https://www.halvorsen.blog

Datalogging and Monitoring System

Hans-Petter Halvorsen

Background

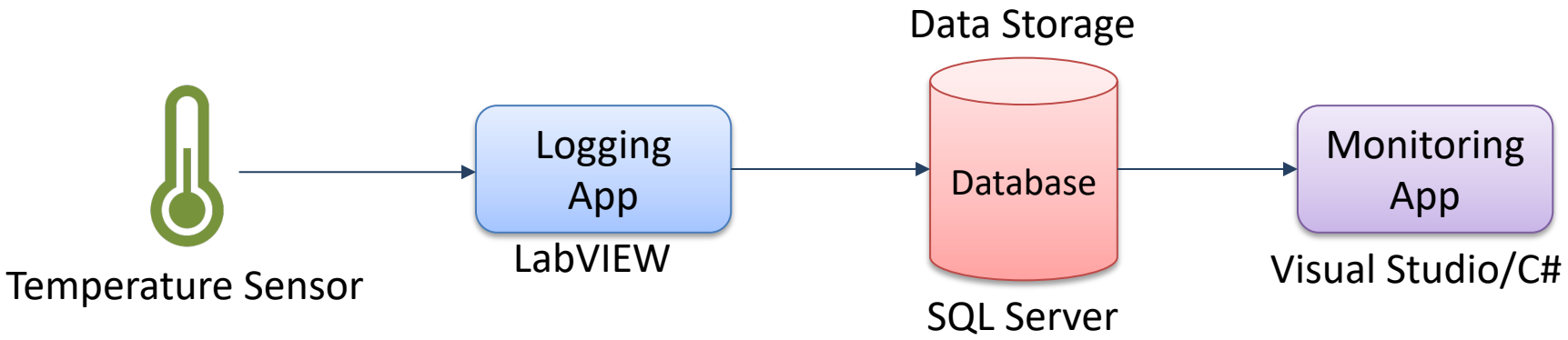
- You work as a **System Engineer** for a System Engineering Company.
- An Industrial Production Company has announced a competition between several selected System Engineering companies to perform a preliminary Project.
- **Your Assignment is to develop a Datalogging and Monitoring System** Prototype/PoC.
- The system should consist of a **SQL Server Database**, a **Datalogging Application** and a **Data Monitoring Application**.
- You need to design a **general and flexible Database structure** that is suitable for the system.
- To create proper and user-friendly **GUI/HMI** is an important part of the Prototype.
- **The delivery is a Technical Report** where you shall give an overview of the entire system made, including the Methods used and the Results archived.
- The PoC and the Report will be an important foundation for decision making within the company when it comes to the final implementation of the system sometime in the future. Note! Multiple System Engineering companies have been given this opportunity, so it is important that you "**Add Value**" and stand out compared to the others in order to be selected as the final Contractor.

System Requirements

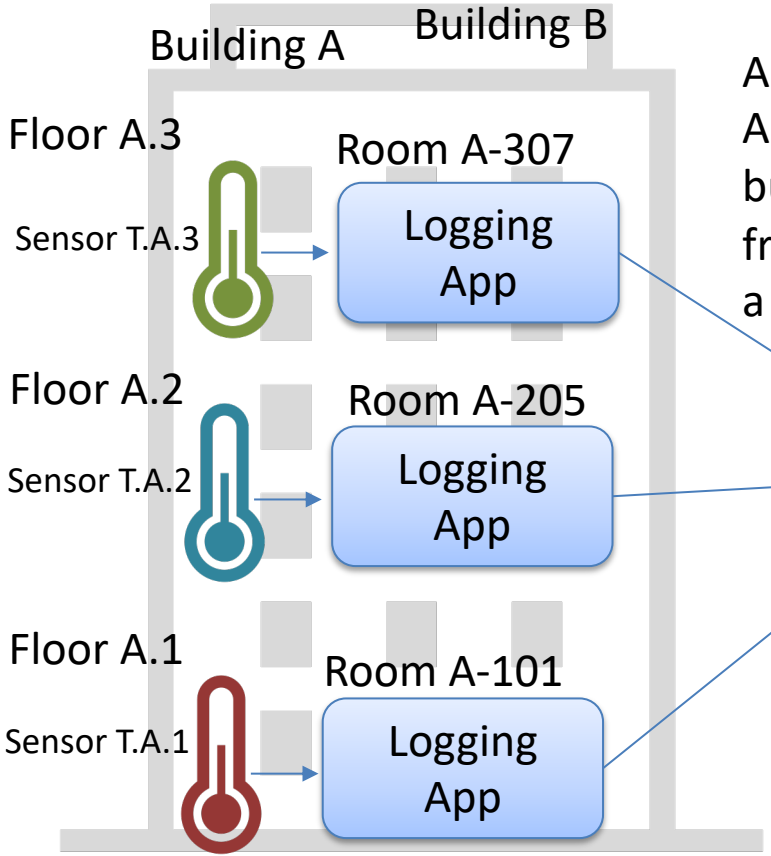
- Design the Database using **erwin Data Modeler**.
  - The Table structure need to be flexible, and it should be possible to store data from different Temperature Sensors and Statistics Data. At least 3-5 Database Tables are required.
- Implement Tables, Views, Stored Procedures and Triggers using **SQL Server**.
  - Triggers should be made to convert from Celsius to Fahrenheit and to store Statistics Data in the Database like Mean, Standard Deviation, Max and Min
- Create a Datalogging Application using **LabVIEW**.
  - The Application should read data from the Temperature Sensor and store the data into the Database. Create proper GUI including showing a Chart, Real-Time Data and Historical Data, etc.
  - The Data should be stored in the Database using a Stored Procedure
- Create a Data Monitoring Application using **Visual Studio/C#**.
  - The Application should read data from the Database and present them in a user-friendly and intuitive way with real-time data, historical data and statistics data. Create and use at least 1 Database View.
  - You can choose between a standard Windows Forms Desktop Application or ASP.NET Core Web Application

These are the complete requirements for the assignment. The rest of this document contains resources like additional information, code examples, tips and tricks, step-by step instructions, etc. that you can use at your own discretion.

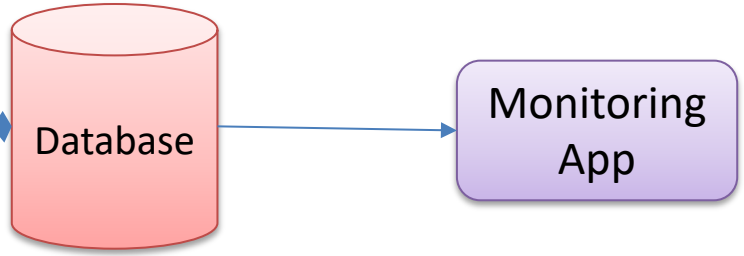
Basic System Overview



Use Case Scenario (Alt1)



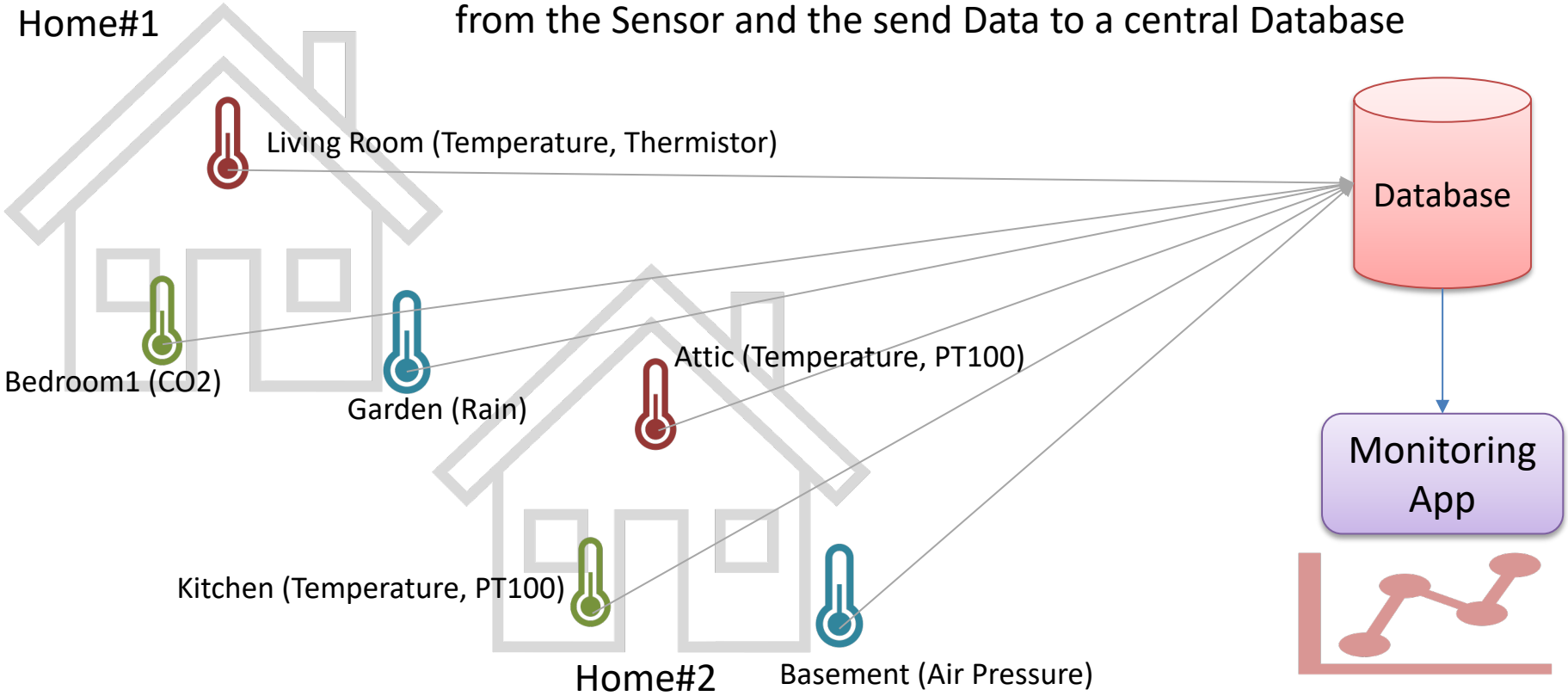
Assume that you can have multiple Data Logging Applications that are in different places inside multiple buildings which are logging Temperature Data (or Data from other Sensors, CO2, etc.) and store the Data inside a common Database



