

Telemark University College

Department of Electrical Engineering, Information Technology and Cybernetics



# Database Communication in LabVIEW

HANS-PETTER HALVORSEN, 2011.02.11

SQL - Example1.vi Block Diagram	
<u> Eile E</u> dit <u>V</u> iew <u>P</u> roject <u>O</u> perate <u>T</u> ools <u>W</u> indow <u>H</u> elp	EX-1
🗘 🐼 🔘 💵 😰 🕵 🐜 🗃 🗊 13pt Dialog Font 🔤 🚛 🖬 🐨 🚧	2
	^
SQL Query Data	
(abc)	
	ferror
PROVIDER=SQLOLEDB; DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=TEST	
	~
	2.:

## Preface

This document explains the basic concepts of a database system and how to communicate with a database from LabVIEW.

You should have some basic knowledge about LabVIEW, e.g., the "**An Introduction to LabVIEW**" training. This document is available for download at <u>http://home.hit.no/~hansha/</u>.

In addition to LabVIEW Professional Development System, you need to install the "LabVIEW Database Connectivity Toolkit".

For more information about LabVIEW and Databases, visit my Blog: <u>http://home.hit.no/~hansha/</u>

Some text in this document is based on text from <u>www.wikipedia.org</u> and "LabVIEW Database Connectivity Toolkit User Manual".

## **Table of Contents**

Pı	eface	
Та	ble of	Contentsiii
1	Intr	oduction to LabVIEW
	1.1	Dataflow programming1
	1.2	Graphical programming1
	1.3	Benefits2
2	Dat	abase Systems
	2.1	RDBMS Components
	2.2	Data warehouse 4
	2.3	Relational Database 4
	2.4	Real-time databases
	2.5	Database Management Systems 5
	2.6	MDAC
	2.6.	1 ODBC
	2.6.	2 OLE DB
	2.6.	3 ADO (ActiveX Data Objects) 6
3	Rela	ational Databases
	3.1	Tables7
	3.2	Unique Keys and Primary Key7
	3.3	Foreign Key9
	3.4	Views
	3.5	Functions
	3.6	Stored procedures

	3.7	Trig	gers	11
4	Stru	ucture	ed Query Language (SQL)	12
	4.1	Que	ries	12
	4.2	Data	a manipulation	13
	4.3	Data	a definition	14
	4.4	Data	a types	14
	4.4.	1	Character strings	14
	4.4	.2	Bit strings	15
	4.4.	.3	Numbers	15
	4.4.	.4	Date and Time	15
5	Dat	abase	e Modelling	16
	5.1	ER D	Diagram	16
	5.2	Mic	rosoft Visio	17
	5.3	EXE	RCISES	18
c				
6	Mic	rosof	t SQL Server	20
6	Mic 6.1		t SQL Server	
D		Intro		20
D	6.1	Intro Req	oduction	20 20
D	6.1 6.2	Intro Req SQL	oduction	20 20 20
D	6.1 6.2 6.3	Intro Req SQL Advo	oduction uirements Server Express	20 20 20 21
0	<ul><li>6.1</li><li>6.2</li><li>6.3</li><li>6.4</li></ul>	Intro Req SQL Advo SQL	oduction	20 20 20 21 21
0	<ul> <li>6.1</li> <li>6.2</li> <li>6.3</li> <li>6.4</li> <li>6.5</li> </ul>	Intro Req SQL Advo SQL Crea	oduction	20 20 21 21 21
0	<ul> <li>6.1</li> <li>6.2</li> <li>6.3</li> <li>6.4</li> <li>6.5</li> <li>6.6</li> </ul>	Intro Req SQL Advo SQL Crea Back	oduction	20 20 21 21 22 22
6	<ul> <li>6.1</li> <li>6.2</li> <li>6.3</li> <li>6.4</li> <li>6.5</li> <li>6.6</li> <li>6.7</li> </ul>	Intro Req SQL Advo SQL Crea Back Exar	oduction	20 20 21 21 22 24 25
7	<ul> <li>6.1</li> <li>6.2</li> <li>6.3</li> <li>6.4</li> <li>6.5</li> <li>6.6</li> <li>6.7</li> <li>6.8</li> <li>6.9</li> </ul>	Intro Req SQL Advo SQL Crea Back Exar Exar	oduction	20 20 21 21 22 24 25 27

	7.2	Example Database	. 28
	7.3	Exercises	. 30
8	ODE	3C	. 32
	8.1	What is ODBC?	. 32
	8.2	Create an ODBC Connection in "ODBC Data Source Administrator"	. 32
	8.3	Get data into Excel using your ODBC Connection	. 34
9	Lab	VIEW Database Connectivity Toolkit	. 38
	9.1	Connect to the Database	. 39
	9.1.	1 DSN	. 40
	9.1.	2 UDL	. 41
	9.1.	3 Connection String	. 42
	9.2	Reading Data from the Database	. 42
	9.3	Writing Data to the Database	. 46
	9.4	Creating and Dropping Tables	. 48
	9.5	Using the Database Connectivity Toolkit Utility VIs	. 49
	9.6	Performing Advanced Database Operations	. 50
10	) Crea	ating and Using Tables	. 52
	10.1	Exercises	. 55
11	L Crea	ating and Using Views	. 56
	11.1	Exercises	. 59
12	2 Crea	ating and using Stored Procedures	. 60
	12.1	Exercises	. 62
13	8 Crea	ating and Using Triggers	. 63
	13.1	Exercises	. 66
14	l Crea	ating and using Functions	. 67
	14.1	Exercises	. 67

15	SQL	Toolkit	68
15	5.1	Installation	68

## **1Introduction to LabVIEW**

LabVIEW (short for **Lab**oratory **V**irtual Instrumentation **E**ngineering **W**orkbench) is a platform and development environment for a visual programming language from National Instruments. The graphical language is named "G". Originally released for the Apple Macintosh in 1986, LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, various flavors of UNIX, Linux, and Mac OS X. The latest version of LabVIEW is version LabVIEW 2009, released in August 2009. Visit National Instruments at <u>www.ni.com</u>.

The code files have the extension ".vi", which is a abbreviation for "Virtual Instrument". LabVIEW offers lots of additional Add-Ons and Toolkits.

#### 1.1 Dataflow programming

The programming language used in LabVIEW, also referred to as G, is a dataflow programming language. Execution is determined by the structure of a graphical block diagram (the LV-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution. Multi-processing and multi-threading hardware is automatically exploited by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution.

### 1.2 Graphical programming

LabVIEW ties the creation of user interfaces (called front panels) into the development cycle. LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector panel. The last is used to represent the VI in the block diagrams of other, calling VIs. Controls and indicators on the front panel allow an operator to input data into or extract data from a running virtual instrument. However, the front panel can also serve as a programmatic interface. Thus a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the given node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

The graphical approach also allows non-programmers to build programs simply by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW

programming environment, with the included examples and the documentation, makes it simple to create small applications. This is a benefit on one side, but there is also a certain danger of underestimating the expertise needed for good quality "G" programming. For complex algorithms or large-scale code, it is important that the programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the possibility of building stand-alone applications. Furthermore, it is possible to create distributed applications, which communicate by a client/server scheme, and are therefore easier to implement due to the inherently parallel nature of G-code.

#### 1.3 Benefits

One benefit of LabVIEW over other development environments is the extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments and buses are included or are available for inclusion. These present themselves as graphical nodes. The abstraction layers offer standard software interfaces to communicate with hardware devices. The provided driver interfaces save program development time. The sales pitch of National Instruments is, therefore, that even people with limited coding experience can write programs and deploy test solutions in a reduced time frame when compared to more conventional or competing systems. A new hardware driver topology (DAQmxBase), which consists mainly of G-coded components with only a few register calls through NI Measurement Hardware DDK (Driver Development Kit) functions, provides platform independent hardware access to numerous data acquisition and instrumentation devices. The DAQmxBase driver is available for LabVIEW on Windows, Mac OS X and Linux platforms.

## 2Database Systems

A database is an integrated collection of logically related records or files consolidated into a common pool that provides data for one or more multiple uses.

One way of classifying databases involves the type of content, for example: bibliographic, full-text, numeric, and image. Other classification methods start from examining database models or database architectures.

The data in a database is organized according to a database model. The relational model is the most common.

A Database Management System (**DBMS**) consists of software that organizes the storage of data. A DBMS controls the creation, maintenance, and use of the database storage structures of organizations and of their end users. It allows organizations to place control of organization-wide database development in the hands of Database Administrators (DBAs) and other specialists. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way.

Database management systems are usually categorized according to the database model that they support, such as the network, relational or object model. The model tends to determine the query languages that are available to access the database. One commonly used query language for the relational database is SQL, although SQL syntax and function can vary from one DBMS to another. A great deal of the internal engineering of a DBMS is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products.

#### 2.1 RDBMS Components

A Relational Database Management System (DBMS) consists of the following components:

- Interface drivers A user or application program initiates either schema modification or content modification. These drivers are built on top of SQL. They provide methods to prepare statements, execute statements, fetch results, etc. An important example is the ODBC driver.
- **SQL engine** This component interprets and executes the SQL query. It comprises three major components (compiler, optimizer, and execution engine).
- **Transaction engine** Transactions are sequences of operations that read or write database elements, which are grouped together.
- **Relational engine** Relational objects such as Table, Index, and Referential integrity constraints are implemented in this component.

• **Storage engine** - This component stores and retrieves data records. It also provides a mechanism to store metadata and control information such as undo logs, redo logs, lock tables, etc.

### 2.2 Data warehouse

A data warehouse stores data from current and previous years — data extracted from the various operational databases of an organization. It becomes the central source of data that has been screened, edited, standardized and integrated so that it can be used by managers and other end-user professionals throughout an organization.

### 2.3 Relational Database

A relational database matches data using common characteristics found within the data set. The resulting groups of data are organized and are much easier for people to understand.

For example, a data set containing all the real-estate transactions in a town can be grouped by the year the transaction occurred; or it can be grouped by the sale price of the transaction; or it can be grouped by the buyer's last name; and so on.

Such a grouping uses the relational model (a technical term for this is schema). Hence, such a database is called a "relational database."

The software used to do this grouping is called a relational database management system. The term "relational database" often refers to this type of software.

Relational databases are currently the predominant choice in storing financial records, manufacturing and logistical information, personnel data and much more.

Strictly, a relational database is a collection of relations (frequently called tables).

### 2.4 Real-time databases

A real-time database is a processing system designed to handle workloads whose state may change constantly. This differs from traditional databases containing persistent data, mostly unaffected by time. For example, a stock market changes rapidly and dynamically. Real-time processing means that a transaction is processed fast enough for the result to come back and be acted on right away. Real-time databases are useful for accounting, banking, law, medical records, multi-media, process control, reservation systems, and scientific data analysis. As computers increase in power and can store more data, real-time databases become integrated into society and are employed in many applications

### 2.5 Database Management Systems

There are Database Management Systems (DBMS), such as:

- Microsoft SQL Server
- Oracle
- Sybase
- dBase
- Microsoft Access
- MySQL from Sun Microsystems (Oracle)
- DB2 from IBM
- etc.

This document will focus on Microsoft Access and Microsoft SQL Server.

### 2.6 MDAC

The Microsoft Data Access Components (**MDAC**) is the framework that makes it possible to connect and communicate with the database. MDAC includes the following components:

- **ODBC** (Open Database Connectivity)
- OLE DB
- ADO (ActiveX Data Objects)

MDAC also installs several data providers you can use to open a connection to a specific data source, such as an MS Access database.

#### 2.6.1 ODBC

**Open Database Connectivity (ODBC)** is a native interface that is accessed through a programming language that can make calls into a native library. In MDAC this interface is defined as a DLL. A separate module or driver is needed for each database that must be accessed.

#### 2.6.2 OLE DB

**OLE** allows MDAC applications access to different types of data stores in a uniform manner. Microsoft has used this technology to separate the application from the data store that it needs to access. This was done because different applications need access to different types and sources of data, and do not necessarily need to know how to access technology-specific functionality. The technology is conceptually divided into consumers and providers. The consumers are the applications that need access to the data, and the provider is the software component that exposes an OLE DB interface through the use of the Component Object Model (or COM).

#### 2.6.3 ADO (ActiveX Data Objects)

ActiveX Data Objects (ADO) is a high level programming interface to OLE DB. It uses a hierarchical object model to allow applications to programmatically create, retrieve, update and delete data from sources supported by OLE DB. ADO consists of a series of hierarchical COM-based objects and collections, an object that acts as a container of many other objects. A programmer can directly access ADO objects to manipulate data, or can send an SQL query to the database via several ADO mechanisms.

## **3Relational Databases**

A relational database matches data using common characteristics found within the data set. The resulting groups of data are organized and are much easier for people to understand.

For example, a data set containing all the real-estate transactions in a town can be grouped by the year the transaction occurred; or it can be grouped by the sale price of the transaction; or it can be grouped by the buyer's last name; and so on.

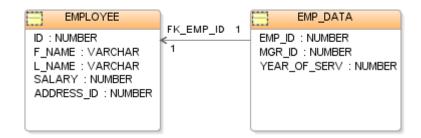
Such a grouping uses the relational model (a technical term for this is schema). Hence, such a database is called a "relational database."

The software used to do this grouping is called a relational database management system. The term "relational database" often refers to this type of software.

Relational databases are currently the predominant choice in storing financial records, manufacturing and logistical information, personnel data and much more.

#### 3.1 Tables

The basic units in a database are tables and the relationship between them. Strictly, a relational database is a collection of relations (frequently called tables).



#### 3.2 Unique Keys and Primary Key

In relational database design, a **unique key** or primary key is a candidate key to uniquely identify each row in a table. A unique key or primary key comprises a single column or set of columns. No two distinct rows in a table can have the same value (or combination of values) in those columns. Depending on its design, a table may have arbitrarily many unique keys but at most one primary key. A unique key must uniquely identify all possible rows that exist in a table and not only the currently existing rows. Examples of unique keys are Social Security numbers or ISBNs.

A **primary key** is a special case of unique keys. The major difference is that for unique keys the implicit NOT NULL constraint is not automatically enforced, while for primary keys it is enforced. Thus, the values in unique key columns may or may not be NULL. Another difference is that primary keys must be defined using another syntax.

Primary keys are defined with the following syntax:

```
CREATE TABLE table_name (

id col INT,

col2 CHARACTER VARYING(20),

...

CONSTRAINT tab_pk PRIMARY KEY(id_col),

...
```

If the primary key consists only of a single column, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (
id_col INT PRIMARY KEY,
col2 CHARACTER VARYING(20),
...
```

The definition of unique keys is syntactically very similar to primary keys.

Likewise, unique keys can be defined as part of the CREATE TABLE SQL statement.

Or if the unique key consists only of a single column, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (

id_col INT PRIMARY KEY,

col2 CHARACTER VARYING(20),

...

key_col SMALLINT UNIQUE,

...

)
```

#### 3.3 Foreign Key

In the context of relational databases, a foreign key is a referential constraint between two tables. The foreign key identifies a column or a set of columns in one table that refers to a column or set of columns in another table. The columns in the referencing table must be the primary key or other candidate key in the referenced table. The values in one row of the referencing columns must occur in a single row in the referenced table. Thus, a row in the referencing table cannot contain values that don't exist in the referenced table. This way references can be made to link information together and it is an essential part of database normalization. Multiple rows in the referencing table may refer to the same row in the referenced table. Most of the time, it reflects the one (master table, or referenced table) to many (child table, or referencing table) relationship.

The referencing and referenced table may be the same table, i.e. the foreign key refers back to the same table. Such a foreign key is known as self-referencing or recursive foreign key.

A table may have multiple foreign keys, and each foreign key can have a different referenced table. Each foreign key is enforced independently by the database system. Therefore, cascading relationships between tables can be established using foreign keys.

Improper foreign key/primary key relationships or not enforcing those relationships are often the source of many database and data modeling problems.

Foreign keys can be defined as part of the CREATE TABLE SQL statement.

```
CREATE TABLE table_name (
   id INTEGER PRIMARY KEY,
   col2 CHARACTER VARYING(20),
   col3 INTEGER,
   ...
   CONSTRAINT col3_fk FOREIGN KEY(col3)
        REFERENCES other_table(key_col),
   ...)
```

If the foreign key is a single column only, the column can be marked as such using the following syntax:

```
CREATE TABLE table_name (
   id INTEGER PRIMARY KEY,
   col2 CHARACTER VARYING(20),
   col3 INTEGER REFERENCES other_table(column_name),
    ...)
```

#### 3.4 Views

In database theory, a view consists of a stored query accessible as a virtual table composed of the result set of a query. Unlike ordinary tables in a relational database, a view does not form part of the physical schema: it is a dynamic, virtual table computed or collated from data in the database. Changing the data in a table alters the data shown in subsequent invocations of the view.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table
- Views can join and simplify multiple tables into a single virtual table •
- Views can act as aggregated tables, where the database engine aggregates data (sum, average etc) and presents the calculated results as part of the data
- Views can hide the complexity of data; for example a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data it presents
- Views can limit the degree of exposure of a table or tables to the outer world •

Syntax:

```
CREATE VIEW <ViewName>
AS
```

### **3.5 Functions**

In SQL databases, a user-defined function provides a mechanism for extending the functionality of the database server by adding a function that can be evaluated in SQL statements. The SQL standard distinguishes between scalar and table functions. A scalar function returns only a single value (or NULL), whereas a table function returns a (relational) table comprising zero or more rows, each row with one or more columns.

User-defined functions in SQL are declared using the CREATE FUNCTION statement.

Syntax:

```
CREATE FUNCTION <FunctionName>
      (@Parameter1 <datatype>,
      @ Parameter2 <datatype>,
      ...)
RETURNS <datatype>
AS
```

#### 3.6 Stored procedures

10

A stored procedure is executable code that is associated with, and generally stored in, the database. Stored procedures usually collect and customize common operations, like inserting a tuple into a relation, gathering statistical information about usage patterns, or encapsulating complex business logic and calculations. Frequently they are used as an application programming interface (API) for security or simplicity.

Stored procedures are not part of the relational database model, but all commercial implementations include them.

Stored procedures are called or used with the following syntax:

CALL	procedure()
01122	P=00000010 ()

or

#### EXECUTE procedure(...)

Stored procedures can return result sets, i.e. the results of a SELECT statement. Such result sets can be processed using cursors by other stored procedures by associating a result set locator, or by applications. Stored procedures may also contain declared variables for processing data and cursors that allow it to loop through multiple rows in a table. The standard Structured Query Language provides IF, WHILE, LOOP, REPEAT, CASE statements, and more. Stored procedures can receive variables, return results or modify variables and return them, depending on how and where the variable is declared.

### 3.7 Triggers

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for keeping the integrity of the information on the database. For example, when a new record (representing a new worker) added to the employees table, new records should be created also in the tables of the taxes, vacations, and salaries.

The syntax is as follows:

```
CREATE TRIGGER <TriggerName> ON <TableName>
FOR INSERT, UPDATE, DELETE
AS
```

# 4Structured Query Language (SQL)

SQL (Structured Query Language) is a database computer language designed for managing data in relational database management systems (RDBMS).

### 4.1 Queries

The most common operation in SQL is the query, which is performed with the declarative SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database.

Queries allow the user to describe desired data, leaving the database management system (DBMS) responsible for planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("\*") can also be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses that include:

- The **FROM** clause which indicates the table(s) from which data is to be retrieved. The FROM clause can include optional JOIN subclauses to specify the rules for joining tables.
- The WHERE clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.
- The **GROUP BY** clause is used to project rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
- The **HAVING** clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
- The **ORDER BY** clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined.

#### Example:

The following is an example of a SELECT query that returns a list of expensive books. The query retrieves all rows from the Book table in which the price column contains a value greater than 100.00. The result is sorted in ascending order by title. The asterisk (\*) in the select list indicates that all columns of the Book table should be included in the result set.

SELECT \* FROM Book WHERE price > 100.00 ORDER BY title;

The example below demonstrates a query of multiple tables, grouping, and aggregation, by returning a list of books and the number of authors associated with each book.

```
SELECT Book.title,count(*) AS Authors
FROM Book
JOIN Book_author ON Book.isbn = Book_author.isbn
GROUP BY Book.title
```

Example output might resemble the following:

```
Title Authors
SQL Examples and Guide 4
The Joy of SQL 1
An Introduction to SQL 2
Pitfalls of SQL 1
```

#### 4.2 Data manipulation

The Data Manipulation Language (DML) is the subset of SQL used to add, update and delete data.

The acronym **CRUD** refers to all of the major functions that need to be implemented in a relational database application to consider it complete. Each letter in the acronym can be mapped to a standard SQL statement:

Operation	SQL
Create	INSERT
Read (Retrieve)	SELECT
<b>U</b> pdate	UPDATE
Delete (Destroy)	DELETE

**Example: INSERT** 

INSERT adds rows to an existing table, e.g.,:

```
INSERT INTO My_table field1, field2, field3)
VALUES ('test', 'N', NULL)
```

#### Example: UPDATE

UPDATE modifies a set of existing table rows, e.g.,:

```
UPDATE My_table
SET field1 = 'updated value'
WHERE field2 = 'N'
```

#### **Example: DELETE**

DELETE removes existing rows from a table, e.g.,:

DELETE FROM My\_table WHERE field2 = 'N'

#### 4.3 Data definition

The **Data Definition Language (DDL)** manages table and index structure. The most basic items of DDL are the CREATE, ALTER, RENAME and DROP statements:

- **CREATE** creates an object (a table, for example) in the database.
- **DROP** deletes an object in the database, usually irretrievably.
- ALTER modifies the structure an existing object in various ways—for example, adding a column to an existing table.

#### **Example: CREATE**

Create a Database Table

#### 4.4 Data types

Each column in an SQL table declares the type(s) that column may contain. ANSI SQL includes the following datatypes.

#### 4.4.1 Character strings

- CHARACTER(n) or CHAR(n) fixed-width n-character string, padded with spaces as needed
- CHARACTER VARYING(n) or VARCHAR(n) variable-width string with a maximum size of n characters
- NATIONAL CHARACTER(n) or NCHAR(n) fixed width string supporting an international character set
- NATIONAL CHARACTER VARYING(n) or NVARCHAR(n) variable-width NCHAR string

#### 4.4.2 Bit strings

- BIT(n) an array of n bits
- BIT VARYING(n) an array of up to n bits

#### 4.4.3 Numbers

- INTEGER and SMALLINT
- FLOAT, REAL and DOUBLE PRECISION
- NUMERIC(precision, scale) or DECIMAL(precision, scale)

#### 4.4.4 Date and Time

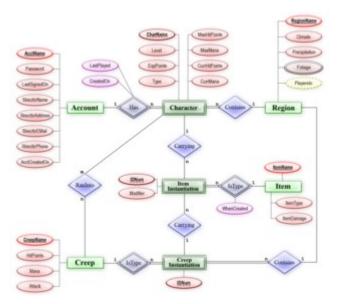
- DATE
- TIME
- TIMESTAMP
- INTERVAL

## **5Database Modelling**

### 5.1 ER Diagram

In software engineering, an Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.

Diagrams created using this process are called entity-relationship diagrams, or ER diagrams or ERDs for short.



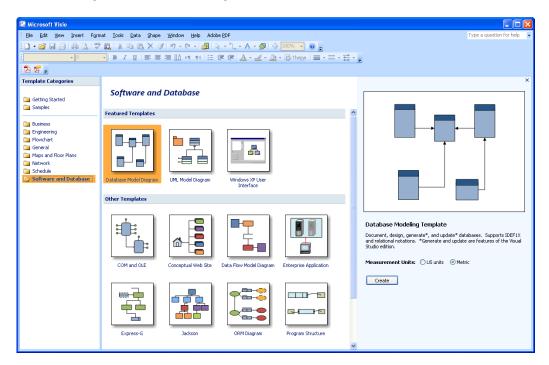
There are many ER diagramming tools. Some of the proprietary ER diagramming tools are ERwin, Enterprise Architect and Microsoft Visio.

Microsoft SQL Server has also a built-in tool for creating Database Diagrams.

									6	
Server Management Studio									l	
File Edit View Debug Table Designer Database Diagram	Tools Window Community	Help								
일 New Query 🕞 📑 😂 🗐 🚭 🛒										
🛅 🔚 💱 🖾 🕬 📓 🤋 alb   Table View + 🏂 🔫	🔁 🔝 😒 🕰 100%	·	릚 🗉 🔤 🖕							
Object Explorer 🗸 🕂 📈	PC88235.TEST - Customer Ta	ble Script.sq5.TEST	(sa (54)) SQLQue	ry3.sql	TEST (sa (52))* 📉 Obj	ect Explorer Details	-	× Properties		+ 4 ×
Connect • 📑 📑 = 🍸 🖓								[Dgm] Custome		-
PC88235 (SQL Server 10.0.2531 - sa)	CUSTOMER			PF	RODUCT			21 21		_
🖃 🧰 Databases	Column Name	Data Type	Allow Nulls		Column Name	Data Type	Allow Nulls	(Identity)		_
🖃 🚞 System Databases	CustomerId	int		8	ProductId	int		(Name)	Customer	
imaster     imaster     ima Tables	FirstName	varchar(50)			ProductName	varchar(50)		Database Des		
Gues     System Tables	LastName	varchar(50)			ProductDescription	varchar(50)		Database Name	TEST	
H in Views	Address	varchar(50)			Price	float		Owner	dbo	
III 🥅 Synonyms	Phone	varchar(50)			ProductCode	varchar(50)	<b>V</b>	Server Name	pc88235	
Department     D	PostCode	varchar(50)								
B Security	PostAddress	varchar(50)	<ul> <li>Image: Construction</li> </ul>							
H 📔 model		Torchar(00)								
🗷 🧻 msdb			_	<			>			
🗉 🧻 tempdb	<		>			8				
		4				Ĭ				
dbo.Customer		Š				Š				
🖃 🦢 Tables	ORDER			OR	DER DETAIL					
Games System Tables     do.CUSTOMER	Column Name	Data Type	Allow Nulls		Column Name	Data Type	Allow Nulls			
H dbb.CCDSTCMER	😵 OrderId	int		8	OrderDetailId	int				
do.ORDER_DETAIL	OrderNumber	varchar(50)			OrderId	int				
dbo.PRODUCT	OrderDescription	varchar(50)			ProductId	int				
Wews	CustomerId	int								
De Synonyms     De Synonyms     De Synonyms				_						
Gervice Broker	_			-00						
🗉 🧰 Security										
🗉 🧰 Security	<	11	>							
Server Objects     Teolication										
Anagement										
								(Identity)		
								-		
							•	<b>₽</b>		
Item(s) Saved										

#### 5.2 Microsoft Visio

Microsoft Visio is a diagramming program for creating different kinds of diagrams. Visio have a template for creating Database Model Diagrams.



🖥 Database Model.vsd - Micro	osoft Visio	
🗐 Eile Edit Yiew Insert f	F <u>o</u> rmat <u>T</u> ools <u>D</u> ata	a D <u>a</u> tabase <u>S</u> hape <u>W</u> indow <u>H</u> elp
🗋 • 📂 🛃 🚑   🖨 🖏 🖤	° 🕰   X 🗈 🕰	Reverse Engineer C A
Arial 🚽 12pt	• B <i>I</i> <u>U</u>	🗐 Show Related Tables 🛛 🖉 🗸 🌆 🗸
12 12 -		Refresh Model
Shapes ×		10 <u>View</u>
Search for Shapes:	530	Import
Type your search here 🛛 👻 🔁		User Defined <u>Types</u>
🗄 Entity Relationship (Metric)		Options •
Entity Relations		PK <u>Customerld</u>
View L Parent to category	560	FirstName LastName Address

In the Database menu Visio offers lots of functionality regarding your database model.

"Reverse Engineering" is the opposite procedure, i.e., extraction of a database schema from an existing database into a database model in Microsoft Visio.

### 5.3 EXERCISES

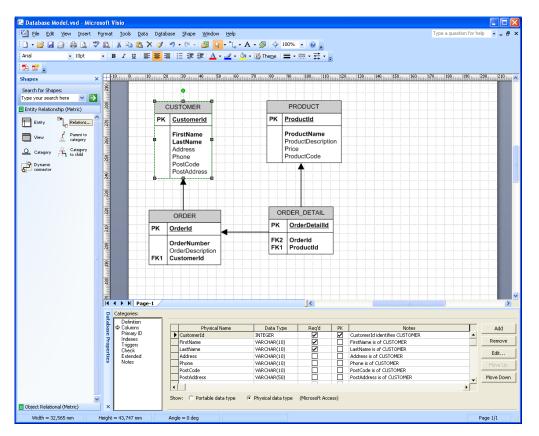
#### Exercise: Database Diagram

Create the following tables in an ER Diagram using MS Visio.

- CUSTOMER
  - o Customerid (PK)
  - o FirstName
  - LastName
  - Address
  - o Phone
  - PostCode
  - o PostAddress
- PRODUCT
  - $\circ$  Productid (PK)
  - ProductName
  - ProductDescription
  - o Price
  - ProductCode

- ORDER
  - o Orderld (PK)
  - o OrderNumber
  - OrderDescription
  - o Customerid (FK)
- ORDER\_DETAIL
  - OrderDetailId (PK)
  - OrderId (FK)
  - Productid (FK)

Database Diagram:



## 6Microsoft SQL Server

#### 6.1 Introduction

Microsoft SQL Server is a relational model database server produced by Microsoft. Its primary query languages are T-SQL and ANSI SQL.

The latest version is Microsoft SQL Server 2008.

Microsoft SQL Server homepage: www.microsoft.com/sqlserver

The Microsoft SQL Server comes in different versions, such as:

- SQL Server Developer Edition
- SQL Server Enterprise Edition
- SQL Server Web Edition
- SQL Server Express Edition
- Etc.

The SQL Server Express Edition is a freely-downloadable and -distributable version.

#### 6.2 Requirements

In order to install SQL Server 2008, you need:

- Microsoft .NET Framework 3.5 SP1
- Windows Installer 4.5
- Windows PowerShell 1.0

Note: You must have administrative rights on the computer to install Microsoft SQL Server 2008.

#### 6.3 SQL Server Express

The SQL Server Express Edition is a freely-downloadable and -distributable version.

However, the Express edition has a number of technical restrictions which make it undesirable for large-scale deployments, including:

- Maximum database size of 4 GB per. The 4 GB limit applies per database (log files excluded); but in some scenarios users can access more data through the use of multiple interconnected databases.
- Single physical CPU, multiple cores
- 1 GB of RAM (runs on any size RAM system, but uses only 1 GB)

SQL Server Express offers a GUI tools for database management in a separate download and installation package, called **SQL Server Management Studio Express**.

### 6.4 AdventureWorks

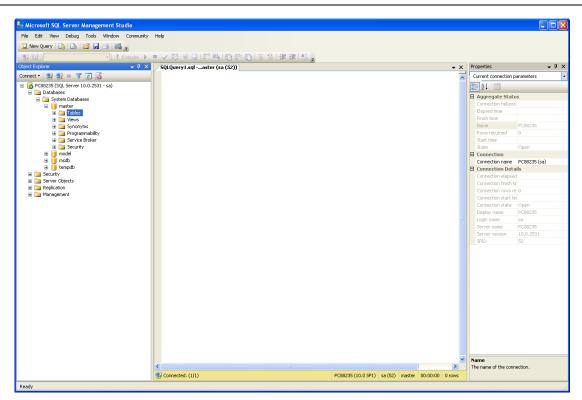
The AdventureWorks is a sample Database with lots of examples, etc.

You should install this sample Database because some of the examples in this document will use the AdventureWorks database.

### 6.5 SQL Server Management Studio

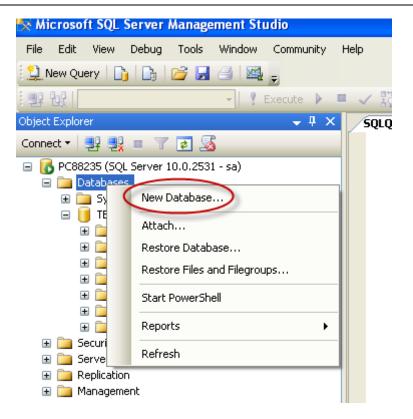
SQL Server Management Studio is a GUI tool included with SQL Server for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. As mentioned earlier, version of SQL Server Management Studio is also available for SQL Server Express Edition, for which it is known as SQL Server Management Studio Express.

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.



#### 6.6 Create a new Database

It is quite simple to create a new database in Microsoft SQL Server. Just right-click on the "Databases" node and select "New Database..."



There are lots of settings you may set regarding your database, but the only information you must fill in is the name of your database:

🚦 New Database					
Select a page I General	<u> S</u> Script 🝷 🛐	Help			
Options Filegroups	Database name:				
	Owner:		<default></default>		
	Use full-text in	ndexing			
	Database files:				
	Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
		Rows Data	PRIMARY	3	By 1 MB, unrestricted growth
	_log	Log	Not Applicable	1	By 10 percent, unrestricted gr
Connection					
Server: PC88235					
Connection: sa					
View connection properties					
Progress					
Ready	<	1111			>
A CRAN			C	Add	Remove
					OK Cancel

### 6.7 Backup/Restore

An important task in database systems is to take backup of the database with regular intervals, e.g., during the night when the system is not in use.

Database backup and Restore:

🐻 PC88235 (SQL Se	rver 10.0.2531 - sa)	
Databases		
🕀 🧰 System D 🖃 间 TEST	atabases	
	New Database	
	New Query	
	Script Database as	
±	Tasks 🕨	Detach
±	Policies +	Take Offline
±	Facets	Bring Online
	Start PowerShell	Shrink <b>&gt;</b>
± 🗖	Reports •	Back Up
😐 🛄 E 🚞 Securit	Rename	Restore +
⊞ 🧰 Lo¢ ⊞ 📬 Ser	Delete	Generate Scripts
	Refresh	Publish using Web Service
🖃 🚞 Server	Properties	Import Data
🕀 🧰 Baunger		Export Data
표 🚞 Linked Se	rvers	
📕 Back Up Database -	- TEST	
Back Up Database - Select a page		
	Script + 👔 Help	
Select a page 🚰 General	Script - 💽 Help	
Select a page 🚰 General	Script - 🚺 Help Source Database:	
Select a page	Script • 💽 Help Source Database: Recovery model:	SIMPLE
Select a page	Script • 💽 Help Source Database: Recovery model: Backup type:	
Select a page	Script • 💽 Help Source Database: Recovery model:	SIMPLE
Select a page	Script  Fight Help  Source  Database:  Recovery model:  Backup type:  Copy Only Backup	SIMPLE
Select a page	Script	SIMPLE
Select a page	Script	SIMPLE VILLE
Select a page	Script	SIMPLE Full
Select a page	Script • 💽 Help Source Database: Recovery model: Backup type: Copy Only Backup Backup component: • Database Files and filegroups: Backup set Name:	SIMPLE VILLE
Select a page General Options	Script - The Help Source Database: Recovery model: Backup type: Copy Only Backup Backup component: Database Files and filegroups: Backup set Name: Description:	SIMPLE VILLE
Select a page	Script	SIMPLE Full  TEST-Full Database Backup
Select a page General Options	Script	SIMPLE Full  TEST-Full Database Backup  days
Select a page General Options	Script    Source  Database:  Recovery model:  Backup type:  Copy Only Backup  Backup component:  Database  Files and filegroups:  Backup set  Name:  Description:  Backup set will expire:  After:  On:  Destination  Back up to:	SIMPLE Full  TEST-Full Database Backup  TEST-Full Database Backup  0 0 02.11.2009  Disk Tape
Select a page General Options	Script	SIMPLE Full TEST-Full Database Backup
Select a page General Options	Script	SIMPLE Full  TEST-Full Database Backup  TEST-Full Database Backup  0 0 02.11.2009  Disk Tape
Select a page General Options Connection Server: PC88235 Connection: sa View connection prope	Script	SIMPLE Full TEST-Full Database Backup
Select a page General Options Connection Server: PC88235 Connection: sa PView connection proper Progress	Script • 💽 Help Source Database: Recovery model: Backup type: Copy Only Backup Backup component: O Database Files and filegroups: Backup set Name: Description: Backup set will expire: O After: O Dr: Destination Back up to: c:\Program Files\Microsoft S	SIMPLE Full TEST-Full Database Backup D 0 0 0 0 0 0 0 0

### 6.8 Example Database

Examples and exercises in this training are based on some basic tables. The Example Database consists of the following Tables:

- CUSTOMER
  - o Customerid (PK)
  - FirstName
  - LastName
  - o Address
  - o Phone
  - PostCode
  - o PostAddress
- PRODUCT
  - Productid (PK)
  - ProductName
  - ProductDescription
  - o Price
  - ProductCode
- ORDER
  - Orderld (PK)
  - OrderNumber
  - o OrderDescription
  - CustomerId (FK)
- ORDER\_DETAIL
  - OrderDetailld (PK)
  - OrderId (FK)
  - ProductId (FK)

	Column Name	Data Type	Allow Nulls	1		Column Name	Data Type	Allow Nulls
P	CustomerId	int			8	ProductId	int	
	FirstName	varchar(50)				ProductName	varchar(50)	
_	LastName	varchar(50)				ProductDescription	varchar(50)	✓
	Address	varchar(50)	Image: A start of the start			Price	float	✓
	Phone	varchar(50)	Image: A start of the start			ProductCode	varchar(50)	✓
	PostCode	varchar(50)	<b>~</b>					
	PostAddress	varchar(50)	<b>~</b>					
<		1111 <b>1111</b>			<			
	PDER				<u> </u>			) ()
	RDER	8	)>		<u> </u>	DER_DETAIL	8	Allow Nulls
OR	Column Name	ģ	Allow Nulls		ORI	DER_DETAIL Column Name OrderDetailId	Data Type	Allow Nulls
OR	Column Name	Data Type	)>		ORI	Column Name	0 8 Data Type	Allow Nulls
OR	Column Name OrderId	Data Type int varchar(50)	Allow Nulls		ORI	Column Name OrderDetailId	Data Type	
OR	Column Name OrderId OrderNumber	Data Type	Allow Nulls		ORI	Column Name OrderDetailId OrderId	Data Type int int	

### 6.9 Exercises

#### **Exercise: New Database**

Create a new Database in MS SQL Server called TEST\_SQLSERVER.

#### **Exercise: Database Diagram**

Create the tables in the Example Database using the Diagram Designer Tool in Microsoft SQL Server.

#### **Exercise: Database Script**

Create the tables in the Example Database Tables using SQL Code. Save the Tables as a SQL Script file (.sql). Use The Query Tool in Microsoft SQL Server.

Exercise: ODBC

Create an ODBC connection for the Database.

## 7 Microsoft Office Access

### 7.1 Introduction

Microsoft Office Access, previously known as Microsoft Access, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the Microsoft Office suite of applications and is included in the Professional and higher versions for Windows. Access stores data in its own format based on the Access Jet Database Engine.

Microsoft Access is used by programmers and non-programmers to create their own simple database solutions.

Microsoft Access is a file server-based database. Unlike client-server relational database management systems (RDBMS), e.g., Microsoft SQL Server, Microsoft Access does not implement database triggers, stored procedures, or transaction logging. All database tables, queries, forms, reports, macros, and modules are stored in the Access Jet database as a single file. This makes Microsoft Access useful in small applications, teaching, etc. because it is easy to move from one computer to another.

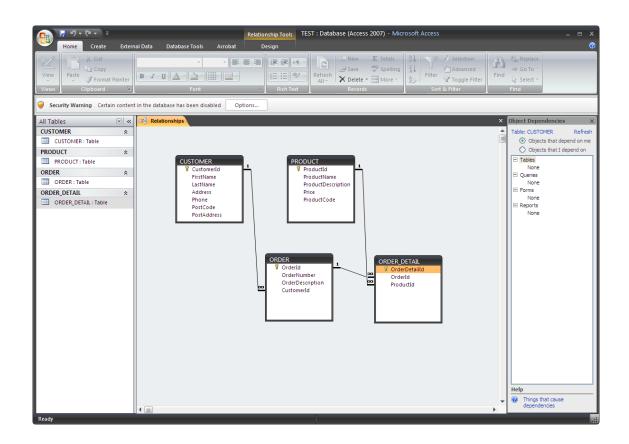
### 7.2 Example Database

I will present an example database in Microsoft Access 2007 which will be used in some of the examples and exercises in this document.

The database consists of the following tables:

- CUSTOMER
  - o Customerid (PK)
  - FirstName
  - LastName
  - o Address
  - o Phone
  - PostCode
  - PostAddress
- PRODUCT
  - Productid (PK)
  - ProductName

- ProductDescription
- o Price
- ProductCode
- ORDER
  - Orderld (PK)
  - OrderNumber
  - OrderDescription
  - Customerid (FK)
- ORDER\_DETAIL
  - OrderDetailId (PK)
  - OrderId (FK)
  - Productid (FK)



ODBC Connection:

Administrative Tools  $\rightarrow$  Data Sources (ODBC)

🐼 ODBC Data Source Adm	inistrator 🛛 💽 🔀							
User DSN System DSN File System Data Sources:	DSN   Drivers   Tracing   Connection Pooling   About   							
Name Default_Database LabVIEW Test Xtreme Sample Database 20	Driver       Add         National Instruments Citadel 5 [       Microsoft Access Driver (*.mdb)         Microsoft Access Driver (*.mdb)       Remove         08 Microsoft Access Driver (*.mdb)       Configure							
<	>							
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.								
	OK Avbryt Bruk Hjelp							

### 7.3 Exercises

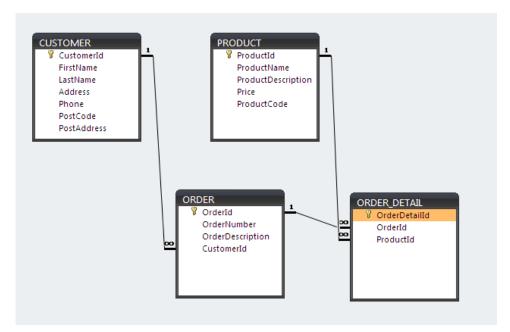
#### **Exercise: Database**

Create a new Database in MS Access called TEST.

B F H) + (¥ + → → Home Create Ext	ernal Data	Database Tools		e Tools TES	T : Database (A	ccess 2007) - 1	vicrosoft Acces		
View Paste Format Painte	Calibri		· ≡ ≡ 3		Refresh All *		Totals Spelling More →	Filter	Find Replace ⇒ Go To ~ Find
Security Warning Certain cont		_							
ll Tables	_	CustomerId -			Address •	Phone •		× PostAddres: • Add N	Object Dependencies
UUSTOMER CUSTOMER : Table RODUCT PRODUCT : Table DRDER ORDER : Table DRDER, DETAIL ORDER, DETAIL ORDER, DETAIL : Table	* *	* <u>New</u>							Table: LUSTONER Regent on Objects that i depend on Objects that i depend on Digets that i depend on Digets None El Roms None El Reports None
									Help
		Record: H + 1 of 1	IN N W N	Filter Search		Ш			<ul> <li>Things that cause dependencies</li> </ul>

#### **Exercise: Database Tables**

Create the tables in the Example Database Tables using the Diagram Designer Tool in Microsoft SQL Server.



### Exercise: ODBC

Create an ODBC connection for the Database.

## 80DBC

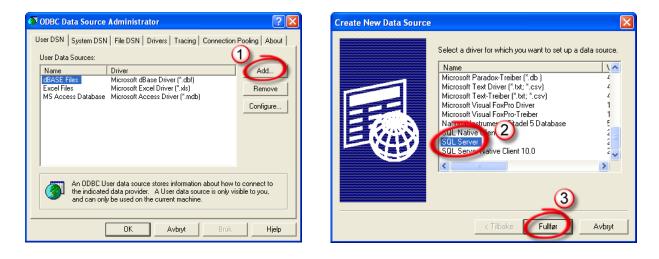
## 8.1 What is ODBC?

In computing, Open Database Connectivity (ODBC) provides a standard software API method for using database management systems (DBMS). The designers of ODBC aimed to make it independent of programming languages, database systems, and operating systems.

## 8.2 Create an ODBC Connection in "ODBC Data Source Administrator"

### Follow these steps:

Add a new Data Source and select the SQL Server driver:



Type a Name for your Connection and your SQL Server Name. You find your Server name as shown below:

Create a New Data Source to SQL Server	🧏 Microsoft SQL Server Management Studio
Select a dimension       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?       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:         Server:       PC88235\DEVELOPMENT          Fullier       Neste >	File       Edit       View       Tools       Window       Community       Help         Object       Explore       Image: Community       Image: Comm

Select SQL Server authentication and type your sa password (System Administrator). You defined the password for the sa user during the setup procedure of SQL Server:

Create a New Data Source to SQL Server	Create a New Data Source to SQL Server
Select driver to the login ID? With Windows NT 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 Client Configuration Client Configuration Connect to SQL Server to obtain default settings for the additional configuration options. Connect to SQL Server to obtain default settings for the additional configuration options. Connect to SQL Server to obtain default settings for the additional configuration options.	<ul> <li>Change the default database to:</li> <li>Change the default database</li></ul>
< Tilbake Neste > Avbryt Hjelp	<tilbake neste=""> Avbryt Hjelp</tilbake>

Complete your configuration and Test your data source to see if its OK:

Create a New Data Source to SQL Server	ODBC Microsoft SQL Server Setup
Create a New Data Source to SQL Server  Create a New Data Source to SQL Server  Change the language of SQL Server system messages to:  English  Change the language of SQL Server system messages to:  English  Use strong encryption for data  Use regional settings when outputting currency, numbers, dates and times.  Save long running queries to the log file:  CNDOCUME~T\hansha\LOCALS~T\Temp\QUER Browse  Long query time (milliseconds): 30000	ODBC Microsoft SQL Server Setup         Image: Configuration:           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 Long Running Settings: No           Use Integrated Security: No         Use Regional Settings: No           Prepared Statements Option: Drop temporary procedures on disconnect         Data Source View: No
C.\DOCUME~1\hansha\LOCALS~1\Temp\STATS     Browse      C.\DOCUME~1\hansha\LOCALS~1\Temp\STATS     Browse      C.Tilbake Fullfor Avbryt Hjelp	Use ANSI Quoted Identifiers: Yes Use ANSI Null, Paddings and Warnings: Yes Data Encryption: No Test Data Source OK Cancel

If you get this message you have succeeded:

SQL Server ODBC Data Source Test	×
Test Results	
Microsoft SQL Server ODBC Driver Version 03.85.1132	
Running connectivity tests	
Attempting connection Connection established Verifying option settings Disconnecting from server	
TESTS COMPLETED SUCCESSFULLY!	
(1)	
ОК	

# 8.3 Get data into Excel using your ODBC Connection

The purpose is to use Excel as a client and get data into Excel from your SQL Server.

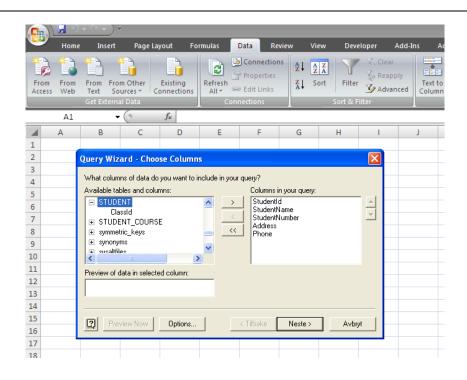
**Step 1:** Open Excel and go to the Data section:

	📙 າງ ·	· (° -	) <del>-</del>				$\frown$				Во	ok1 - Mi	crosoft	Excel								
<u> </u>	Home	Inse	rt Pa	age Layout	Forn	nulas 🧍	Data	Review	w Vi	iew	Deve	loper	Add-In:	s A	crobat	Team					0	) _ @
From Access	Web	Text 🔪	rom Oth			Refresh	Di Conne Propei Se Edit Li	rties		ort	Filter	V Clea	ply	Text to Column		es Validatio	n *	l Entropy of the second		Ungroup Su	111 - <sub>1</sub>	show Detai Hide Detail
_		Get Ext	75	From SQL Create a co				Inter Terre			Sort & F	ilter				Data To	ols			Out	line	
	G4			into Excel	as a Table	e or Pivot	Table repo	ort.	ort data													
1	A	В		From Ana Create a co Import dat	onnection	n to a SQ				ube.	н	1	1	I	К	L	Μ	N	0	Р	Q	R
2 3				From XML Open or m			Excel.															
4 5			×	From Data Import dat	ta for an i	unlisted t	ormat by	using th	e Data													
6 7 8		$\langle$		From Micr Import dat Ouery Wiz	rosoft Que ta for an u	e <mark>ry</mark> unlisted f		using th	e Micros	oft	>											
9		0									~											
10																						
11																						
12																						
13																						
14																						
15 16																						
10																						
18																						
19																						
20																						
21																						
22																						
12			1	Sheet3	100								Í									

Step 2: Select your ODBC connection

Ca		• (* • ) •								
<u> </u>	Home	Insert	Page La	ayout Fo	rmulas	Data Re	view	View [	Developer	Add-Ins A
Fro				Existing connections	Refresh All •	Connection Properties Edit Links nections	ns Art Art Art		Cle Kore Kore Ad & Filter	
	A1	•	0	$f_{x}$						
	А	В	С	D	E	F	G	Н	- I.	J
1		_								
2		Choose	Data Sour	ce						
3		_	,							
4		Datab	ases Queri	es   OLAP Cu	bes			OK		
5 6			Data Source sTest*	e>				Cance	I	
7		dBASI	E Files*					Browse.		
8		Detau Excel	lt_Database' Files*	ĸ						
9		LabVI MS Ar	EW* ccess Datab	200 <sup>×</sup>				Options.		
10		SCHO	OLTEST*					Delete		
11		Xtrem	e Sample Da	tabase 2008*						
12		2	✓ Use t	he Query Wiza	ard to create/	edit queries				
13			,							
14										
15										

Step 3: Select your Table(s)



### Step 4: Insert Data into Excel

Query Wizard - Finish	
What would you like to do next?	
Return Data to Microsoft Office Excel	Save Query
O View data or edit query in Microsoft Query	
2	< Tilbake Fullfør Avbryt

The results should look something like this:

	<b>   -</b> 19 - 1	(∾ • ) ⊽								
	Home	Insert	Page Lay	out	Formulas	Data	Review	View	Developer	Add
Pa	Stee Copy Copy Format Clipboard	Painter	Calibri B I L	- 1 [	• 🕭 • 🛕				Vrap Text Nerge & Cent	ter +
	K21	-	6	$f_{\mathcal{K}}$						
	А		В		С		D	E	F	G
1	StudentId 💌	Studen	tName 💌	Studen	tNumber 🔽	Address	s 🔽	Phone 💌		
2	3	Barak C	)bama	333333	33333	White H	louse 12	45667722		
3	2	Jens St	oltenberg	222222	22222	Pilstred	et 45	66778899		
4	1	John Cl	eese	1111111	11111	Pilstred	et 12	12345678		
5	4	Kurt Ni	lsen	4444444	14444	Karl Joh	an 34	44332277		
6										
7										
8										
9										
10										

## 9LabVIEW Database Connectivity Toolkit

LabVIEW offers an additional Toolkit called "LabVIEW Database Connectivity Toolkit". With this toolkit you can communicate with different databases, such as SQL Server, Oracle, etc.

Functions Palette: Connectivity → Database

Database					×
🔒 🔍 Search	S View 🕶				
DB	DB	DB	DB		
DB Tools Ope	DB Tools Clos	DB Tools Inse	DB Tools Sele	Utility	
DB 🗰	XIII DB			ADV •	
DB Tools Cre	DB Tools Dro	Database Var		Advanced	

The following list describes the main features of the Database Connectivity Toolkit:

- Works with any provider that adheres to the Microsoft ActiveX Data Object (ADO) standard.
- Works with any database driver that complies with ODBC or OLE DB.
- Maintains a high level of portability. In many cases, you can port an application to another database by changing the connection information you pass to the DB Tools Open Connection VI.
- Converts database column values from native data types to standard Database Connectivity Toolkit data types, further enhancing portability.
- Permits the use of SQL statements with all supported database systems, even non-SQL systems.
- Includes VIs to retrieve the name and data type of a column returned by a SELECT statement.
- Creates tables and selects, inserts, updates, and deletes records without using SQL statements.

Some of the text in this chapter is based on the "LabVIEW Database Connectivity Toolkit User Manual".

## 9.1 Connect to the Database

Before you can access data in a table or execute SQL statements, you must establish a connection to a database. You may use different methods in order to connect to the database:

- ODBC Data Source Name (DSN)
- Universal Data Link (UDL)
- Connection String

These different methods are explained below.

For all of these methods, you will use the same VI:

Context Help	X
DB Tools Open Connection.vi	^
userID connection information prompt? (f) error in (no error) password	
Opens a database connection using the connection information path and returns a connection reference. If <b>prompt?</b> is set to TRUE, LabVIEW displays a dialog box to set up the connection. Wire data to the <b>connection information</b> input to determine the polymorphic instance to use or manually select the instance.	
Detailed help	~
₫ <b>Ъ?</b> <	.::

Connecting to a database is where most errors occur because each database management system (DBMS) uses different parameters for the connection and different levels of security. The different standards also use different methods of connecting to databases. For example, ODBC uses Data Source Names (DSN) for the connection, whereas the Microsoft ActiveX Data Object (ADO) standard uses Universal Data Links (UDL) for the connection. The "DB Tools Open Connection.vi" VI supports all these methods for connecting to a database.

When you are finished with reading from the database and writing to the database, you should always close the Connection. Use the "DB Tools Close Connection.vi".

Context Help	×
DB Tools Close Connection.vi	^
connection reference	
error in (no error)	
Closes a database connection by destroying its associated <b>connection reference</b> .	
Detailed help	~
· 🕫 💩 🕐 🔇	

### 9.1.1 DSN

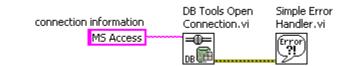
A **DSN (ODBC Data Source Name (DSN))** is the name of the data source, or database, to which you are connecting. The DSN also contains information about the ODBC driver and other connection attributes including paths, security information, and read-only status of the database. Two main types of DSNs exist: machine DSNs and file DSNs. Machine DSNs are in the system registry and apply to all users of the computer system or to a single user. DSNs that apply to all users of a computer system are system DSNs. DSNs that apply to single users are user DSNs. A file DSN is a text file with a .dsn extension and is accessible to anyone with proper permissions. File DSNs are not restricted to a single user or computer system. Use the ODBC Data Source Administrator to create and configure DSNs.

In the Control Panel, Administrative Tools, you find the ODBC Data Source Administrator tool.

Ø	ODBC D	ata Source /	dmini	strator				? 🛛
ſ	Jser DSN	System DSN	File DS	N Drivers	Tracing	Connect	ion Poolir	ng About
	System D	ata Sources:						
	Name			Driver				Add
	Default_	Database		National Ins	struments	Citadel 5 [	_	
	LabVIEV	•		Microsoft A				Remove
	Xtreme 9	Sample Databas	e 2008:	Microsoft A	ccess Driv	ver (*.mdbj		
								Configure
	<					>		
		An ODBC Sy	etem dal	ta source sto	res inform	ation about	t how to r	connect to
		the indicated	data pro	ovider. A Sy	stem data	source is t	visible to	all users
		on this machi	ne, inclu	uding NT serv	vices.			
		Г	ОК		\vbryt	Bru	ık 📘	Hjelp
		L	011					

### Example: DSN

This Example specifies a DSN called MS Access to open a connection to that specific database.



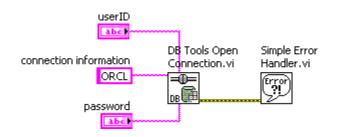
### Example: DSN from File

You can use a path to specify a file DSN. This example specifies a path to a file DSN called "access.dsn" to open a connection to the database.



### Example: DSN with UserId and Password

Most Database systems (DBMS – Database Management Systems) also require a UserId and a Password.



### 9.1.2 UDL

Whereas you must create a DSN to connect to a database using ODBC, you use **UDL (Universal Data Link)** to connect to databases that use ADO and OLE DB.

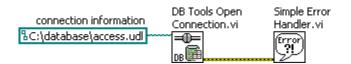
A UDL is similar to a DSN in that it describes more than just the data source. A UDL specifies what OLE DB provider is used, server information, the user ID and password, the default database, and other related information.

In order to create a new UDL file, create an empty text file and change the file extension of this document from .txt to .udl. You then can double-click the UDL file to display the Data Link Properties dialog box.

🗒 Datakoblingsegenskaper 🛛 🗙					
Tjeneste Tilkobling Avansert Alt					
Velg dataene du vil koble til:					
OLE DB-leverandør(er)					
MediaCatalogDB OLE DB Provider MediaCatalogMergedDB OLE DB Provider					
MediaCatalogWebDB OLE DB Provider Microsoft Jet 4.0 OLE DB Provider					
Microsoft Office 12.0 Access Database Engine OLE DB Pro-					
Microsoft OLE DB Provider for Analysis Services 9.0 Microsoft OLE DB Provider For Data Mining Services					
Microsoft OLE DB Provider for Indexing Service Microsoft OLE DB Provider for Internet Publishing					
Microsoft OLE DB Provider for ODBC Drivers					
Microsoft OLE DB Provider for OLAP Services 8.0 Microsoft OLE DB Provider for Oracle					
Microsoft OLE DB Provider for Search Microsoft OLE DB Provider for SQL Server					
Microsoft OLE DB Simple Provider					
MSDataShape					
Neste >>					
OK Avbryt Hjelp					

### **Example: UDL**

Connect to a Database using UDL:



### 9.1.3 Connection String

Rather than including an existing UDL in an application, you also can use an ODBC connection string with the Microsoft ActiveX Data Object (ADO) standard.

A connection string is written like this:

PROVIDER=SQLOLEDB;DATA
SOURCE=server name;UID=user name;PWD=password;DATABASE=database name;

You could use more parameters, but the parameters used above are the most common ones.

## 9.2 Reading Data from the Database

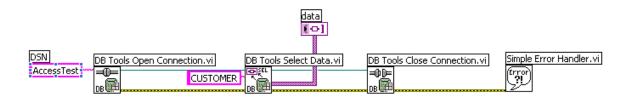
Reading data from a database table is similar to writing data to the database. You open a connection to the database, select the data from a table, and then close the connection.

The "DB Tools Select Data.vi" is use	d to read data from the Database:
--------------------------------------	-----------------------------------

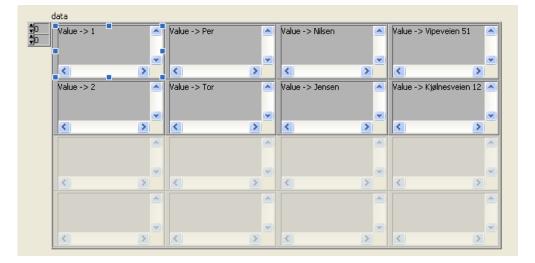
Context Help	×
DB Tools Select Data.vi	^
connection reference connection reference out table data columns columns colum	
Selects data from the <b>table</b> in the database identified by <b>connection reference</b> using the columns supplied in the <b>columns</b> array.	
Detailed help	~
₫ <b>Ъ</b> ?<	:

### **Example: Select Data from MS Access**

The following example gets data from the CUSTOMER table in MS Access.



The Front Panel looks like this:

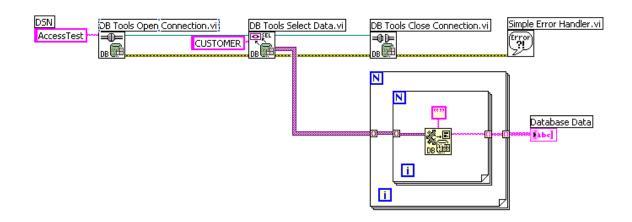


Notice in Figures 5-4 and 5-5 that the database data is returned as a two-dimensional array of variants. As the name implies, the Microsoft ActiveX Data Object (ADO) standard is based on ActiveX, which defines variants as its data types. Variants work well in languages such as Visual Basic that are not strongly typed. Because LabVIEW is strongly typed, you must use the Database Variant To Data

function to convert the variant data to a LabVIEW data type before you can display the data in standard indicators such as graphs, charts, and LEDs.

### **Example: Select Data from MS Access**

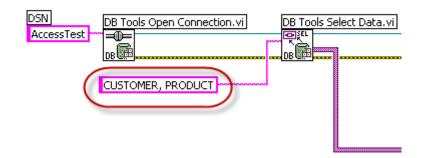
The following example gets data from the CUSTOMER table in MS Access and converts the data to text.



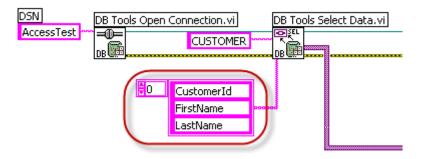
The Front Panel looks like this:

1	Per	Nilsen	Vipeveien 51	12345678	1234	Porsgrunn	1
2	Tor	Jensen	Kjølnesveien 12	45678932	5566	Bergen	

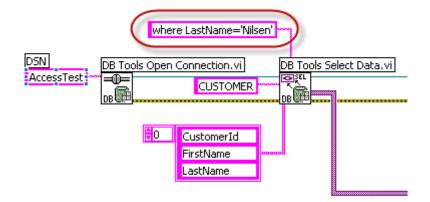
You may read from more than one table if you use a comma-delimited string to specify multiple table names:



You may select which columns you want to read by using the "Columns" input:

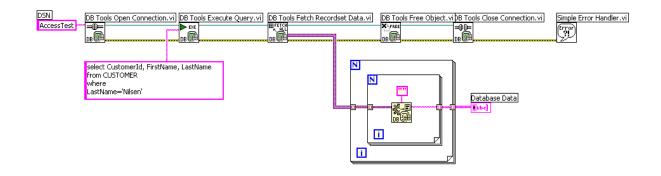


You may also restrict which data to receive using the "optional Clause" input:



### **Example: Read Data**

Using some VIs from the "Advanced" palette, create the following example:



### 9.3 Writing Data to the Database

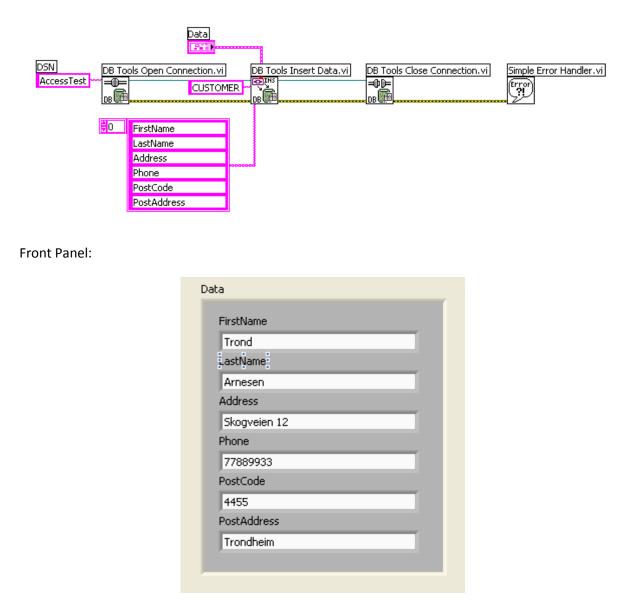
Writing data to a database with the LabVIEW Database Connectivity Toolkit is similar to reading data to a file. You open a connection, insert the data, and close the connection when you are finished.

The "DB Tools Insert Data.vi" is used to write data to the Database:

Context Help	X
DB Tools Insert Data.vi	^
create table? (f)	
connection reference out table columns columns columns columns flatten cluster? (f)	
Inserts a new row into the table in the database identified by the <b>connection reference</b> .	
Detailed help	~
₫ <b>₿</b> ?<	>:

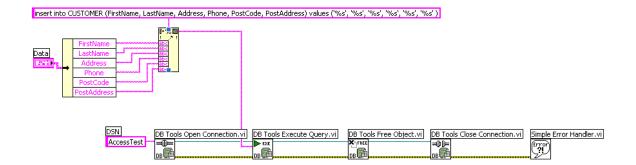
### **Example: Write Data**

Create the following block diagram:



### **Example: Write Data**

Create the following block diagram using some VIs from the "Advanced" palette.

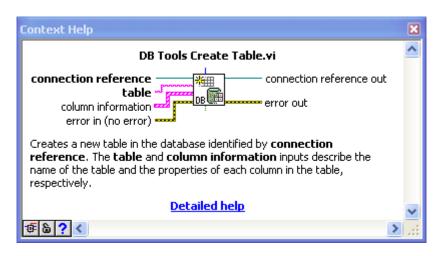


## 9.4 Creating and Dropping Tables

You may use standard SQL syntax in order to create:

CREATE TABLE <TableName> (...)

Or you may use the "DB Tools Create Table.vi" in order to create a table.



You may use standard SQL syntax in order to drop tables (delete tables):

```
DROP TABLE <TableName>
```

Or you may use the "DB Tools Drop Table.vi" in order to drop/delete a table.

Context Help	×
DB Tools Drop Table.vi	^
connection reference connection reference out table reference out error in (no error)	
Deletes the specified <b>table</b> from the database identified by <b>connection reference</b> .	
<u>Detailed help</u>	
······································	•

## 9.5 Using the Database Connectivity Toolkit Utility VIs

In the "**Utility**" palette there are several useful VIs for getting more information about tables, saving to text files, etc.

Utility		*************	×
🔒 🔍 Search	View 🔨		
			<b>−€</b> ≝∔
DB Tools List	DB Tools List	DB Tools Set	DB Tools Get
FEB 12	CSTR5		<b>⊞+</b> ⊞
DB	DB 🎆	DB 🎆	DB
DB Tools For	DB Tools Dat	DB Tools Sav	DB Tools Loa

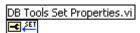
Here is a short description of the VIs located in the "Utility" palette:



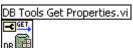
This VI lists the tables in the database identified by connection reference.

### DB Tools List Columns.vi

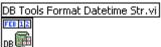
This VI lists the columns present in table. The column information includes the name, the data type, and the defined size of the column.



This VI sets properties on the object as determined by the inputs.



This VI gets properties of the object as determined by the inputs.



This VI Returns a string containing the formatted date and time, and identifies the string as a date/time string so other VIs can interpret it.



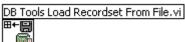
DB

This VI begins, commits, or rolls back a transaction for any type of

reference.

#### DB Tools Save Recordset To File.vi ♥→■

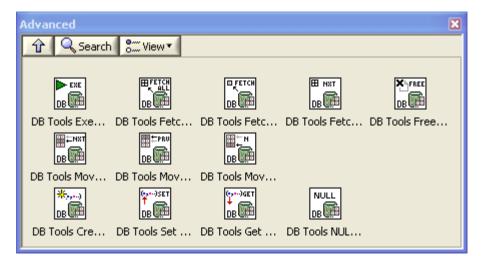
This VI saves the recordset identified by the recordset reference to either an XML or ADTG file. The ADTG file format is a proprietary format that only the LabVIEW Database Connectivity Toolkit can interpret. The ADTG format results in a smaller file than the XML format.



This VI loads a recordset from a file and returns a recordset reference that identifies this recordset. You can retrieve data from this recordset like any other recordset, but some properties might not be available on this recordset.

## 9.6 Performing Advanced Database Operations

When creating real programs you will soon need some of the VIs in the "Advanced" palette.



Here is a short description of some of the VIs located in the "Advanced" palette:



This VI Executes an SQL query and returns a recordset reference that you must eventually free with the DB Tools Free Object VI.



This VI retrieves the data in the recordset identified by the recordset reference input. You can convert each element in the array to its native LabVIEW data type using the "Database Variant To Data function".

This VI frees an object by destroying its associated reference and returns a

different reference object.

## 10 Creating and Using Tables

The SQL syntax for creating a Table is as follows:

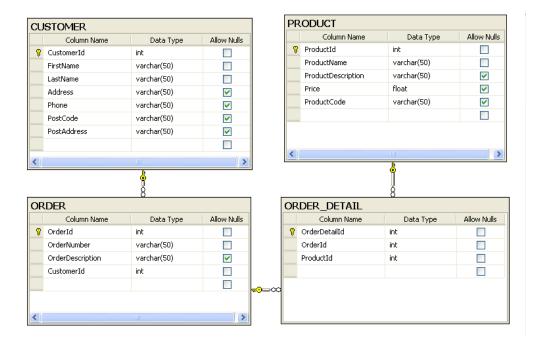
CREATE TABLE <TableName>
(
 <ColumnName> <datatype>
...
)

The SQL syntax for inserting Data into a Table is as follows:

INSERT INTO <TableName> (<Column1>, <Column2>, ...)
VALUES(<Data for Column1>, <Data for Column2>, ...)

### **Example: Insert Data into Tables**

We will insert some data into our tables:



The following SQL Query inserts some example data into these tables:

```
--CUSTOMER
INSERT INTO [CUSTOMER]
([FirstName], [LastName], [Address], [Phone], [PostCode], [PostAddress]) VALUES
('Per', 'Nilsen', 'Vipeveien 12', '12345678', '1234', 'Porsgrunn')
GO
INSERT INTO [CUSTOMER]
([FirstName],[LastName],[Address],[Phone],[PostCode],[PostAddress]) VALUES
('Tor', 'Hansen', 'Vipeveien 15', '77775678', '4455', 'Bergen')
GO
INSERT INTO [CUSTOMER]
([FirstName], [LastName], [Address], [Phone], [PostCode], [PostAddress]) VALUES
('Arne', 'Nilsen', 'Vipeveien 17', '12345778', '4434', 'Porsgrunn')
GO
--PRODUCT
INSERT INTO [PRODUCT]
([ProductName], [ProductDescription], [Price], [ProductCode]) VALUES ('Product
A', 'This is product A', 1000, 'A-1234')
GO
INSERT INTO [PRODUCT]
([ProductName], [ProductDescription], [Price], [ProductCode]) VALUES ('Product
B', 'This is product B', 1000, 'B-1234')
GO
INSERT INTO [PRODUCT]
([ProductName], [ProductDescription], [Price], [ProductCode]) VALUES ('Product
C', 'This is product C', 1000, 'C-1234')
GO
--ORDER
INSERT INTO [ORDER] ([OrderNumber], [OrderDescription], [CustomerId]) VALUES
('10001', 'This is Order 10001', 1)
GO
INSERT INTO [ORDER] ([OrderNumber], [OrderDescription], [CustomerId]) VALUES
('10002', 'This is Order 10002', 2)
GO
INSERT INTO [ORDER] ([OrderNumber], [OrderDescription], [CustomerId]) VALUES
('10003', 'This is Order 10003', 3)
GO
--ORDER DETAIL
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (1, 1)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (1, 2)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (1, 3)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (2, 1)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (2, 2)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (3, 3)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (3, 1)
GO
INSERT INTO [ORDER DETAIL] ([OrderId], [ProductId]) VALUES (3, 2)
GO
```

INSERT INTO [ORDER\_DETAIL] ([OrderId],[ProductId]) VALUES (3, 3)
GO

### Executing the following Queries then gives:

select \* from CUSTOMER

		📰 Results 🛅 Messages						
	 FirstName	LastName	Address	Phone	PostCode	PostAddress		
1 1	Per	Nilsen	Vipeveien 12	12345678	1234	Porsgrunn		
2 2	Tor	Hansen	Vipeveien 15	77775678	4455	Bergen		
3 3	Arne	Nilsen	Vipeveien 17	12345778	4434	Porsgrunn		

select \* from PRODUCT

🔠 Results 🛅 Messages					
	ProductId	ProductName	ProductDescription	Price	ProductCode
1	1	Product A	This is product A	1000	A-1234
2	2	Product B	This is product B	1000	B-1234
3	3	Product C	This is product C	1000	C-1234

### select \* from [ORDER]

🔢 Results 📑 Messages					
	OrderId	OrderNumber	OrderDescription	CustomerId	
1	1	10001	This is Order 10001	1	
2	2	10002	This is Order 10002	2	
3	3	10003	This is Order 10003	3	

select \* from ORDER\_DETAIL

🔲 F	🔠 Results 📑 Messages				
	OrderDetailId	Orderld	ProductId		
1	1	1	1		
2	2	1	2		
3	3	1	3		
4	4	2	1		
5	5	2	2		
6	6	3	3		
7	7	3	1		
8	8	3	2		
9	9	3	3		

## 10.1 Exercises

Run the queries above from LabVIEW.

# 11 Creating and Using Views

In database theory, a view consists of a stored query accessible as a virtual table composed of the result set of a query. Unlike ordinary tables in a relational database, a view does not form part of the physical schema: it is a dynamic, virtual table computed or collated from data in the database. Changing the data in a table alters the data shown in subsequent invocations of the view.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table
- Views can join and simplify multiple tables into a single virtual table
- Views can act as aggregated tables, where the database engine aggregates data (sum, average etc) and presents the calculated results as part of the data
- Views can hide the complexity of data; for example a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data it presents
- Depending on the SQL engine used, views can provide extra security
- Views can limit the degree of exposure of a table or tables to the outer world

Just as functions (in programming) can provide abstraction, so database users can create abstraction by using views. In another parallel with functions, database users can manipulate nested views, thus one view can aggregate data from other views.

Syntax:

CREATE VIEW <ViewName> AS ...

### Create a VIEW:

Step 1: Create a new View

😽 Microsoft SQL Server Manage	ement Studio	
File Edit View Tools Window	Community	Help
🏩 New Query 📑 📑 😭	3 🗠 🚽	
Object Explorer		- ↓ × 🚺 Ob
Connect 🕶 📑 📑 🔤 🖉 🛃	3	•
		Nam

### Step 2: Add your tables

疑 Microsoft SQL Server Management Studio	
File Edit View Debug Query Designer Tools Window Community Help	
😫 New Query 🕞 🕞 😂 🛃 🎒 🌉 👷	
C III 28 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Object Explorer • # × PC88235\DEVELT - dbo.View	● Object Explorer Details
Connect 🕶 🛃 🛃 🔳 🍸 🍒	<u>-</u>
Contemporary Cont	
🕫 🦢 System Databases	?X
Add Table      Database Diagrams	
Tables     Tables     Views     Functions     Synonyms	
H	
🗉 🔳 dbo.SCHOOL	×
	Output Sort Type Sort Or
🗷 🦢 Synonyms	
🗃 🧫 Security	
🗃 🦕 Replication	
Management	
Refresh Add	Close
	8
14 4 0 of 0 > >1	
Output	≁ # ×
Ready	

### Step 3: Add your columns

Server Managemen Managemen												
File Edit View Debug Query Design		s Window Com	munity He	elp								
🎎 New Query 📑 🗟 😂 🦪												
2 💷 🗷 🚦 📍 😽 🕼 🛅	÷											
Object Explorer 🚽 🗸 🗙	PC8	3235\DEVEL db	o.View_1*	Object Explo	orer Details					• ×	Properties	≁ ü ×
Connect • 📑 🛃 = 🍸 📓		E CLASS	_		CHOOL	-				^	[Col] ClassNa	ime 🔹
PC88235\DEVELOPMENT (SQL Server		* (All Columns)			All Columns)						8≣ 2↓ □	
Databases Image: System Databases		GassId		i Sch	vir Columns) icoliid						(Identity)	
E II TEST		<ul> <li>ClassName</li> </ul>		Sch 🖉	ioolName						(Name)	ClassName
😠 🧰 Database Diagrams		SchoolId	-	xo/							View Designation	
🖃 🧰 Tables											Allow Nulls	Yes
⊞ 📴 System Tables     ⊞ 💷 dbo.CLASS										_	Collation Data Type	Danish_Norwegla
	<									>	Length	50
🖃 🦳 Views											Precision	0
😠 🧰 System Views		Column	Alias	Table		Sort Type	Sort Order	Filter	Or		Scale	0
🗄 🧰 Synonyms 😠 🧰 Programmability	<u>۲</u>	SchoolName		SCHOOL	~						Size	50
Gervice Broker		ClassName		CLASS						~		
🗉 🧰 Security	<				1					>		
B 🔤 Securky B 🥁 Server Objects B 🔄 Replication B 🥁 Management	SELECT											
	<								1	>		
<		0 of 0								-		
Output										• 4 ×		
Output										<b>₩</b> 4 X		
											1	
											(Identity)	
Ready												

### Step 4: Save it

퉗 Microsoft SQL Server Management St	udio		
File Edit View Debug Query Designer	Tools Window Community Help		
😫 New Query 🕞 🕞 🥩 🗔 🕑 🖼			
Object Explorer 🗸 🗣 🔪	PC88235\DEVEL dbo.View_1* Object Explorer Details ->	Properties	<b>*</b> ‡ ×
Connect 🕶 📑 📑 📲 🖉 🍒 📃 🔪	🔲 CLASS 🔲 🔳 SCHOOL 📕	[Col] ClassNa	ame 🔹
E 6 PC88235\DEVELOPMENT (SQL Server	* (Al Columns)	- 121 ⊡	
Databases Image: Section 2 and Sec	GlassId /2 Schoolid	(Identity)	
E TEST	Clastitame	(Name)	ClassName
🗉 🚞 Database Diagrams		Allow Nulls	gner Yes
Tables Tables System Tables		Collation	Danish Norwegian
🗉 💷 dbo.CLASS		Data Type	varchar
🗄 🛄 dbo.SCHOOL 🧃		Length	50
🖃 🧰 Views 🛞 🚞 System Views	Column Allas Table Output Sort Tupe Sort Order Filter Or	Precision     Scale	0
E Synonyms	SchoolName Choose Name 🔀	Size	50
Programmability	ClassName		
🖲 🧰 Service Broker	Citical difficition of the view.	~	
🗷 🦢 Security 💦 👘	LECT dbo.SCHOOL.SchoolNan		
	OM dbo. CLASS INNER JOIN dbo. SCHOOL ON dbo		
Kepication     Management			
		_	
		-	
<		>	
< > I*	4 4 0 of 0 b b b b a 0		
Output	+ ₽ ;	×	
		(Identity)	
			<b>Na</b>

Using the VIEW in a Query:

😽 Microsoft SQL Server Management	Studio		
File Edit View Query Debug Tool	; Window Community Help		
😫 New Query 📑 📑 🚰 🛃	🕰 <u>-</u>		
	! Execute ▶ = ✔ \$7		
Object Explorer	PC88235\DEVELT - dbo.CLA55 SQLQuery1.sqlTE5T (sa (52))* PC88235\DEVELT - dbo.MyView	= ×	Properties 🚽 🕂 🗙
Connect - 📑 💐 = 🍸 🍒	select * from MyView	_	Current connection parameters -
PC88235\DEVELOPMENT (SQL Server			8 2↓ □
🖃 🧰 Databases			
🗉 🧰 System Databases			Aggregate Status     Connection fa
🖃 间 TEST 😠 🧰 Database Diagrams			Elapsed time 00:00:00.0312500
Database Diagrams     Tables			Finish time 04.05.2010 14:13
표 🚞 System Tables			Name PC88235\DEVELO
😟 💷 dbo.CLASS			Rows returne 2
🗄 📃 dbo.SCHOOL			Start time 04.05.2010 14:13
E 🚞 Views E 🗀 System Views		*	State Open
	<	>	Connection Connection n: PC88235\DEVELO
			Connection Details
🗉 🚞 Programmability		]	Connection el 00:00:00.031250
🗉 🧰 Service Broker	SchoolName ClassName		Connection fir 04.05.2010 14:13
🗈 🧰 Security	1 TUC SCE		Connection rc 2
Security     Security     Server Objects	2 TUC SCE2		Connection st 04.05.2010 14:13
E Replication			Connection st Open
🗄 🧰 Management			Display name PC88235\DEVELO
			Login name sa
			Server name PC88235\DEVELO Server versior 10.0.2531
			SPID 52
			5/10 52
	Query executed successfully. PC88235\DEVELOPMENT (10.0 SP1) sa (52) TEST 0	0:00:00 2 rows	
Output		<b>→</b> ╄ ×	
			Name The name of the connection.
			momane of the connection.
		- 1 - 1	
Ready	Ln 1	Col 21	Ch 21 INS .:

## 11.1 Exercises

Create a simple view based on the example tables and run the view from LabVIEW.

## 12 Creating and using Stored Procedures

A stored procedure is a subroutine available to applications accessing a relational database system. Typical uses for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures are used to consolidate and centralize logic that was originally implemented in applications. Large or complex processing that might require the execution of several SQL statements is moved into stored procedures, and all applications call the procedures only.

A stored procedure is a precompiled collection of SQL statements and optional control-of-flow statements, similar to a macro. Each database and data provider supports stored procedures differently. Stored procedures offer the following benefits to your database applications:

**Performance**—Stored Procedures are usually more efficient and faster than regular SQL queries because SQL statements are parsed for syntactical accuracy and precompiled by the DBMS when the stored procedure is created. Also, combining a large number of SQL statements with conditional logic and parameters into a stored procedure allows the procedures to perform queries, make decisions, and return results without extra trips to the database server.

**Maintainability**—Stored Procedures isolate the lower-level database structure from the application. As long as the table names, column names, parameter names, and types do not change from what is stated in the stored procedure, you do not need to modify the procedure when changes are made to the database schema. Stored procedures are also a way to support modular SQL programming because after you create a procedure, you and other users can reuse that procedure without knowing the details of the tables involved.

**Security**—When creating tables in a database, the Database Administrator can set EXECUTE permissions on stored procedures without granting SELECT, INSERT, UPDATE, and DELETE permissions to users. Therefore, the data in these tables is protected from users who are not using the stored procedures.

Stored procedures are similar to user-defined functions. The major difference is that functions can be used like any other expression within SQL statements, whereas stored procedures must be invoked using the CALL statement.

The syntax for creating a Stored Procedure is as follows:

```
CREATE PROCEDURE <ProcedureName>
@<Parameter1> <datatype>
```

### **Example: Create a Stored Procedure**

This Procedure gets Customer Data based on a specific Order Number.

```
IF EXISTS (SELECT name
       FROM sysobjects
       WHERE name = 'sp_CustomerOrders'
       AND type = 'P')
     DROP PROCEDURE sp CustomerOrders
GO
CREATE PROCEDURE sp CustomerOrders
@OrderNumber varchar(50)
AS
/*-----
                              _____
Last Updated Date: 2009.11.03
Last Updated By: hans.pr.halvorsen@hit.no
Last Updated By:
Description:
                   Get Customer Information from a specific Order Number
-----*/
SET NOCOUNT ON
declare @CustomerId int
select @CustomerId = CustomerId from [ORDER] where OrderNumber = @OrderNumber
select CustomerId, FirstName, LastName, [Address], Phone from CUSTOMER where
CustomerId=@CustomerId
SET NOCOUNT OFF
GO
```

#### **Example: Using a Stored Procedure**

Using the Stored procedure like this

exec sp CustomerOrders '10002'

gives the following result:

🔲 Results	🛅 Messag	es		
Custo	merld   FirstN	lame   LastNam	e Address	Phone
1 2	Tor	Hansen	Vipeveien 15	77775678

## 12.1 Exercises

Run the Stored Procedure created above from LabVIEW.

# 13 Creating and Using Triggers

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for keeping the integrity of the information on the database. For example, when a new record (representing a new worker) added to the employees table, new records should be created also in the tables of the taxes, vacations, and salaries.

Triggers are commonly used to:

- prevent changes (e.g. prevent an invoice from being changed after it's been mailed out)
- log changes (e.g. keep a copy of the old data)
- audit changes (e.g. keep a log of the users and roles involved in changes)
- enhance changes (e.g. ensure that every change to a record is time-stamped by the server's clock, not the client's)
- enforce business rules (e.g. require that every invoice have at least one line item)
- execute business rules (e.g. notify a manager every time an employee's bank account number changes)
- replicate data (e.g. store a record of every change, to be shipped to another database later)
- enhance performance (e.g. update the account balance after every detail transaction, for faster queries)

The major features of database triggers, and their effects, are:

- do not accept parameters or arguments (but may store affected-data in temporary tables)
- cannot perform commit or rollback operations because they are part of the triggering SQL statement
- can cancel a requested operation
- can cause mutating table errors, if they are poorly written.

Microsoft SQL Server supports triggers either after or instead of an insert, update, or delete operation.

The syntax is as follows:

```
CREATE TRIGGER <TriggerName> on <TableName>
FOR INSERT, UPDATE, DELETE
AS
... Create your Code here
GO
```

- Replace <TriggerName> with the Name of your Trigger
- Replace <TableName> with the Name of your Table

Define when the Trigger should be execute

- If the Trigger should be executed only when you insert data into the table: FOR INSERT
- If the Trigger should be executed only when you update data into the table: FOR UPDATE
- If the Trigger should be executed only when you delete data into the table: FOR DELETE
- If the Trigger should be executed when you insert and update data into the table: FOR INSERT, UPDATE
- Etc.

#### **Example: Trigger**

The Example above change the "below" in the Table "SCHOOL" from 'TUC' to 'Telemark University College'

```
CREATE TRIGGER CheckSchoolData on SCHOOL
FOR INSERT, UPDATE
AS
DECLARE
@SchoolName varchar(50)
select @SchoolName=SchoolName from INSERTED
If @SchoolName='TUC'
update SCHOOL set SchoolName='Telemark University College' where
SchoolName=@SchoolName
```

GO

**Note!** Note the use of a temporary table called "**INSERTED**". This temporary table contains the last inserted record into the SCHOOL table

Note! In SQL you define a variable like this

DECLARE @myVariable <datatype>

Example:

```
DECLARE
@SchoolName varchar(10)
```

64

### Note! You have to use the symbol "@" before the name of the variable !!!

Nicrosoft SQL Server Management Studio	
File Edit View Query Debug Tools Window Community Help	
👱    🕞 📓 📽 😹 🖉 🚬 🕐 Insert the Trigger in your Database system by clicking Execute	
21 U TEST · I Decude → = ◇ 23 単目 21 単词 43 個句 三 2 単単 45 。	
Object Explorer • 4 × PC68235(DEVELT-dbo.SCHOOL SQLQuery2.sqlTEST (sa (54))* SQLQuery1.sqlTEST (sa (53))*	Properties 👻 🖡 🗙
Connet + 🔢 🛃 = 🝸 📓 CREATE TRIGGER CheckSchoolData on SCHOOL	Current connection parameters -
B CREESSIDEVELOPMENT (SQL Server 10.0.2531 - 15	
🖃 🦲 Databases	Aggregate Status
B System Databases DECLARE DECLARE	Connection fa
B Database Diagrams (85cho) Market var char (50)	Elapsed time 00:00:00.0156250
	Finish time 04.05.2010 15:30 Name PC88235\DEVELO
B _ System Tables	Rows returne 0
a do. SCHOOL	Start time 04.05.2010 15:30
B Columns If @SchoolName='TUC'	State Open
Keys     update SCHOOL set SchoolName='Telemark University College' where SchoolName=@SchoolName	Connection
	Connection n: PC88235\DEVELO
☑ ChedSchooData	Connection el 00:00:00.0156250
<sup>™</sup> Your Trigger will be	Connection fir 04.05.2010 15:30
	Connection rc 0
	Connection st 04.05.2010 15:30 Connection st Open
🗄 🔁 Programmability <	Display name PC88235\DEVELO
B genty      Bounty     Bounty     Bounty	Login name sa
Security     Command(s) completed successfully.	Server name PC88235\DEVELO
🗷 🦢 Server Objects	Server versior 10.0.2531 SPID 53
B ■ Replication B ■ Management 3 If everything is OK you get this message	SPID 53
	1
Query executed successfully.     PC68235(DEVELOPMENT (10.0 SP1) sa (53) TEST 00:00:00 0 rows	
Output 🗸 🗸 🗸 🗸 🗸	c
Auto-attach to process '[836] [SQL] PC88235' on machine 'PC88235' succeeded.	
The thread 'PC88235\DEVELOPHENT [53]' (0x22e8) has exited with code 0 (0x0).	
The program '[836] [SQL] PC88235: PC88235:DEVELOPMENT' has exited with code 0 (0x0).	
	Name
	The name of the connection.
	2
Ready Ln 14 Col 3	Ch 3 INS .:

Below we see how we create a Trigger from the "SQL Server Management Studio":

### Check if the Trigger is working as expected:

Procedure:

Step 1: Check the data in your table before you do anything, e.g.:

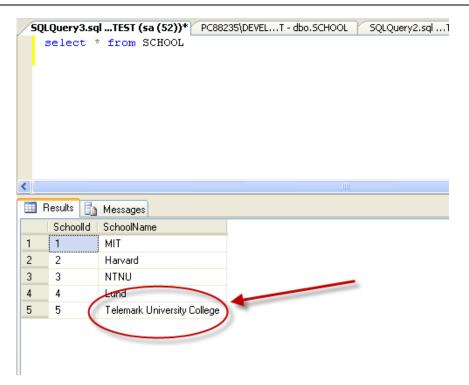
select \* from SCHOOL

Step 2: Insert some test data into your table, e.g.:

insert into SCHOOL (SchoolId, SchoolName) values (5, 'TUC')

Step 3: Check the data has been updated according to your code in the Trigger:

select \* from SCHOOL



 $\rightarrow$  As you see the data you inserted into the table has been automatically been changed by the Trigger

## 13.1 Exercises

Create a Trigger that adds "+47" to all Phone numbers in the CUSTOMER table.

Test and see if the Trigger works properly by inserting and updating some data in the CUSTOMER table.

## 14 Creating and using Functions

In SQL databases, a user-defined function provides a mechanism for extending the functionality of the database server by adding a function that can be evaluated in SQL statements. The SQL standard distinguishes between scalar and table functions. A scalar function returns only a single value (or NULL), whereas a table function returns a (relational) table comprising zero or more rows, each row with one or more columns.

Stored Procedures vs. Functions:

- Only functions can return a value (using the RETURN keyword).
- Stored procedures can use RETURN keyword but without any value being passed[1]
- Functions could be used in SELECT statements, provided they don't do any data manipulation and also should not have any OUT or IN OUT parameters.
- Functions must return a value, but for stored procedures this is not compulsory.
- A function can have only IN parameters, while stored procedures may have OUT or IN OUT parameters.
- A function is a subprogram written to perform certain computations and return a single value.
- A stored procedure is a subprogram written to perform a set of actions, and can return multiple values using the OUT parameter or return no value at all.

User-defined functions in SQL are declared using the CREATE FUNCTION statement.

### 14.1 Exercises

Create a simple function that finds number of order for a specific customer and use it in the following query:

"Select FirstName, LastName, fn\_NumberOfOrders(CustomerId) from CUSTOMER"

## 15 SQL Toolkit

I have made a simple and easy to-use SQL Toolkit. The SQL Toolkit is available for download from my Blog: <a href="http://home.hit.no/~hansha/">http://home.hit.no/~hansha/</a>

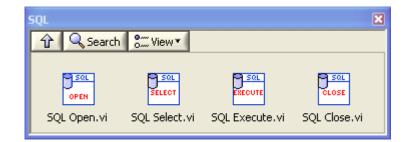
## 15.1 Installation

The installation procedure is as follows:

- 1. Download the zip file SQLToolkit.zip from my Blog
- 2. Unzip the file
- 3. Copy "SQLToolkit.mnu" to ...\LabVIEW 2009\menus\Categories\
- 4. Copy "SQLToolkit.llb" to ...\LabVIEW 2009\vi.lib\
- 5. The SQL Toolkit is ready to use and in the Functions palette in LabVIEW a new palette named "SQL" will appear.



The SQL Toolkit palette in available in LabVIEW:



The SQL Toolkit contains the following VIs:

### 

**SQL Open.vi**" - This VI open a connection to the database specified in the Connection string. The connection string may be as follows:

"PROVIDER=SQLOLEDB; DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=xxx"

You need to replace the "xxx" with the parameters from your database.

### SQL

"SQL Select.vi" -This VI get data from the database specified in the SQL Query. The output is a 2D string array with data.

**SQL Execute.vi**" - This VI executes a Query with no return Data, e.g., an INSERT statement

**SQL Close.vi**" - This VI Close the connection to the database opened by "SQL Open.vi"

Two examples are also included:

"SQL – Example 1.vi" – This example selects data from a table. The example uses "SQL Select.vi" in order to get data from the database.

Front panel:

😰 SQL - Example1.vi Front Panel								
Elle Edit View Project Operate Tools Window Help								
◇ ② ● II 13pt Application Font  マ ジーマ デーマ ジーマ								
							^	
Hans-Petter Halvorsen hans.p.halvorsen@hit.no Telemark University College Department of Electrical Engineering, Information Tec	:hnology and	I	ト 古 雪 茶 Høgskolen i T	elemark				
SQL Query		Data						
select * from CUSTOMER	() o	1	Per	Nilsen	Vipeveien 12	12345678		
	() o	2	Tor	Hansen	Vipeveien 15	77775678		
	Ť	3	Arne	Nilsen	Vipeveien 17	12345778		
		4	Jonr	Blund	Vipeveien 14	12366678		
							~	
<							<b>&gt;</b>	

Block Diagram:

SQL - Example1.vi Block Diagram	
Elle       Edit       Yiew       Project       Operate       Iools       Window       Help         Image: State of the	2 E8-1
SQL Query     Data       SQL Query     Data       Labe P     SQL Query       PROVIDER=SQLOLEDB; DATA SOURCE=xxx; JUID=xxx; PWD=xxx; DATABASE=TEST     SQL Query	(tro)
	× > ;;

E8-2

"SQL – Example 2.vi" – This example inserts data into a table. The example uses "SQL Execute.vi" in order to insert data into the database. No data is returned.

Front panel:

😫 s	iQL - Example2.vi Front P	anel		×
Eile	<u>E</u> dit <u>V</u> iew <u>P</u> roject <u>O</u> perate	<u>T</u> ools <u>W</u> indow <u>H</u> elp		X-2
	수 🕹 🔵 💵 13pt 4	pplication Font 🔍 🚛 🖬 🖬 🕊	2	n-c
				^
-	Hans-Petter Halvorse hans.p.halvorsen@hit.no	n Tig	) <u> </u>	
Т	Felemark University College		Høgskolen i Telemark	
	Department of Electrical Engi	neering, Information Technology and Cybernetics		
	SQL Query			
		stName],[LastName],[Address],[Phone],[PostCode],[Post	Address])	Ξ
	VALUES ('Jonr', 'Blund', 'Vipeveia	n 14', '12366678', '1234', 'Porsgrunn')		
				~
<			>	1.11

### Block Diagram:

SQL - Example 2. vi Block Diagram	
Eile Edit View Project Operate Tools Window Help	88-2
SQL Query	≡.
PROVIDER=SQLOLEDB; DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=TEST	
	<ul><li>✓</li><li>&gt; .::</li></ul>

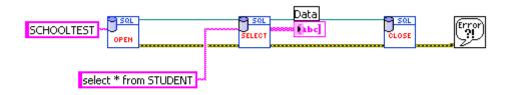
### Example: Get Data into LabVIEW using SQL Toolkit

Download the SQL toolkit from the <u>Homepage of the Database Lab</u> and follow the instructions in the ReadMe file.

On the Functions palette on your Block Diagram the following palette should appear:

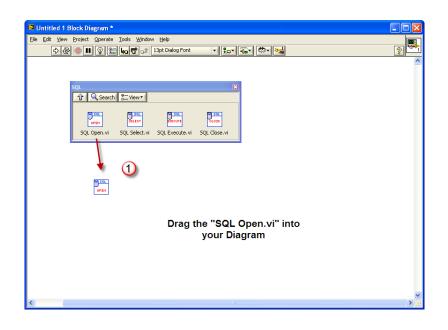
Statechart				
SQL	SQL	SQL Select.vi	SQL Execute.vi	SQL Close, vi

Here is a simple example of how you get data from the database into LabVIEW:

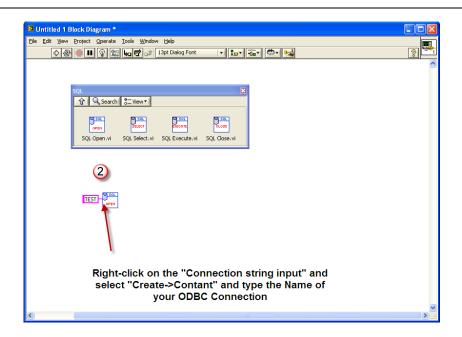


The procedure is as follows:

### Step 1:



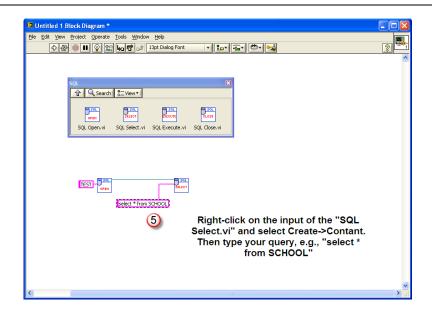
Step 2:



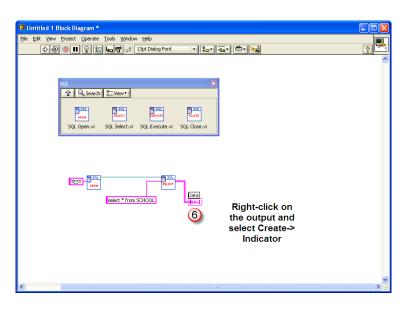
### Step 3 and 4:

12 Untitled 1 Block Diagram *	
Ele Edit Yew Broject Operate Iools Window Help 수 없	2
Storent SQL Select.vi Together Connect the "SQL Select.vi" together	~
	> .::

### Step 5:



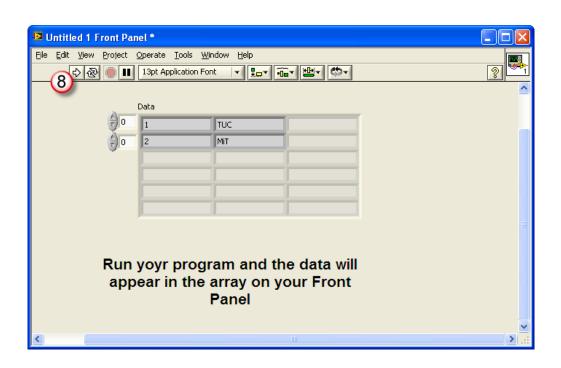
### Step 6:



Step 7:

E Untitled 1 Block Diagram * Ele Edit Vjew Project Operate Iools Window Help	
다 같다. 가 아 다 아 다 아 가 아 가 아 가 아 가 아 가 아 가 아 가 아	2
SOL SOL SOL SOL Open.vi SQL Select.vi SQL Execute.vi SQL Close.vi SOL Open.vi SQL Select.vi SQL Execute.vi SQL Close.vi Belect * from SCHOOL Select * from SCHOOL	
	<u> </u>

### Step 8:





**Telemark University College** 

Faculty of Technology

Kjølnes Ring 56

N-3914 Porsgrunn, Norway

www.hit.no

Hans-Petter Halvorsen, M.Sc.

**Telemark University College** 

Department of Electrical Engineering, Information Technology and Cybernetics

Phone: +47 3557 5158

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

Blog: <u>http://home.hit.no/~hansha/</u>

Room: B-237a