PRECISION** | **FLOAT(p)** |
| **32 bit integer** | **INTEGER** | **INTEGER** |
| **Fixed point number** | **NUMERIC(p,s)** | **NUMERIC(p,s)** |
| **Date includes year, month, and day** | **DATE** | **DATE** |
| **Date and time with fractional seconds** | **TIMESTAMP(p)** | **DATETIME, DATETIME2(p)** |
| **Date and time with time zone** | **TIMESTAMP(p) WITH TIME ZONE** | **DATETIMEOFFSET(p)** |
| **Date and time** | **TIMESTAMP(0)** | **SMALLDATETIME** |
| **Unsigned integer, 0 to 255 (8 bit)** | **SMALLINT** | **TINYINT** |
| **UUID (16 byte)** | **CHAR(16)** | **UNIQUEIDENTIFIER** |
| **Automatically updated binary data** | **BYTEA** | **ROWVERSION** |
| **Currency amount (32 bit)** | **MONEY** | **SMALLMONEY** |
| **Variable length binary data, <= 2GB** | **BYTEA** | **IMAGE** |
| **Geometric types** | **POINT, LINE, LSEG, BOX, PATH, POLYGON, CIRCLE** | **GEOMETRY** |

**Creation of Tables:**

1. Create Table

A table or relation can be created in a database by CREATE TABLE statement of SQL.

CREATE TABLE<table name>

( <attribute name> <datatype> [size] [constraint],

<attribute name> <datatype> [size] [constraint],

…..

);

Constraints are for example:

Type Description

NULL/NOT NULL Specifies, if a column can or cannot have NULL values.

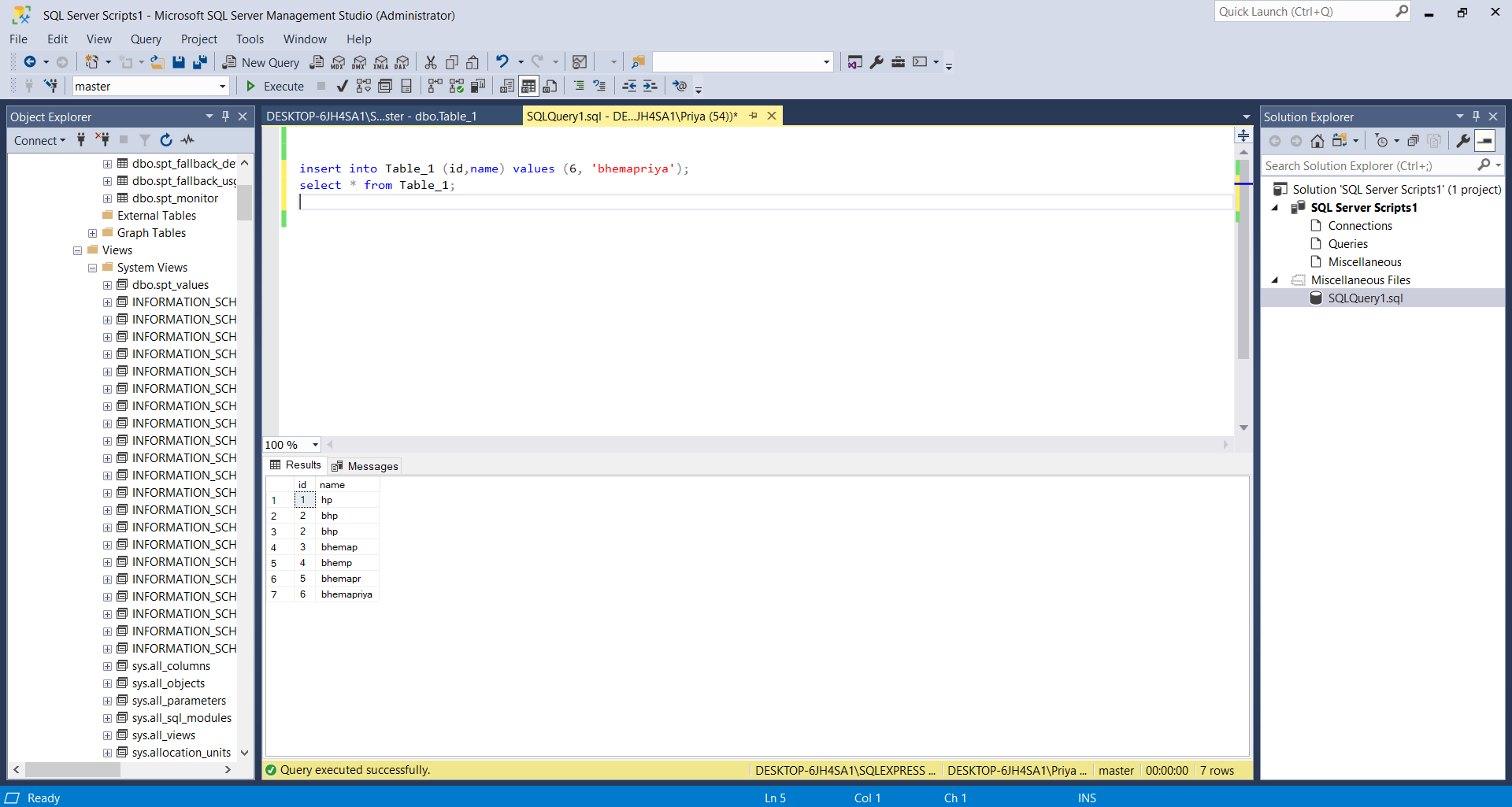
UNIQUE Value of a column have to be unique.

