PEAK IT 003

SQL Server Database Project

How to develop and test SQL Server databases

using Visual Studio database project

Török Mihály

AgileHub

Instructor's bio...

PEAK IT 003

Török Mihály

e-mail: mihaly.torok@gmail.com

LinkedIn: https://www.linkedin.com/in/mihaly-torok-b3433366

AgileHub: https://agilehub.ro/

Slack PeakIT: https://peakit003.slack.com/ - #curs-17oct-mihaly-torok

Expertise: Database development and query tuning

AgileHub

Diamond Sponsors:

Trimble

Atos

SIEMENS
Ingenuity for life

nagarro

Partners:

coresi

Universitatea Transilvania din Brașov

Valorile AgileHub

INDEPENDENȚA

- Trainerii și participanții vin pe cont propriu, fără a promova vreo firmă
- Nimeni nu reprezentă interesele nici unei firme
- La începutul cursului, când ne prezentăm, spunem care e rolul și experiența noastră, fără a specifica firma la care lucrăm

EDUCAȚIA GRATUITĂ

- Cursurile sunt gratuite pentru toți participanții

VOLUNTARIATUL

- Toți trainerii sunt voluntari

ÎMPĂRTĂȘIREA EXPERIENȚEI PROPRII

- Majoritatea formatorilor **NU** sunt traineri profesioniști
- Formatorii lucrează în IT și au multă experiență practică în domeniul pe care îl predau

Participant Introduction

PEAK IT 003

- Name
- Course relevant experience
- Did you use the SQL Server Database project?

AgileHub

What to expect

PEAK IT 003

- PowerPoint slides
- Participation in class labs
- Instructor demonstrations
- Copying of lab files for independent study is permissible

The examples are based on AdventureWorks sample databases

The code is shared on **GitHub:** https://github.com/MihalyTorok/PeakIT003.git

AgileHub

Agenda

PEAK IT 003

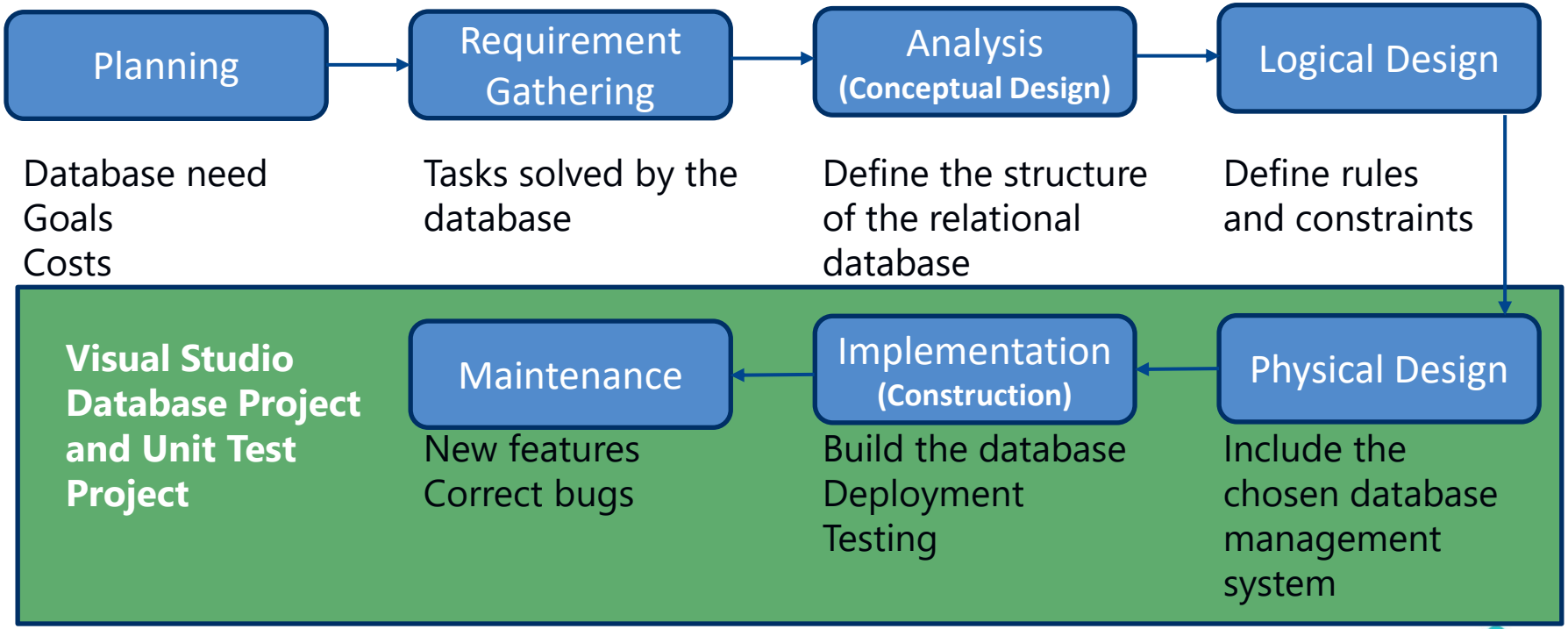
1. SQL Server Database Project
2. Creating the database project
3. Implementing database objects
4. Building the project
5. Publishing the project
6. Inserting initial data into the database
7. Incremental deployment
8. Unit testing

AgileHub

1. SQL Server Database Project

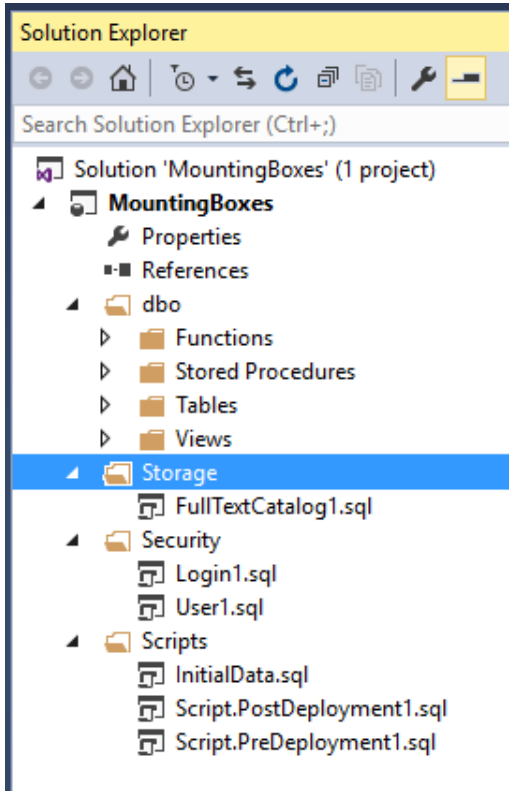
PEAK IT 003

Database Development Lifecycle



AgileHub

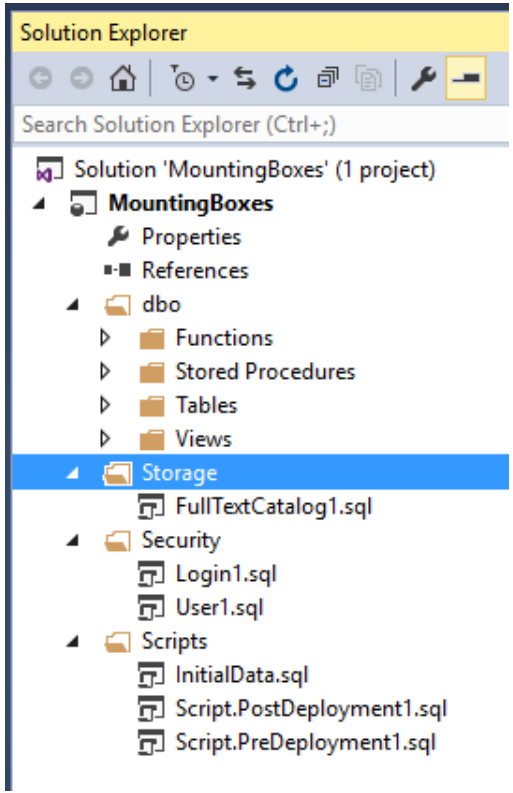
SQL Server Database Project



Recommended project's structure

- The database project doesn't have a mandatory structure.
- It is recommended to define folders for each different database schema. The **dbo** folder contains all database objects from the default database schema.
- The database objects are saved in folders based on their type:
  - **Functions**
  - **Stored Procedures**
  - **Tables**
  - **Views**

SQL Server Database Project



The **Scripts** folder:

- Fill scripts - insert the initial data
- *PostDeploy* scripts – manage the database update
- Data movement scripts used for new version deployment.
- *PreDeploy* scripts – executed before each database update.

Other folders as needed by the application:

- **Security** information
- **Storage** information as full text catalogs.

2. Creating the database project

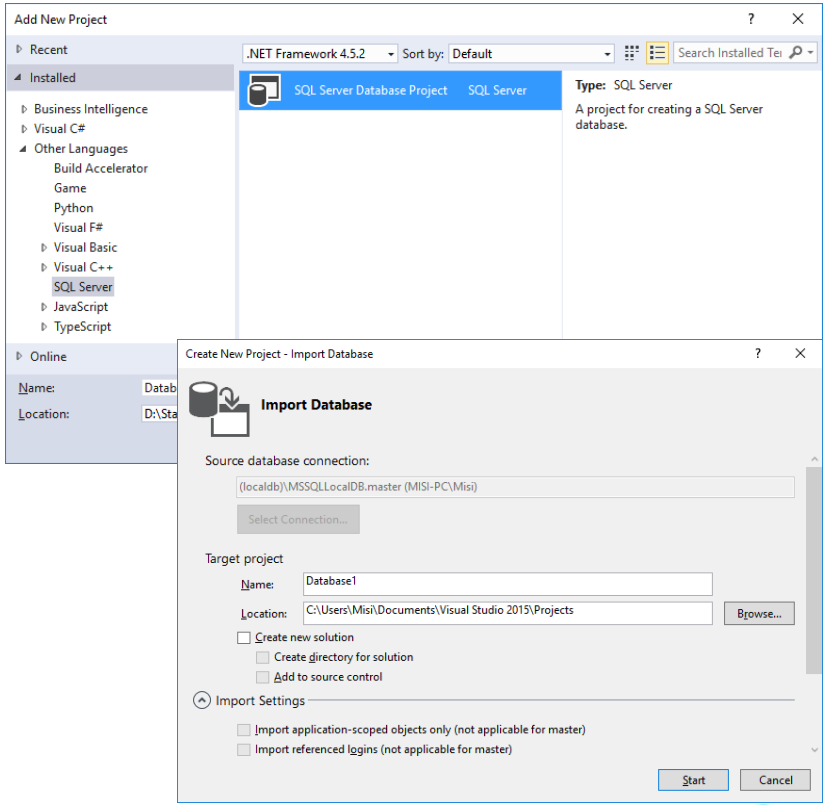
PEAK IT 003

Create a new database project from scratch:

- **Add / New Project** option from the solution's context menu.
- Select the **SQL Server Database Project** template from the *Other Languages / SQL Server section*.

Reverse engineer an existing database:

- From the **SQL Server Object Explorer** select the database
- Select the **Create New Project** option.



</> AgileHub

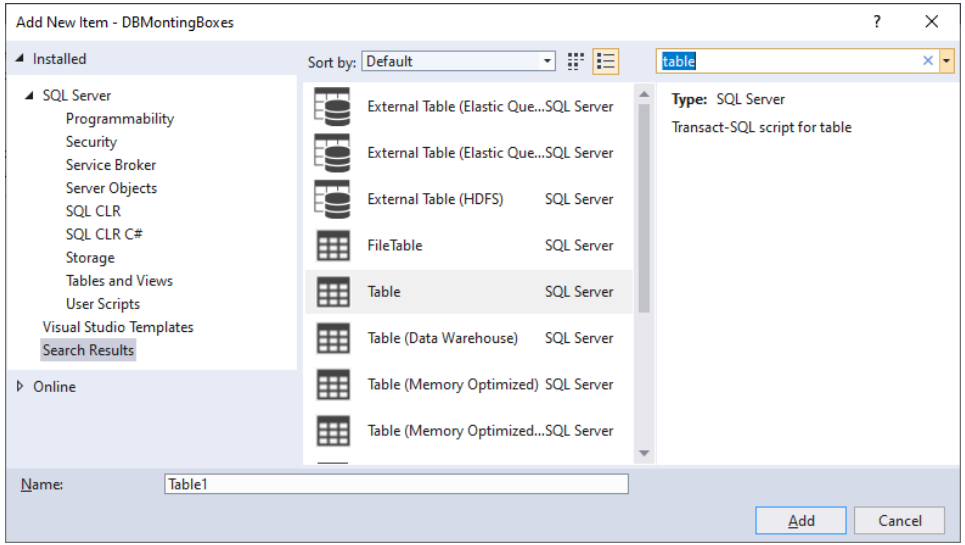
3. Implementing database objects

PEAK IT 003

The SQL Server database project provides templates for each database object.

To add a database object follow the steps:

- Select the project folder where the new object should be added.
- Select **Add -> New Item** context menu item
- Select the needed template from the *SQL Server section*.



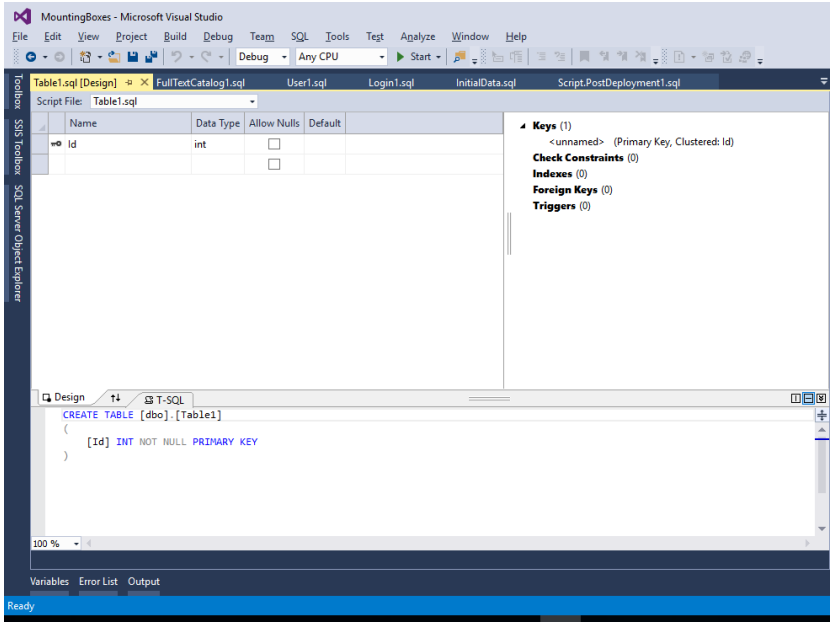
AgileHub

Implementing database objects

PEAK IT 003

Table:

- Select the **Table** template.
- Using the designer from the left side implement:
  - Columns
  - Column Types and lengths
  - NULL constraints of columns
  - Default values



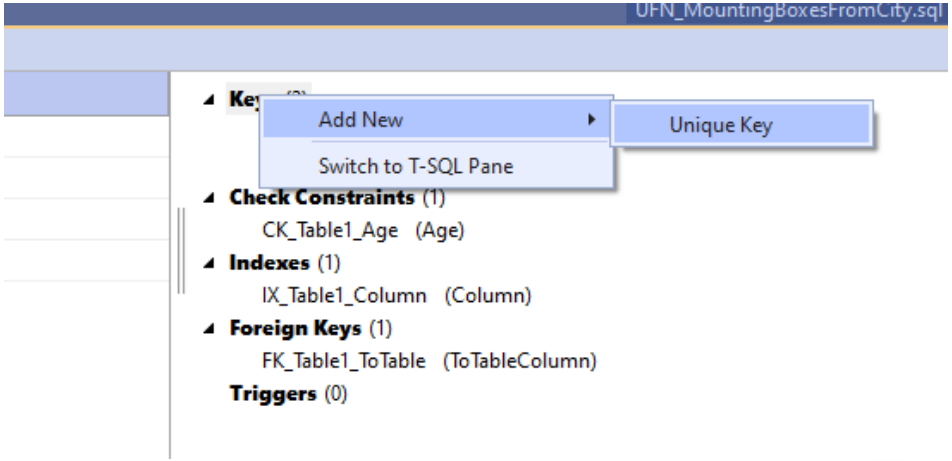
AgileHub

Implementing database objects

PEAK IT 003

Table:

- Add **keys**, **foreign keys**, **check constraints**, **indexes**, using the context menu on the corresponding table object section from the right side.
- In the bottom side of the screen you can edit manually the SQL statement snippet added by the designer



</> AgileHub

Implementing database objects

PEAK IT 003

Coding Style

- Two-part names needed when schema is not the default **dbo**
- Use indentation
- Do not use aliases for tables if possible
- Do not use special characters or whitespaces in names or aliases

```
CREATE PROCEDURE ListStudentByCity(@city NVARCHAR(100))
AS
BEGIN
      SET NOCOUNT ON;
      SELECT Score.StudentId
         ,Student.Name
         ,MAX(Score.Score) AS MaxScore
      FROM dbo.Score
         INNER JOIN dbo.Student
            ON Score.StudentId = Student.StudentId
      WHERE Student.City = @city
      GROUP BY Score.StudentId, Student.Name
      HAVING MAX(Score.Score) >= 7
      ORDER BY MaxScore, Student.Name;
END
```

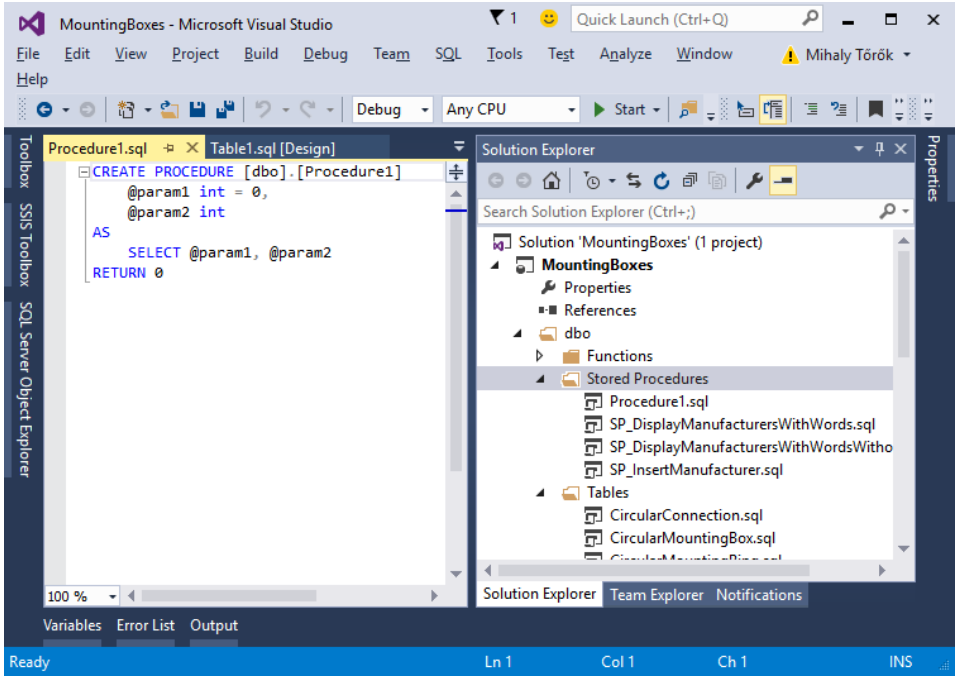

AgileHub

Implementing database objects

PEAK IT 003

Stored Procedure:

- Select the **Stored Procedure** template.
- Edit the stored procedure's content using the inserted code snippet.



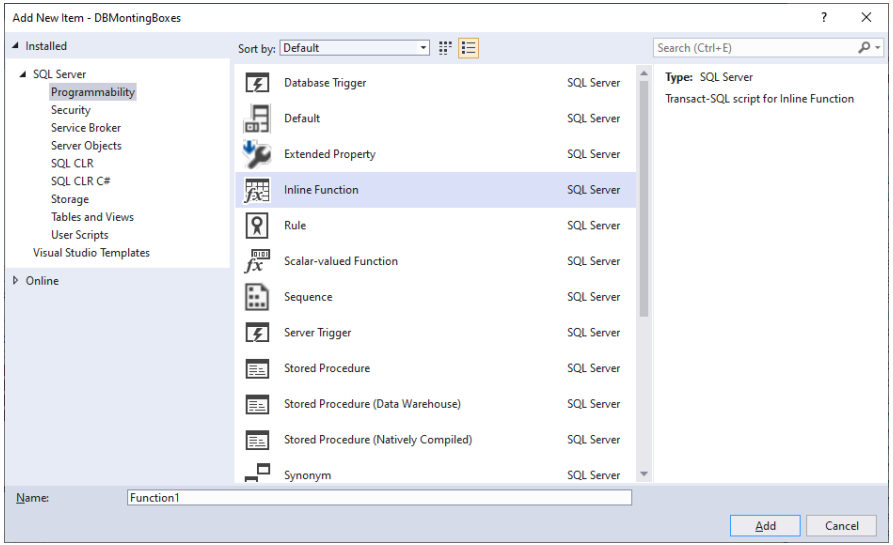
AgileHub

Implementing database objects

PEAK IT 003

Other database objects

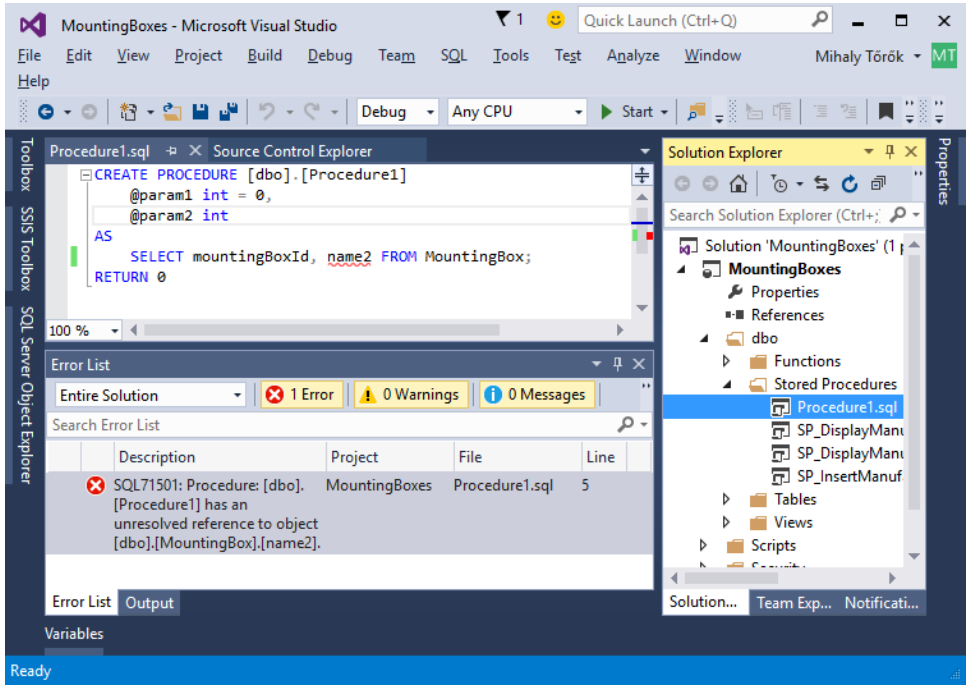
- Functions
- Triggers
- Views
- Logins and Users



AgileHub

4. Building the project

- Built as any other Visual Studio project.
- The database objects are checked for syntactic and object reference errors.
- The errors are displayed in the **Error List** panel and the **Output** panel.
- The Output panel has hints about additional errors.



Building the project

PEAK IT 003

Build configurations

- Build configurations are supported
- Selectable from the Visual Studio ribbon

Resulted files after build

- <project_name>`.dacpac` file located in the bin folder
- Used for database deployment.

Project validation

- Publishing to a database server does the final check of the project
- Remark: files that are set with build *None* are not checked during build step.

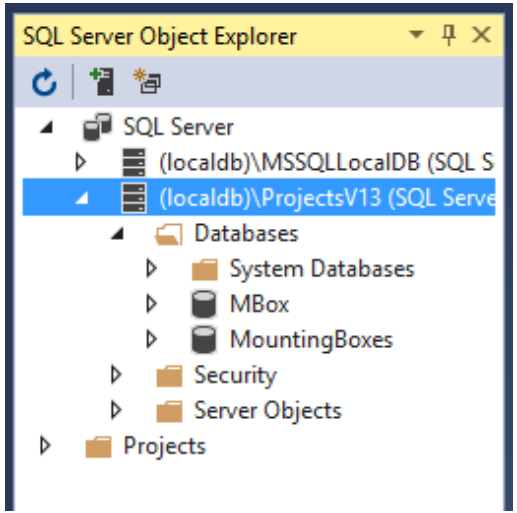
</> AgileHub

5. Publishing the project

PEAK IT 003

Run the project

- The **Start** button deploys the database to the **`localdb`** SQL Server instance.
- The **`localdb`** SQL Server instance can be explored in the **SQL Server Object Explorer** from Visual Studio.



Limitations:

- Some features, like full text search, are not supported by **`localdb`**.

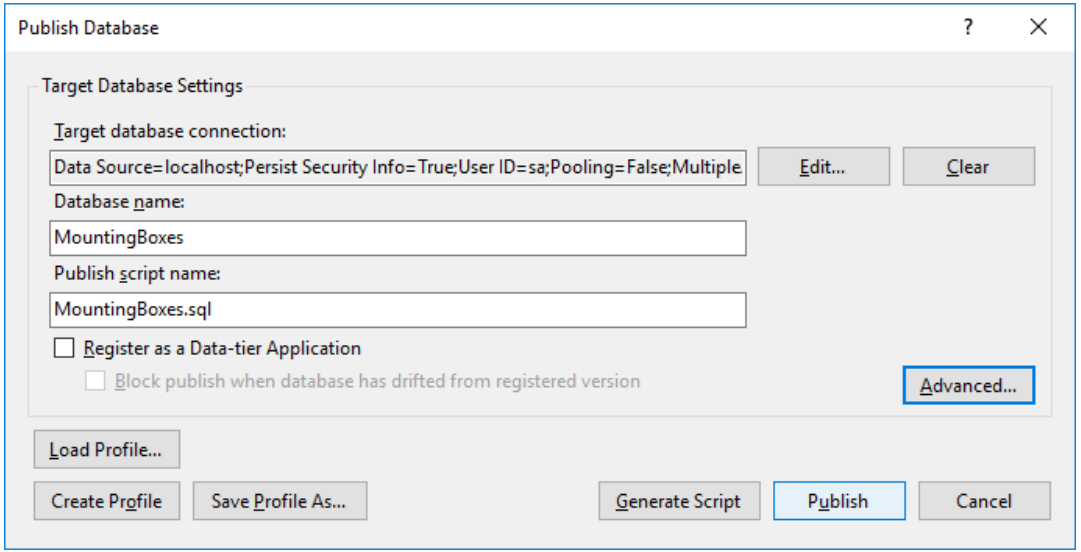
AgileHub

Publishing the project

PEAK IT 003

Publish to database server

- Click the **Publish** option from the project's context menu.
- In the **Publish Database** window edit the database connection
- **Publish**



Result:

- The target database server is upgraded to the new schema.
- The data contained in the database are migrated.
- Possible data loss fails the operation.

AgileHub

Publishing the project

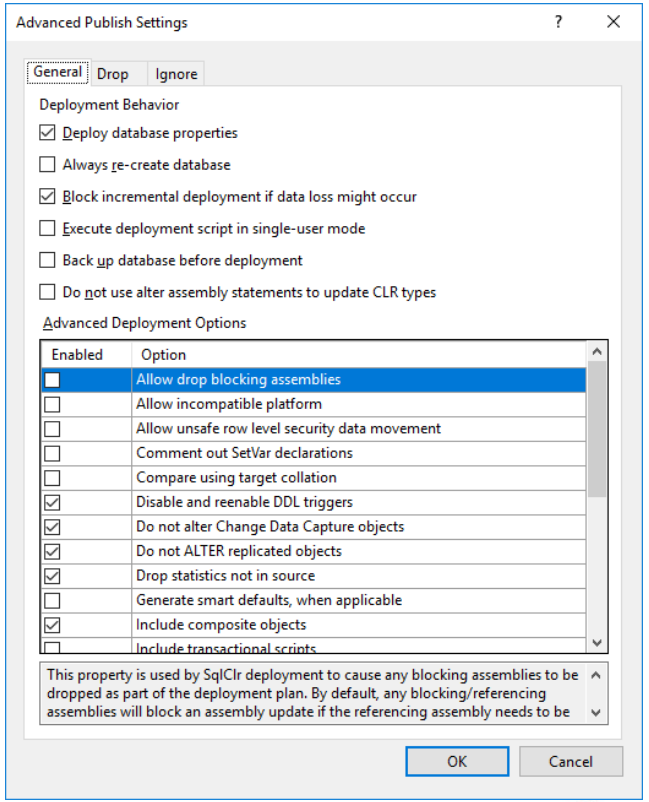
PEAK IT 003

Publish profile

- Describes the rules how the database project is published
- Different publish profiles for each maintained database.

Defining the profile

- Click the **Create Profile** button from the **Publish Database** window
- Edit the profile using the **Advanced** button.



</>

AgileHub

6. Inserting initial data into the database

PEAK IT 003

- Initial data is inserted after the database schema deployment.
- Implemented by the **`PostDeployment`** script.

Rules:

- Only one *PostDeployment* script in the database project.
- The **`SQLCMD`** syntax can be used to include multiple files.

```
:r .\myfile.sql
```

- Build action: **`PostDeploy`** for the post deployment script

  `None` for the included scripts
- The scripts should be idempotent.

AgileHub

Inserting initial data into the database

PEAK IT 003

Initial data insertion example:

PostDeployment.sql file:

```
:r .\DefineLocalizations.sql
```

DefineLocalizations.sql file:

```
MERGE INTO Localizations AS tgt
USING (VALUES('IE'), ('NL'), ('RO')) AS src(name)
ON tgt.name = src.name
WHEN NOT MATCHED THEN INSERT(name) VALUES(src.name)
WHEN NOT MATCHED BY SOURCE THEN DELETE;
```


AgileHub

7. Incremental deployment

PEAK IT 003

Requirements of a database upgrade

- Needed for the deployment of a new version of the application
- Implemented using the database project
- The project is published the target database.
- The user's existing data should be preserved unaltered.

How to solve this?

Use the **PostDeployment** and the **PreDeployment** scripts.

AgileHub

Incremental deployment

PostDeployment script

- Inserts new initial data related to the current application version
- Handles the data movement in case of schema changes.
- Relays on data saved by the PreDeployment script if needed.

PreDeployment script

- Is executed before the database schema upgrade step.
- Saves user data in order to load them back after the schema upgrade.
- It is used together with the PostDeployment script.

Incremental deployment

Rules related to the **PreDeployment** script:

- A single PreDeployment script may exists in the database project.
- The **SQLCMD** syntax can be used to include multiple files

```
:r .\myfile.sql
```

- Build action: **PreDeploy** for the pre deployment script

  None for the included scripts

- The existence of the object that is referred to should be checked. Remember, the database may not exists when this script is run.

8. Unit testing

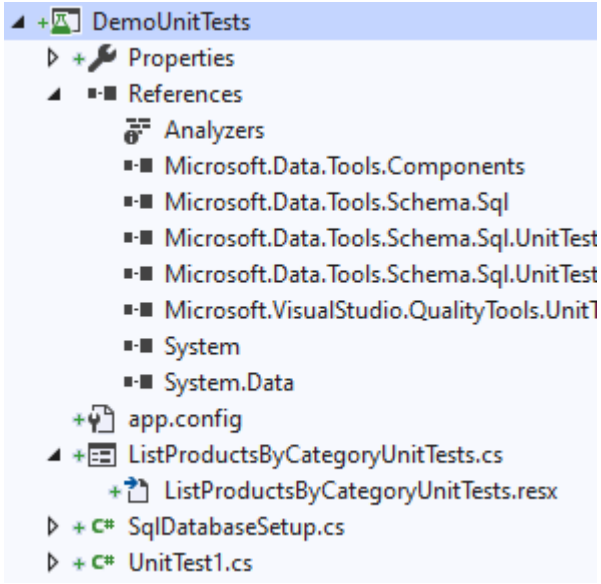
PEAK IT 003

Create the Unit Test Project

Each database from the solution should have its Unit Test project.

- When you add the first unit test the project is also created
  - Name the project
- Configure connection strings using

  SQL Server Test Configuration…

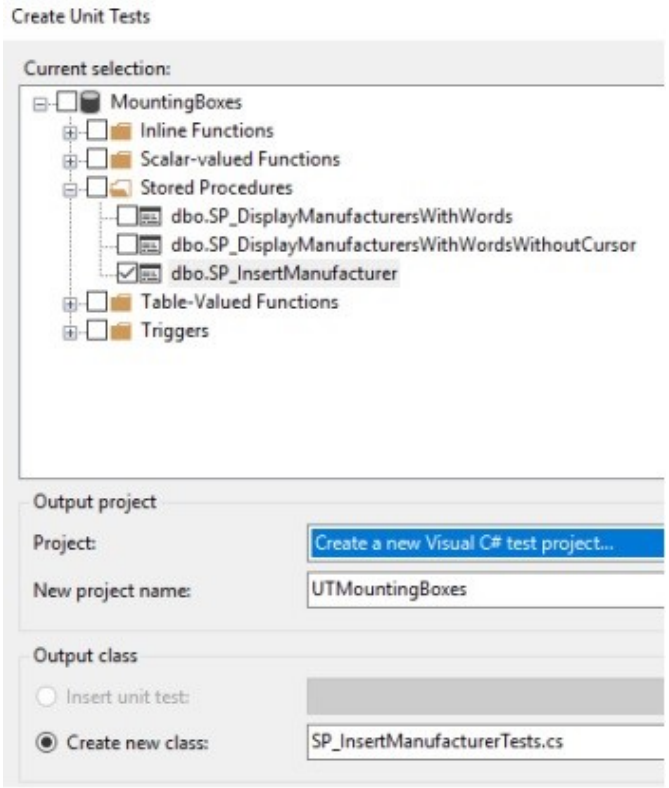


AgileHub

Unit testing

Add the unit test class

- Select a stored procedure from the project listed in **SQL Server Object Explorer**
- Use the **Create Unit Tests...** option from the context menu.
- Name the test class.
- The test is created and the selected stored procedure's **EXECUTE** statement is generated.



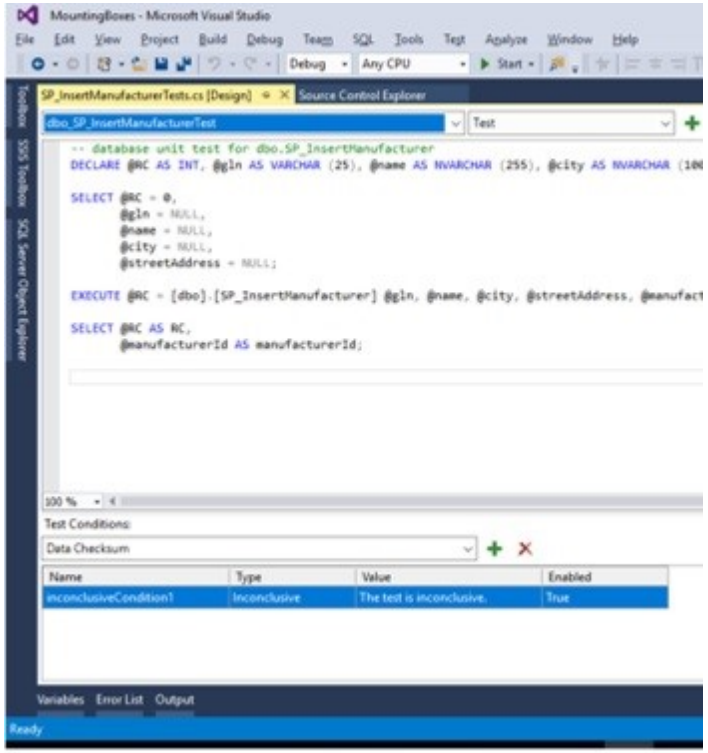
Unit testing

Implement the test

A unit test definition contains three parts:

- **Pre-test** implements T-SQL statements necessary to prepare the database into the state needed for the test
- **Test** part which implements the test's logic
- **Post-test** which cleans the changes made by the test.

In the test part any returned data set can be used for the test's validation.



Unit testing

Validate the test

- The **Test Conditions** panel is used to define the test conditions.
- The data returned by any statement can be validated.

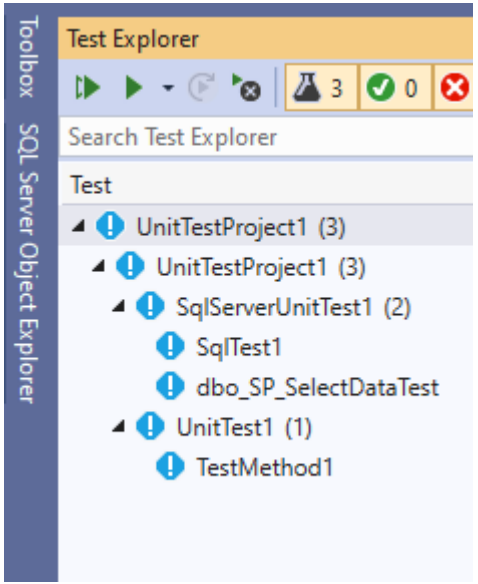
Test conditions:

- Data Checksum
- Empty ResultSet
- Execution Time
- Expected Schema
- Inconclusive
- Not Empty ResultSet
- Row Count
- Scalar Value

Unit testing

Run the test

- Deploy the database project to a server
- Configure the connection using the **SQL Server Test Configuration…** option
- The database tests are run using **MSTest** as any other unit test.



SQL Server Database Project – More info

PEAK IT 003

Working with Database Projects – MSDN

How to: Create a New Database Project - MSDN

SqlPackage.exe - MSDN

Download SQL Server Data Tools (SSDT) – Microsoft Docs

SQL Server Data Tools – MSDN

Walkthrough: Creating and Running a SQL Server Unit Test – MSDN

Introducing Unit Testing for SQL Server Database Projects - Jill McClenahan

AgileHub

Conclusions

PEAK IT 003

1. SQL Server Database Project
2. Creating the database project
3. Implementing database objects
4. Building the project
5. Publishing the project
6. Inserting initial data into the database
7. Incremental deployment
8. Unit testing

AgileHub

Questions

PEAK IT 003

1. SQL Server Database Project
2. Creating the database project
3. Implementing database objects
4. Building the project
5. Publishing the project
6. Inserting initial data into the database
7. Incremental deployment
8. Unit testing

Slack PeakIT: https://peakit003.slack.com/ - #curs-17oct-mihaly-torok
GitHub: https://github.com/MihalyTorok/PeakIT003.git

AgileHub

Feedback

PEAK IT 003

http://bit.ly/peakit003-feedback

Completați acum

Durează 2-3 minute

Feedback anonim - pentru formator si AgileHub

AgileHub