

Copyright © 2014 by Software Craftsmanship Guild.

All rights reserved. No part of these materials may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the Software Craftsmanship Guild. For permission requests, write to the Software Craftsmanship Guild, addressed “Attention: Permissions Coordinator,” at the address below.

Software Craftsmanship Guild

526 S. Main St, Suite 609

Akron, OH 44311



Intro to Data Definition Language (DDL)

Software Craftsmanship Guild

Lesson Goals

- Learn how to create database and tables
- Learn to configure primary keys
- Learn to define relationship constraints
- Learn about the basic data types available in SQL Server

Creating a New Database

- We have the option to create new databases either through the Management Studio interface or with SQL.
- The management studio interface is a wizard:
 - Right Click the Databases folder in the explorer, and choose 'new database'
 - Most wizard windows in SQL Server have a scripts button that allows your choices to be scripted. This is a great option for saving SQL syntax for later

New Database

Select a page

- General
- Options
- Filegroups

Script Help

Database name:

Owner:

☒ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize
	Rows ...	PRIMARY	3	By 1 MB, Unlimited
_log	Log	Not Applicable	1	By 10 percent, Unlimited

Connection

Server:
DATOR-ERIC\SQL2012

Connection:
Dator-Eric\Eric

[View connection properties](#)

Progress

Ready

Add Remove

OK Cancel

Once we name our database...

- We should set the recovery model. Which model you choose will determine how backups and restores work. We will only concern ourselves with two
 - Simple- There are no log backups and space in the log is automatically managed so there is no need to free up transaction log space. However any changes since the last backup will be lost.
 - Full – Requires log backups. You can recover up to a point in time and typically there will be little to no data lost in a recovery situation

New Database

Select a page

- General
- Options
- Filegroups

Script Help

Collation: <default>

Recovery model: Full

Compatibility level: SQL Server 2012 (110)

Containment type: None

Other options:

Auto Shrink: False

Auto Update Statistics: True

Auto Update Statistics Asynchronously: False

Containment

Default Fulltext Language LCID: 1033

Default Language: English

Nested Triggers Enabled: True

Transform Noise Words: False

Two Digit Year Cutoff: 2049

Cursor

Close Cursor on Commit Enabled: False

Default Cursor: GLOBAL

FILESTREAM

FILESTREAM Directory Name:

FILESTREAM Non-Transacted Access: Off

Miscellaneous

Allow Snapshot Isolation: False

ANSI NULL Default: False

Allow Snapshot Isolation

Connection

Server: DATOR-ERIC\SQL2012

Connection: Dator-Eric\Eric

[View connection properties](#)

Progress

Ready

OK Cancel

File Groups

- These let you separate data across multiple disks for performance reasons. We won't be covering this in the beginner's course.
- Consult an experienced DBA before messing with this.

Wait a tick, what the heck is a log?

Why does SQL think it's so important?

- Every SQL Server database has a transaction log that records all transactions and the database modifications made by each transaction.
- This means every time you make a change to data there are 3 writes:
 - The action to occur
 - The real data modification
 - The commit that the real data has been updated
- In this way if you submit a modification to the data and someone kicks the cord out of the server in the middle, the database can see uncommitted transactions in the log and roll them back, preserving the clean state of your data!

So what logging strategy should I choose?

- Typically full for production with frequent (depends on loss tolerance of the business) log backups
- In development, we usually use simple because a nightly backup is sufficient.

Common Log SNAFUs by Developers

1. Not backing up the log

- a) Since SQL Server needs it to recover, it won't overwrite the log unless it gets backed up. So if you forget your log will grow forever until you run out of disk space someday.

