

Understanding the GUID data type in SQL Server

May 3, 2018 by [Ben Richardson](#)

What is a GUID?

GUID is a 16 byte binary SQL Server data type that is globally unique across tables, databases, and servers. The term GUID stands for Globally Unique Identifier and it is used interchangeably with UNIQUEIDENTIFIER.

To create a GUID in SQL Server, the NEWID() function is used as shown below:

```
SELECT NEWID()
```

Execute the above line of SQL multiple times and you will see a different value every time. This is because the NEWID() function generates a unique value whenever you execute it.

To declare a variable of type GUID, the keyword used is UNIQUEIDENTIFIER as mentioned in the script below:

```
DECLARE @UNI UNIQUEIDENTIFIER  
SET @UNI = NEWID()  
  
SELECT @UNI
```

As mentioned earlier, GUID values are unique across tables, databases, and servers. GUIDs can be considered as global primary keys. Local primary keys are used to uniquely identify records within a table. On the other hand, GUIDs can be used to uniquely identify records across tables, databases, and servers.



The Problem GUID Solves

Let's see what issues we face if we have redundant records within tables across different databases and how GUID solves these issues.

Execute the following script.

```
CREATE DATABASE EngDB
GO

USE EngDB
GO

CREATE TABLE EnglishStudents
(
    Id INT PRIMARY KEY IDENTITY,
    StudentName VARCHAR (50)
)
GO

INSERT INTO EnglishStudents VALUES ('Shane')
INSERT INTO EnglishStudents VALUES ('Jonny')
```

In the script above, we create a database named "EngDB". We then create a table "EnglishStudents" within this database. The table has two columns: Id and StudentName. The Id column is the primary key column and we set it to auto increment using Identity as the constraint. Finally, we insert two records for students called 'Shane' and 'Jonny' into the "EnglishStudents" table.

Now if you select all the records from the "EnglishStudents" table, you should see the following output:

Id	StudentName
1	Shane
2	Jonny



Now, let's create another database `MathDB`, create a table `MathStudents` in the DB and insert some records into the table. Execute the following script to do so.

```
CREATE DATABASE MathDB
GO

USE MathDB
GO

CREATE TABLE MathStudents
(
    Id INT PRIMARY KEY IDENTITY,
    StudentName VARCHAR (50)
)
GO

INSERT INTO MathStudents VALUES ('Sally')
INSERT INTO MathStudents VALUES ('Edward')
```

The `MathStudents` table of the `MathDB` should have the following records.

Id	StudentName
1	Sally
2	Edward

Now if you select all the records from the `EnglishStudents` table of the `EngDB` and `MathStudents` table of the `MathDB`, you will see that records from both tables will have the same values for the primary key columns `Id`. Execute the following script to see this result:

```
SELECT * FROM EngDB.dbo.EnglishStudents
SELECT * FROM MathDB.dbo.MathStudents
```

You will see the following output in SQL Server Management Studio



Id	StudentName
----	-------------



1	1	Shane
2	2	Jonny

	Id	StudentName
1	1	Sally
2	2	Edward

The student records from different tables that exist in two different databases have the same value for the Id column. This is the default behavior of SQL Server.

Now let's create a new table "Student" that contains the union of all the records from the MathStudents table and EnglishStudents table. Execute the following script:

```
USE EngDB
GO

CREATE TABLE Students
(
    Id INT PRIMARY KEY,
    StudentName NVARCHAR (50)
)
GO

INSERT INTO Students
SELECT * FROM EngDB.dbo.EnglishStudents
UNION ALL
SELECT * FROM MathDB.dbo.MathStudents
```

In the above script, we create a new table "Students" in the EngDB. This table contains Id and StudentName columns.

If you try to run the above script, you will see an error:

```
Msg 2627, Level 14, State 1, Line 11
Violation of PRIMARY KEY constraint 'PK__Students__3214EC07C6AB2BA8'.
| Cannot insert duplicate key in object 'dbo.Students'. The duplicate key value is (1).
The statement has been terminated.
```



This error is due to both the MathStudents and EnglishStudents table having the same values for the Id column which is also the primary key column for the newly created Students table. Therefore, when we try to insert the union of the records from MathStudents and EnglishStudents tables, the "Violation of PRIMARY KEY constraint" error occurs. Execute the following script to see what we are actually trying to insert in the Students table.

```
SELECT * FROM EngDB.dbo.EnglishStudents
UNION ALL
SELECT * FROM MathDB.dbo.MathStudents
```

	Id	StudentName
1	1	Shane
2	2	Jonny
3	1	Sally
4	2	Edward

However, what if we want records to have unique values across multiple databases? For instance, we want that the Id column of the EnglishStudents table and the MathStudents table to have unique values, even if they belong to different databases. This is when we need to use the GUID data type.

You can see that students Shane and Sally both have Ids of 1 while Jonny and Edward both have Ids of 2. This causes the violation of primary key constraint for the Students table.

Solution with GUID

Now, let's see how GUID can be used to solve this issue

Let's create a table EngStudents1 within the EngDB but this time we change the data type of the Id column from INT to UNIQUEIDENTIFIER. To set a default value for the column we will use the default keyword and set the default value as the value returned by the 'NEWID()' function.

This will ensure that whenever a new record is inserted in the EngStudents1 table, by default, the NEWID() function generates a unique



value for the Id column. When inserting the records, we simply have to specify "default" as value for the first column. This will insert a default unique value to the Id column. Execute the following script to create EngStudents1 table:

```
USE EngDB
GO

CREATE TABLE EnglishStudents1
(
    Id UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
    StudentName VARCHAR (50)
)
GO

INSERT INTO EnglishStudents1 VALUES (default, 'Shane')
INSERT INTO EnglishStudents1 VALUES (default, 'Jonny')
```

Now if you select all the records from EnglishStudents1 table, you will a result that looks like this:

Id	StudentName
4B900A74-E2D9-4837-B9A4-9E828752716E	Jonny
AEDC617C-D035-4213-B55A-DAE5CDFCA366	Shane

Note: Your values for the Id column will be different from the ones shown in the above table, because they are generated randomly on the fly. However, they should be globally unique.

In the same way, create another table MathStudents1 in the MathDB database. Execute the following script:

```
USE MathDB
GO

CREATE TABLE MathStudents1
(
    Id UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
    StudentName VARCHAR (50)
)
```



```
/
GO

INSERT INTO MathStudents1 VALUES (default, 'Sally')
INSERT INTO MathStudents1 VALUES (default, 'Edward')
```

Again if you try to retrieve all the records from the MathStudents1 table of the MathDB database, you will see results similar to the one below:

Id	StudentName
69121893-3AFC-4F92-85F3-40BB5E7C7E29	Sally
CB77CCE6-C2CB-471B-BDD4-5DAC8C93B756	Edward

Now we have globally unique values in the Id columns of both the EnglishStudents1 and MathStudents1 table. Let's create a new Table named Students1 and just as we did before, try to insert the union of the records from EnglishStudents1 and MathStudents1. This time we will see that there will be no "Violation of PRIMARY KEY constraint" error, since the values in the Id column of both EnglishStudents1 and MathStudents1 are unique across both the EngDB and MathDB databases.

```
USE EngDB
GO

CREATE TABLE Students1
(
    Id UNIQUEIDENTIFIER PRIMARY KEY,
    StudentName NVARCHAR (50)
)
GO

INSERT INTO Students1
SELECT * FROM EngDB.dbo.EnglishStudents1
UNION ALL
SELECT * FROM MathDB.dbo.MathStudents1
```

You can see in the above script that the type of the Id column is UNIQUEIDENTIFIER. Run the above script and then try to retrieve all the records from the Students1 table and you should see results similar to the following:



Id	StudentName
----	-------------

69121893-3AFC-4F92-85F3-40BB5E7C7E29	Sally
--------------------------------------	-------

CB77CCE6-C2CB-471B-BDD4-5DAC8C93B756	Edward
--------------------------------------	--------

4B900A74-E2D9-4837-B9A4-9E828752716E	Jonny
--------------------------------------	-------

AEDC617C-D035-4213-B55A-DAE5CDFCA366	Shane
--------------------------------------	-------

You can see that using GUID, we are able to insert a union of records from two different databases into a new table without any "Violation of PRIMARY KEY constraint" error.

References

- [Using UNIQUEIDENTIFIER](#)
- [Video on GUID and UNIQUEIDENTIFIER](#)
- [GUID vs INT Debate](#)

Other great articles from Ben

[Difference between Identity & Sequence in SQL Server](#)

[What is the difference between Clustered and Non-Clustered Indexes in SQL Server?](#)

[Understanding the GUID data type in SQL Server](#)

See more

To boost SQL coding productivity, check out these [free SQL tools](#) for SSMS and Visual Studio including T-SQL formatting, refactoring, auto-complete, text and data search, snippets and auto-replacements, SQL code and object comparison, multi-db script comparison,

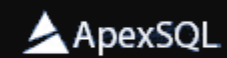


object decryption and more

An introduction to ApexSQL Complete



The best **tools** in life are **free!**





Ben Richardson

Ben Richardson runs Acuity Training a leading provider of SQL training the UK. It offers a full range of SQL training from introductory courses through to advanced administration and data warehouse training – [see here](#) for more details. Acuity has offices in London and Guildford, Surrey. He also blogs occasionally on Acuity's [blog](#)

[View all posts by Ben Richardson](#)

Related Posts:

1. [SQL Server Data Type Conversion Methods and performance comparison](#)
2. [SQL varchar data type deep dive](#)
3. [Understanding SQL Server's TRY_PARSE and TRY_CONVERT functions](#)
4. [Understanding Skewed Data in SQL Server](#)
5. [Sequence Objects in SQL Server](#)

Data types

53,948 Views

6 Comments

SQL Shack

 Login ▾

 Recommend 12

 Tweet

 Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



Puran Kandpal • 4 months ago

Excellent information  

1 Like · 0 Replies · 0 Shares



1 ^ | v • Reply • Share ›



Ben Richardson → Puran Kandpal • 3 months ago

Hi Puran, Glad to hear that you found this usefl.

^ | v • Reply • Share ›



Eduardo Torres • 10 months ago

You're so smart, thank u so much, I applied this information in my job, and worked perfect,

1 ^ | v • Reply • Share ›



Ben Richardson → Eduardo Torres • 10 months ago

I'm not sure about that but thank you for the kind comments on the article

1 ^ | v • Reply • Share ›



Jack Pine • 15 hours ago

Very clear and readable explanation of GUID values in SQL Server. Thanks very much for this article.

^ | v • Reply • Share ›



Bhavleen Singh • 2 months ago

very usefull

^ | v • Reply • Share ›

ALSO ON SQL SHACK

SQL Server in Azure Kubernetes Service (AKS)

2 comments • 2 months ago



Sudheer — Great article. Steps are in detail
Avatar

Change default database file and backup paths in SQL Server on Linux

2 comments • 6 months ago



Jozo Slejko — You have a little error, setting
Avatar

<https://www.sqlshack.com/understanding-the-guid-data-type-in-sql-server/>

SQL Server monitoring tools for memory performance

1 comment • 4 months ago



manu — Nicely explained. DMV named
Avatar `sys.dm_os_process_memory` (introduced in SQL 2008) definitely helps by providing physical/virtual memory notifications.

Power BI Desktop and Python; like Peanut Butter and Chocolate



1 comment • 5 months ago



Shrikant — Can we edit python script in power bi service after
Avatar publishing to manipulate the query?



[Available](#) filelocation.defaultlogdir parameter for both data and log. For data [Available](#) publishing to manipulate the graph. it should be filelocation.defaultdatadir parameter.

 [Subscribe](#)  [Add Disqus to your site](#) [Add Disqus](#) [Add Disqus](#)  [Disqus' Privacy Policy](#) [Privacy Policy](#) [Privacy Policy](#) [Privacy Policy](#)