PRIMARY KEY Defines it to be a primary key.

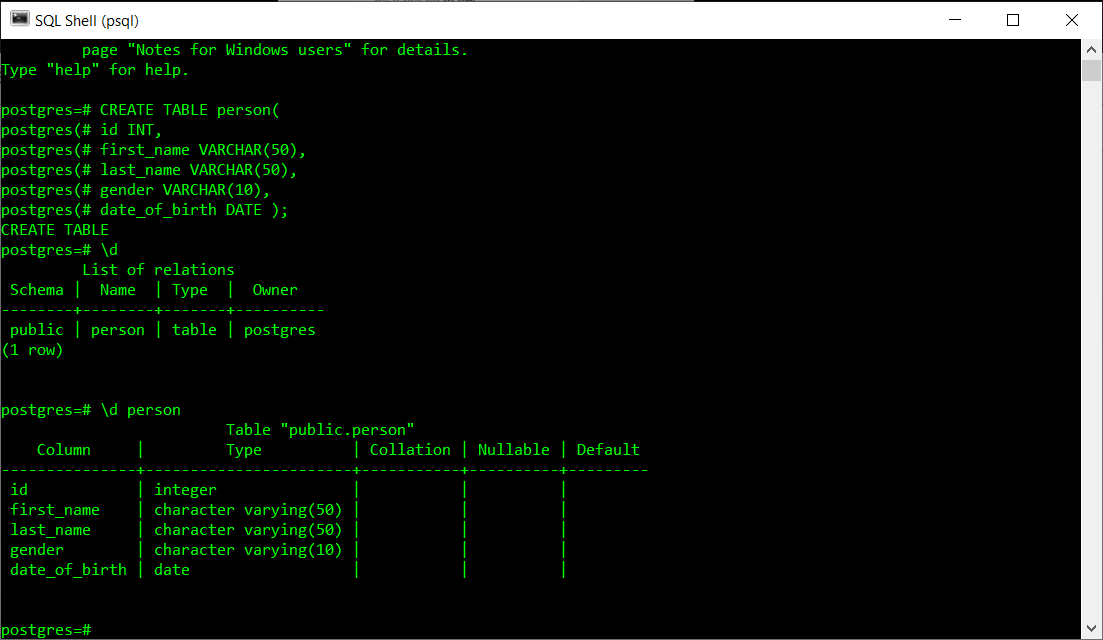
DEFAULT It prevents null values in a row.

CHECK Explicitly defines a condition that each row must satisfy.

Using Microsoft SQL:



Using Postgresql:



TRIGGERS

A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, DELETE or TRUNCATE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

From my learning

* A postgresql trigger is a function invoked automatically whenever an event associated with a table occurs.
* A event could be INSERT,UPDATE,DELETE or TRUNCATE
* A trigger is a special user-defined function that binds a table.
* You can specify whether the trigger is invoked before or after an event

Types of triggers:

1. Row trigger
2. Statement trigger

Benefits of using trigger:

1. Maintains data integrity rules
2. Check activities of applications accessing the database

Drawbacks of using triggers

1. Remembers trigger exists
2. Understand the logic

Postgresql trigger features:

* Postgresql fires trigger for truncate event.
* Postgresql allows you to define statement-level trigger on views.
* Postgresql requires you to define a user-defined function as the action of trigger.

Creating a trigger:

SYNTAX

CREATE TRIGGER trigger\_name [BEFORE|AFTER|INSTEAD OF] event\_name

ON table\_name

[

-- Trigger logic goes here....

];

Syntax for update option of the triggers:

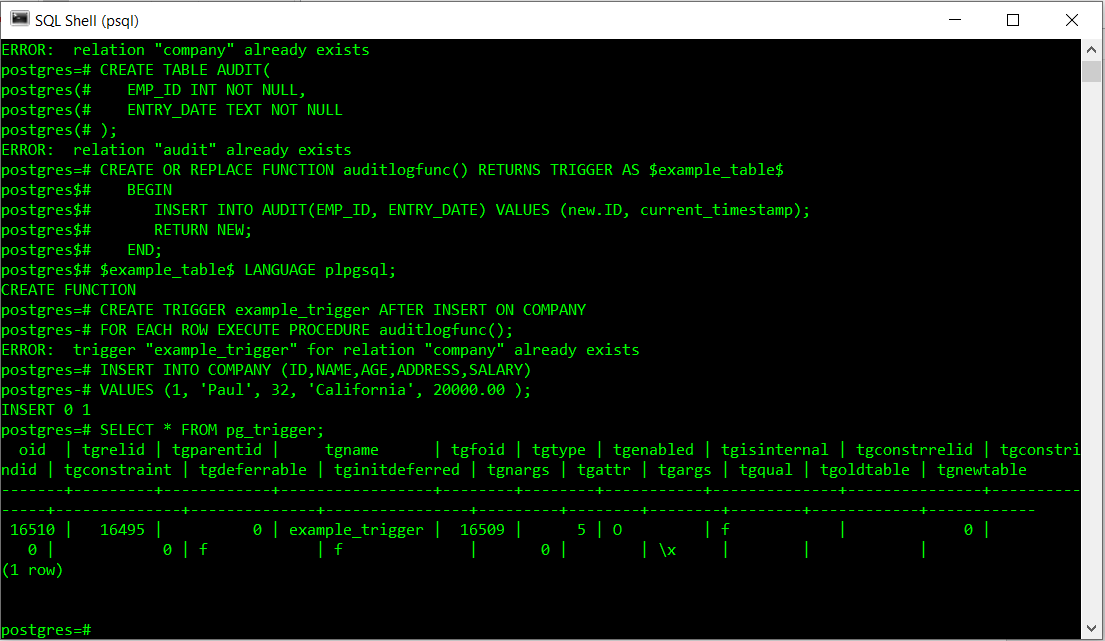
CREATE TRIGGER trigger\_name [BEFORE|AFTER] UPDATE of column\_name ON table\_name

[

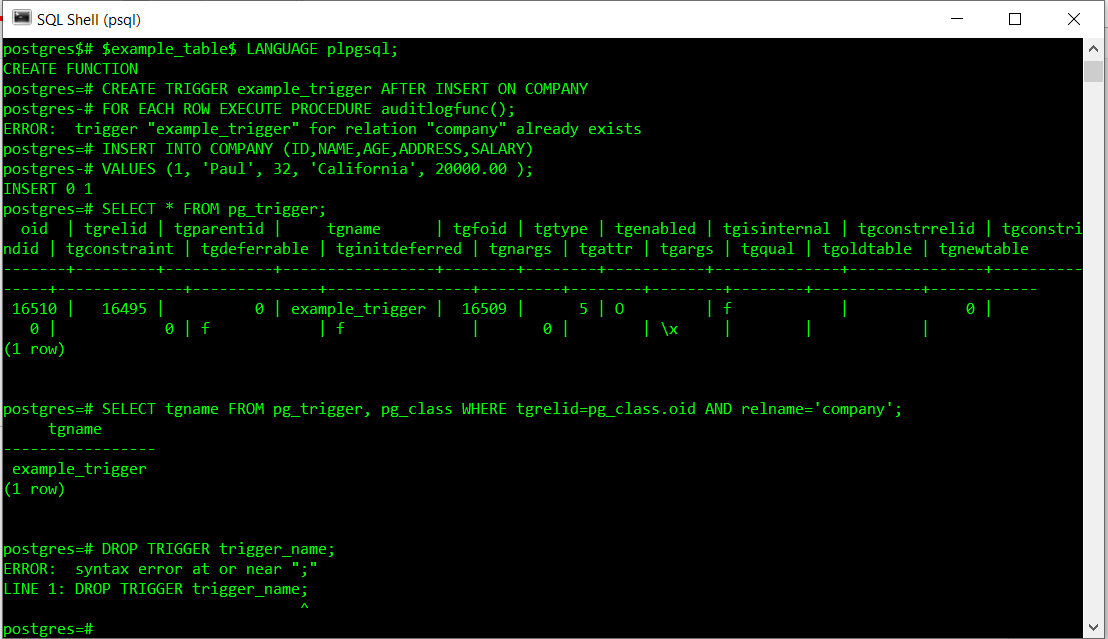
-- Trigger logic goes here....

];

A screenshot of the trigger creation with the listing of the triggers.



Listing and dropping of triggers created:



Stored Procedures:

1. Create a stored procedure:

Stored procedures in Microsoft SQL Server are similar to procedures in other programming languages in that they can:

* Accept input parameters and return multiple values in the form of output parameters to the calling procedure or batch.
* Contain programming statements that perform operations in the database, including calling other procedures.
* Return a status value to a calling procedure or batch to indicate success or failure (and the reason for failure).

You can use the Transact-SQL EXECUTE statement to run a stored procedure. Stored procedures are different from functions in that they do not return values in place of their names and they cannot be used directly in an expression.

The benefits of using stored procedures in SQL Server rather than Transact-SQL programs stored locally on client computers are:

* They are registered at the server.
* They can have security attributes (such as permissions) and ownership chaining, and certificates can be attached to them.

Users can be granted permission to execute a stored procedure without having to have direct permissions on the objects referenced in the procedure.

* They can enhance the security of your application.

Parameterized stored procedures can help protect your application from SQL Injection attacks.

* They allow modular programming.

You can create the procedure once, and call it any number of times in your program. This can improve the maintainability of your application and allow applications to access the database in a uniform manner.

* They are named code allowing for delayed binding.

This provides a level of indirection for easy code evolution.

* They can reduce network traffic.

An operation requiring hundreds of lines of Transact-SQL code can be performed through a single statement that executes the code in a procedure, rather than by sending hundreds of lines of code over the network.