2. Putting the log on the same drive as the data

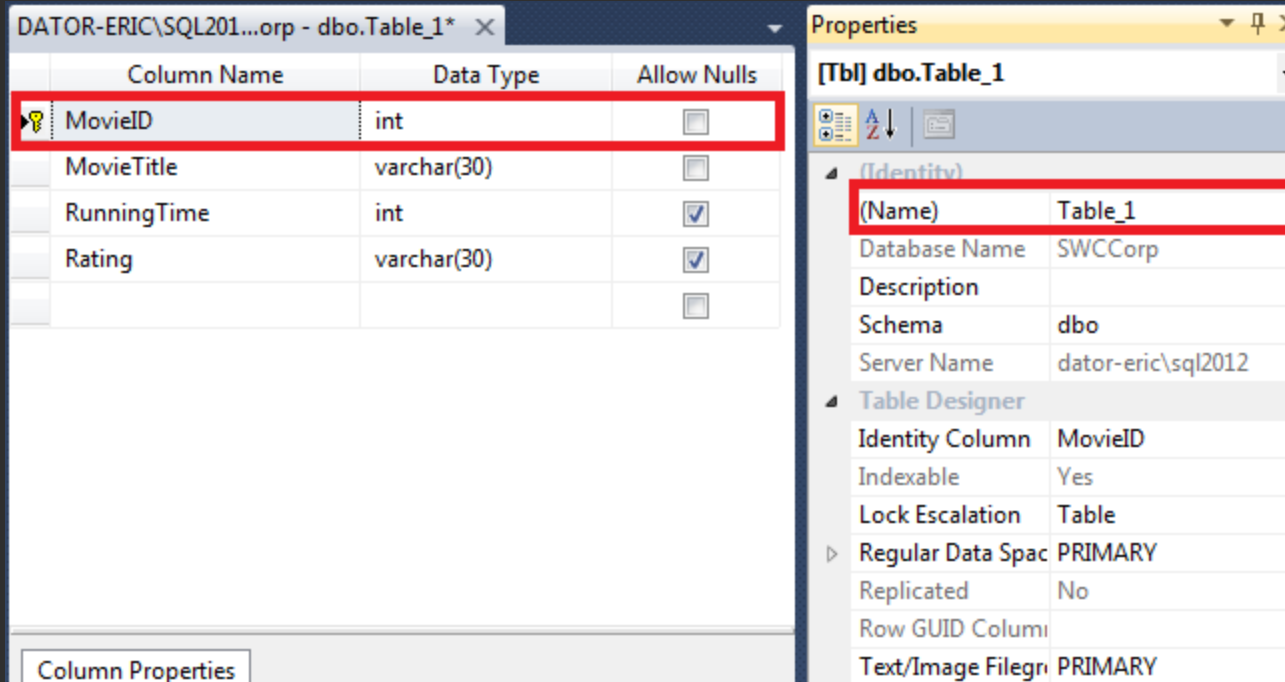
- a) Can you say write contention? I knew you could.

Creating Tables

- To Create a table we can either right click the Tables folder in our database or write a CREATE TABLE script.
- Let's look at the wizard first

Create Table in SSMS

- We can enter column names and Data Types for the columns, as well as tell SQL Server whether they are required (Allow Nulls Unchecked).
- The properties window allows us to set a table name and we can right click on a column to set it as a primary key



Column Properties

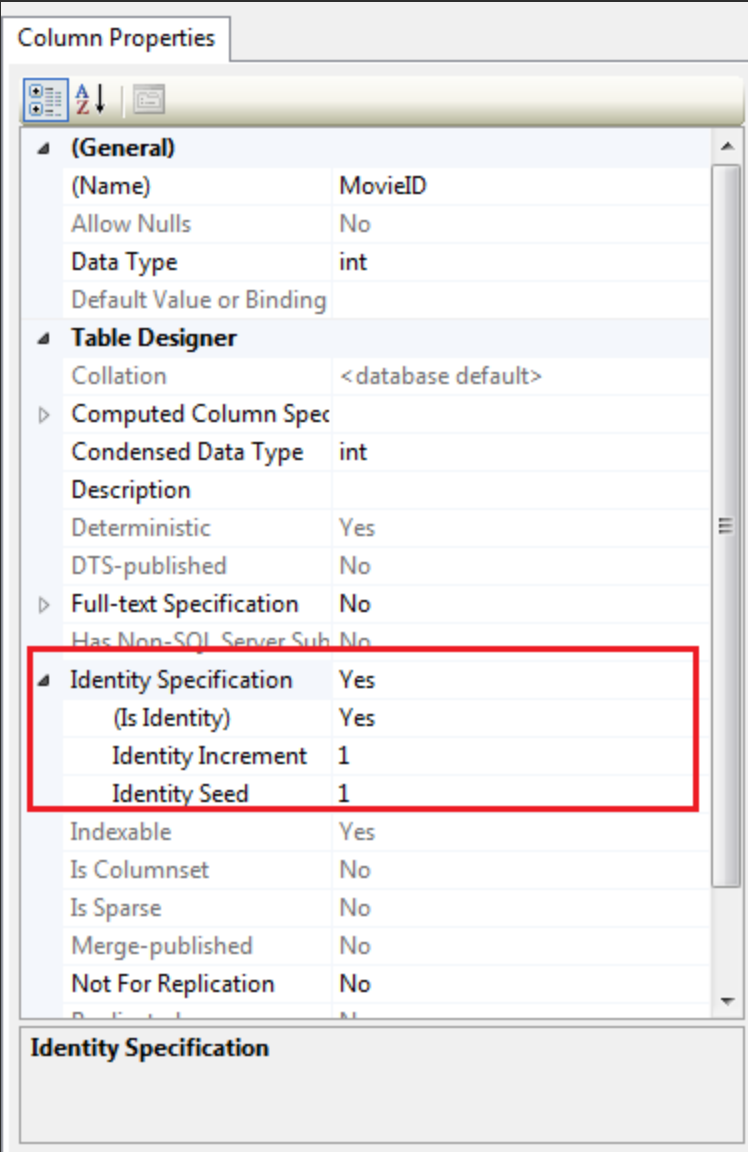
The column properties has a variety of advanced functions, but I want to draw your attention to the Identity Specification property.

If set to yes, this instructs SQL Server to manage the column and automatically increment the field.

The Identity Increment is how many numbers between values (right now it will count up by 1)

The Identity Seed is the starting number (sometimes companies like to do things like start invoices at 1000)

We most often set identity columns as primary keys



Column Properties	
(General)	
(Name)	MovieID
Allow Nulls	No
Data Type	int
Default Value or Binding	
Table Designer	
Collation	<database default>
Computed Column Specification	
Condensed Data Type	int
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subtype	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes
Is Columnset	No
Is Sparse	No
Merge-published	No
Not For Replication	No

Identity Specification

Why AutoNumbers For Keys?

There are several reason why this is typical

1. Privacy- Keys often have to be referenced in code, so we don't like to use personal data (like social security number) because it isn't safe to put into website urls, etc.
2. Fragmentation- The primary key by default exists as a *Clustered Index*
 - a) A clustered index is the physical order of the data on the disk, so by using an auto number key we can ensure new data is always inserted at the end.
 - b) If the primary key/clustered index is a text field it can force the database to insert things in order, and sometimes have to move data around to make space. So usually it is best for lower fragmentation and higher performance.

By the way

- We can have multiple columns make up a primary key. The only requirement is that the column (or combination of columns) has to be unique. Creating a duplicate record raises an error.
- When multiple columns make up a primary key we call them *combination keys*. They are common in many-to-many relationships.

Here's the SQL Syntax

- [dbo] is the *owner* of the table.
- All tables must have an owner. DBO is a special schema called “database owner”.
- Most times this is the default and what you will find in the wild, sometimes in high security scenarios different roles are assigned as owners

```
CREATE TABLE [dbo].[Movies](  
    [MovieID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,  
    [MovieTitle] [varchar](30) NOT NULL,  
    [RunningTime] [int] NULL,  
    [Rating] [varchar](30) NULL  
)
```

SQL Data Types (Text)

Data Type	Description	Size
Char(n)	Fixed-length character string. Max 8,000 characters	n bytes
Varchar(n)	Variable-length character string, Max 8,000 characters	1 byte per character + 2 bytes overhead
Varchar(max)	Variable-length max 1,073,741,824 characters	1 byte per character + 2 bytes overhead
Text	Variable-length max 2GB of text data	1 byte per character + 2 bytes overhead
XML	XML formatted data up to 2GB in size	

* There is also nchar, nvarchar, and ntext which store Unicode characters and binary, varbinary, and image which store Binary data

SQL Data Type (Numeric)

Data Type	Description	Size
Bit	True/False stored as 0/1	
Tinyint	0 to 255	1 Bytes
SmallInt	-32,768 to 32,767	2 Bytes
Int	-2,147,483,648 to 2,147,483,647	4 Bytes
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 Bytes
Decimal(p,s)	Fixed precision and scale numbers -10 ³⁸ to 10 ³⁸ -1 P is the max number of digits (left and right) S is the number of digits to the right of the decimal point	5-17 bytes

SQL Data Type (Numeric)

Data Type	Description	Size
Smallmoney Money	In legacy versions these were for currency, these are deprecated and should be moved to decimal	
Float(n)	Floating point precision -1.79E + 308 to 1.79E + 308 n determines 4 or 8 bytes. 24 for 4, 53 for 8. Default is 24	4 or 8 Bytes
Real	Floating point precision -3.40E + 38 to 3.40E + 38	4 Bytes
Uniqueidentifier	Not really numeric, it's a GUID (globally unique identifier). Useful for multiple database syncing scenarios	16 Bytes

SQL Data Type (Date And Time)

Data Type	Description	Size
datetime	Jan 1, 1753 to Dec 31, 9999 Accuracy of time to 3.33 milliseconds	8 Bytes
Datetime2	Jan 1, 0001 to Dec 31, 9999 Accuracy of time to 100 nanoseconds	6-8 Bytes
Smalldatetime	Jan 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 Bytes
Date	Jan 1, 0001 to Dec 31, 9999	3 bytes
Time	Time only, 100 nanoseconds precision	3-5 bytes
Datetimeoffset	Same as datetime2 but with time zone offset info	8-10 bytes
Timestamp	Unique number that gets updated every time a row is created or modified. Used for concurrency/auditing	8 Bytes

Combo Keys And Foreign Keys

As mentioned earlier, we can have combination primary keys, typically in many-to-many relationships

Foreign Keys put a constraint on the table that a key field must exist in the table that it references

In this example our movies can have many actors, and actors can belong to many movies. This is a classic many-to-many relationship.

We handle this by creating a join table to handle the cross references. Notice that MovieActors is a combination key, that is an actor can only be in a movie one time, but can be in many movies

```
CREATE TABLE [dbo].[Actors](
    [ActorID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
    [FirstName] [varchar](30) NOT NULL,
    [LastName] [varchar](30) NOT NULL
)

CREATE TABLE [dbo].[MovieActors](
    [MovieID] [int] NOT NULL,
    [ActorID] [int] NOT NULL,
    Constraint PK_MovieActors PRIMARY KEY(MovieID, ActorID)
)

ALTER TABLE [dbo].[MovieActors]
    ADD CONSTRAINT FK_MovieActors_Movies FOREIGN KEY (MovieID)
    REFERENCES [dbo].[Movies] (MovieID)

ALTER TABLE [dbo].[MovieActors]
    ADD CONSTRAINT FK_MovieActors_Actors FOREIGN KEY (ActorID)
    REFERENCES [dbo].[Actors] (ActorID)
```

Now let's do it using the database designer

DEMO

Fin

- Next up- Data Modification Queries!