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



Additional Resources

Hans-Petter Halvorsen

Table of Contents

Table of Contents

Introduction

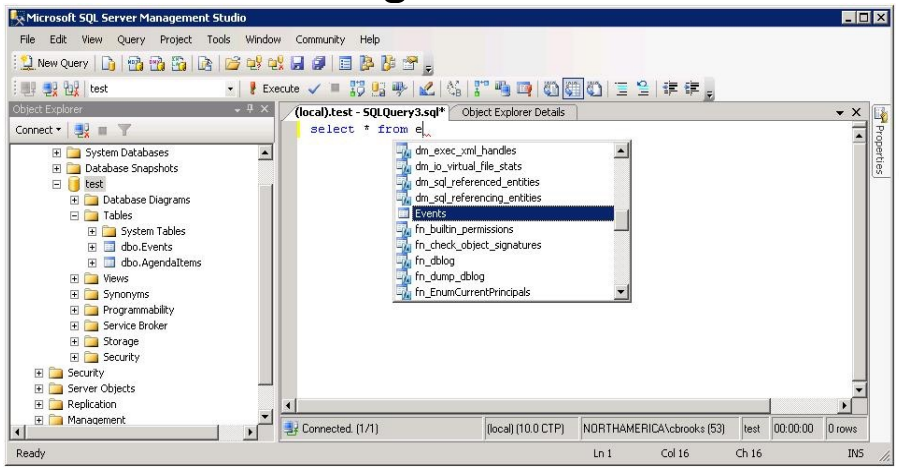
Hans-Petter Halvorsen

Table of Contents

Lab Overview

erwin Database Design & Modelling

SQL Server Management Studio



Create Tables

Database Management

Write/Read Data

Microsoft **SQL Server** Database

NATIONAL INSTRUMENTS LabVIEW

Visual Studio

We will create Applications in LabVIEW & Visual Studio that Write and Read data to/from the Database

Software

erwin

NATIONAL INSTRUMENTS LabVIEW

Visual Studio

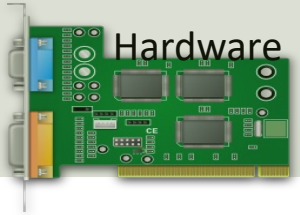
- **erwin Data Modeler** (Academic Edition, free download from Internet)
- **SQL Server** Express Edition (Download for free from Internet)
- LabVIEW
- DAQmx Driver Software
- **LabVIEW SQL Toolkit** (Free download)
- Visual Studio

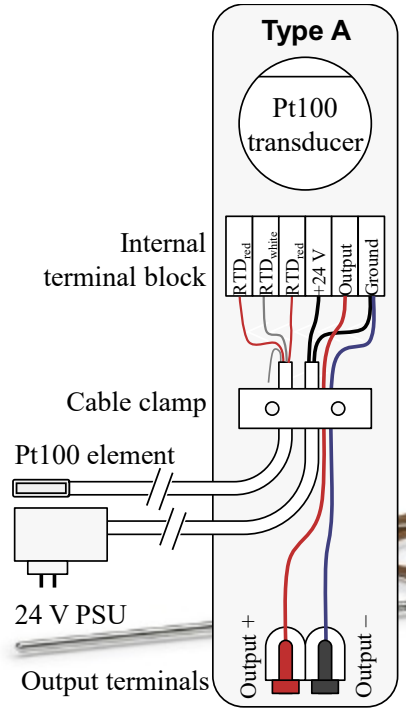
Microsoft® SQL Server®

Make sure to install the necessary Software before you go to the laboratory!

Greater Challenge? Make it possible to change between different Temperature Sensors or use multiple Sensors

Hardware





PT100 Temperature Sensor Device

USB-6008 I/O Module

TMP36

Thermistor

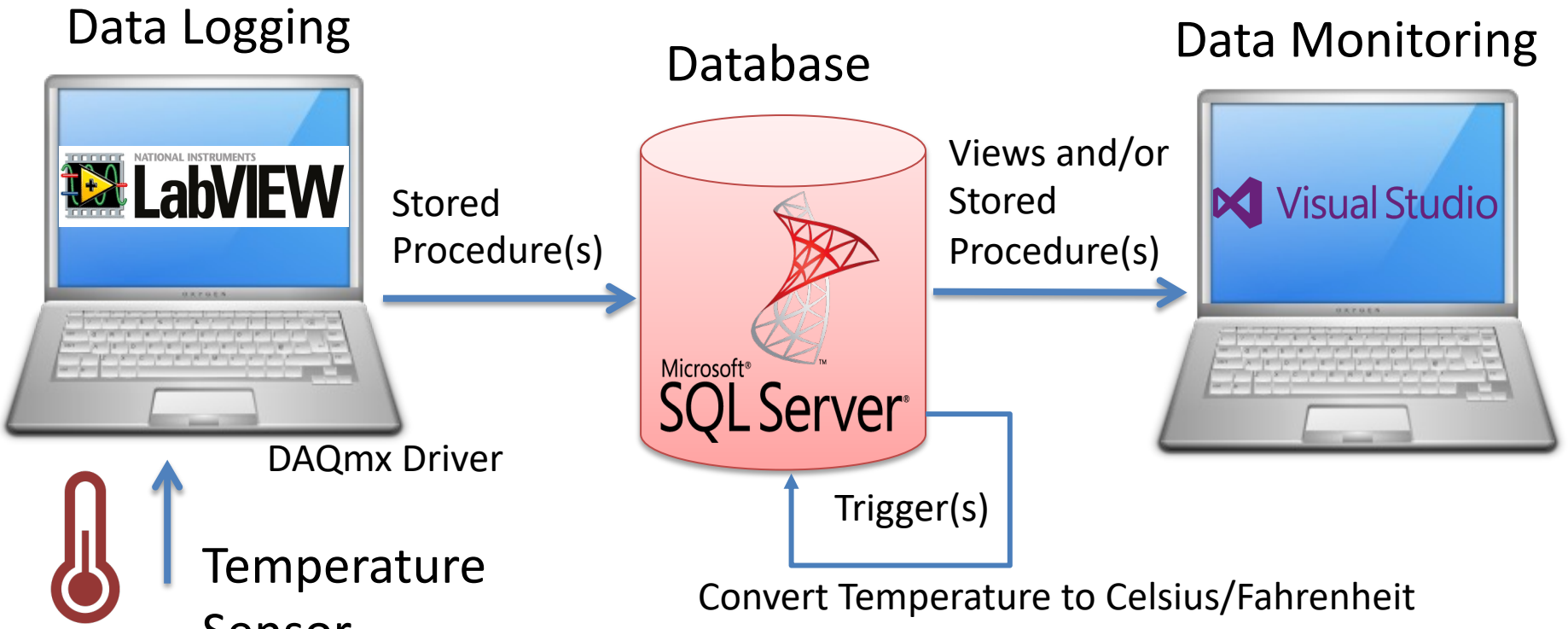
PC

TC-01 Thermocouple

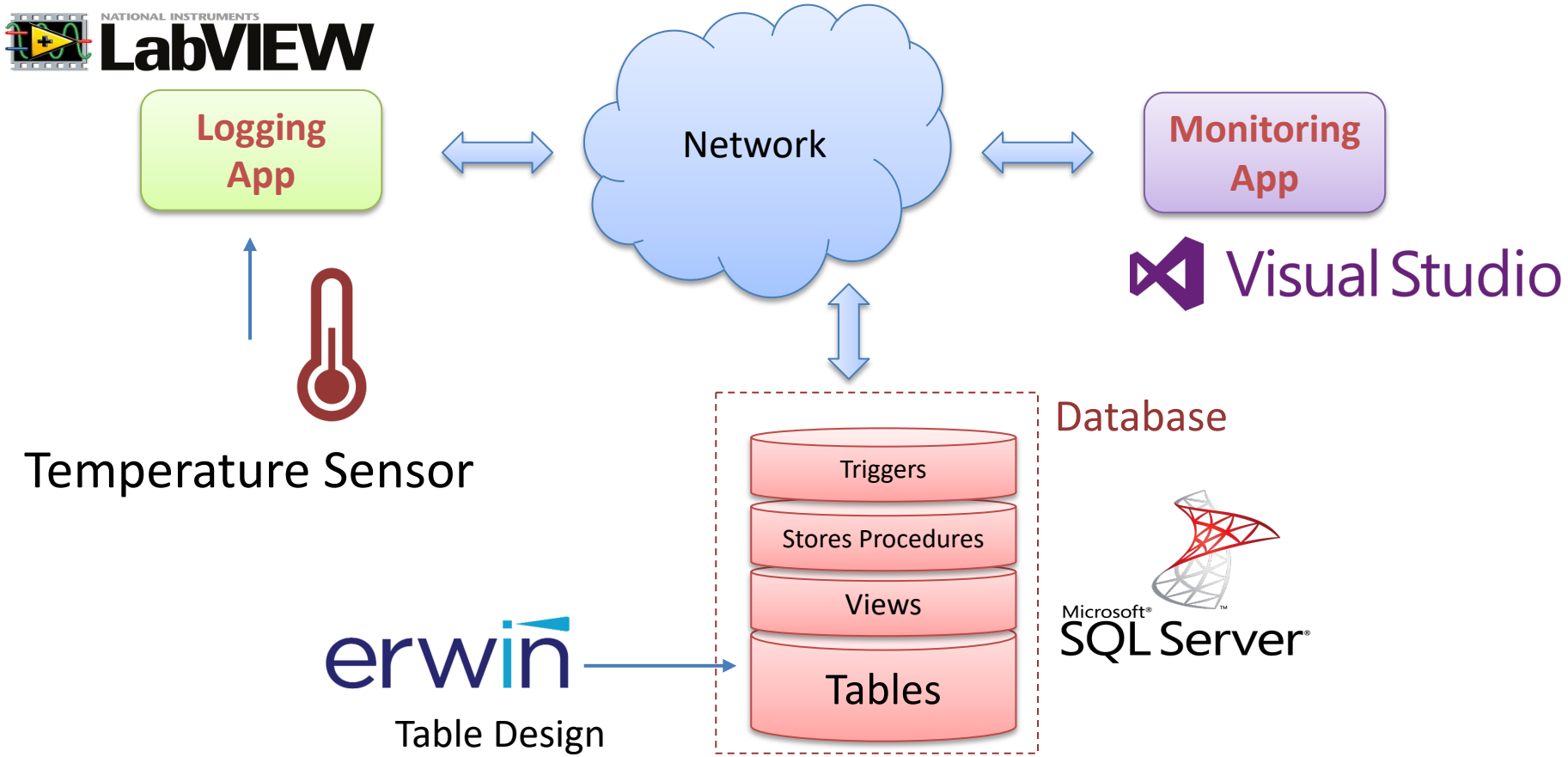
We will use a TC-01 Thermocouple Temperature Sensor, TMP36, Thermistor and PT-100 Temperature Sensors and a USB-6008 I/O Module

Note! If you don't have a Temperature Sensor available, you can create and use a Simulator

System Overview



System Overview



Learning Goals

- Learn key concepts within **Database Systems**
- Learn **Database Modelling**
- Learn **Structured Query Language (SQL)**
- Learn practical skills and **implementation of Database Systems**
- Learn more Programming (LabVIEW, C#)
- Learn about Hardware-Software Interactions
- Learn Practical Skills and Implementations in general
- Learn Software Installation in general, which can be cumbersome with many pitfalls
- Learn to use and create Software in general
- Learn to Design and Develop Software needed by a given client
- Problem Solving: Learn to solve unexcepted Problems during Development of a given System

The teacher have not done all the Tasks in detail, so he may not have all the answers! That's how it is in real life also!

Very often it works on one computer but not on another. You may have other versions of the software, you may have installed it in the wrong order, etc...
In these cases Google is your best friend!

The Teacher dont have all the answers (very few actually ☹)!! Sometimes you just need to “Google” in order to solve your problems, Collaborate with other Students, etc. Thats how you Learn!

Troubleshooting & Debugging

Use the **Debugging Tools** in your Programming IDE.
Visual Studio, LabVIEW, etc. have great Debugging Tools! Use them!!

“Google It”!

You probably will find the answer on the Internet

Another person in the world probably had a similar problem

Use available Resources such as User Guides, Datasheets, Textbooks, Tutorials, Examples, Tips & Tricks, etc.

My System is not Working??

Multimeter, etc.

Use Microsoft Teams

Check your electric circuit, electrical cables, DAQ device, etc. Check if the wires from/to the DAQ device is correct. Are you using the same I/O Channel in your Software as the wiring suggest? etc.

Lab Assignment Guidelines

- Make sure to read the whole assignment before you start to solve any of the problems.
- If you miss assumptions for solving some of the problems, you may define proper assumptions yourself.
- The Tasks described in the Assignment are somewhat loosely defined and more like guidelines, so feel free to interpret the Tasks in your own way with a personalized touch.
- Feel free to **Explore**! Make sure to **Add Value** and **Creativity** to your Applications!
- Try to add some extra value and be creative compared to the simplified examples given by me, in that way you learn so much more.

Lab Assignment Guidelines

- Think about the Lab Assignment as a small real-life industrial Project, and not a set of tasks or exercises.
- What does the company that hire you expect from you when you deliver this project? What kind of Quality is expected?
- Try to see your work in a larger context than just a Lab Assignment or a set of exercises.
- Try to see the big picture. The tasks within the assignment are just just small building blocks that ends up with a fully working system.
- It is recommended that you make a Work Plan and a System Sketch that gives you an overview of what YOU should do

Lab Work Requirements

- Make sure to see the "**Big picture**" – you don't need to document every single step you have made. Focus on what's important (your final system).
- Your GUIs is important! - make sure to make them user friendly and intuitive. You create this on behalf of someone that are going to use your applications.
- Make sure to always add **Units** in your GUI, charts, documentation, etc.
- **Presenting values with 4+ decimals makes no sense!** E.g., a temperature sensor is not that accurate. You can easily change number of decimals that you present in your GUI in LabVIEW, C#, etc.
- The **Quality** of the LabVIEW code is important. Make sure to use "straight lines" in your LabVIEW code, etc. The code should also flow from left to right, not opposite direction. You create this on behalf of someone that are going to use your applications. Neat code makes it easier to develop, maintain, find code errors, etc.
- In general, make sure that you take some pride in your applications and the work that you do. It's not about getting finished as soon as possible. The mission is to learn as much as possible within a given timeframe. Try to change the mindset.
- To improve the LabVIEW code, please see this video: LabVIEW Applications using State Machine: https://youtu.be/-b9St8wNhpQ

https://www.halvorsen.blog

erwin Data Modeler

Database Modelling and Design

Hans-Petter Halvorsen

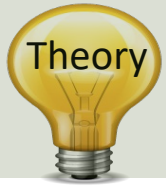
Table of Contents

erwin Data Modeler Academic Edition

- Database Modelling Tool for creating ER (Entity Relationship) diagrams
- Students and teachers of approved accredited academic institutions can apply for the Academic Edition of erwin Data Modeler to use on their personal computers.
- Its free, but you need to apply for it - so it may take 2-5 days before you get it!
- Make sure to do this ASAP
- https://www.erwin.com/register/129709/

erwin

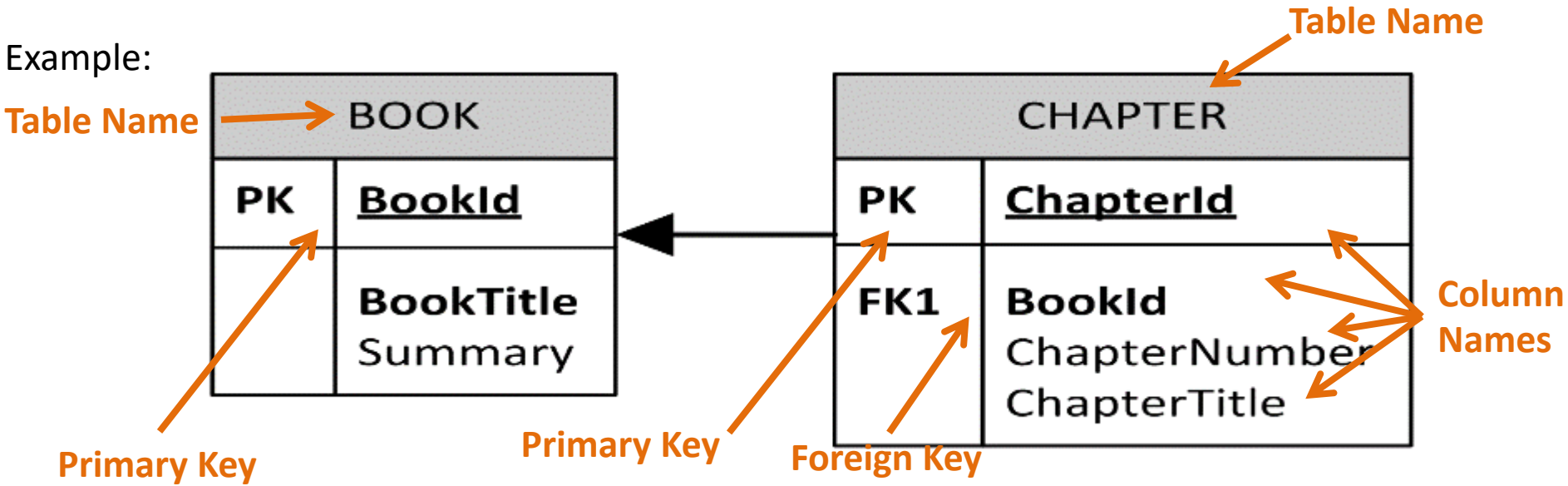
ER Diagram



ER Diagram (Entity-Relationship Diagram)

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

Example:



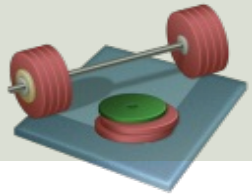
Relational Database. In a relational database all the tables have one or more relation with each other using Primary Keys (PK) and Foreign Keys (FK). Note! You can only have one PK in a table, but you may have several FK's.

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”, ...)

It is strongly recommended that you follow these guidelines!

Database System



- Create the overall Specifications and Design for your System
- Start by Design the Database Tables using ERwin and create a SQL Script
- Implement the Tables in SQL Server, e.g., using a SQL Script generated in ERwin
- 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

1

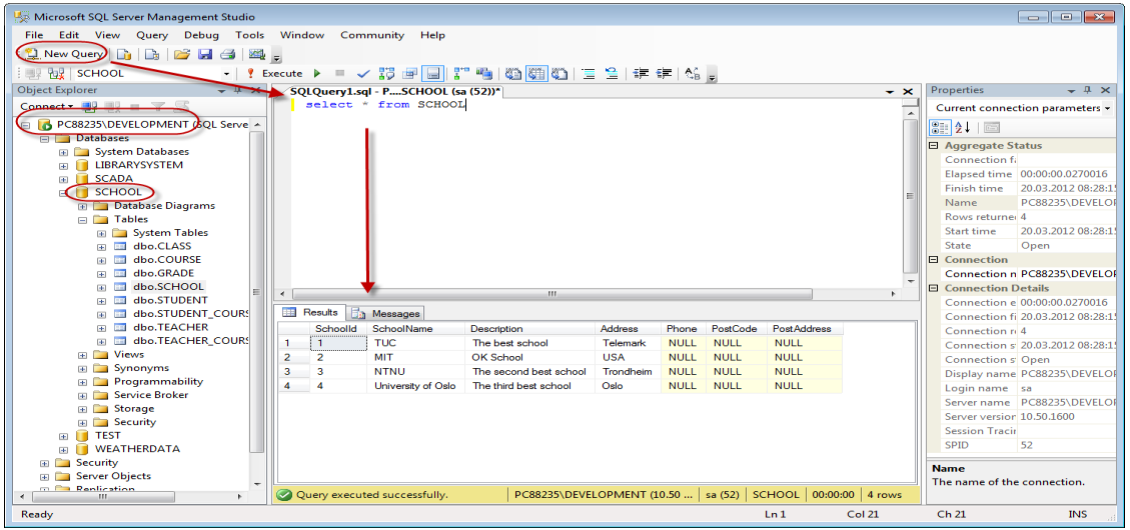
SQL Server Database Engine and Repository

The Data Storage

Note! These are 2 separate modules you need to install

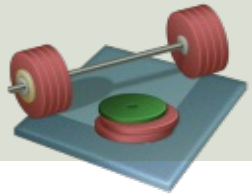
2

SQL Server **Management Studio**



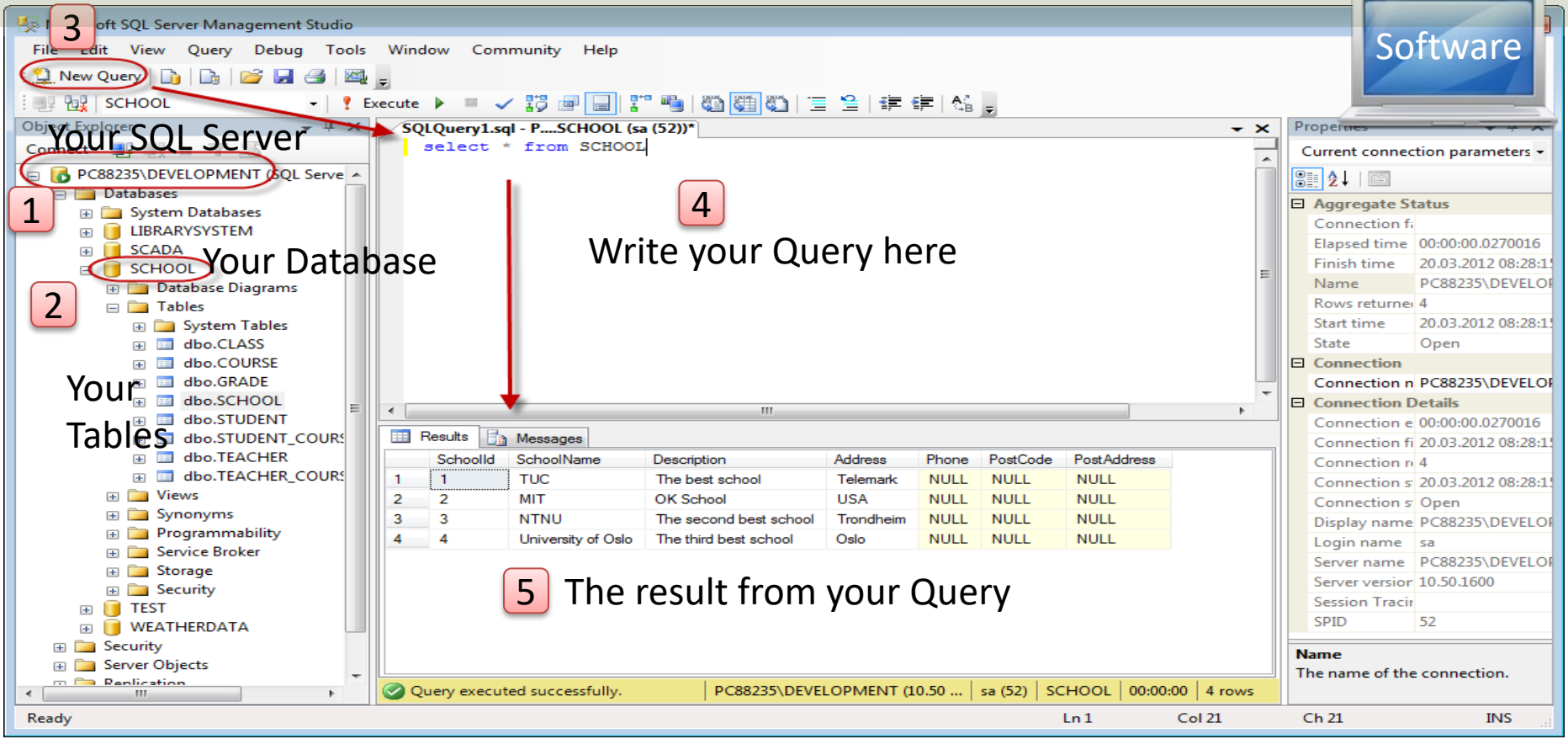
A graphical interface to the Database Engine where you can create tables and manipulate data, etc.

Microsoft SQL Server

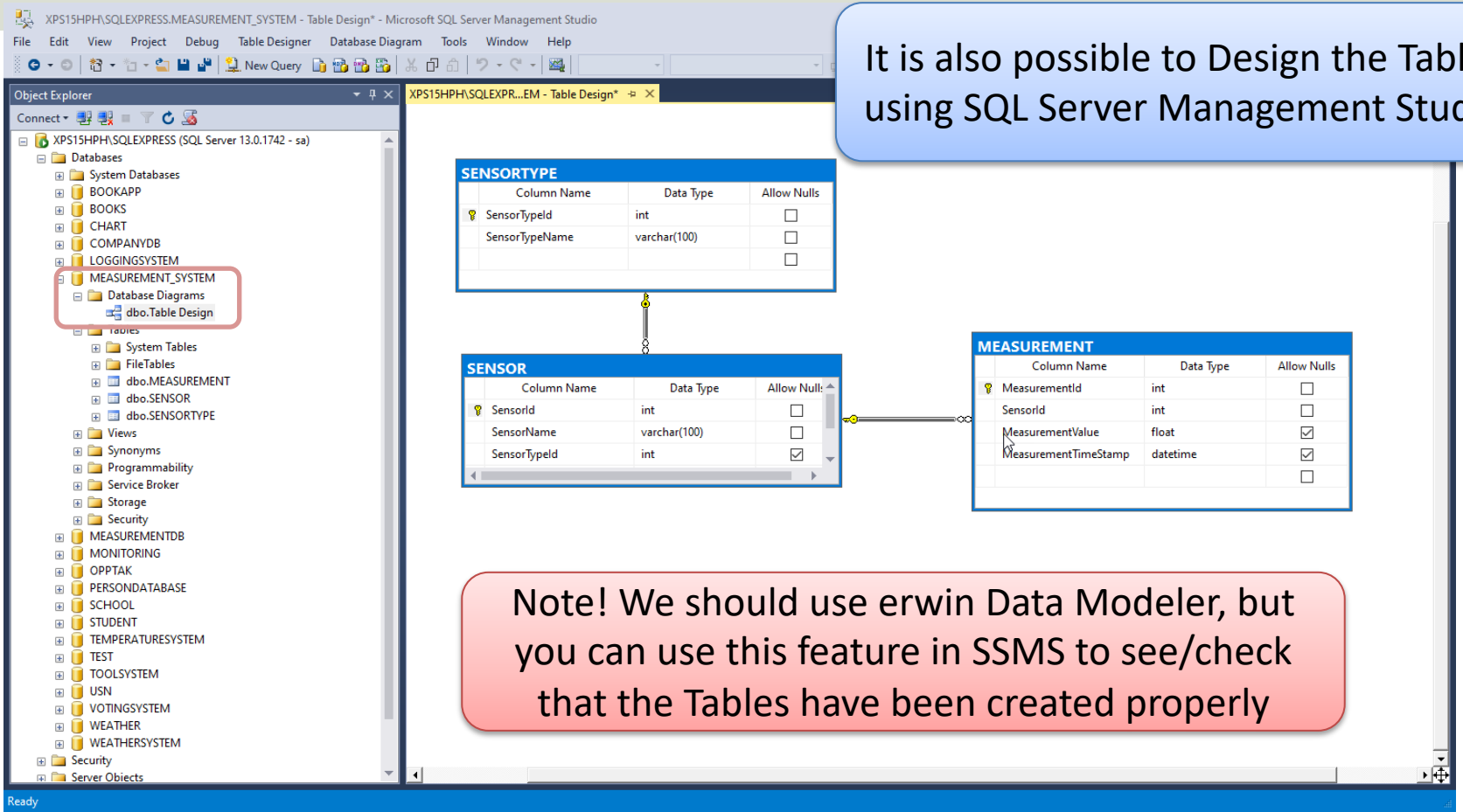


- Start by **Design the Database Tables using ERwin** and create a SQL Script
- **Implement the Tables in SQL Server**, e.g., using a SQL Script generated in ERwin
- **Create necessary Views, Stored Procedures and Triggers** within the SQL Server Management Studio.
  - Put each of them into a .sql file.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

Microsoft SQL Server Management Studio



Database Design in SQL Server Management Studio

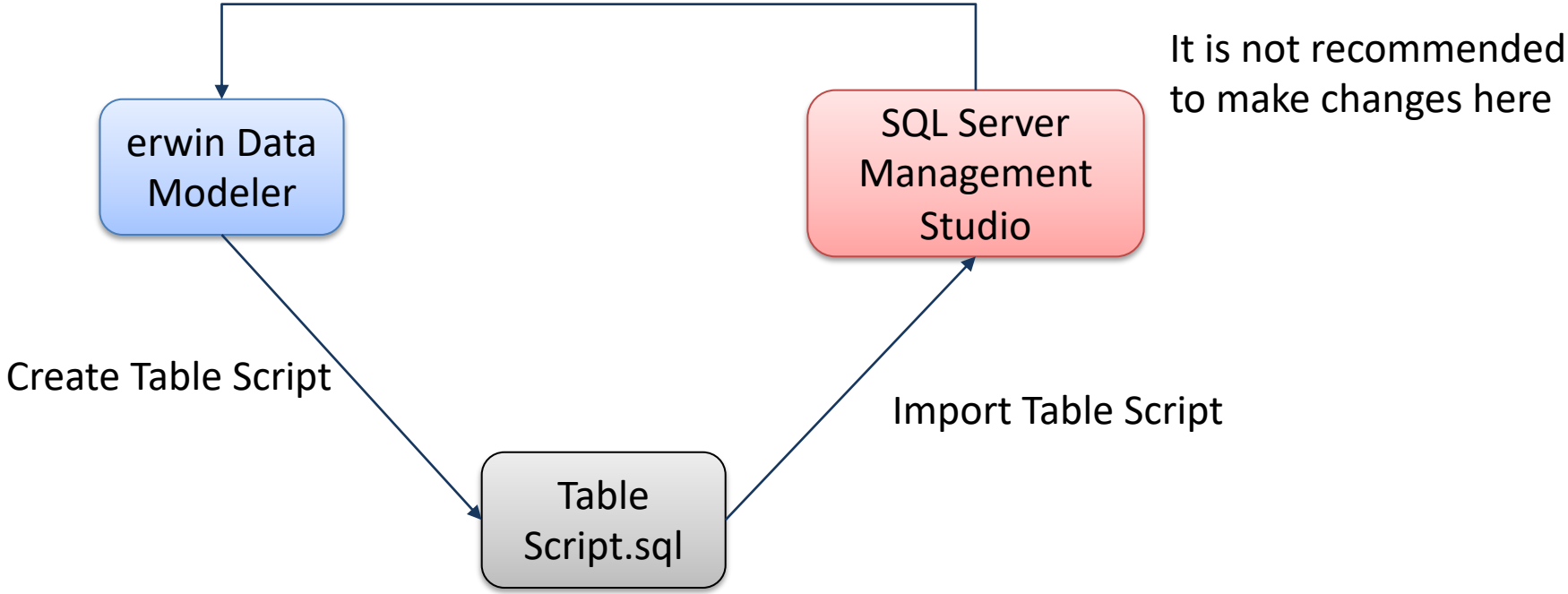


It is also possible to Design the Tables using SQL Server Management Studio

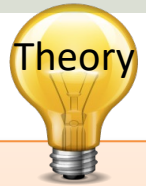
Note! We should use erwin Data Modeler, but you can use this feature in SSMS to see/check that the Tables have been created properly

Database Design and Implementation

Need to make some improvements? Update the Table Design in erwin Data Modeler



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

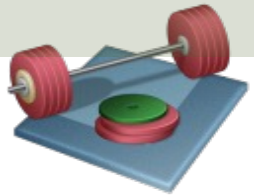
- **delete** from STUDENT **where** SchoolId=3

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 in order 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 multiple 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
```

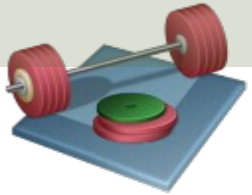
	SchoolId	SchoolName	CourseId	CourseName	Description
1	1	TUC	1	Industrial IT	The best course ever
2	1	TUC	2	Control with Implementation	Control Theory
3	1	TUC	3	Systems and Control Laboratory	Practical Lav course

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 use a Stored Procedure in order to save Measurement Data to the Database.
- The Datalogging App should use a Stored Procedure in order 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

1

Create Stored Procedure:

Stored Procedures


```
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:

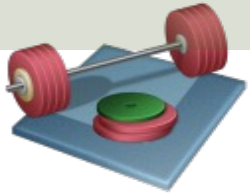
```
execute StudentGrade 'John Wayne', 'SCE2006', 'B'
```

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

```
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

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

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

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.

https://www.halvorsen.blog

Datalogging using LabVIEW

Hans-Petter Halvorsen

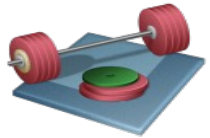
Table of Contents

Datalogging using LabVIEW

Start by Design and Implement the Database Tables using ERwin

erwin

Database Design & Modelling



Create Stored Procedure(s) and Triggers in SQL Server

Temperature Sensor

DAQ

Data

Stored Procedure(s)

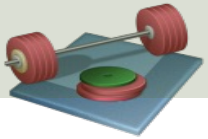
Microsoft SQL Server Database

Triggers

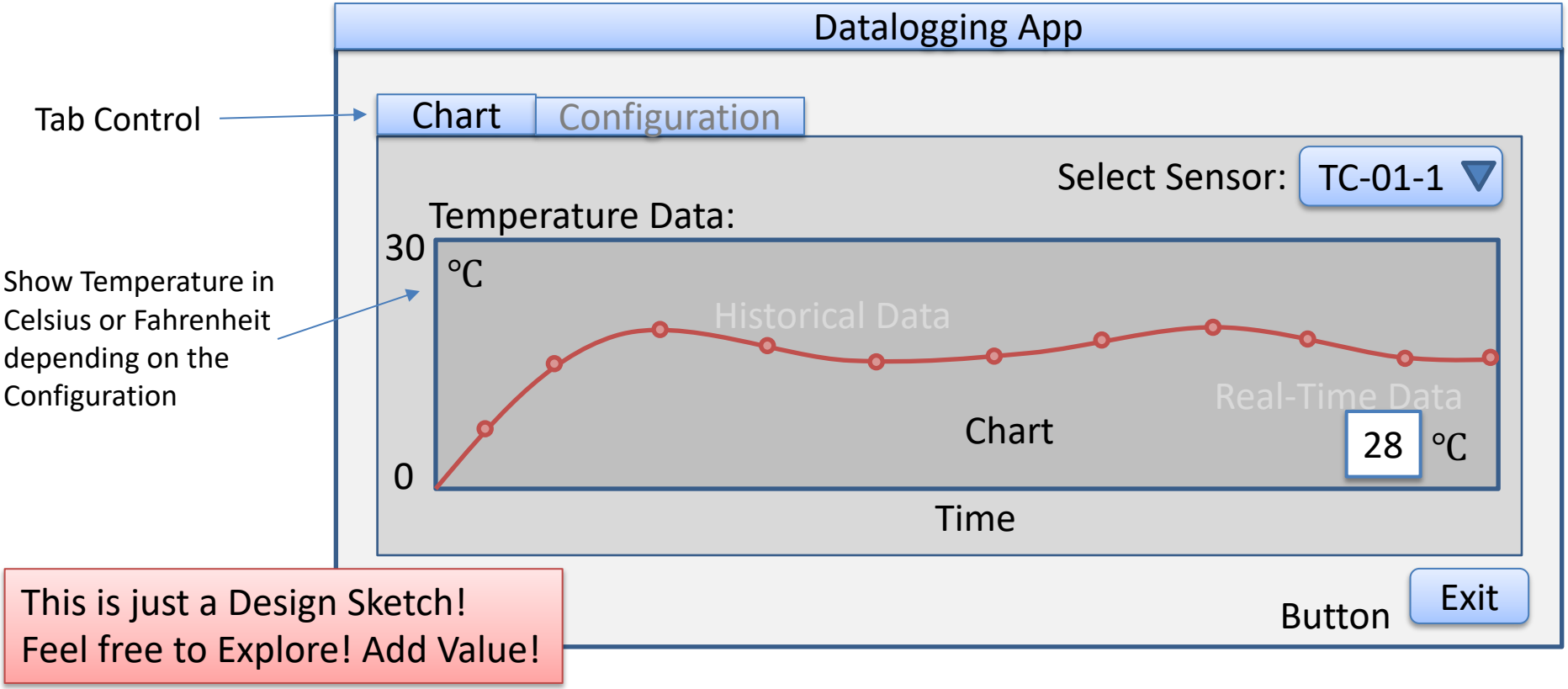
Convert Temperature to Celsius/Fahrenheit

Calculate Average, Max, Min Temperature Data

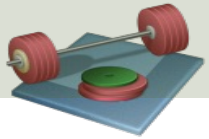
LabVIEW HMI Example



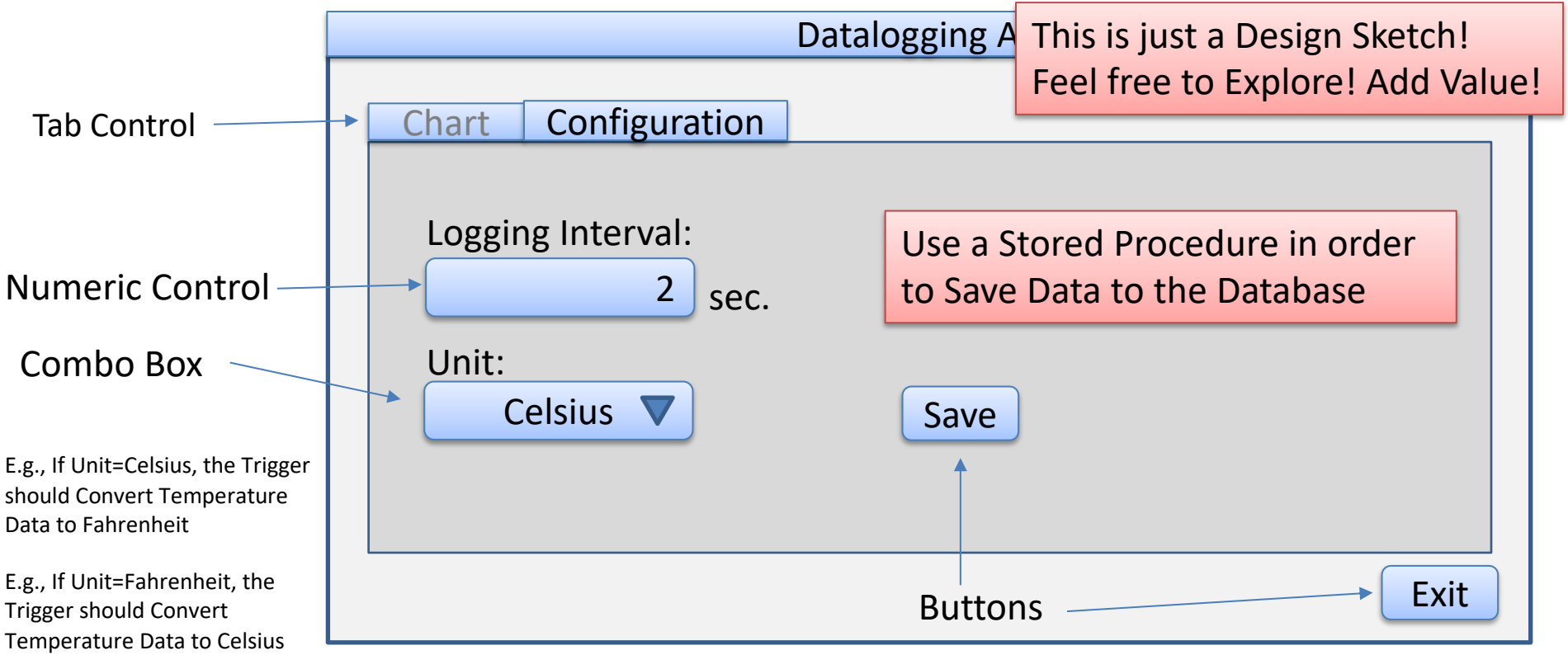
The Temperature Data from the Sensors(s) should be stored in the Database



LabVIEW HMI Example

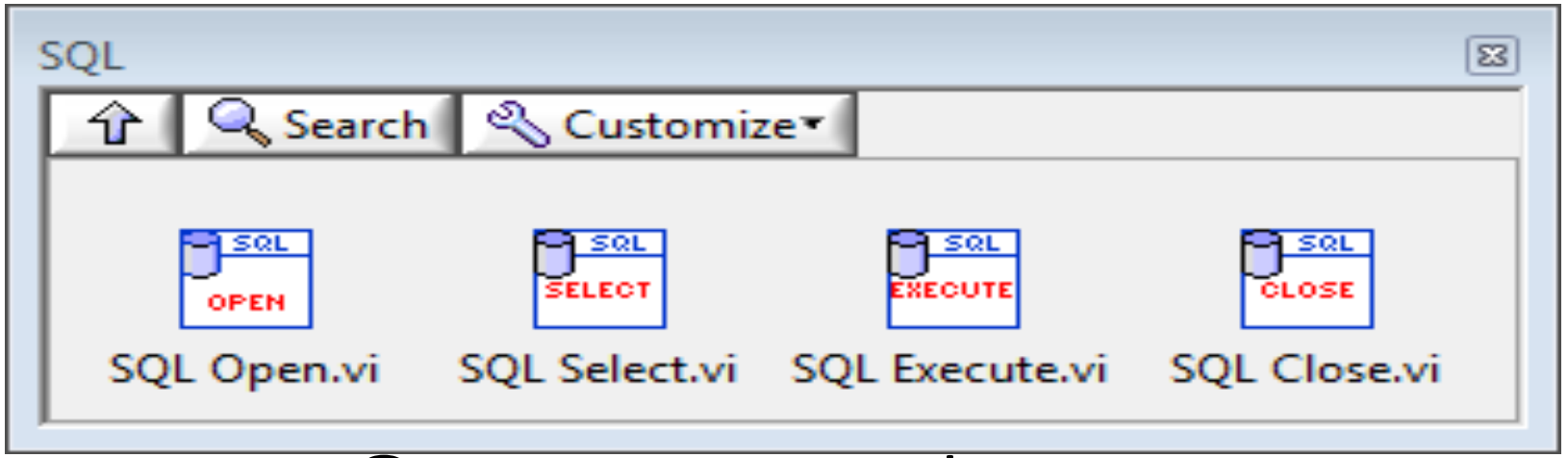


The Temperature Data from the Sensors(s) should be stored in the Database



LabVIEW SQL Toolkit

For Easy Database Communication with LabVIEW



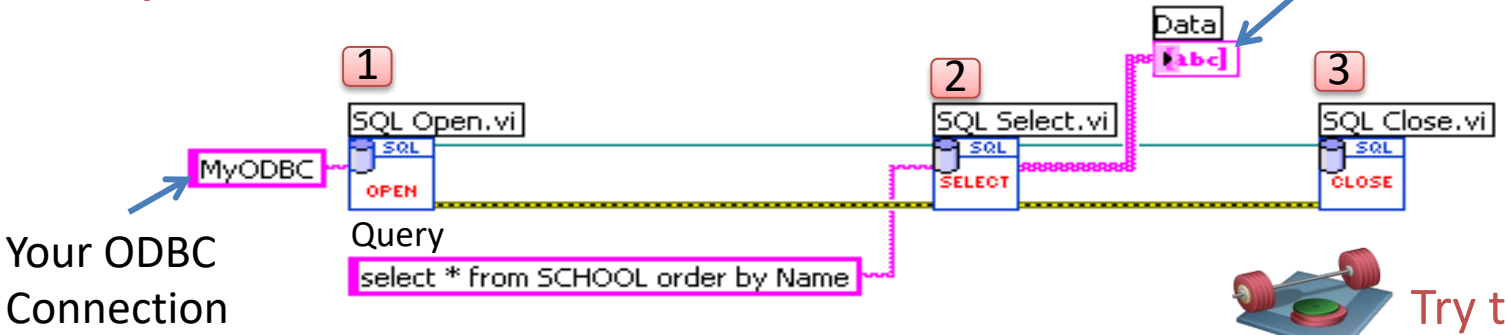
Download for free here:

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

LabVIEW SQL Toolkit Example

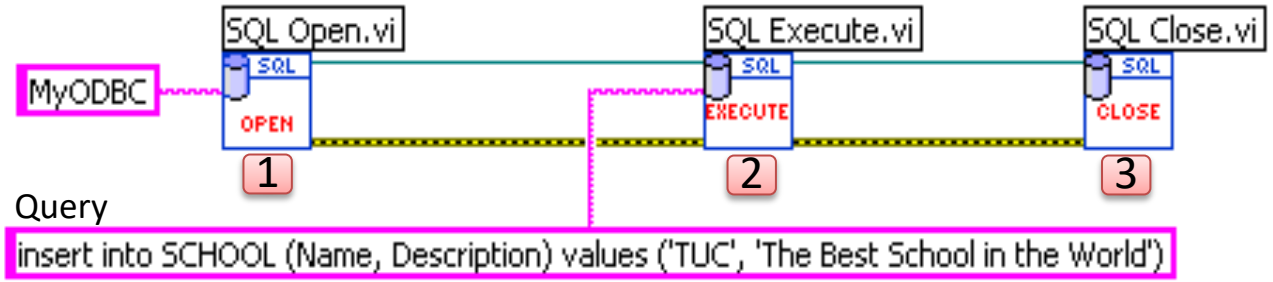
Easy Access to Database Systems from LabVIEW

Example 1: Get Data from Database into LabVIEW:



Try these Examples

Example 2: Write Data to Database from LabVIEW:



Connect to Database

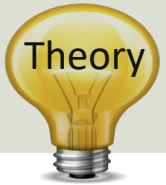
- Alt 1: Use ODBC
  - Setup your Database connection using a Wizard (“ODBC Data Source Administrator”)
- Alt 2: Use Connection String directly
  - Alt 2.2: SQL Server Authentication:

    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

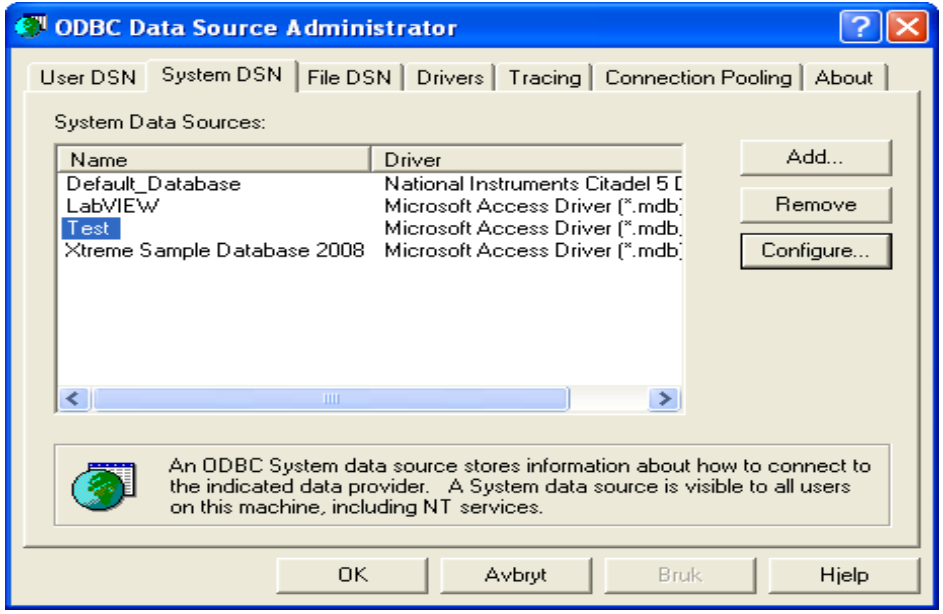
See Examples on next slides...

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.

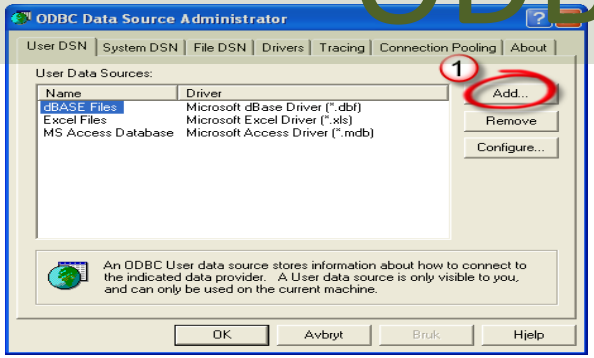
Control Panel → Administrative Tools → Data Sources (ODBC)

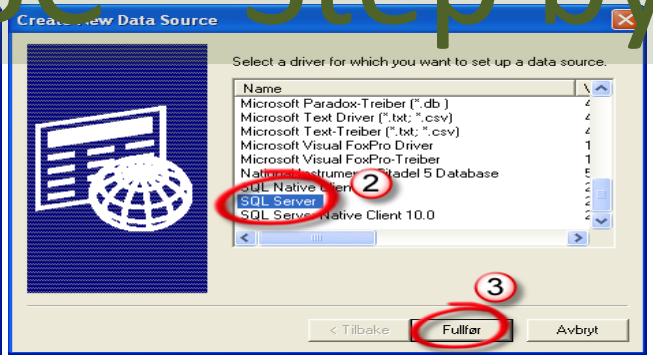


We will use this ODBC Connection later in LabVIEW in order to open the Database Connection from LabVIEW

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

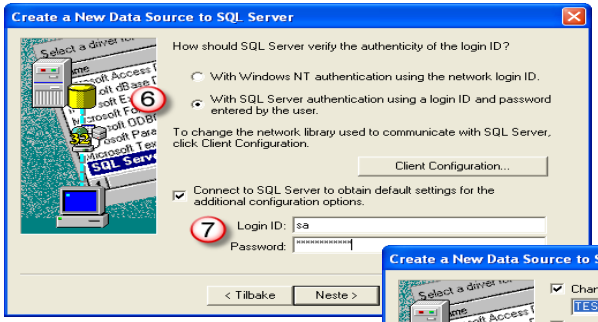
ODBC – Step by Step



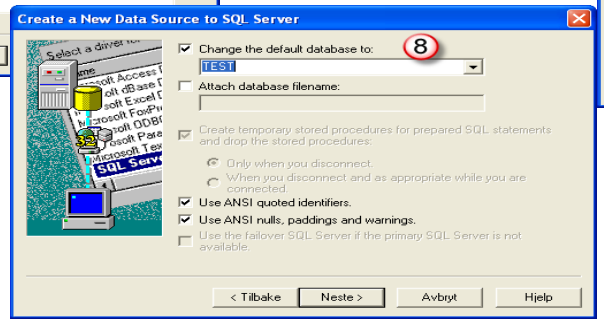


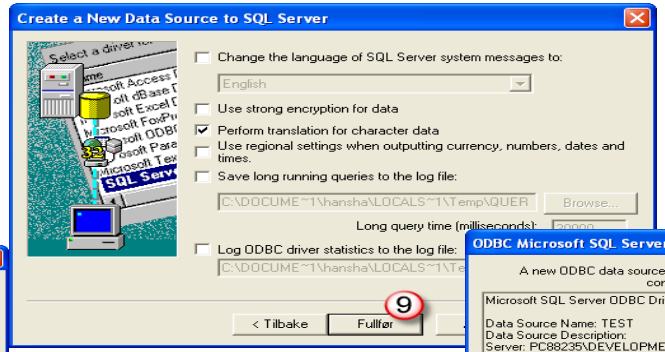
The Name of your ODBC Connection

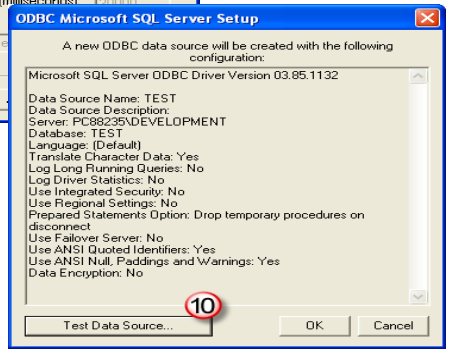
The Name of your SQL Server



Select the Database you are using







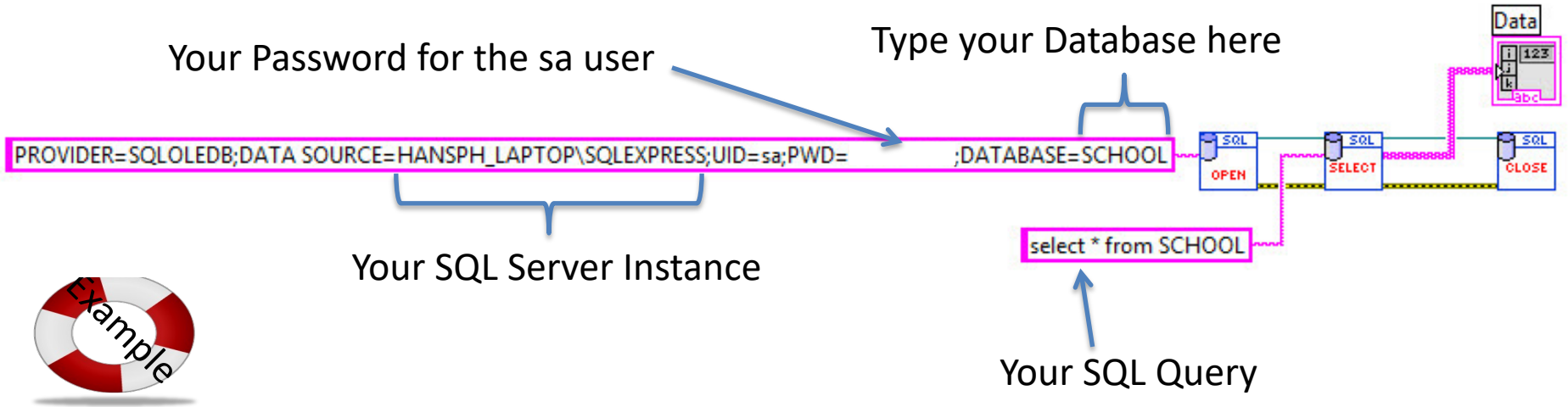
Use either Windows or SQL Server authentication (Windows is simplest to use!)

Test your connection to see if its works

LabVIEW SQL Toolkit Example

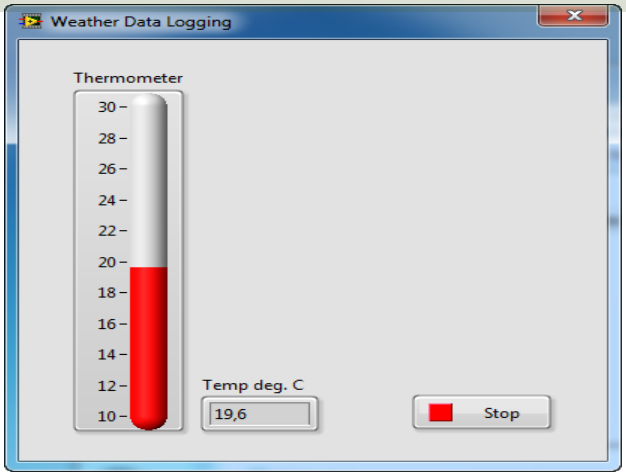
Easy Access to Database Systems from LabVIEW

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



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

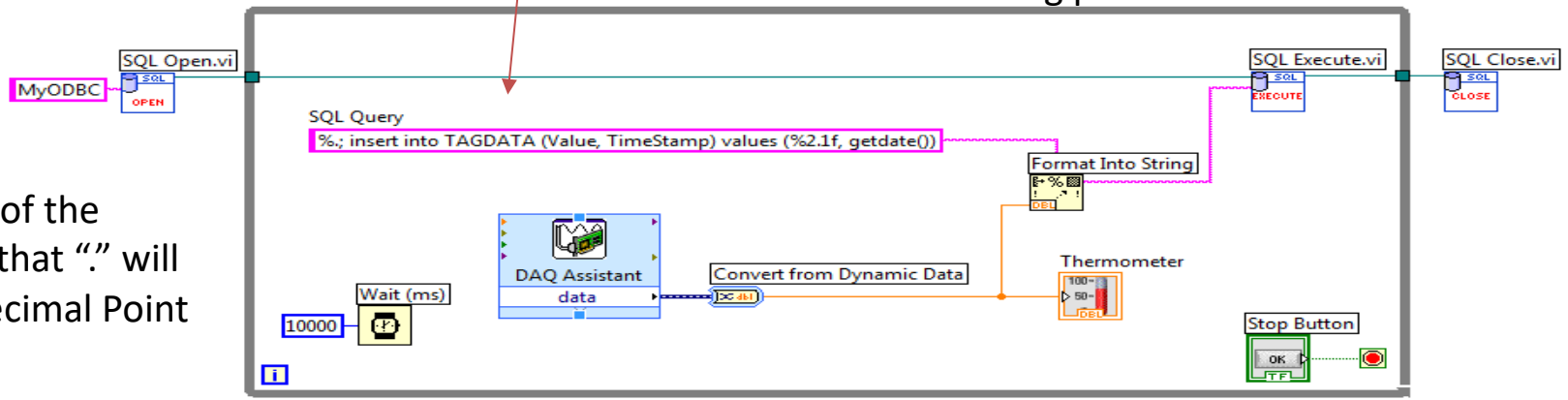
LabVIEW SQL Toolkit Example



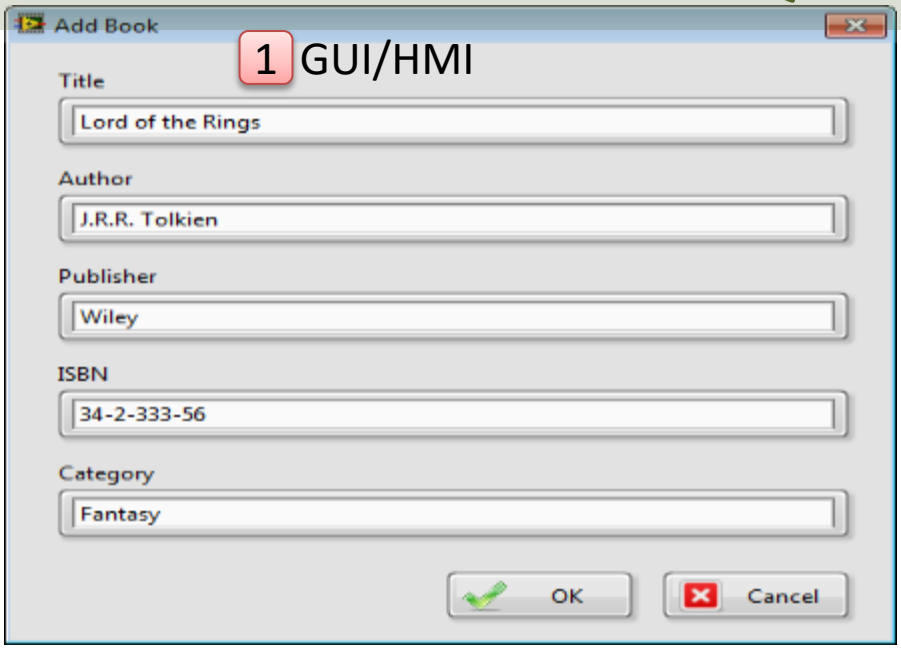
You should use a Stored Procedure for saving the Temperature Data to the Database

"%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

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



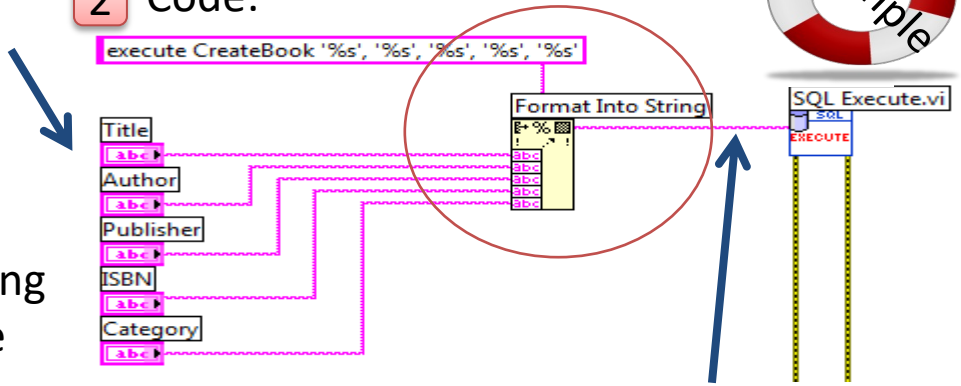
LabVIEW SQL Toolkit Example



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:



Example of Executing a Stored Procedure

3 Resulting SQL Query:

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

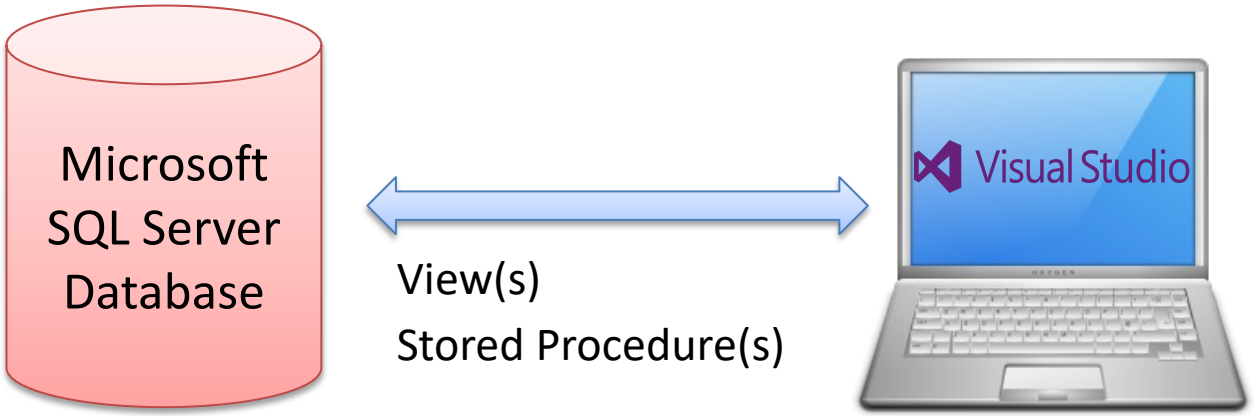
https://www.halvorsen.blog

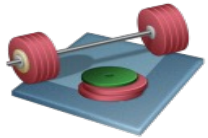
Data Monitoring using Visual Studio/C#

Hans-Petter Halvorsen

Table of Contents

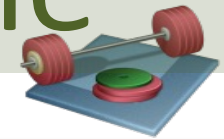
Data Monitoring using Visual Studio/C#





You can create a Desktop Application (WinForm App) or a Web Application (ASP.NET Core App)

Visual Studio HMI Example



Data Monitoring App

Temperature Data:

30 °C

Chart

0

Time

This is just a Design Sketch!
Feel free to Explore! Add Value!

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

DataGridView

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

Alternatives (Choose one):

1. **WinForm 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 to the assignment

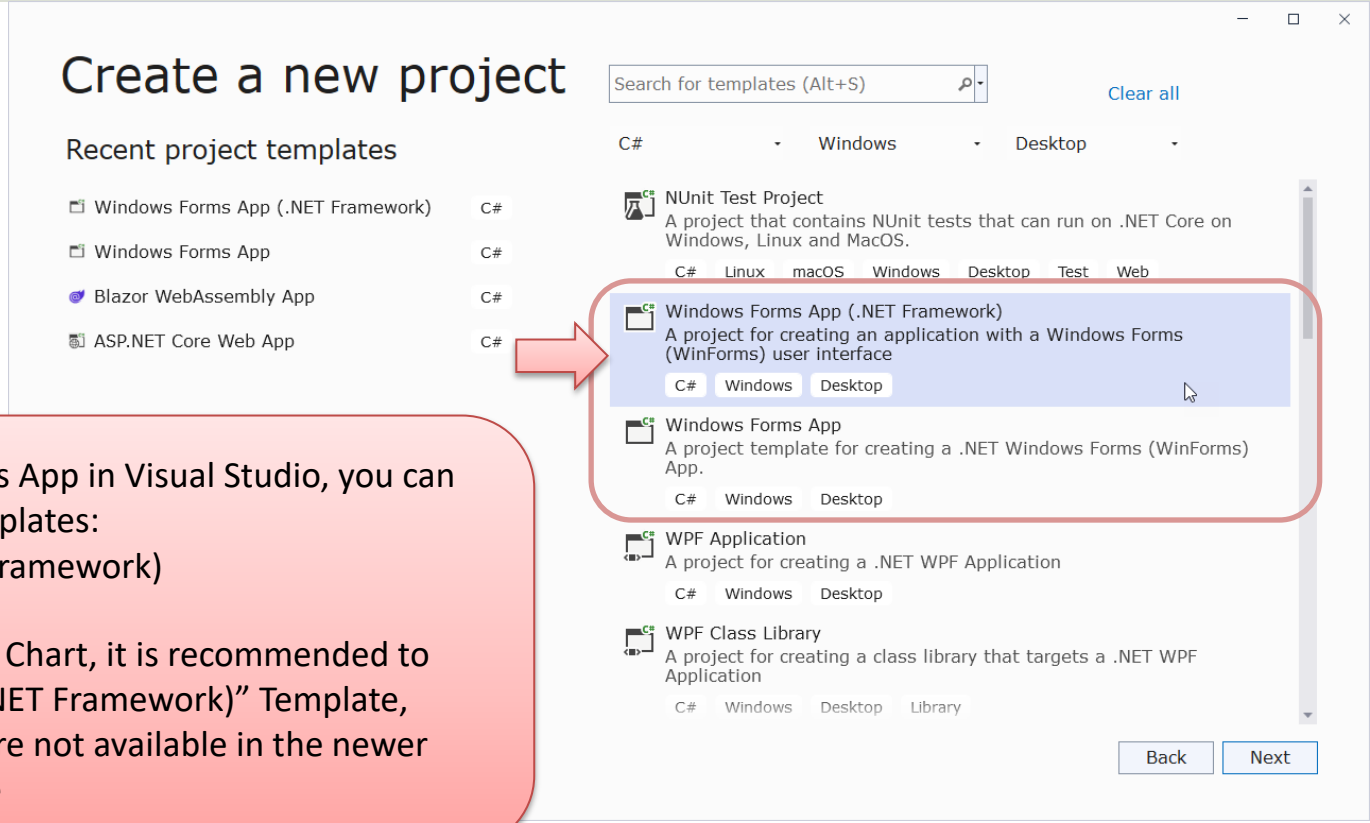
https://www.halvorsen.blog

Windows Forms Desktop Application

Hans-Petter Halvorsen

Table of Contents

Windows Forms App



When creating a Windows Forms App in Visual Studio, you can choose between 2 different Templates:

- Windows Forms App (.NET Framework)
- Windows Forms App

If you are planning on creating a Chart, it is recommended to choose “Windows Forms App (.NET Framework)” Template, because the Chart component are not available in the newer “Windows Forms App” Template

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;
      }
   }
}
```

Timer

1  Timer

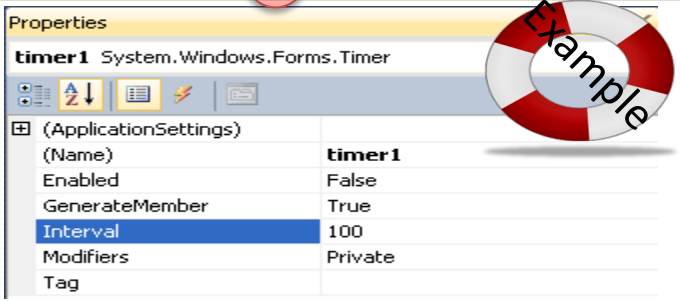
2 **Initialization:**

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

         timer1.Start();
      }
```

Select the “Timer” component in the Toolbox

Double-click on the Timer object in order to create the Event

3 **Properties:**



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.

Charting in Visual Studio

Visual Studio has a Chart control that you can use in Windows Forms Applications

https://msdn.microsoft.com/en-us/library/dd489237.aspx

http://www.i-programmer.info/programming/uiux/2756-getting-started-with-net-charts.html

```
using System.Windows.Forms.DataVisualization.Charting;
...
chart1.Series.Clear();
chart1.Series.Add("My Data");
chart1.Series["My Data"].ChartType=SeriesChartType.Line;
...
int[] x = {1, 2, 3, 4, 5, 6, 7, 8};
int[] y = {20, 22, 25, 24, 28, 27, 24, 26};
for (int i = 0; i < x.Length; i++)
{
   chart1.Series["My Data"].Points.AddXY(x[i],y[i]);
}
```

https://www.halvorsen.blog

ASP.NET Core Web Application

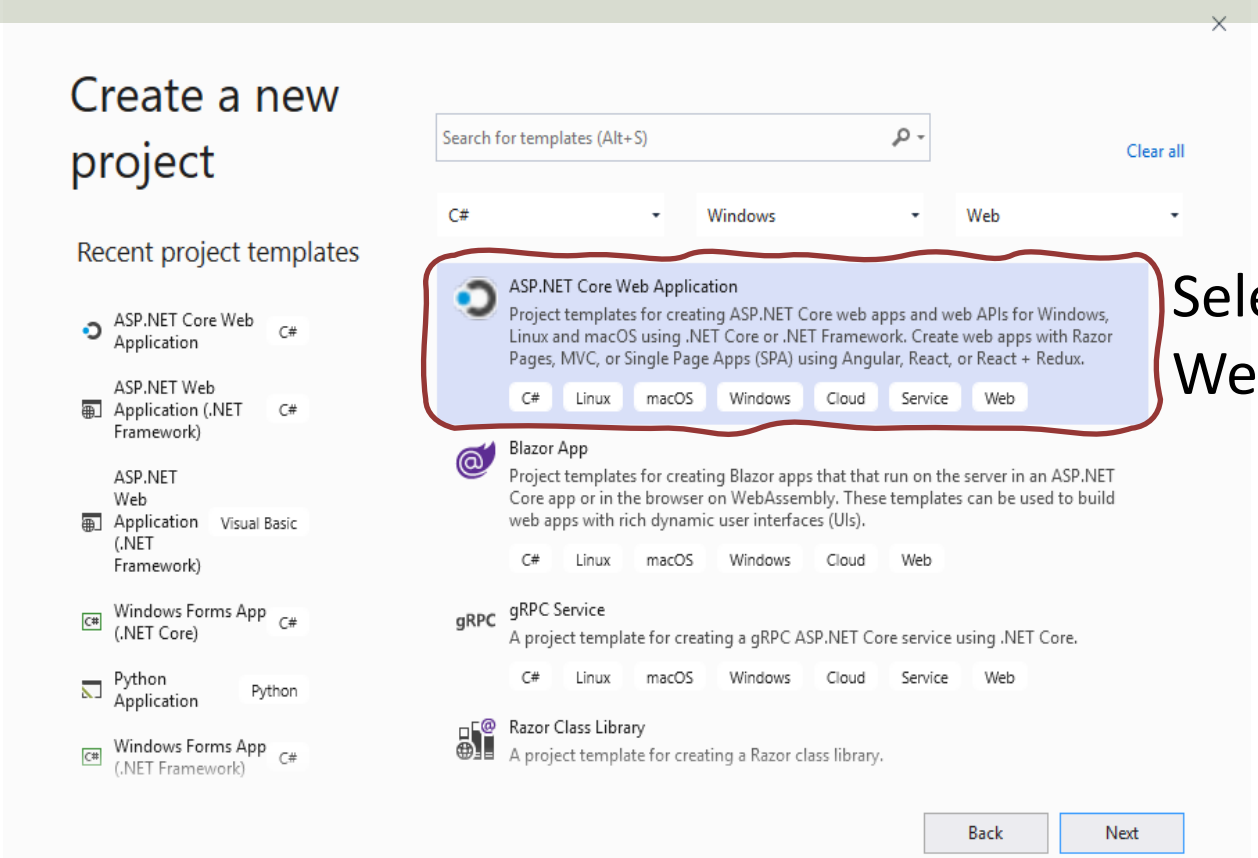
Hans-Petter Halvorsen

Table of Contents

ASP.NET Core Web Application

- ASP.NET is a Web Framework for creating Web Applications
- ASP.NET is integrated with Visual Studio and you will use the C# Programming Language
- .NET Core is cross-platform, meaning it will work on Windows, Linux and macOS.
- ASP.NET Core is Microsoft’s newest baby, and it is the future of Web Programming

ASP.NET Core in Visual Studio



Select the ASP.NET Core Web Application Project

ASP.NET Core Examples

Recommended Videos:

- ASP.NET Core – Introduction: https://youtu.be/zkOtiBcwo8s
- ASP.NET Core - Database Communication: https://youtu.be/0Ta3dQ3rxzs
- ASP.NET Core – Charts: https://youtu.be/mksUls9fx-Q

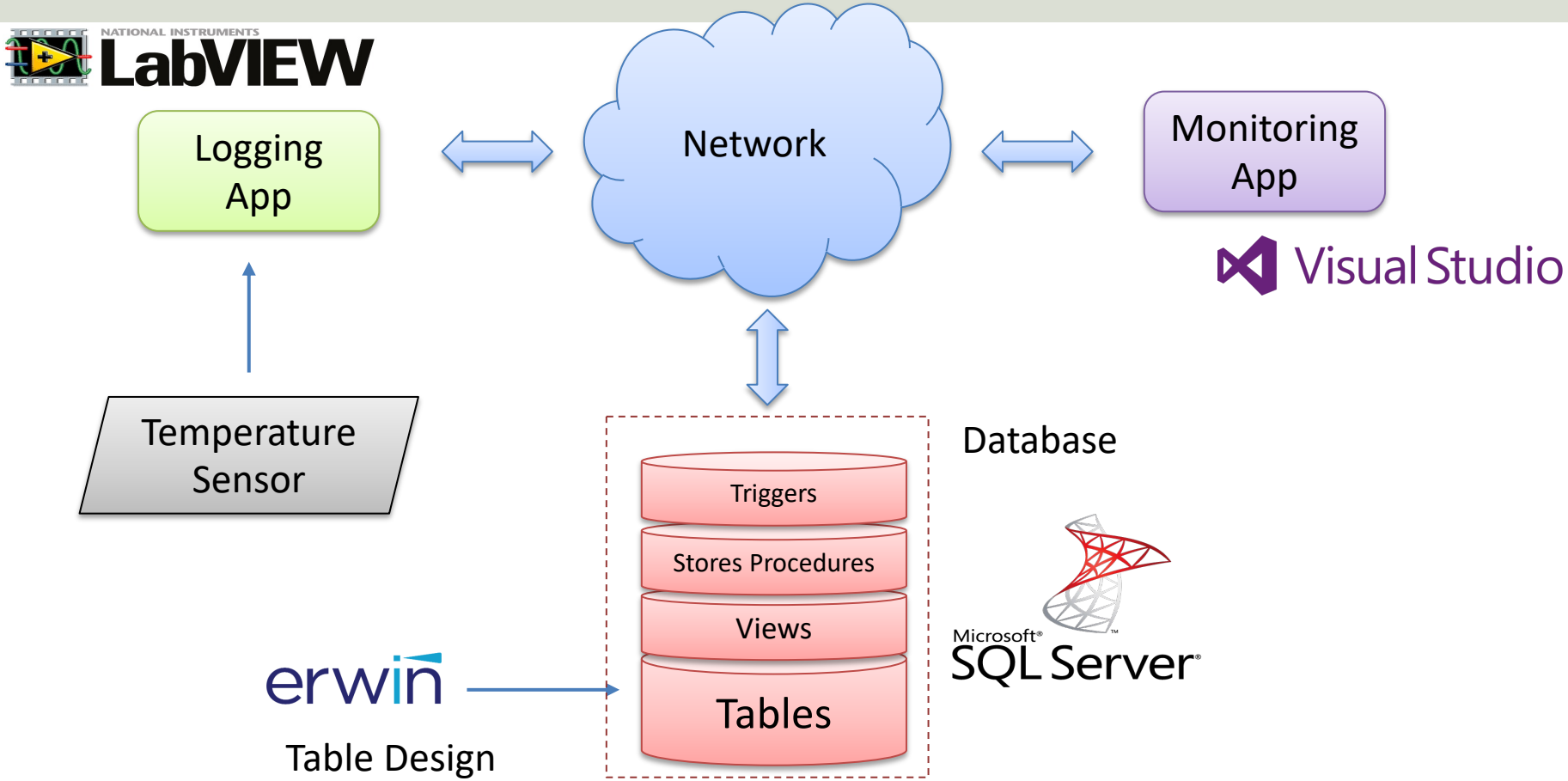
Download Examples here: https://www.halvorsen.blog/documents/programming/web/aspnet

ASP.NET Core Resources

- Textbook
- Videos
- Tutorials
- Example Code

https://www.halvorsen.blog/documents/programming/web/aspnet

This should be your final Solution



Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog