https://www.halvorsen.blog

Database Systems Overview

Hans-Petter Halvorsen

Table of Contents

- Introduction
- Database Modelling and Design
- SQL Server
- Datalogging using LabVIEW
- Data Monitoring Visual Studio/C#
  - WinForm Desktop Application
  - ASP.NET Core Web Application

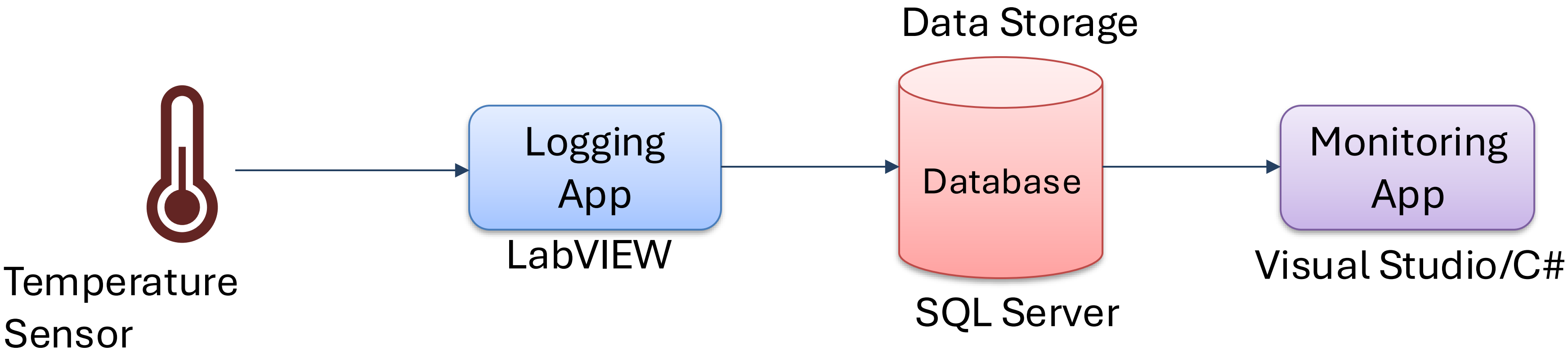
Introduction

Database Systems

- A database is a structured way to store lots of information.
- The information is stored in different tables.
- “Everything” today is stored in a database today, like bank systems, information in web pages and data used by AI, etc.
- In the industry we have, e.g., **Datalogging and Monitoring Systems** and **SCADA Systems**.
- Some popular database systems today are Oracle, MySQL, MariaDB and **Microsoft SQL Server**.
- Typically you start by designing your database and create a so-called Entity Relationship Diagram (ERD).
- There exist many software tools for creating ER diagrams like **DB Designer**, Lucidchart and erwin Data Modeler.

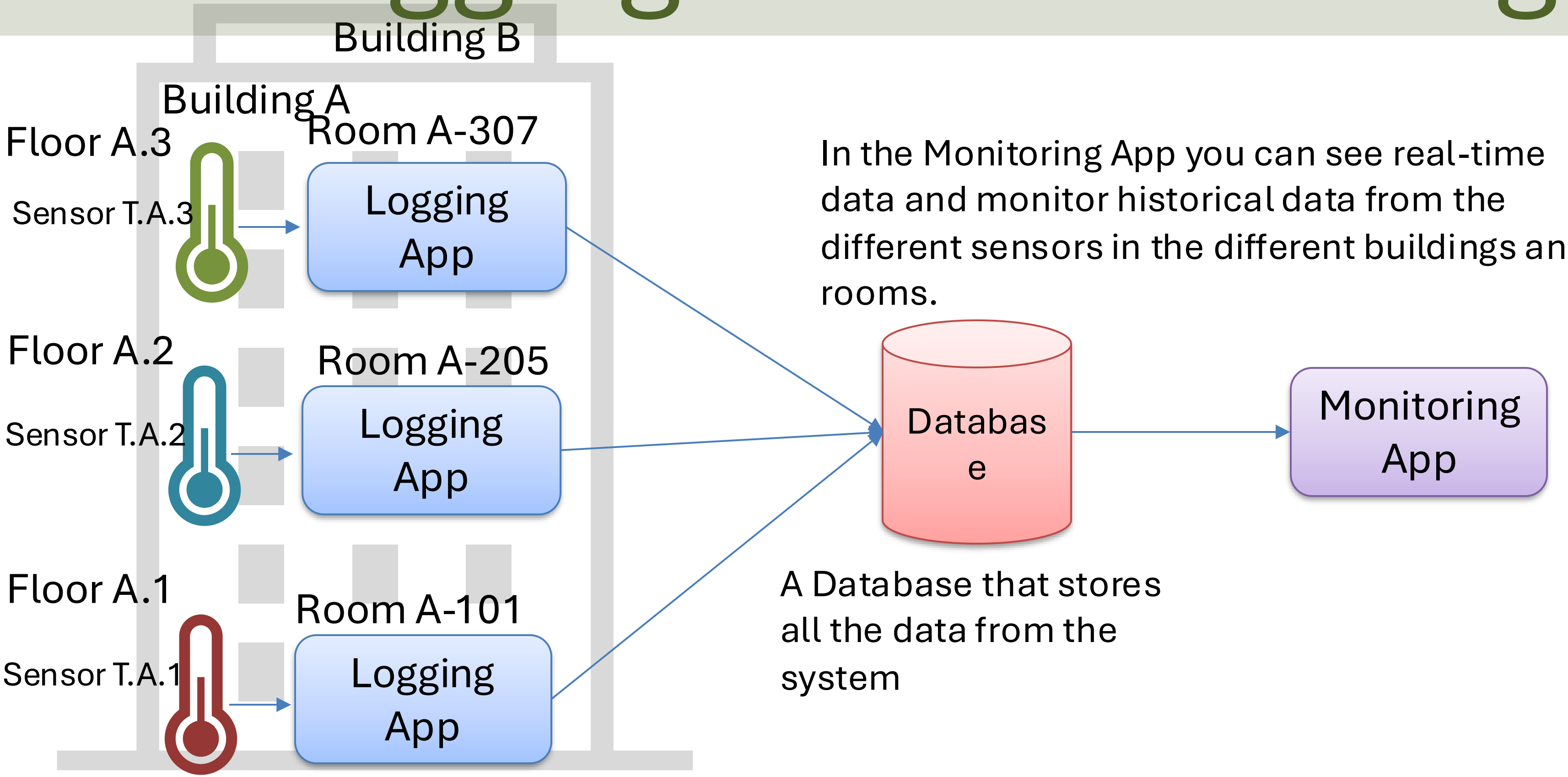
Datalogging and Monitoring

Here you see the core concept of a Datalogging and Monitoring System:



On the next pages you will see different User Case Scenarios for such a System, e.g., Home Automation System, Building Monitoring System, etc.

Datalogging and Monitoring



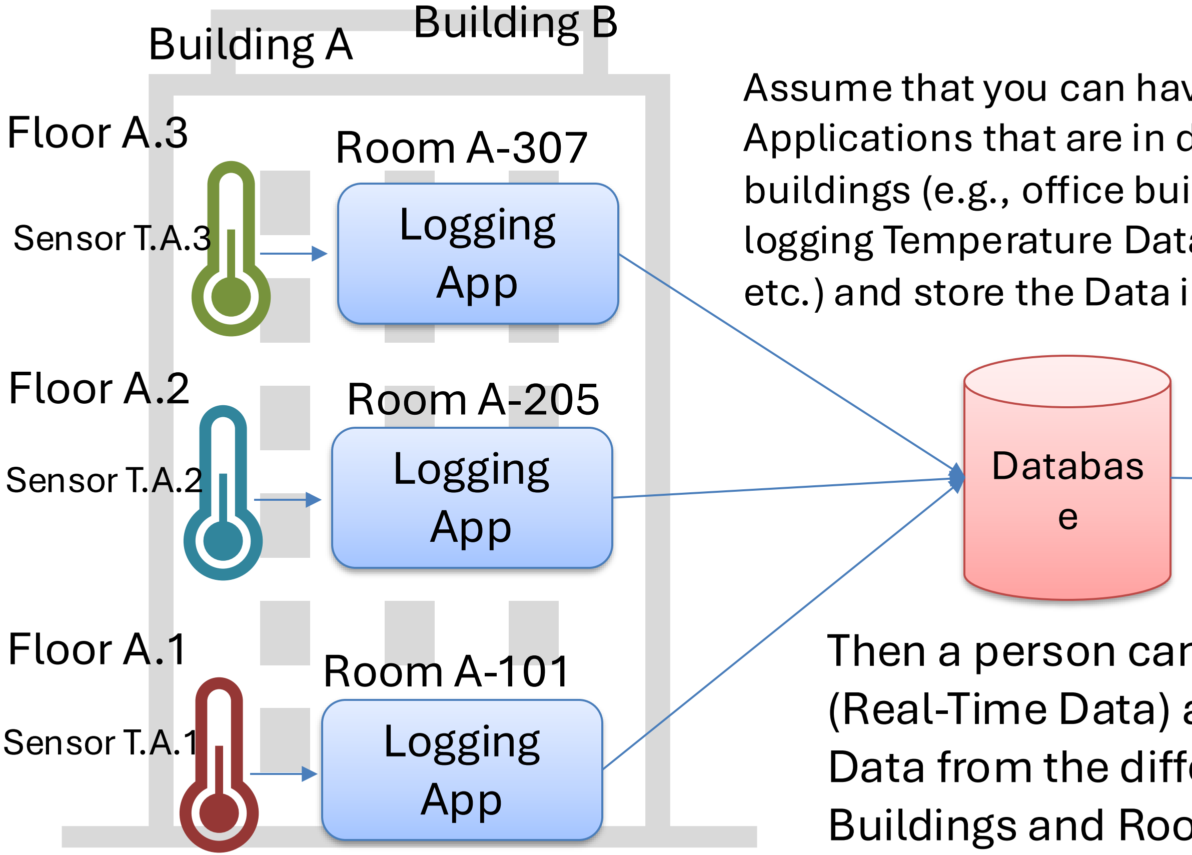
In the Monitoring App you can see real-time data and monitor historical data from the different sensors in the different buildings and rooms.

A Database that stores all the data from the system

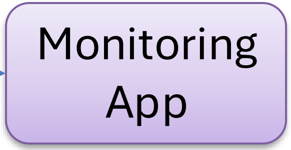
The Logging App is in the different rooms where you want to see and log data from different sensors.

Use Case Scenario (Alt1)

“Building Monitoring System”



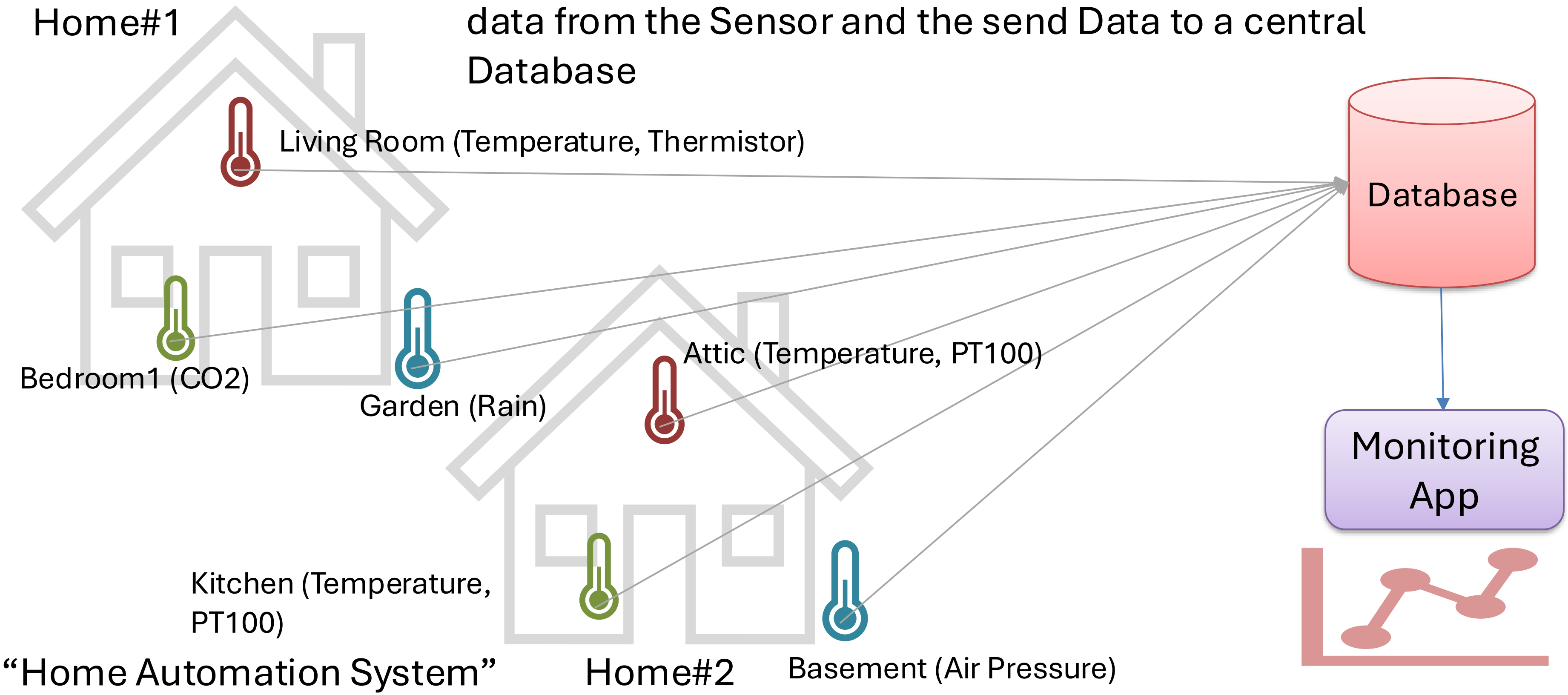
Assume that you can have multiple Data Logging Applications that are in different places inside multiple buildings (e.g., office buildings, factory, etc.) which are logging Temperature Data (or Data from other Sensors, CO2, etc.) and store the Data inside a common Database.



Then a person can sit somewhere and Observe (Real-Time Data) and Monitor (Historical Data) the Data from the different Sensors in the different Buildings and Rooms using the Monitoring App.

Use Case Scenario (Alt2)

Assume each Sensor has a separate Logging App that Log data from the Sensor and the send Data to a central Database



Home#1

Living Room (Temperature, Thermistor)

Bedroom1 (CO2)

Garden (Rain)

Attic (Temperature, PT100)

Kitchen (Temperature, PT100)

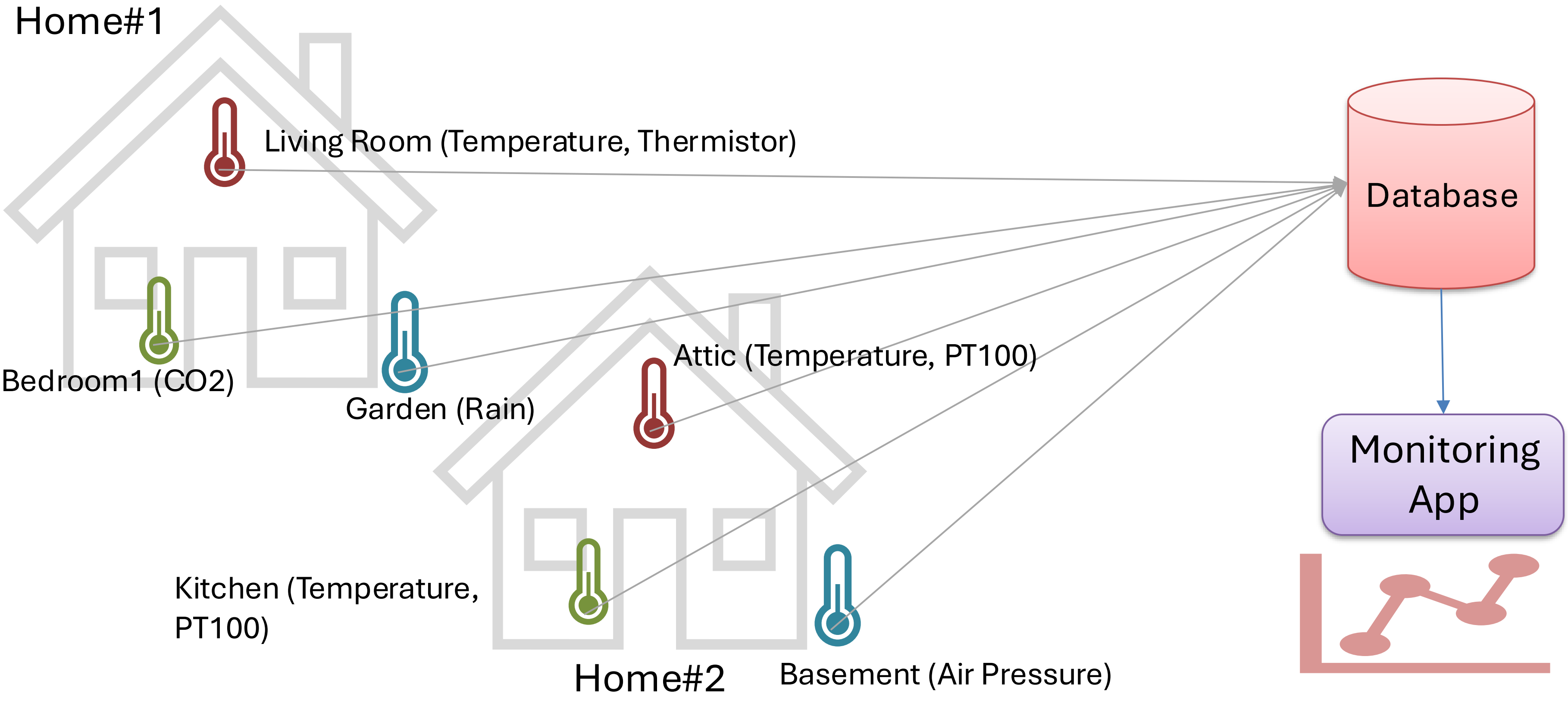
Basement (Air Pressure)

Home#2

Database

Monitoring App

“Home Automation System”


Home#1
Living Room (Temperature, Thermistor)
Bedroom1 (CO2)
Garden (Rain)
Attic (Temperature, PT100)
Kitchen (Temperature, PT100)
Home#2
Basement (Air Pressure)
Database
Monitoring App

Database Design and Imlementation

DB DESIGNER

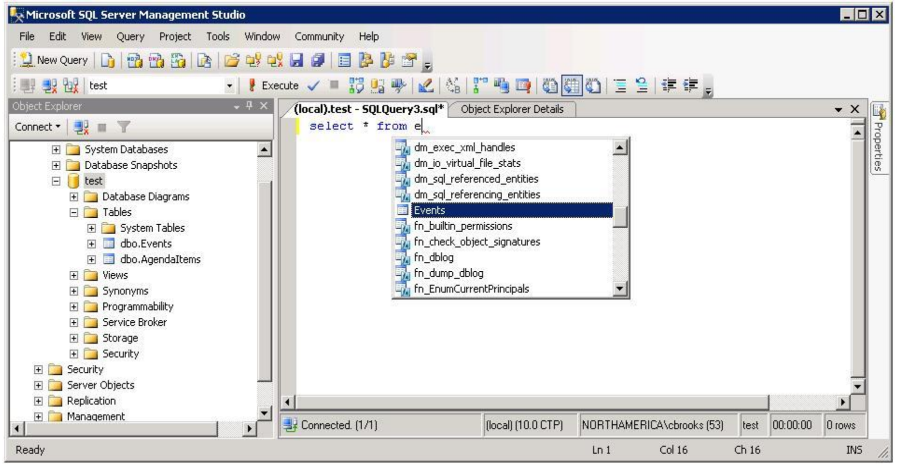
Lucidchart

erwin

Database Design & Modelling

Create Tables

SQL Server Management Studio



```
select * from e
```

Database Management

Microsoft
SQL Server
Database

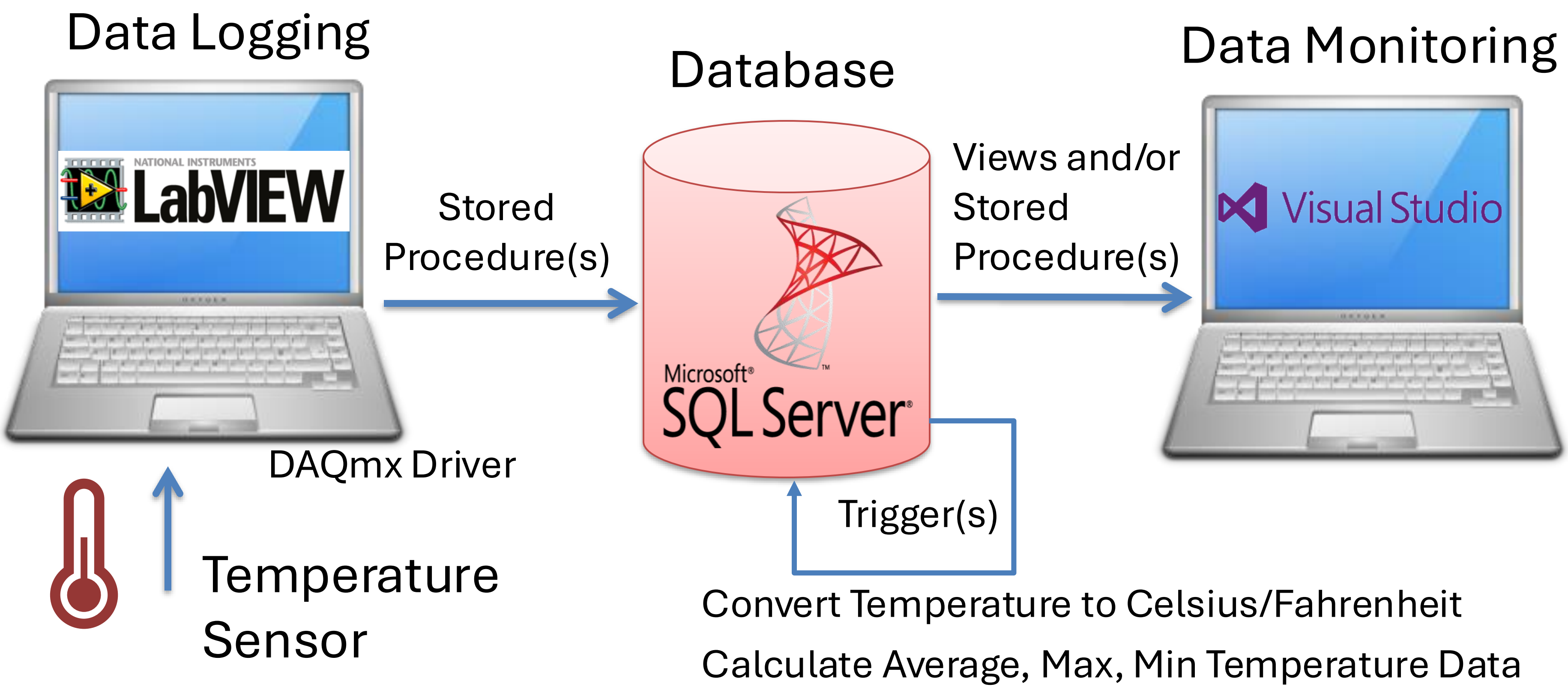
Write/Read Data

NATIONAL INSTRUMENTS LabVIEW

Visual Studio

We can create Applications in LabVIEW & Visual Studio that Writes and Reads Data to/from the Database.

System Overview



Data Logging

Database

Data Monitoring

Stored Procedure(s)

Views and/or Stored Procedure(s)

DAQmx Driver

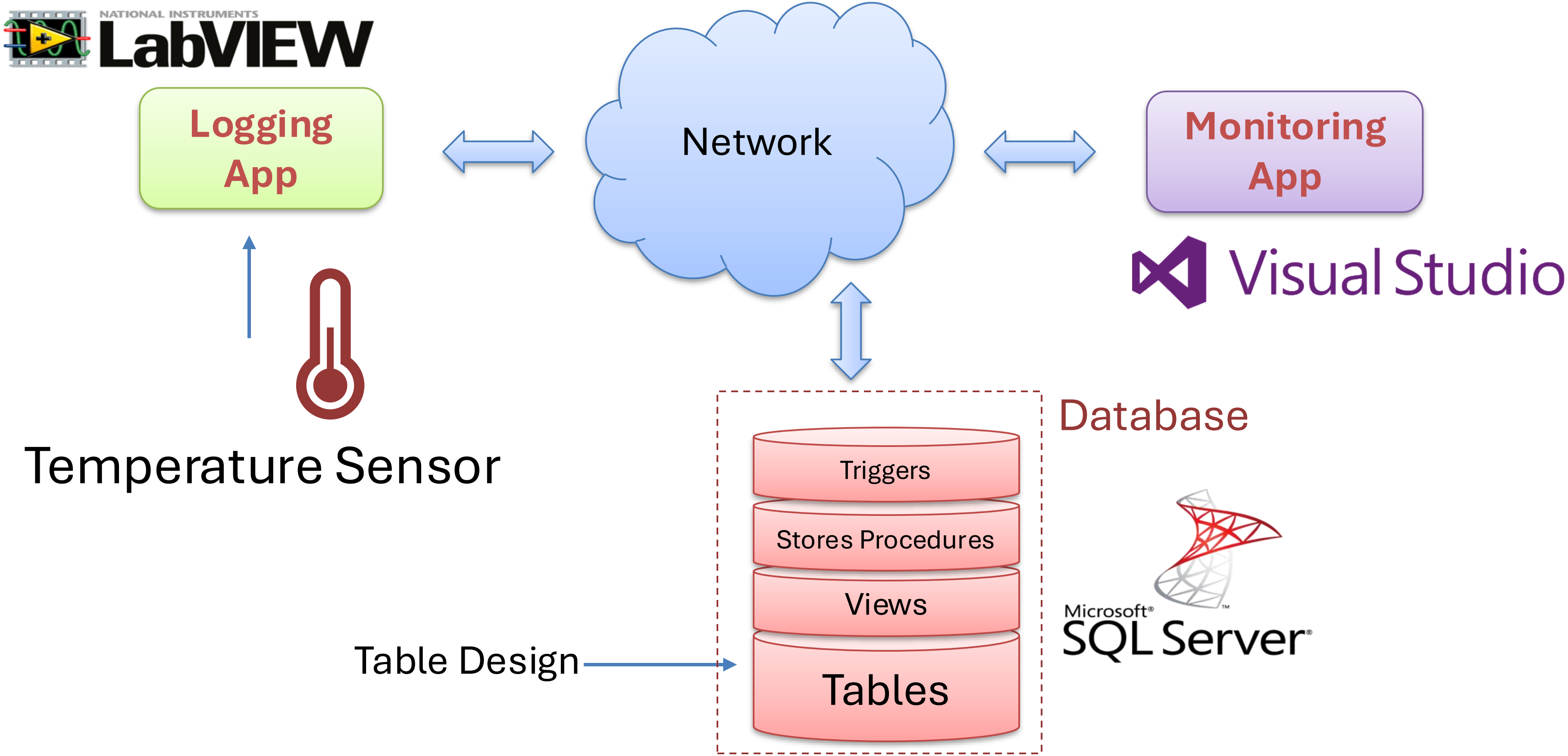
Temperature Sensor

Trigger(s)

Convert Temperature to Celsius/Fahrenheit

Calculate Average, Max, Min Temperature Data

System Overview



Database Modelling and Design

Hans-Petter Halvorsen

Table of Contents

ER Diagram

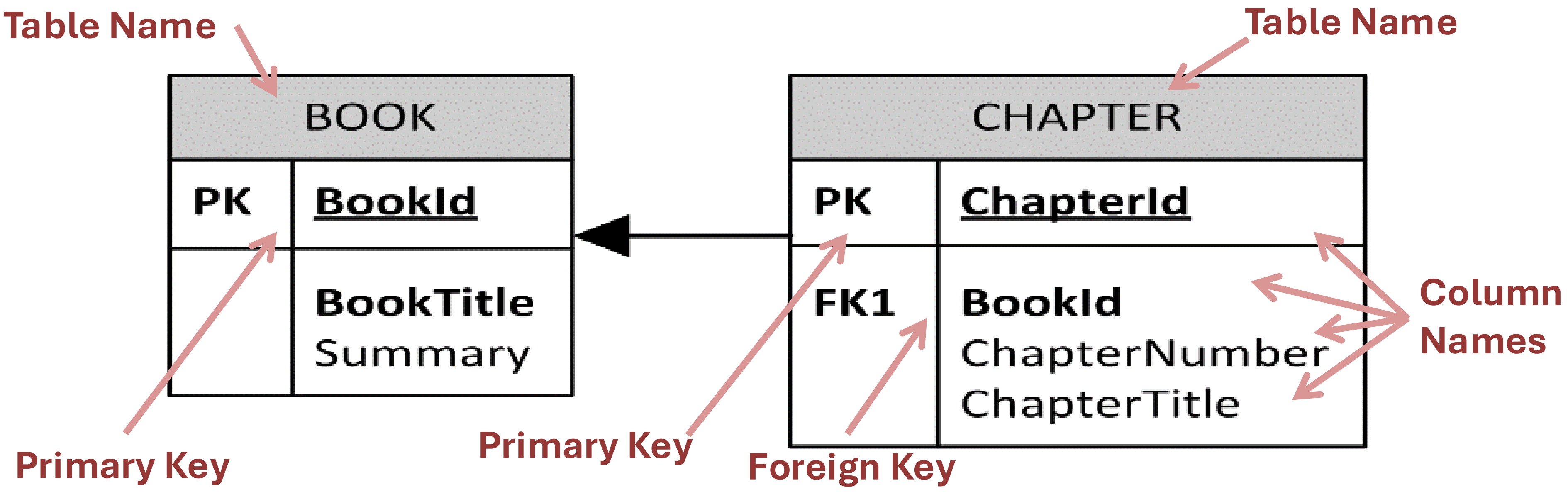


Table Name

Table Name

BOOK	
PK	**<u>BookId</u>**
	BookTitle Summary

CHAPTER	
PK	**<u>ChapterId</u>**
FK1	**BookId** ChapterNumber ChapterTitle

Column Names

Primary Key

Primary Key

Foreign Key

ER Diagram

ER Diagram (Entity-Relationship Diagram)

- Used for Design and Modeling of Databases.
- Specify Tables and **relationship** between them (**Primary Keys** and **Foreign Keys**)

Example:

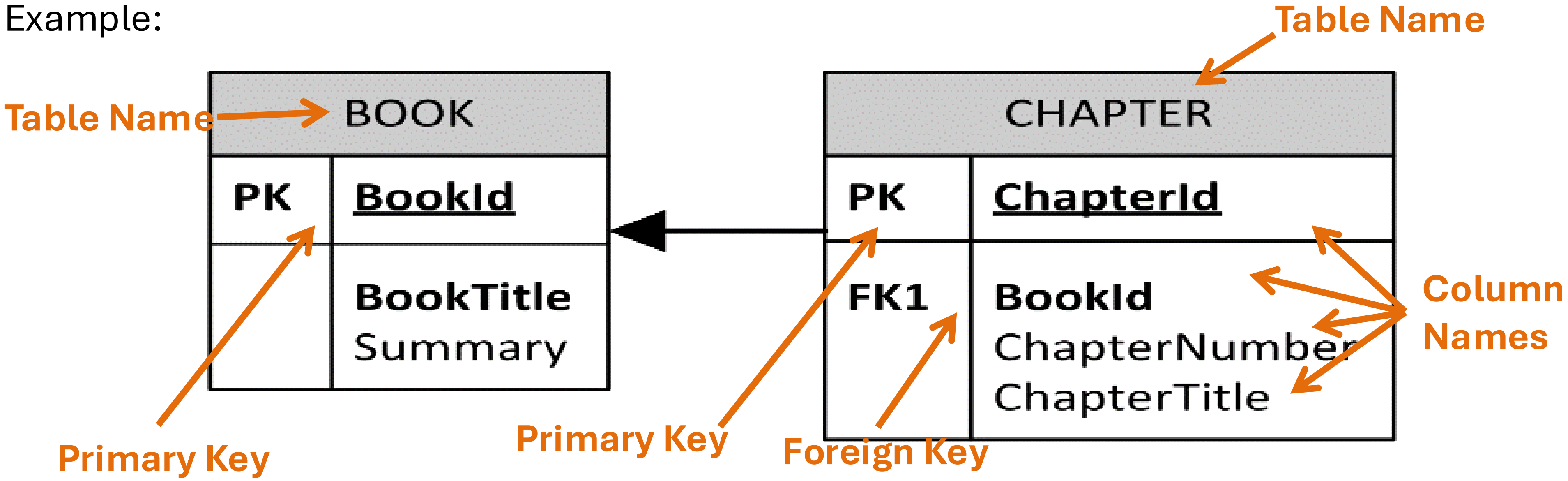


Table Name

BOOK

PK **BookId**

BookTitle
Summary

Table Name

CHAPTER

PK **ChapterId**

FK1 **BookId**
ChapterNumber
ChapterTitle

Column Names

Primary Key

Primary Key

Foreign Key

Relational Database. In a relational database all the tables have one or more relation with each other using Primary Keys (PK) and Foreign Keys (FK). Note! You can only have one PK in a table, but you may have several FK’s.

Database - “Best Practice”

- **Tables**: Use upper case and singular form in table names – not plural, e.g., “STUDENT” (not “students”)
- **Columns**: Use Pascal notation, e.g., “StudentId”
- **Primary Key**:
  - If the table name is “COURSE”, name the Primary Key column “CourseId”, etc.
  - “Always” use Integer and Identity(1,1) for Primary Keys. Use UNIQUE constraint for other columns that needs to be unique, e.g. “RoomNumber”
- Specify **Required** Columns (NOT NULL) – i.e., which columns that need to have data or not
- Standardize on few/these **Data Types**: *int*, *float*, *varchar(x)*, *datetime*, *bit*
- Use English for table and column names
- Avoid abbreviations! (Use “RoomNumber” – not “RoomNo”, “RoomNr”, ...)

Database System

- Typically to start by creating the overall Specifications and Design for your System.
- Them Design the Database Tables using an ERD software and create a SQL Script.
- Then implement the Tables in SQL Server, e.g., using a SQL Script generated from the ERD software.
- Then Create necessary Views, Stored Procedures and Triggers within the SQL Server Management Studio. It is recommended that you save these as separate SQL Files.

https://www.halvorsen.blog

SQL Server

Database Implementation and Structured Query Language (SQL)

Hans-Petter Halvorsen

Table of Contents

Microsoft SQL Server

Note! These are 2 separate modules you need to install

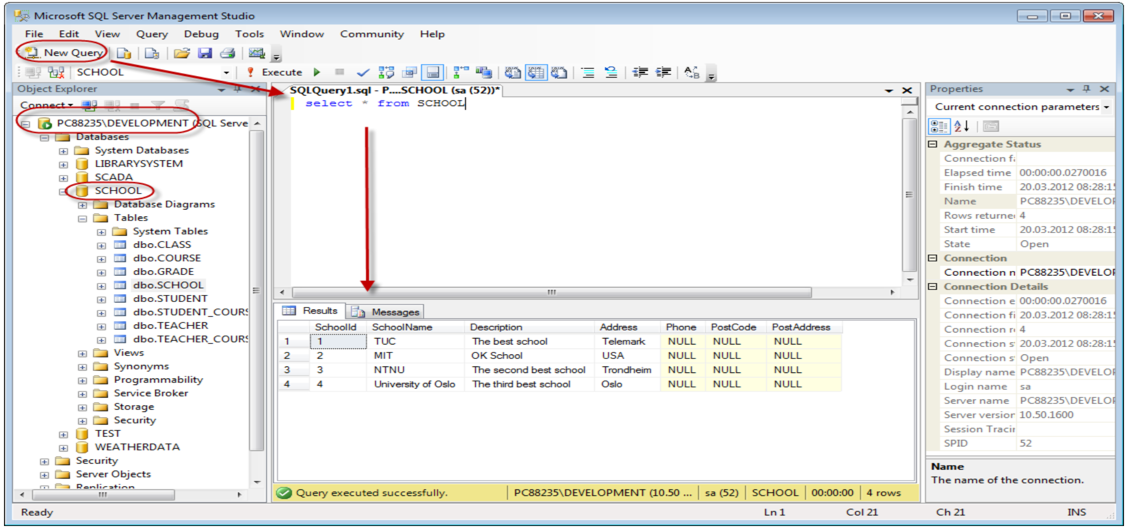
1

SQL Server Database Engine and Repository

The Data Storage

2

SQL Server **Management Studio**



```
select * from SCHOOL
```

	SchoolId	SchoolName	Description	Address	Phone	PostCode	PostAddress
1	1	TUC	The best school	Telemark	NULL	NULL	NULL
2	2	MIT	OK School	USA	NULL	NULL	NULL
3	3	NTNU	The second best school	Trondheim	NULL	NULL	NULL
4	4	University of Oslo	The third best school	Oslo	NULL	NULL	NULL

Query executed successfully. | PC88235\DEVELOPMENT (10.50 ... | sa (52) | SCHOOL | 00:00:00 | 4 rows

Aggregate Status	
Connection fa	
Elapsed time	00:00:00.0270016
Finish time	20.03.2012 08:28:1
Name	PC88235\DEVELOF
Rows returne	4
Start time	20.03.2012 08:28:1
State	Open
Connection	
Connection n	PC88235\DEVELOF
Connection Details	
Connection e	00:00:00.0270016
Connection fi	20.03.2012 08:28:1
Connection r	4
Connection s	20.03.2012 08:28:1
Connection s	Open
Display name	PC88235\DEVELOF
Login name	sa
Server name	PC88235\DEVELOF
Server versior	10.50.1600
Session Tracir	
SPID	52

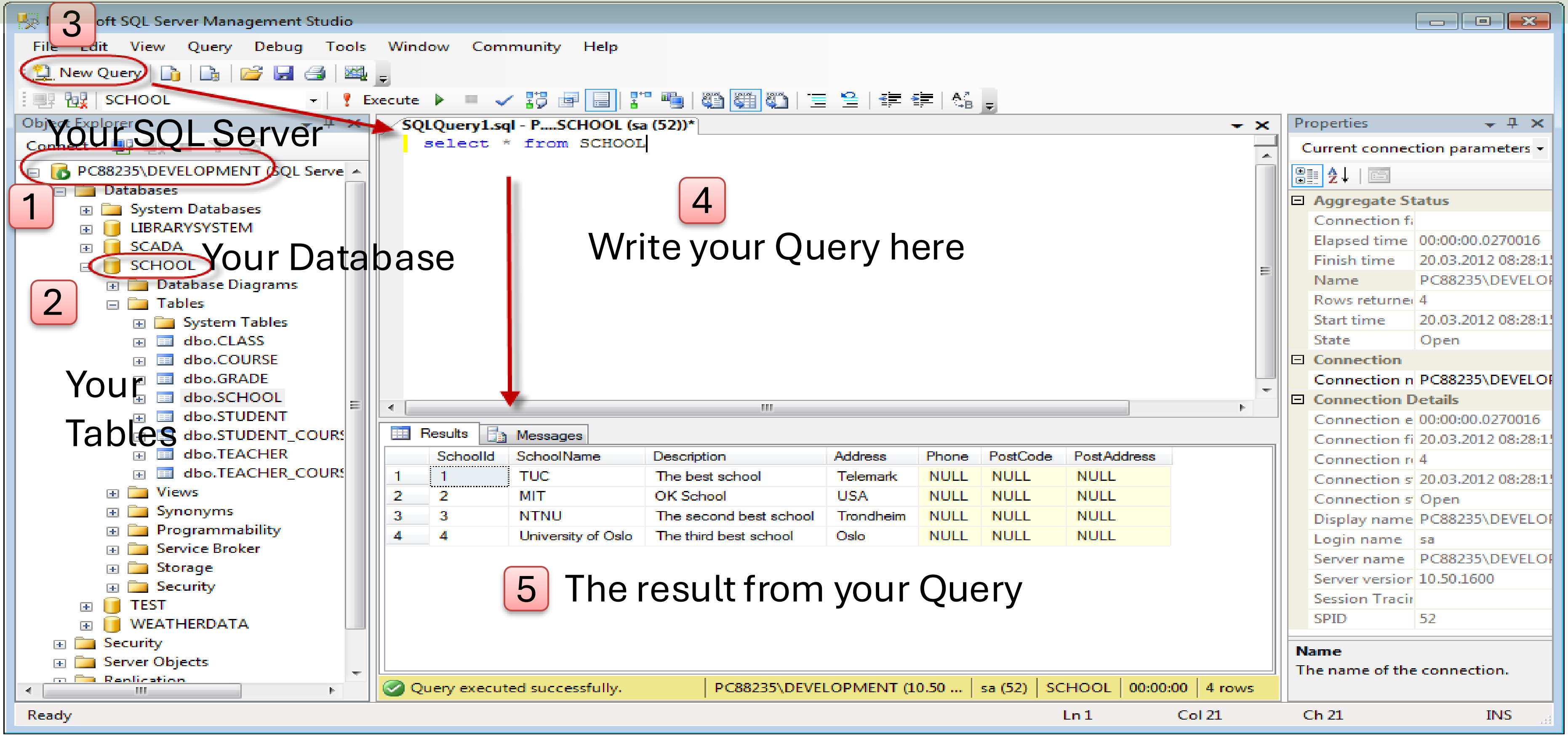
Name
The name of the connection.

A graphical interface to the Database Engine where you can create tables and manipulate data, etc.

Database Design and Implementation

- Start by **Design the Database Tables using an ERD software** and create a SQL Script.
- **Implement the Tables in SQL Server**, e.g., using a SQL Script generated in the ERD software.
- **Create necessary Views, Stored Procedures and Triggers** within the SQL Server Management Studio.
  - Put each of them into a .sql file.
  - You may wait to create them until you need them in the LabVIEW or C# Code.

Microsoft SQL Server Management Studio



3

1 Your SQL Server

2 Your Database

Your Tables

4 Write your Query here

```sql
select * from SCHOOL
```

5 The result from your Query

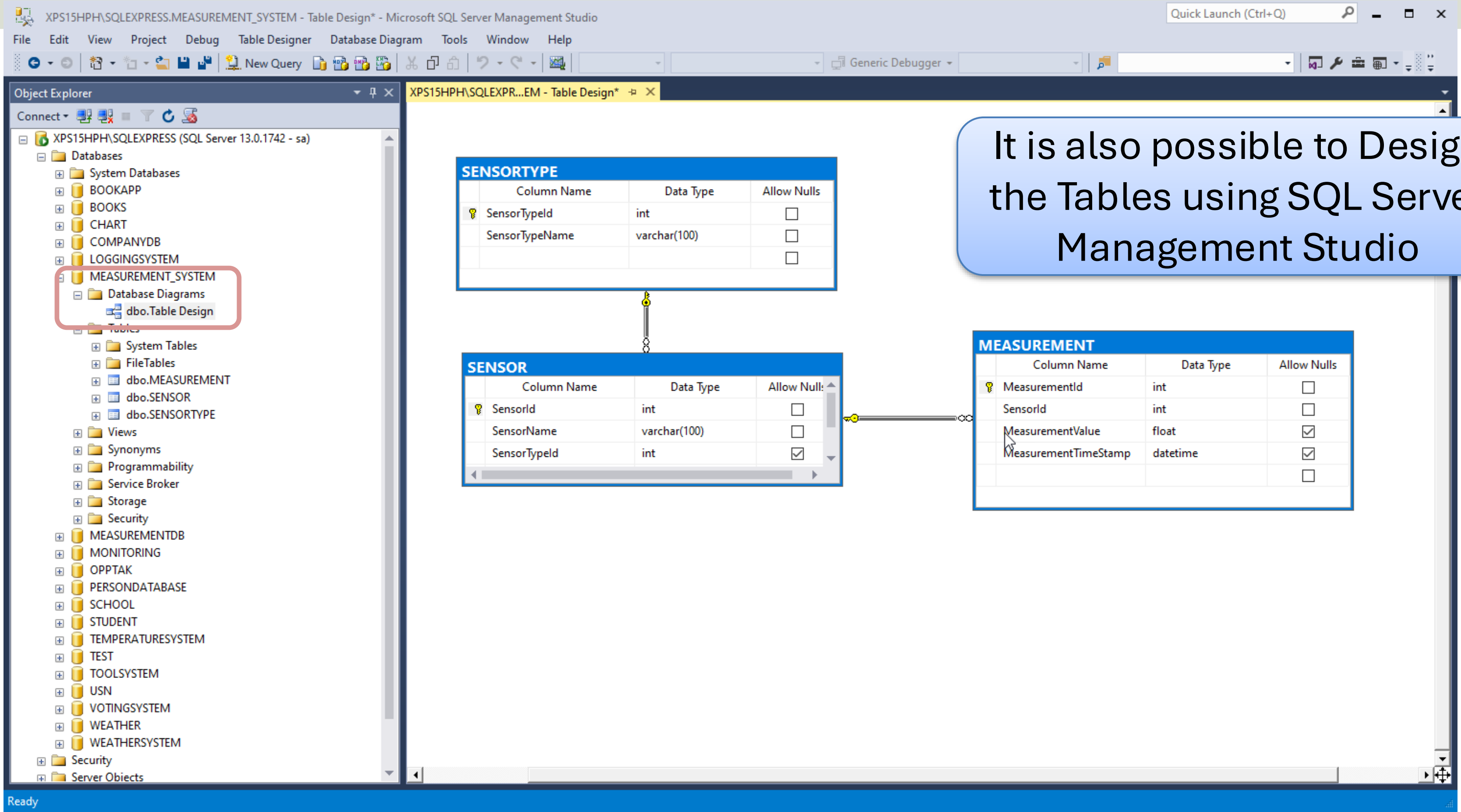
	SchoolId	SchoolName	Description	Address	Phone	PostCode	PostAddress
1	1	TUC	The best school	Telemark	NULL	NULL	NULL
2	2	MIT	OK School	USA	NULL	NULL	NULL
3	3	NTNU	The second best school	Trondheim	NULL	NULL	NULL
4	4	University of Oslo	The third best school	Oslo	NULL	NULL	NULL

Query executed successfully. | PC88235\DEVELOPMENT (10.50 ... | sa (52) | SCHOOL | 00:00:00 | 4 rows

Aggregate Status	
Connection f	
Elapsed time	00:00:00.0270016
Finish time	20.03.2012 08:28:1
Name	PC88235\DEVELOI
Rows returne	4
Start time	20.03.2012 08:28:1
State	Open
Connection	
Connection n	PC88235\DEVELOI
Connection Details	
Connection e	00:00:00.0270016
Connection fi	20.03.2012 08:28:1
Connection r	4
Connection s	20.03.2012 08:28:1
Connection s	Open
Display name	PC88235\DEVELOI
Login name	sa
Server name	PC88235\DEVELOI
Server versior	10.50.1600
Session Tracir	
SPID	52

Name
The name of the connection.

Database Design in SQL Server Management Studio



SENSORTYPE

	Column Name	Data Type	Allow Nulls
🔑	SensorTypeId	int	□
	SensorTypeName	varchar(100)	□
			□

SENSOR

	Column Name	Data Type	Allow Null:
🔑	SensorId	int	□
	SensorName	varchar(100)	□
	SensorTypeId	int	☑

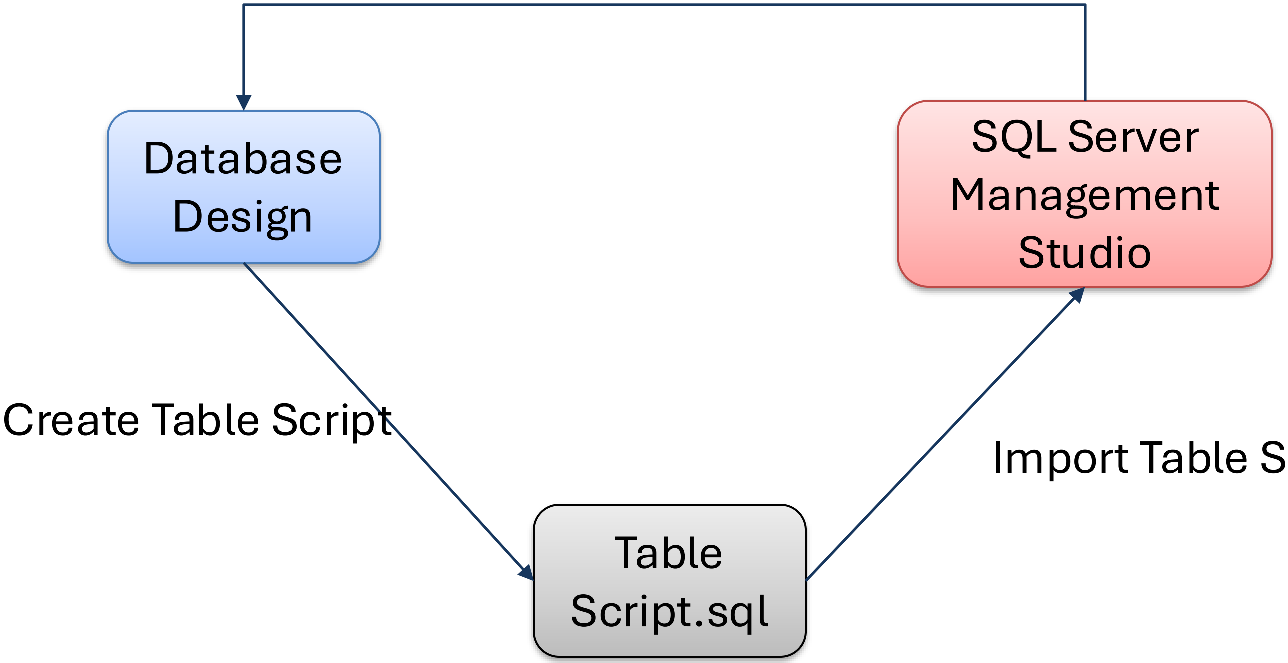
MEASUREMENT

	Column Name	Data Type	Allow Nulls
🔑	MeasurementId	int	□
	SensorId	int	□
	MeasurementValue	float	☑
	MeasurementTimeStamp	datetime	☑
			□

It is also possible to Design the Tables using SQL Server Management Studio

Database Design and Implementation

Need to make some improvements? Update the Table Design and import the Tables again



It is not recommended to make changes here

SQL – Structured Query Language

Query Examples:

```
• insert into STUDENT (Name , Number, SchoolId)
  values ('John Smith', '100005', 1)

• select SchoolId, Name from SCHOOL

• select * from SCHOOL where SchoolId > 100

• update STUDENT set Name='John Wayne' where StudentId=2

• delete from STUDENT where SchoolId=3
```

We have 4 different Query Types: **INSERT**, **SELECT**, **UPDATE** and **DELETE**

Views, Stored Procedures and Triggers

- **Views**: Views are virtual tables for easier access to data stored in multiple tables.
- **Stored Procedures**: A Stored Procedure is a precompiled collection of SQL statements. In a stored procedure you can use if sentence, declare variables, etc.
- **Triggers**: A database trigger is code that is automatically executed in response to certain events on a particular table in a database.

Database Views

- A Database View is a “virtual” table that can contain data from multiple tables
- You probably need to Create and Use one or more Database Views to get Data from the Database, both in the Data Logging App and Data Monitoring App
- It is recommended that you wait to create them until you need them in the LabVIEW or C# Code

Database Views

1 Create View:

```
IF EXISTS (SELECT name
             FROM   sysobjects
             WHERE  name = 'CourseData'
             AND      type = 'V')
          DROP VIEW CourseData
GO

CREATE VIEW CourseData
AS

SELECT
SCHOOL.SchoolId,
SCHOOL.SchoolName,
COURSE.CourseId,
COURSE.CourseName,
COURSE.Description

FROM
SCHOOL
INNER JOIN COURSE ON SCHOOL.SchoolId = COURSE.SchoolId
GO
```

A View is a “virtual” table that can contain data from <u>multiple</u> tables

This part is not necessary – but if you make any changes, you need to delete the old version before you can update it

The Name of the View

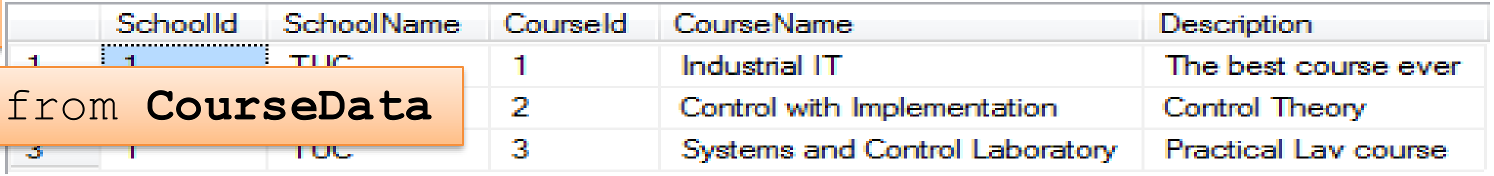
Inside the View you join the different tables together using the **JOIN** operator

You can Use the View as an ordinary table in Queries:

Using the View:

2

```
select * from CourseData
```



	SchoolId	SchoolName	CourseId	CourseName	Description
1	1	TUC	1	Industrial IT	The best course ever
			2	Control with Implementation	Control Theory
3	1	TUC	3	Systems and Control Laboratory	Practical Lav course

Database View Template

```
IF EXISTS (SELECT name
             FROM   sysobjects
             WHERE  name = '<ViewName>'
             AND      type = 'V')
          DROP VIEW <ViewName>
GO

CREATE VIEW <ViewName>
AS

SELECT
<TableName>.<ColumnName>,
<TableName>.<ColumnName>,
<TableName>.<ColumnName>,
<TableName>.<ColumnName>,
<TableName>.<ColumnName>

FROM
<TableName1>
INNER JOIN <TableName2> ON <TableName1>.<PrimKeyColumnName1> = <TableName2>.<PrimKeyColumnName2>
GO
```

Copy to SQL Server Management Studio, save as a SQL File (.sql) as the same name as the View you are going to create. Store all your files on your hard drive.

Stored Procedures

Typically, you need some Stored Procedures:

- The Datalogging App should typically use a Stored Procedure to save Measurement Data to the Database.
- The Datalogging App should typically use a Stored Procedure to save Configuration Data to the Database.
  - Logging Interval
  - Unit (Celsius or Fahrenheit)
- It is recommended that you wait to create them until you need them in the LabVIEW or C# Code

Stored Procedures

1

Create Stored Procedure:

```
IF EXISTS (SELECT name
                FROM  sysobjects
                WHERE  name = 'StudentGrade'
                AND              type = 'P')
              DROP PROCEDURE StudentGrade
GO

CREATE PROCEDURE StudentGrade
@Student varchar(50),
@Course varchar(10),
@Grade varchar(1)

AS

DECLARE
@StudentId int,
@CourseId int

select @StudentId = StudentId from STUDENT where StudentName = @Student

select @CourseId = CourseId from COURSE where CourseName = @Course

insert into GRADE (StudentId, CourseId, Grade)
values (@StudentId, @CourseId, @Grade)
GO
```

A Stored Procedure is like a Method in C# - it is a piece of code with SQL commands that do a specific task – and you reuse it

This part is not necessary – but if you make any changes, you need to delete the old version before you can update it

Procedure Name

Input Arguments

Internal/Local Variables
Note! Each variable starts with @

SQL Code (the "body" of the Stored Procedure)

2

Using the Stored Procedure:

```
execute StudentGrade 'John Wayne', 'SCE2006', 'B'
```

Stored Procedure Template

```
IF EXISTS (SELECT name
              FROM   sysobjects
              WHERE  name = '<StoredProcedureName>'
              AND       type = 'P')
           DROP PROCEDURE <StoredProcedureName>
GO

CREATE PROCEDURE <StoredProcedureName>
@<InputVariable1> <DataType>,
@<InputVariable2> <DataType>
AS

DECLARE
@<InternalVariable1> <DataType>,
@<InternalVariable2> <DataType>

select @<InternalVariable1> = <ColumnName> from <TableName> where <ColumnName> =
@<InputVariable1>

insert into <TableName> (<ColumnName1>, <ColumnName2>, ...) values (@<InternalVariable1>,
@<Inputvariable1>, ...)
GO
```

Copy to SQL Server Management Studio, save as a SQL File (.sql) as the same name as the SP you are going to create. Store all your files on your hard drive.

Database Triggers

You may need one or more Triggers that do, e.g., the following:

- Convert Temperature to Celsius/Fahrenheit
  - E.g., If Unit=Celsius, the Trigger should Convert Temperature Data to Fahrenheit.
  - E.g., If Unit=Fahrenheit, the Trigger should Convert Temperature Data to Celsius.
  - Both Celsius and Fahrenheit values should probably be stored in the Database for easy access later in Monitoring App.
- Calculate Average, Max, Min Temperature Data
  - The Trigger should calculate and store Average(Mean), Max and Min Temperature Data into the Database.
- You may wait to create them until you need them in the LabVIEW or C# Code.

Database Triggers

A Trigger is executed when you insert, update or delete data in a Table specified in the Trigger

```
IF EXISTS (SELECT name
                FROM    sysobjects
                WHERE   name = 'CalcAvgGrade'
                AND         type = 'TR')
            DROP TRIGGER CalgAvgGrade
GO

CREATE TRIGGER CalcAvgGrade ON GRADE
FOR UPDATE, INSERT, DELETE
AS

DECLARE
@StudentId int,
@AvgGrade float

select @StudentId = StudentId from INSERTED

select @AvgGrade = AVG(Grade) from GRADE where StudentId = @StudentId

update STUDENT set TotalGrade = @AvgGrade where StudentId = @StudentId

GO
```

This part is not necessary – but if you make any changes, you need to delete the old version before you can update it

Name of the Trigger

Specify which Table the Trigger shall work on

Specify what kind of operations the Trigger shall act on

Internal/Local Variables

Example

Inside the Trigger you can use ordinary SQL statements, create variables, etc.

SQL Code (The "body" of the Trigger)

Note! "INSERTED" is a temporarily table containing the latest inserted data, and it is very handy to use inside a trigger

Trigger Template

```
IF EXISTS (SELECT name
             FROM   sysobjects
             WHERE  name = '<TriggerName>'
             AND      type = 'TR')
          DROP TRIGGER <TriggerName>
GO

CREATE TRIGGER <TriggerName> ON <TableName>
FOR UPDATE, INSERT, DELETE --Delete the ones not needed
AS

DECLARE
@<InternalVariable1> <DataType>,
@<InternalVariable2> <DataType>

select @Variable1 = Column1 from INSERTED
select @Variable2 = AVG(Column2) from TABLE where Column1 = @Variable1
update TABLE set Column3= @Variabl2e where Column1= @Variable1

GO
```

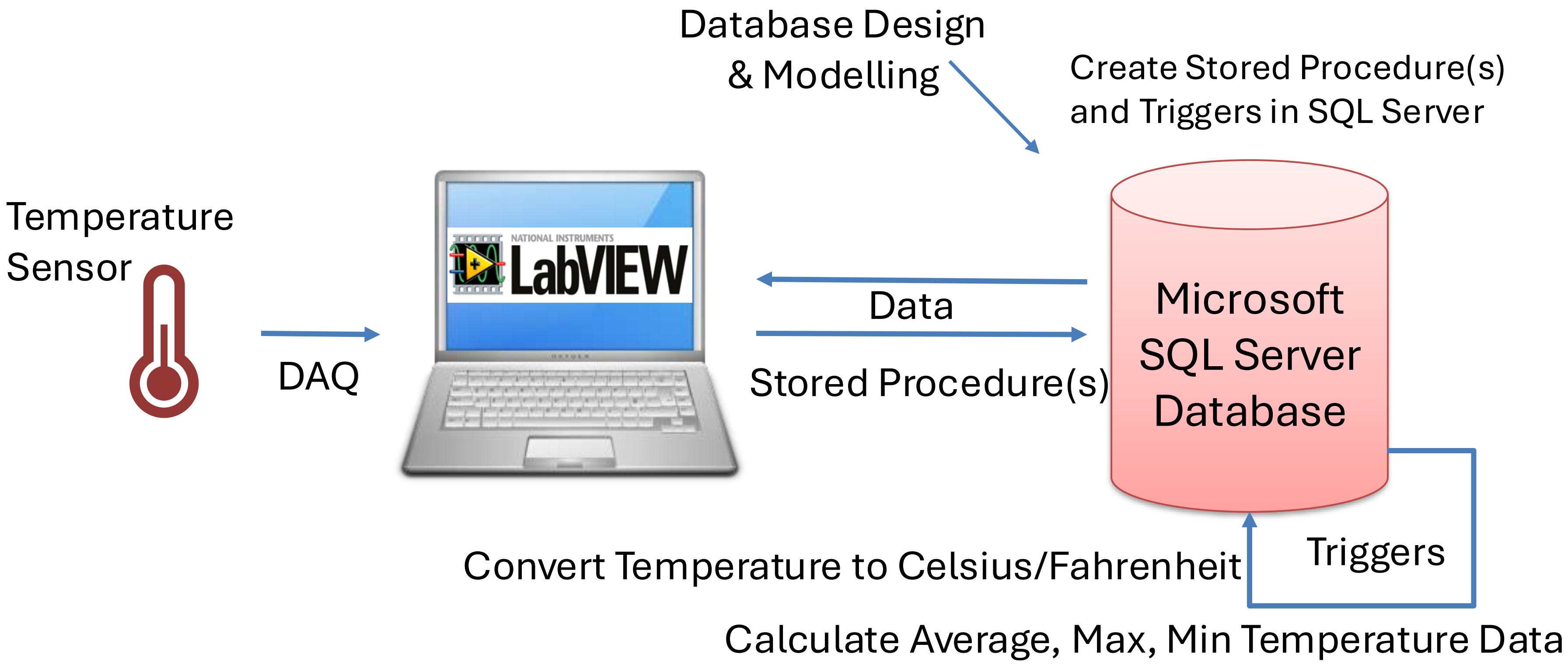
Copy to SQL Server Management Studio, save as a SQL File (.sql) as the same name as the Trigger you are going to create. Store all your files on your hard drive.

Datalogging using LabVIEW

Hans-Petter Halvorsen

Table of Contents

Datalogging using LabVIEW



Temperature Sensor

DAQ

Database Design & Modelling

Create Stored Procedure(s) and Triggers in SQL Server

Data

Stored Procedure(s)

Microsoft SQL Server Database

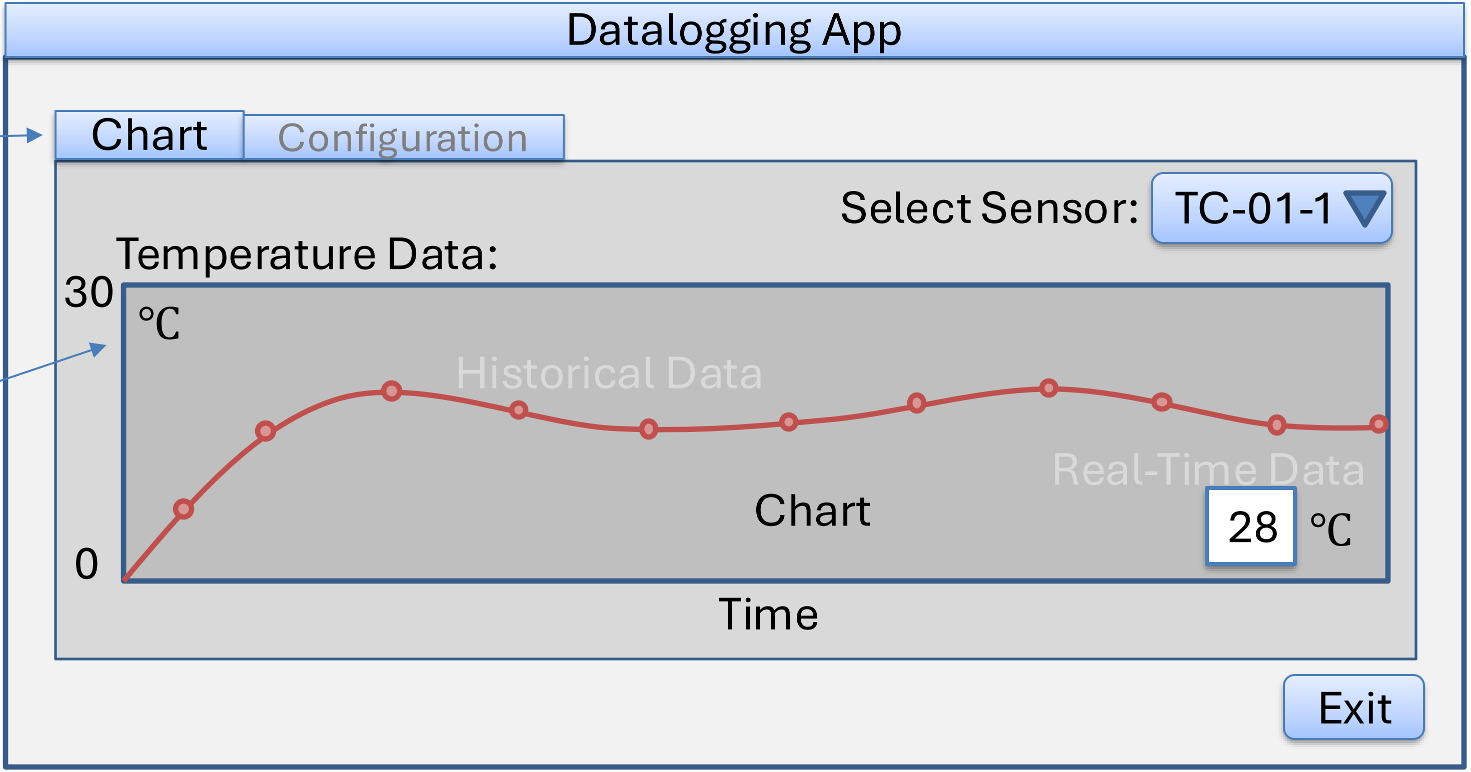
Triggers

Convert Temperature to Celsius/Fahrenheit

Calculate Average, Max, Min Temperature Data

LabVIEW HMI Example

The Temperature Data from the Sensors(s) should typically be stored in the Database

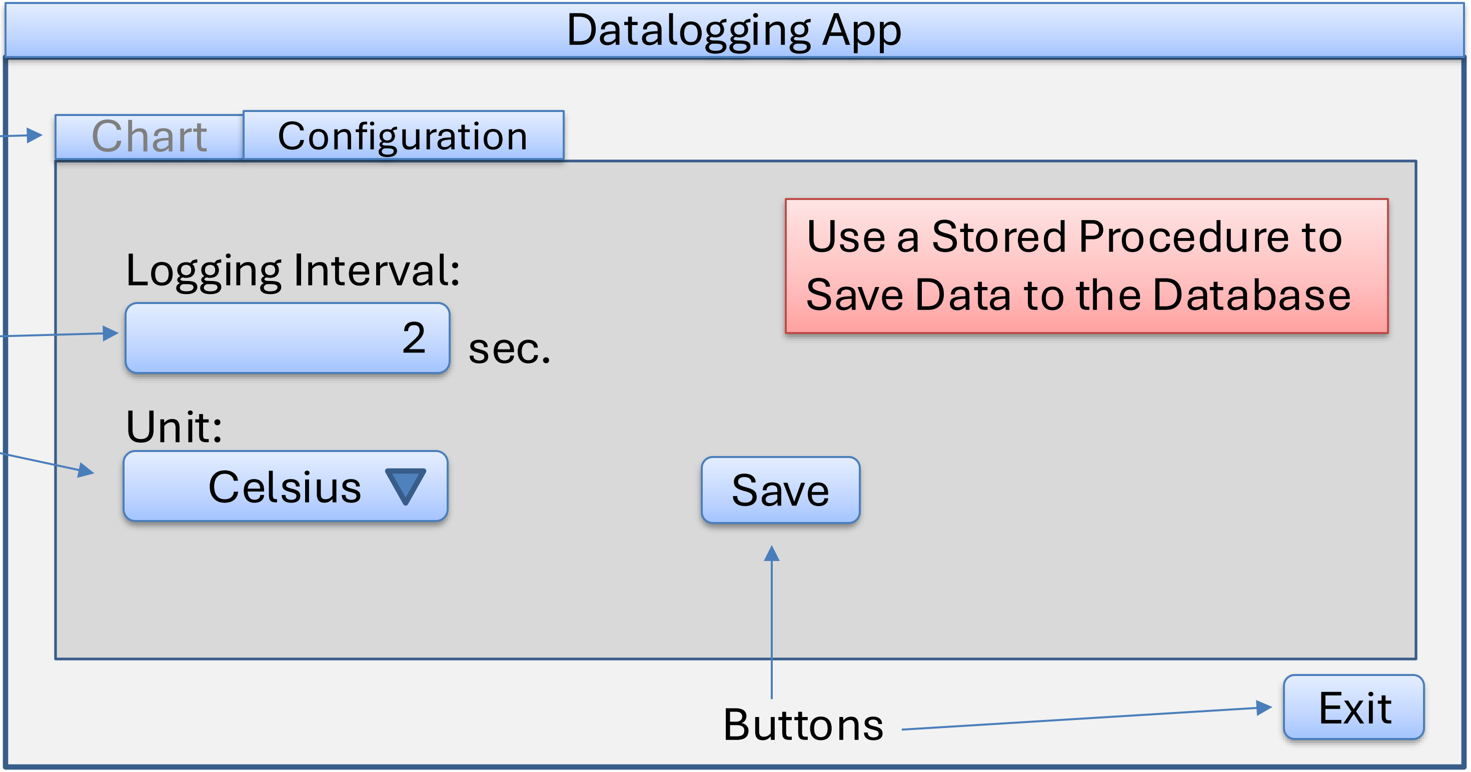


Tab Control

Show Temperature in Celsius or Fahrenheit depending on the Configuration

LabVIEW HMI Example

The Temperature Data from the Sensors(s) should be stored in the Database



Datalogging App

Logging Interval: 2 sec.

Unit: Celsius

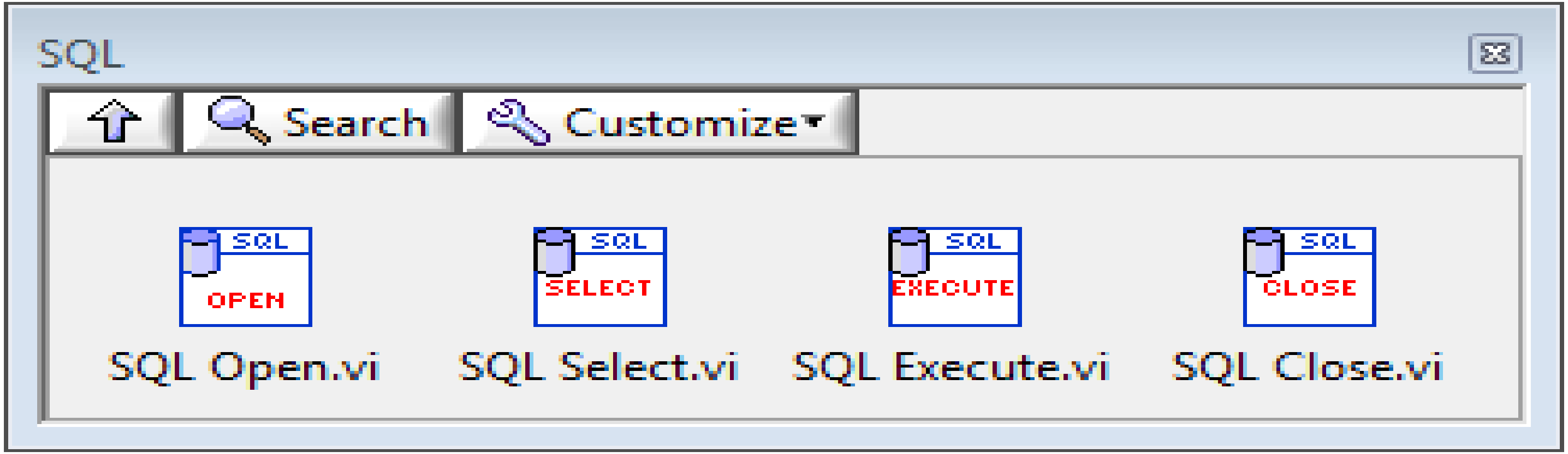
Use a Stored Procedure to Save Data to the Database

E.g., If Unit=Celsius, the Trigger should Convert Temperature Data to Fahrenheit

E.g., If Unit=Fahrenheit, the Trigger should Convert Temperature Data to Celsius

LabVIEW SQL Toolkit

For Easy Database Communication with LabVIEW



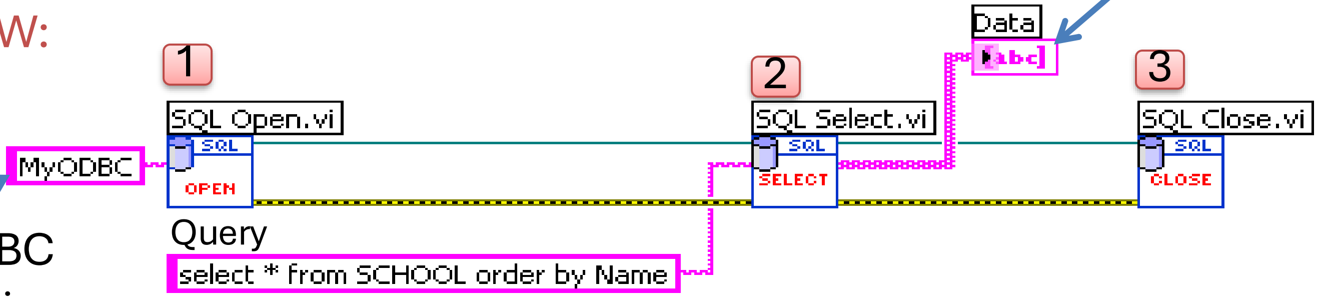
Download for free here:

https://www.halvorsen.blog/documents/technology/database/database_labview.php

LabVIEW SQL Toolkit Example

Easy Access to Database Systems from LabVIEW

Example 1: Get Data from Database into LabVIEW:

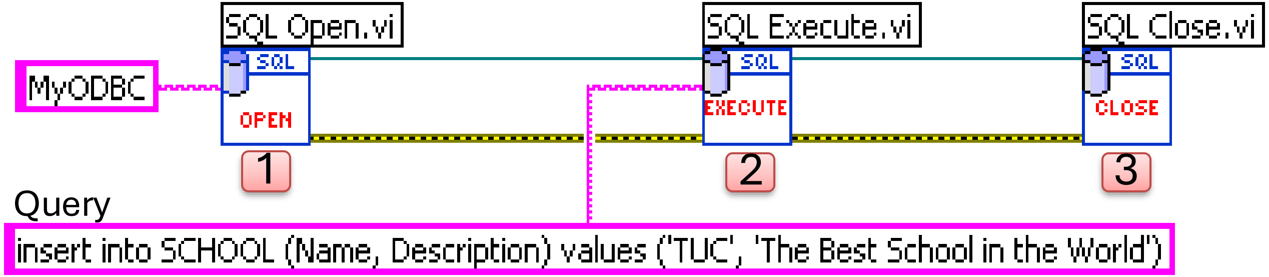


```
select * from SCHOOL order by Name
```

2D Table with Data

Your ODBC Connection

Example 2: Write Data to Database from LabVIEW:



```
insert into SCHOOL (Name, Description) values ('TUC', 'The Best School in the World')
```

Connect to Database

- Alt 1: Use ODBC
  - Setup your Database connection using a Wizard ("ODBC Data Source Administrator")
- Alt 2: Use Connection String directly
  - Alt 2.2: SQL Server Authentication:

    ```
    PROVIDER=SQLOLEDB; DATA SOURCE=COMPUTERNAME\SQLEXPRESS; DATABASE=MEASUREMENTS; UID=sa; PWD=xxx;
    ```

  - Alt 2.1: Windows Authentication:

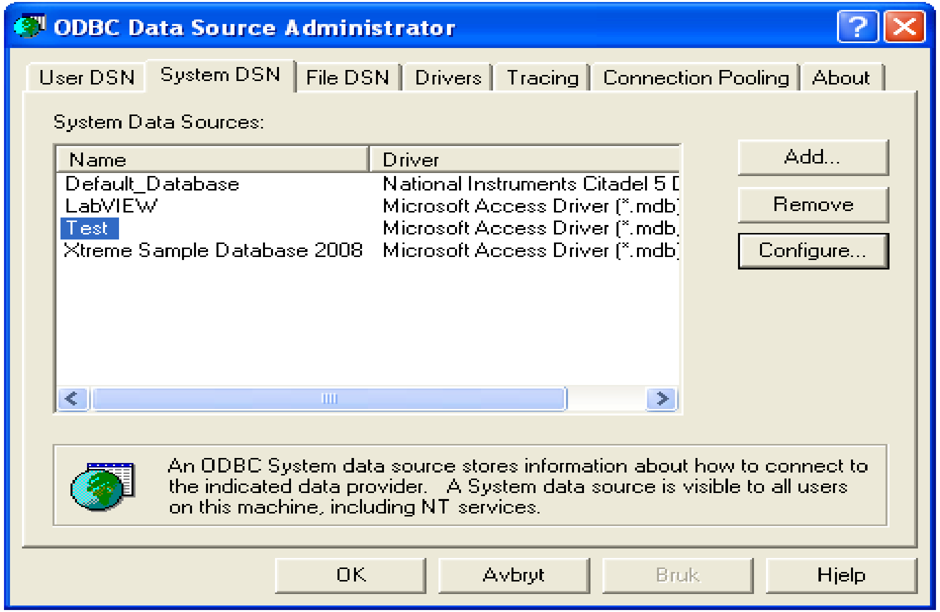
    ```
    Data Source=<dbserver>;Initial Catalog=<dbname>;Trusted_Connection=True
    ```

See Examples on next slides…

ODBC

ODBC (Open Database Connectivity) is a standardized interface (API) for accessing the database from a client. You can use this standard to communicate with databases from different vendors, such as Oracle, SQL Server, etc. The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems.

Control Panel → Administrative Tools → Data Sources (ODBC)



System Data Sources:

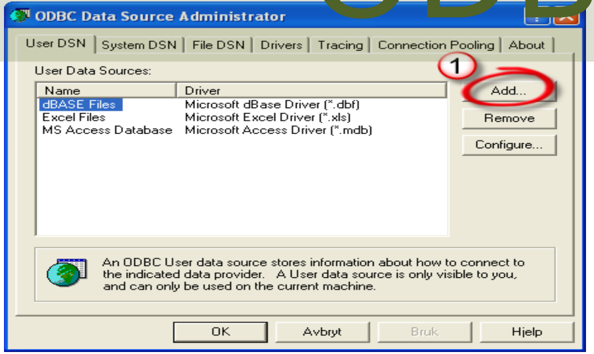
Name	Driver
Default_Database	National Instruments Citadel 5 D
LabVIEW	Microsoft Access Driver (*.mdb)
Test	Microsoft Access Driver (*.mdb,
Xtreme Sample Database 2008	Microsoft Access Driver (*.mdb)

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

We will use this ODBC Connection later in LabVIEW to open the Database Connection from LabVIEW

Note! Make sure to use the **32-bit** version of the ODBC Tool!

ODBC – Step by Step


ODBC Data Source Administrator

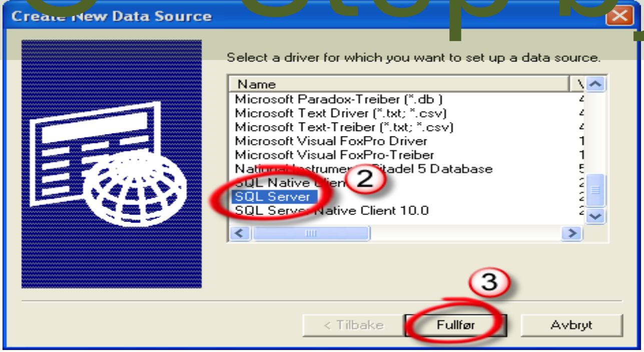
User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
MS Access Database	Microsoft Access Driver (*.mdb)

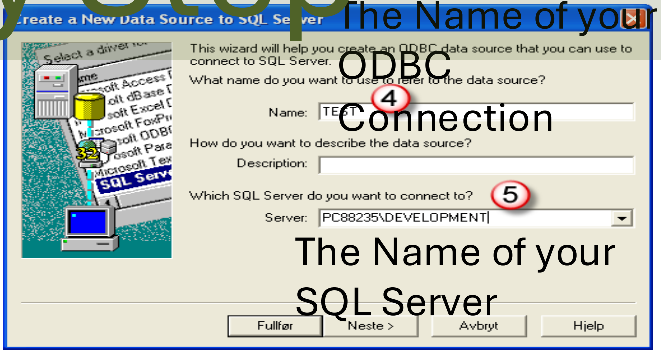
1
Add...
Remove
Configure...

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.


Create New Data Source

Select a driver for which you want to set up a data source.

Name
- Microsoft Paradox-Treiber (*.db)
- Microsoft Text Driver (*.txt; *.csv)
- Microsoft Text-Treiber (*.txt; *.csv)
- Microsoft Visual FoxPro Driver
- Microsoft Visual FoxPro-Treiber
- National Instruments Citadel 5 Database
- SQL Native Client
- SQL Server
- SQL Server Native Client 10.0


Create a New Data Source to SQL Server

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: TEST

How do you want to describe the data source?

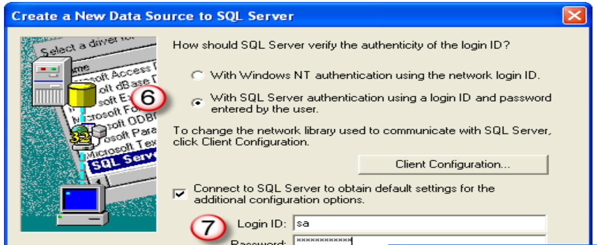
Description:

Which SQL Server do you want to connect to?

Server: PC88235\DEVELOPMENT

The Name of your ODBC Connection

The Name of your SQL Server


Create a New Data Source to SQL Server

How should SQL Server verify the authenticity of the login ID?

○ With Windows NT authentication using the network login ID.

● With SQL Server authentication using a login ID and password entered by the user.

To change the network library used to communicate with SQL Server, click Client Configuration.

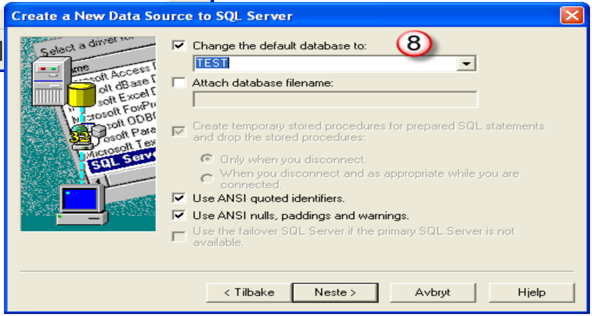
☑ Connect to SQL Server to obtain default settings for the additional configuration options.

Login ID: sa

Password: ************

Select the Database you are using

Use either Windows or SQL Server authentication (Windows is simplest to use!)


Create a New Data Source to SQL Server

☑ Change the default database to: TEST

□ Attach database filename:

☑ Create temporary stored procedures for prepared SQL statements and drop the stored procedures:

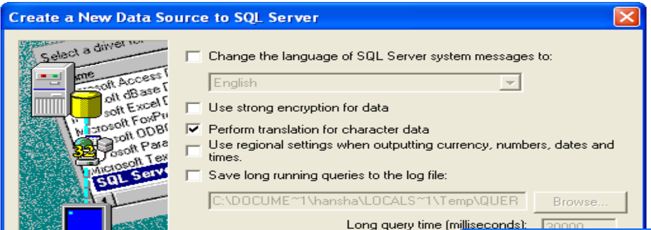
● Only when you disconnect.

○ When you disconnect and as appropriate while you are connected.

☑ Use ANSI quoted identifiers.

☑ Use ANSI nulls, paddings and warnings.

□ Use the failover SQL Server if the primary SQL Server is not available.


Create a New Data Source to SQL Server

□ Change the language of SQL Server system messages to: English

□ Use strong encryption for data

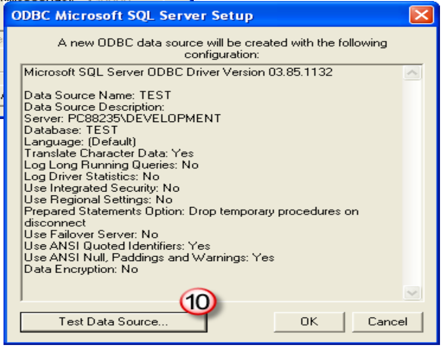
☑ Perform translation for character data

□ Use regional settings when outputting currency, numbers, dates and times.

□ Save long running queries to the log file: C:\DOCUME~1\hansha\LOCALS~1\Temp\QUER

Long query time (milliseconds): 30000

□ Log ODBC driver statistics to the log file: C:\DOCUME~1\hansha\LOCALS~1\Te


ODBC Microsoft SQL Server Setup

A new ODBC data source will be created with the following configuration:

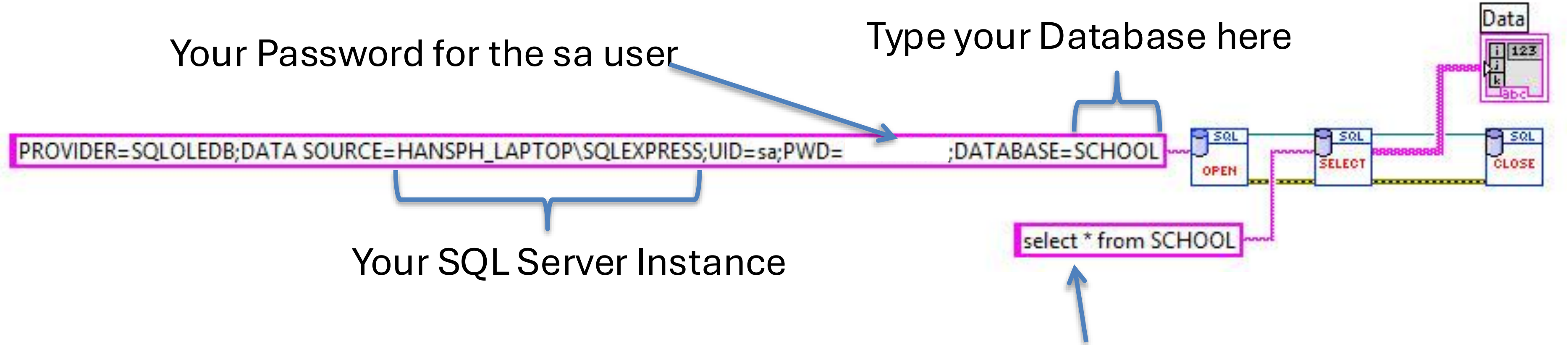
```
Microsoft SQL Server ODBC Driver Version 03.85.1132

Data Source Name: TEST
Data Source Description:
Server: PC88235\DEVELOPMENT
Database: TEST
Language: (Default)
Translate Character Data: Yes
Log Long Running Queries: No
Log Driver Statistics: No
Use Integrated Security: No
Use Regional Settings: No
Prepared Statements Option: Drop temporary procedures on
disconnect
Use Failover Server: No
Use ANSI Quoted Identifiers: Yes
Use ANSI Null, Paddings and Warnings: Yes
Data Encryption: No
```

Test your connection to see if its works

LabVIEW SQL Toolkit Example

Easy Access to Database Systems from LabVIEW

Alternative Solution: Type in the **Connection String** for your Database



Your Password for the sa user

Type your Database here

```
PROVIDER=SQLOLEDB;DATA SOURCE=HANSPH_LAPTOP\SQLEXPRESS;UID=sa;PWD=              ;DATABASE=SCHOOL
```

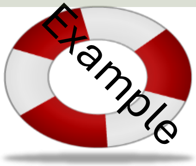
```
select * from SCHOOL
```

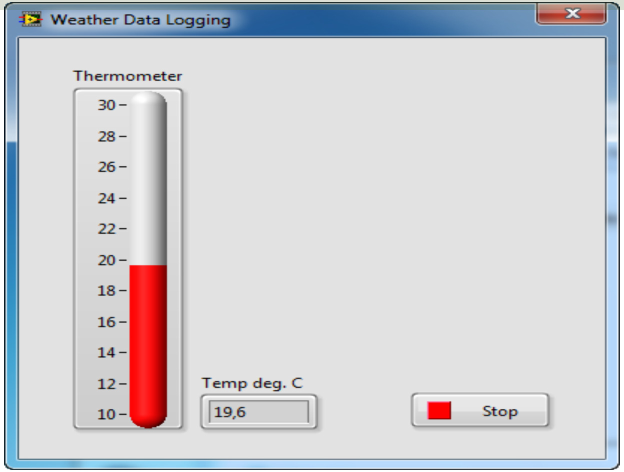
Your SQL Server Instance

Your SQL Query

Note! When using this method, you don’t need to create an ODBC Connection first!

LabVIEW SQL Toolkit Example

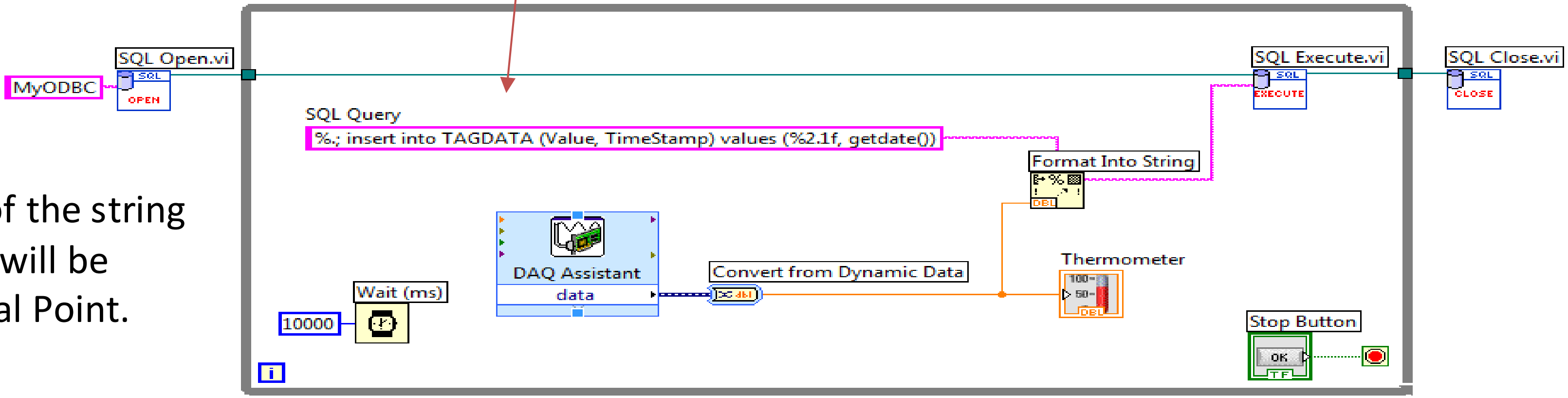




Temp deg. C: 19,6

<u>You</u> should use a Stored Procedure for saving the Temperature Data to the Database

"%2.1f" means that this is replaced with the value that comes from the Sensor with one decimal value. "f" means it is a floating-point value.

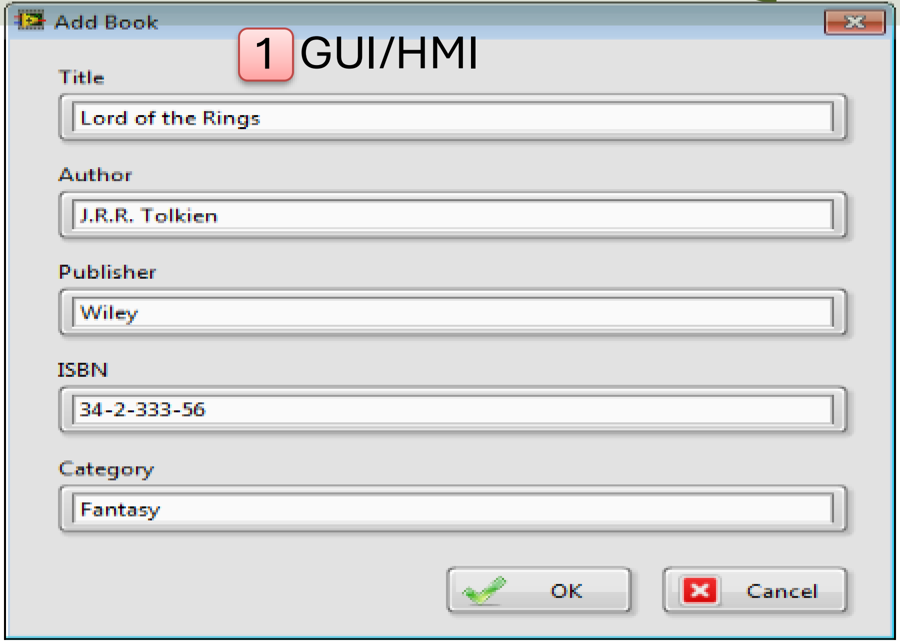


SQL Query

```
%.; insert into TAGDATA (Value, TimeStamp) values (%2.1f, getdate())
```

"%.;" in front of the string means that "." will be used as Decimal Point.

LabVIEW SQL Toolkit Example



1 GUI/HMI

Title: Lord of the Rings

Author: J.R.R. Tolkien

Publisher: Wiley

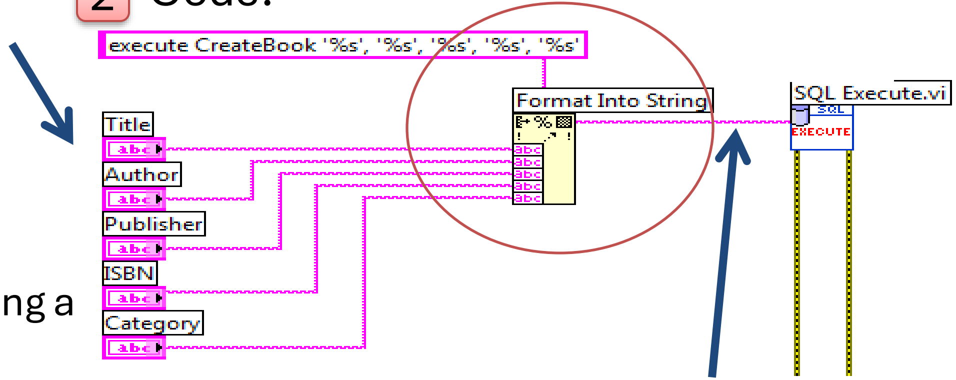
ISBN: 34-2-333-56

Category: Fantasy

If we want to save input data from the user, we can use the “**Format Into String**” function

The **%s** operator will be replaced by the text from the TextBox on the Front Panel. For Numbers we can use **%d** (Integer) or **%f** for Floating-point Number.

2 Code:



```
execute CreateBook '%s', '%s', '%s', '%s', '%s'
```

3 Resulting SQL Query:

Example of Executing a Stored Procedure

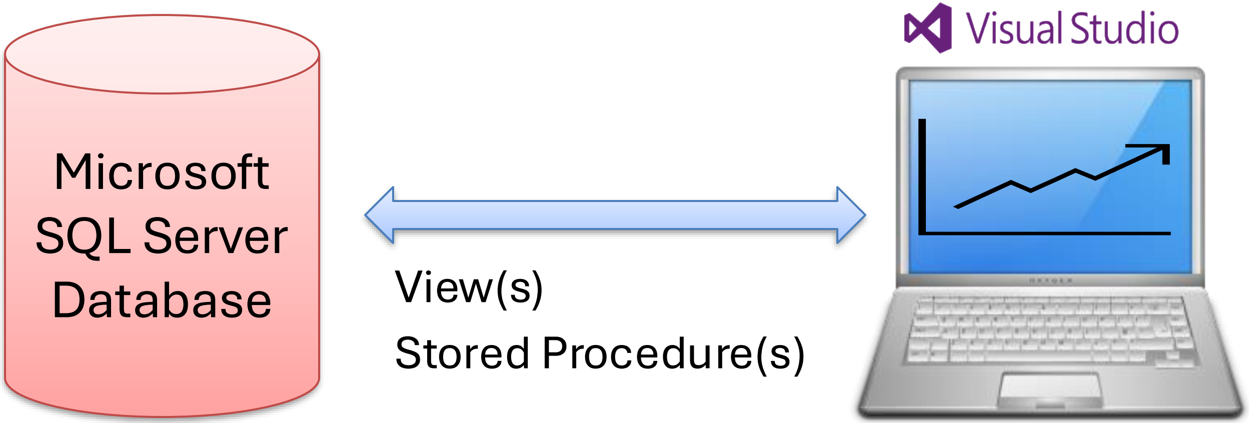
execute **CreateBook** ‘Lord of the Rings', ‘J.R.R. Tolkien', Wiley', '32-2-333-56', 'Fantasy'

Data Monitoring using Visual Studio/C#

Hans-Petter Halvorsen

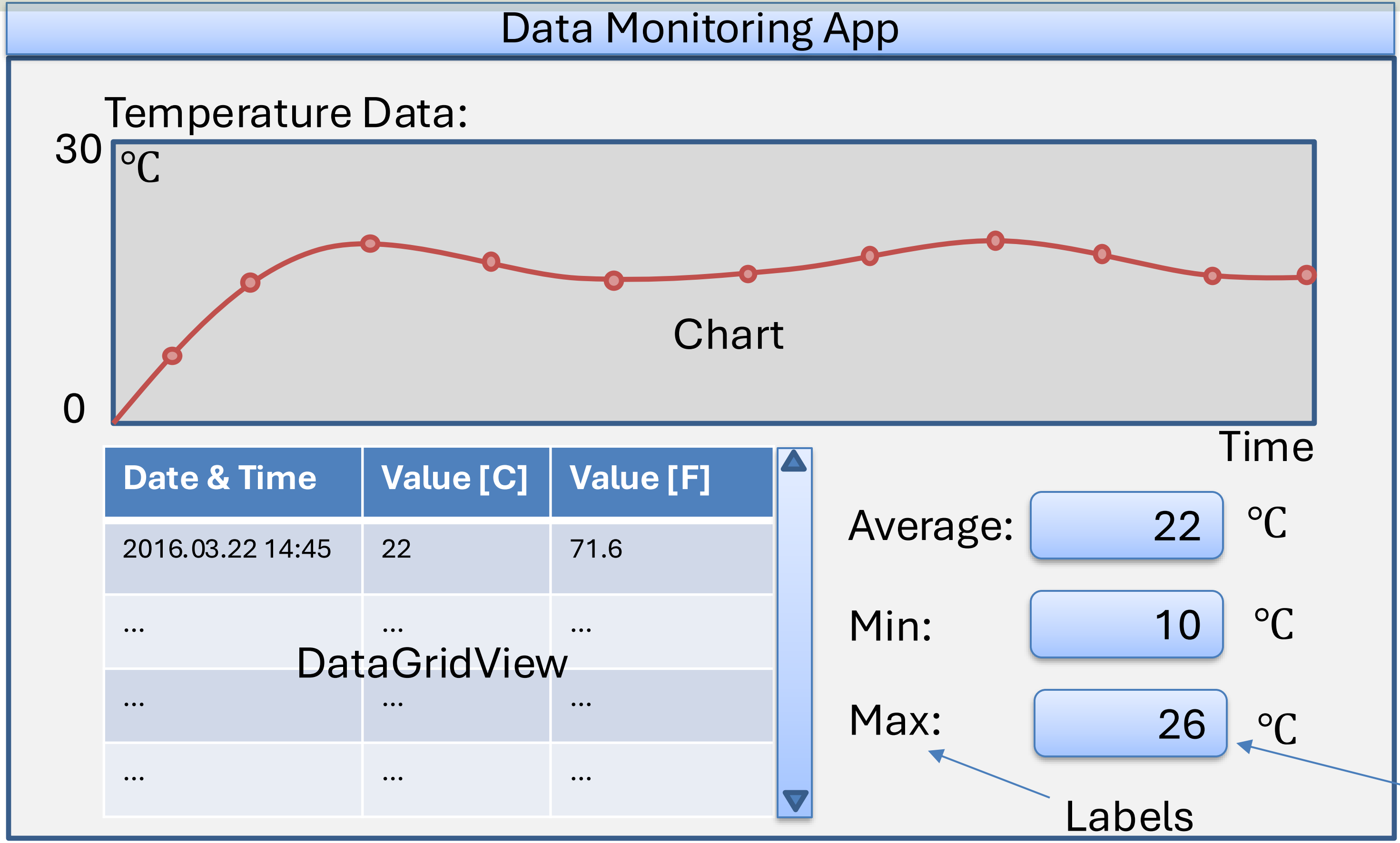
Table of Contents

Data Monitoring using Visual Studio/C#



The Data Monitoring App is typically a Desktop Application (Windows Forms App) or a Web Application (ASP.NET Core App)

Data Monitoring Example



Date & Time	Value [C]	Value [F]
2016.03.22 14:45	22	71.6
...	...	...
...	...	...
...	...	...

Average: 22 °C

Min: 10 °C

Max: 26 °C

Labels

TextBoxes

You should get the Data from the Database

Typically, you get Data from the Database using Views and/or Stored Procedures

Data Monitoring Application

Example of different Alternatives:

1. **Windows Form Desktop Application**
   - This is the “safe” choice and the recommended choice for most of you
2. **ASP.NET Core Web Application**
   - This is the “future” - for those who wants to learn something new and add an extra challenge.

Windows Forms Desktop Application

Hans-Petter Halvorsen

Table of Contents

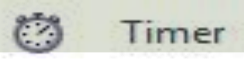
C# Database Example

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;

namespace MonitoringApp.Classes
{
    public class SensorData
    {
        public int SensorDataId { get; set; }
        public double SensorValue { get; set; }
        public DateTime SensorDateTime { get; set; }

        public List<SensorData> GetSensorData()
        {

            string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

            List<SensorData> sensorDataList = new List<SensorData>();

            SqlConnection con = new SqlConnection(connectionString);

            string selectSQL = "select SensorDataId, SensorValue, SensorDateTime from GetSensorData where SensorName ='TC-01'";

            con.Open();
            SqlCommand cmd = new SqlCommand(selectSQL, con);
            SqlDataReader dr = cmd.ExecuteReader();

            if (dr != null)
            {
                while (dr.Read())
                {
                    SensorData sensorData = new SensorData();

                    sensorData.SensorDataId = Convert.ToInt32(dr["SensorDataId"]);
                    sensorData.SensorValue = Convert.ToDouble(dr["SensorValue"]);
                    sensorData.SensorDateTime = Convert.ToDateTime(dr["SensorDateTime"]);

                    sensorDataList.Add(sensorData);
                }
            }
            con.Close();
            return sensorDataList;
        }
    }
}
```

This example retrieves data from a specific sensor

Timer

1



Select the “Timer” component in the Toolbox

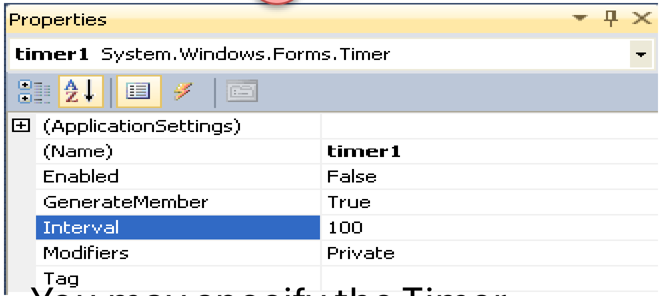
2

Initialization:

```
public Form1()
     {
        InitializeComponent();

        timer1.Start();
     }
```

Double-click on the Timer object to create the Event

Properties: 3


Properties

timer1 System.Windows.Forms.Timer

⊞ (ApplicationSettings)	
(Name)	**timer1**
Enabled	False
GenerateMember	True
Interval	100
Modifiers	Private
Tag	

You may specify the Timer Interval in the Properties Window

4

Timer Event:

```
private void timer1_Tick(object sender, EventArgs e)
     {
       ... //Read from DB
       ... //Formatting
       ... //Plot Data
     }
```

Structure your Code properly!! Define Classes and Methods which you can use here

In Visual Studio you may want to use a Timer instead of a While Loop in order to read values at specific intervals.

ASP.NET Core Web Application

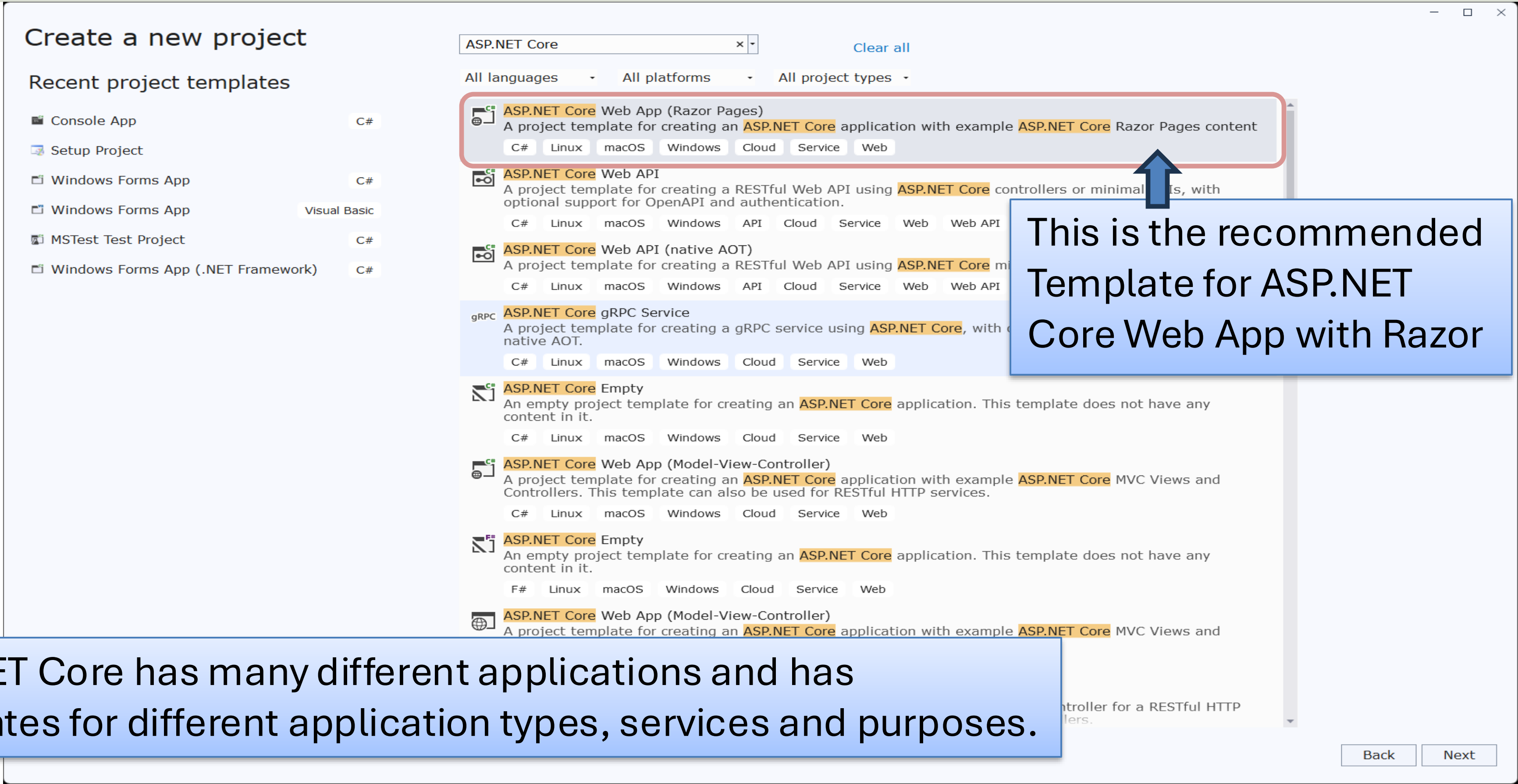
Hans-Petter Halvorsen

Table of Contents

ASP.NET Core

- ASP.NET Core is a framework for web development.
- ASP.NET Core is based on .NET and C#.
- What is the difference between ASP.NET Core and .NET frameworks?
  - ASP.NET Core is specifically designed for web development, while the .NET framework covers a broader range of application types, including Windows desktop, mobile, and web applications.
- In ASP.NET Core Razor code and layout are separated into 2 files; The layout file has the extension “. cshtml”, and the code-behind file has the extension “. cshtml.cs” (where “cs” is short for C#).
- The layout files “. cshtml” use something called Razor syntax and are mixed with HTML.
- ASP, ASP.NET and ASP.NET Core is made by Microsoft.
- Homepage: https://dotnet.microsoft.com/en-us/apps/aspnet

ASP.NET Core Web App with Razor



Create a new project

Recent project templates

- Console App — C#
- Setup Project
- Windows Forms App — C#
- Windows Forms App — Visual Basic
- MSTest Test Project — C#
- Windows Forms App (.NET Framework) — C#

ASP.NET Core ×

Clear all

All languages · All platforms · All project types

ASP.NET Core Web App (Razor Pages)
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Web API
A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal Is, with optional support for OpenAPI and authentication.
C# · Linux · macOS · Windows · API · Cloud · Service · Web · Web API

ASP.NET Core Web API (native AOT)
A project template for creating a RESTful Web API using ASP.NET Core mi
C# · Linux · macOS · Windows · API · Cloud · Service · Web · Web API

ASP.NET Core gRPC Service
A project template for creating a gRPC service using ASP.NET Core, with native AOT.
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
F# · Linux · macOS · Windows · Cloud · Service · Web

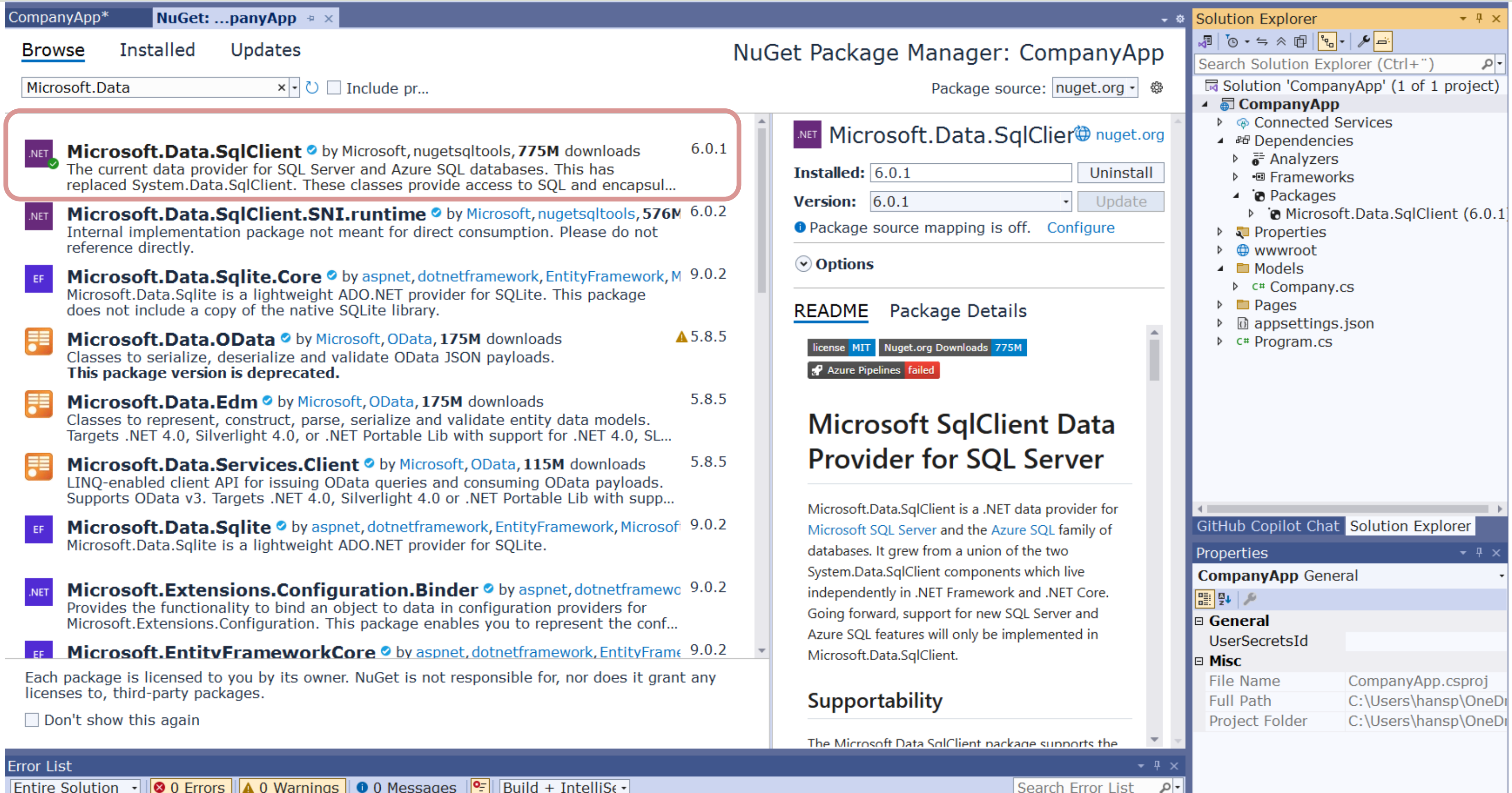
ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and

ntroller for a RESTful HTTP
lers.

This is the recommended Template for ASP.NET Core Web App with Razor

ASP.NET Core has many different applications and has templates for different application types, services and purposes.

NuGet – Database Communication



CompanyApp* | NuGet: ...panyApp

Browse | Installed | Updates

NuGet Package Manager: CompanyApp

Microsoft.Data ☐ Include pr...

Package source: nuget.org

Microsoft.Data.SqlClient by Microsoft, nugetsqltools, **775M** downloads 6.0.1
The current data provider for SQL Server and Azure SQL databases. This has replaced System.Data.SqlClient. These classes provide access to SQL and encapsul...

Microsoft.Data.SqlClient.SNI.runtime by Microsoft, nugetsqltools, **576M** 6.0.2
Internal implementation package not meant for direct consumption. Please do not reference directly.

Microsoft.Data.Sqlite.Core by aspnet, dotnetframework, EntityFramework, M 9.0.2
Microsoft.Data.Sqlite is a lightweight ADO.NET provider for SQLite. This package does not include a copy of the native SQLite library.

Microsoft.Data.OData by Microsoft, OData, **175M** downloads ⚠5.8.5
Classes to serialize, deserialize and validate OData JSON payloads.
This package version is deprecated.

Microsoft.Data.Edm by Microsoft, OData, **175M** downloads 5.8.5
Classes to represent, construct, parse, serialize and validate entity data models. Targets .NET 4.0, Silverlight 4.0, or .NET Portable Lib with support for .NET 4.0, SL...

Microsoft.Data.Services.Client by Microsoft, OData, **115M** downloads 5.8.5
LINQ-enabled client API for issuing OData queries and consuming OData payloads. Supports OData v3. Targets .NET 4.0, Silverlight 4.0 or .NET Portable Lib with supp...

Microsoft.Data.Sqlite by aspnet, dotnetframework, EntityFramework, Microsof 9.0.2
Microsoft.Data.Sqlite is a lightweight ADO.NET provider for SQLite.

Microsoft.Extensions.Configuration.Binder by aspnet, dotnetframewc 9.0.2
Provides the functionality to bind an object to data in configuration providers for Microsoft.Extensions.Configuration. This package enables you to represent the conf...

Microsoft.EntityFrameworkCore by aspnet, dotnetframework, EntityFrame 9.0.2

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Don't show this again

Microsoft.Data.SqlClier nuget.org

Installed: 6.0.1 Uninstall

Version: 6.0.1 Update

Package source mapping is off. Configure

Options

README | Package Details

license MIT | Nuget.org Downloads 775M | Azure Pipelines failed

Microsoft SqlClient Data Provider for SQL Server

Microsoft.Data.SqlClient is a .NET data provider for Microsoft SQL Server and the Azure SQL family of databases. It grew from a union of the two System.Data.SqlClient components which live independently in .NET Framework and .NET Core. Going forward, support for new SQL Server and Azure SQL features will only be implemented in Microsoft.Data.SqlClient.

Supportability

The Microsoft.Data.SqlClient package supports the

Solution Explorer

Search Solution Explorer (Ctrl+¨)

Solution 'CompanyApp' (1 of 1 project)
- **CompanyApp**
  - Connected Services
  - Dependencies
    - Analyzers
    - Frameworks
    - Packages
      - Microsoft.Data.SqlClient (6.0.1
  - Properties
  - wwwroot
  - Models
    - Company.cs
  - Pages
  - appsettings.json
  - Program.cs

GitHub Copilot Chat | Solution Explorer

Properties

CompanyApp General

General	
UserSecretsId	
Misc	
File Name	CompanyApp.csproj
Full Path	C:\Users\hansp\OneDı
Project Folder	C:\Users\hansp\OneDı

Error List

Entire Solution | 0 Errors | 0 Warnings | 0 Messages | Build + IntelliSe | Search Error List

This example retrieves data from an SQL Server Database

```
using Microsoft.Data.SqlClient;

namespace CompanyApp.Models
{
    public class Company
    {
        public int companyId { get; set; }
        public string? companyName { get; set; }
        public string? webSite { get; set; }

        public List<Company> GetCompanies()
        {
            string connectionString = "Data Source=SERVERNAME\\SQLEXPRESS;Initial Catalog=WORK;Integrated
               Security=True;TrustServerCertificate=True";
            SqlConnection con = new SqlConnection(connectionString);
            con.Open();

            string sqlQuery = "select CompanyId, CompanyName, WebSite from COMPANY";

            SqlCommand cmd = new SqlCommand(sqlQuery, con);

            SqlDataReader dr = cmd.ExecuteReader();

            List<Company> compamyList = new List<Company>();

            while (dr.Read())
            {
                Company company = new Company();

                company.companyId = Convert.ToInt32(dr["CompanyId"]);
                company.companyName = dr["CompanyName"].ToString();
                company.webSite = dr["WebSite"].ToString();

                compamyList.Add(company);
            }
            con.Close();
            return compamyList;
        }
    }
}
```

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog