



# Using the database connection

Chris Kuip

Webinar 19 November 2014

# Using the Database Connection, demo based webinar

---

1. Basic: Minimal startup
  1. Declare database table and connect
  2. Read from table and write to table
2. Intermediate: Row filtering
3. Intermediate: Instance selection
4. Intermediate: Making the connection flexible
  1. String parameters for connection and table
  2. Mapping column names to AIMMS identifiers
5. Advanced: leveraging database procedures
6. Advanced: Data Protected and environment query
  1. Create connection function
  2. Directly executing SQL commands
7. (No demo) Advanced: Querying environment

# 1. Database table for ProductDatabase

---

Database

Key column	Non key column
ProductData	
ProductName	ExpectedDemand
Laptops	3
Radios	10
TVs	20
*	0
Set	Parameter

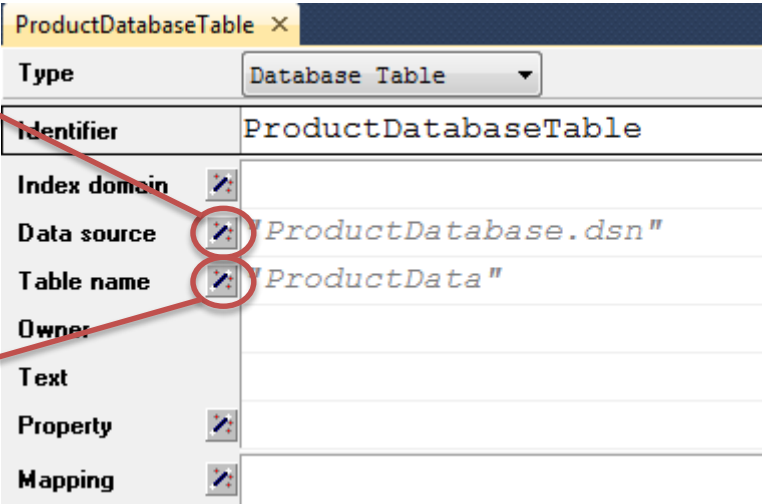
AIMMS

## Summary demo 1:

- Create an AIMMS identifier of type database table and connect it:

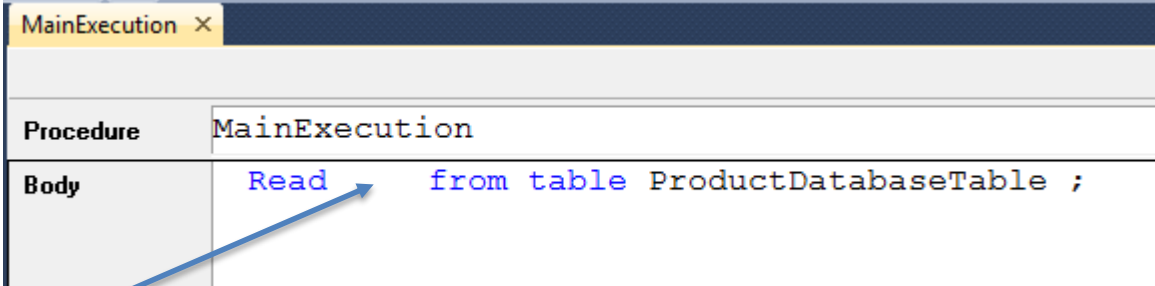
Select from available connections

Select from available tables



ProductDatabaseTable x	
Type	Database Table
Identifier	ProductDatabaseTable
Index domain	
Data source	ProductDatabase.dsn
Table name	ProductData
Owner	
Text	
Property	
Mapping	

- Read from that table:



MainExecution x	
Procedure	MainExecution
Body	Read from table ProductDatabaseTable ;

Column names are mapped to AIMMS identifiers with the same name

## 2. Intermediate: Row filtering

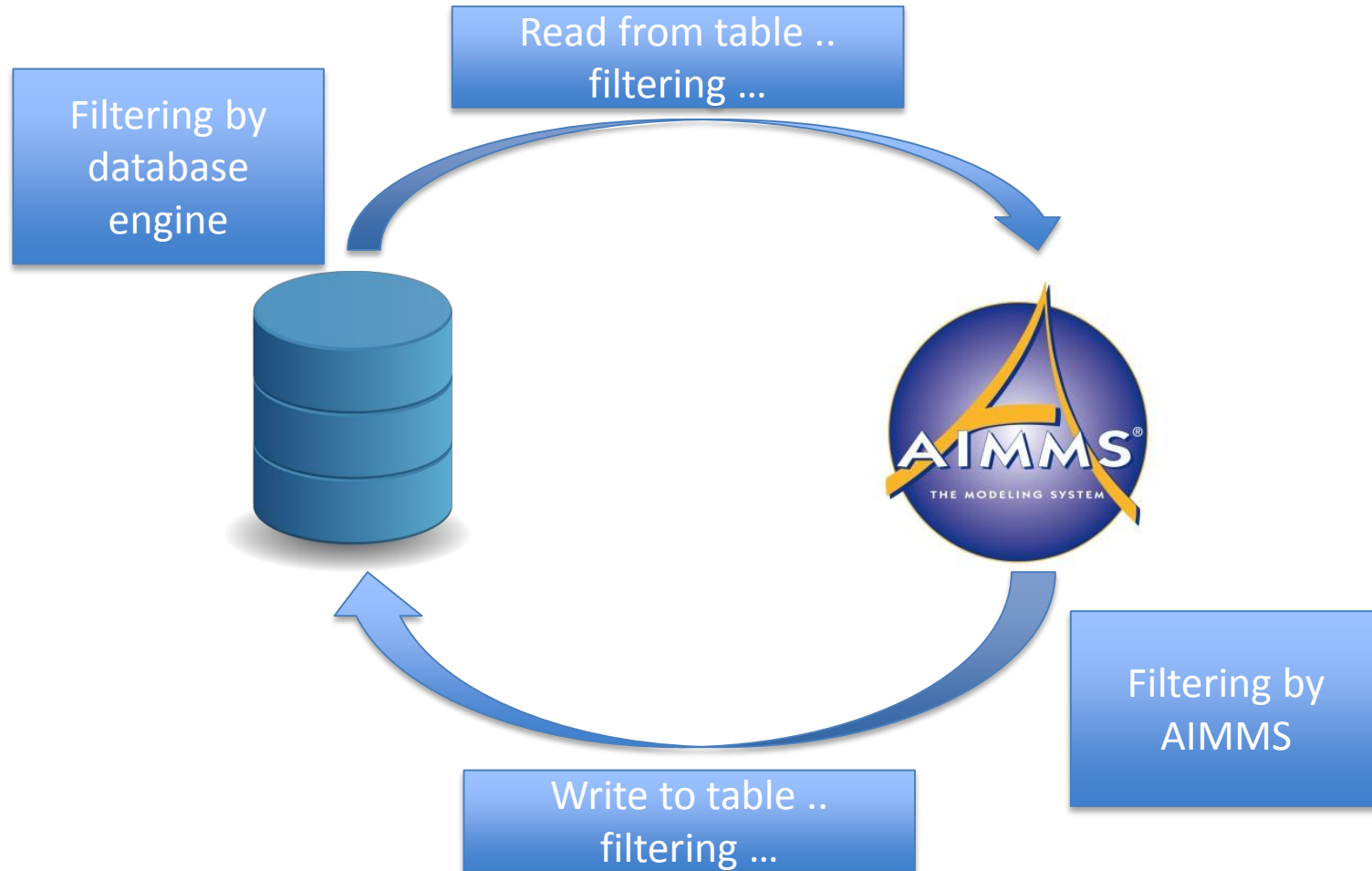
ProductName	ExpectedDemand
Table	1
Chair	5
Radios	10
Laptops	12
TVs	20

Selected rows

```
! Set ProductName = data { Table, Chair }  
read from table ProductTable  
  filtering ProductName ;
```

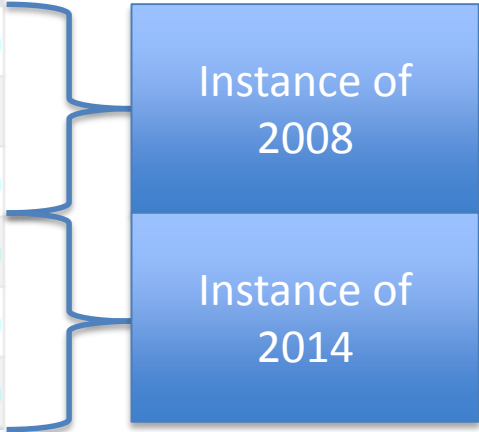
### 3. Filtering at origin: reduce communication overhead

---



### 3. Intermediate: Instance selection

YearNumber	ProductName	ExpectedDemand
2008	Radios	10
2008	Laptops	12
2008	TVs	20
2014	Phones	70
2014	Laptops	50
2014	TVs	20



Obtain collection of instances:

```
Read YearNumber from Table ArchiveProductDatabaseTable ;
```

Select an instance:

```
e_YearNo := first( YearNumber );
```

Actually read that data instance:

```
Read ExpectedDemand from
```

```
Table IndexedArchiveProductDatabaseTable (e_YearNo);
```



## 4. Make connection flexible; use string parameters

---

- Use a string parameter to control the database selected; when the AIMMS app is installed at a different location, the database connection may have a different name as well.

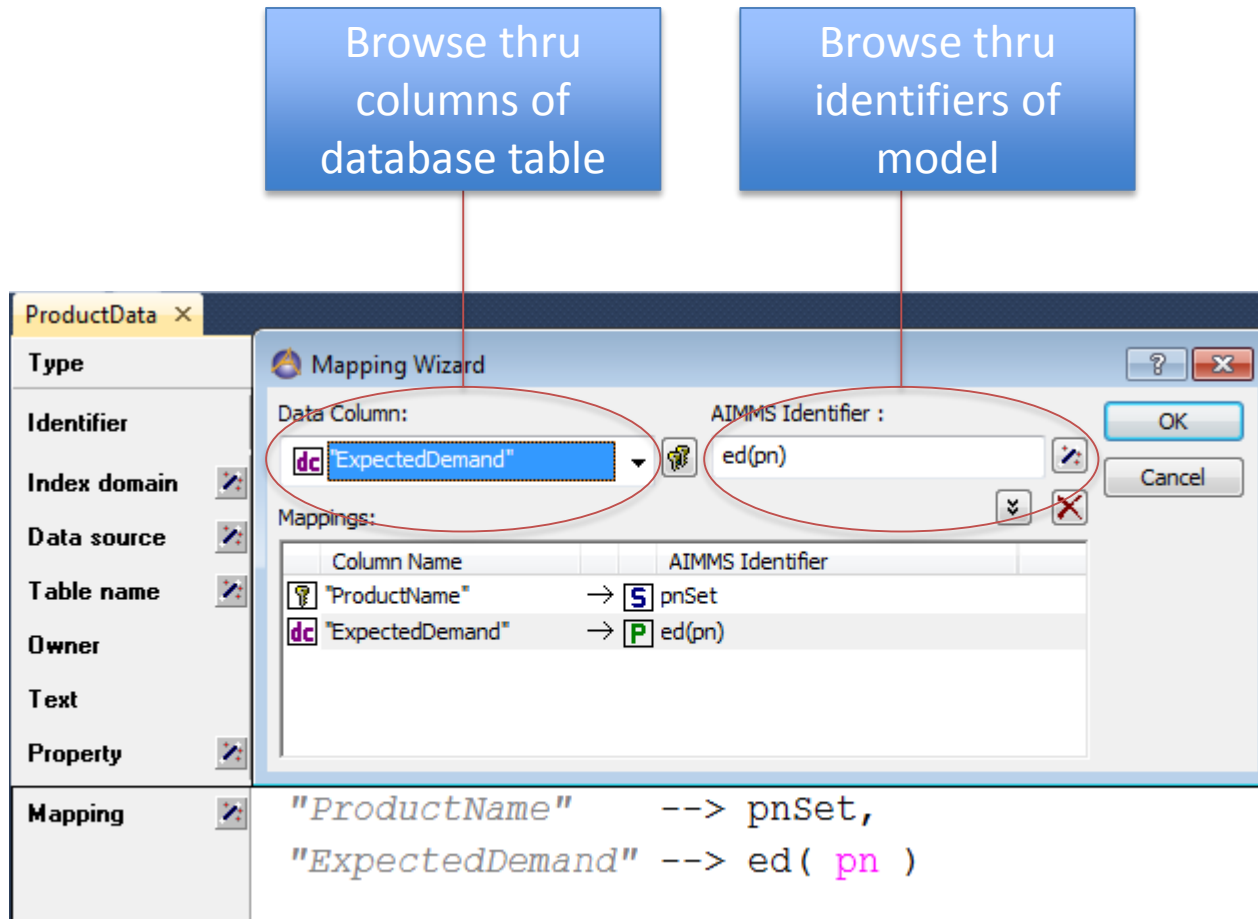


## 4. Intermediate: Making connection flexible

- Use a mapping attribute to map database columns to AIMMS identifiers. For instance, a space is allowed in a database column name, but not in the name of an AIMMS identifier.

Browse thru columns of database table

Browse thru identifiers of model



Mapping Wizard

Data Column: dc "ExpectedDemand"

AIMMS Identifier : ed(pn)

Mappings:

Column Name	AIMMS Identifier
"ProductName"	pnSet
"ExpectedDemand"	ed(pn)

Mapping

```
"ProductName" --> pnSet,  
"ExpectedDemand" --> ed( pn )
```

## 5. Advanced: Leveraging database procedures

- The result set of SQL SELECT queries can be used to read data

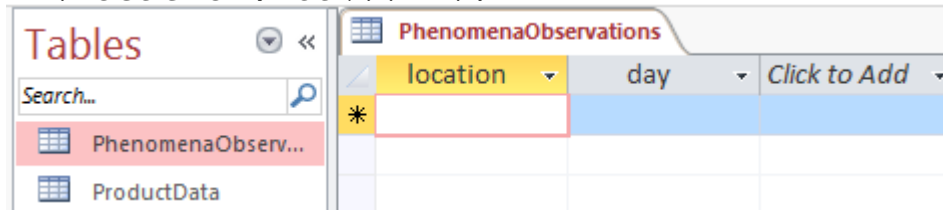
ReadByExecutingQuery ×	
Database procedure	ReadByExecutingQuery
Arguments	
Data source	"ProductOrders.dsn"
<input checked="" type="radio"/> Sql query <input type="radio"/> Stored procedure	<pre>"SELECT Products.ProductName, Sum(Orders.Quantity) AS SumOfQuantity "+ "FROM Products INNER JOIN Orders ON Products.ID = Orders.[Product id] "+ "GROUP BY Products.ProductName;"</pre>
Owner	
Property	UseResultSet
Mapping	<pre>"ProductName" --&gt; ProductName, "SumOfQuantity" --&gt; AmountOrdered</pre>

- Set property UseResultSet
- Use statement "Read from table ReadByExecutingQuery"

- 
- Without result set, just call as any other procedure.

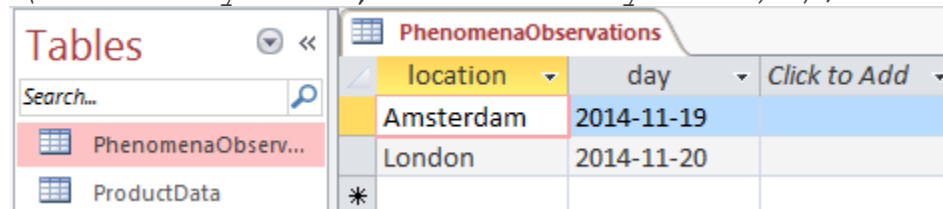
## 6. DirectSQL: Executing allowed SQL statements

- ok := DirectSQL( protected\_connection\_string, "CREATE TABLE PhenomenaObservations(location varchar(255), day varchar(255), PRIMARY KEY(location, day))" );



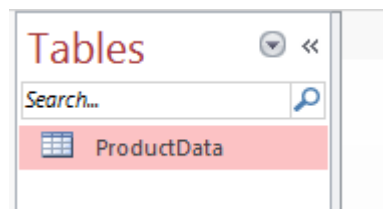
location	day	Click to Add
*		

- ok := DirectSQL(protected\_connection\_string, "INSERT INTO PhenomenaObservations(location, day) VALUES ('" + city + "', '" + someDay + "')");



location	day	Click to Add
Amsterdam	2014-11-19	
London	2014-11-20	
*		

- ok := DirectSQL( protected\_connection\_string, "DROP TABLE PhenomenaObservations" );



ProductData
-------------

## No demo, advanced: Query environment

---

There are AIMMS intrinsic procedures to:

- Query number and name of drivers:

```
noDriverNames := SQLNumberOfDrivers('ODBC');  
while LoopCount <= noDriverNames do  
    DriverName := SQLDriverName('ODBC', LoopCount);  
    SetElementAdd(DriverNames, eDriverName, DriverName);  
endwhile;
```

- Query number and name of tables and views given a database connection

```
SQLNumberOfTables, SQLTableName, SQLNumberOfViews,  
SQLViewName
```

- Query number and name, type of columns given a database table or view.

```
SQLNumberOfColumns, SQLColumnData
```

## Related documentation:

---

- AIMMS Language reference:
  - Chapter 26: The READ and WRITE Statements
  - Chapter 27: Communicating With Databases
- AIMMS Function reference:
  - Chapter 16: Database Functions
- The tech blog of AIMMS:
  - <http://techblog.aimms.com/2012/08/28/connect-to-access-database-file-via-odbc-connection-string/>
  - <http://techblog.aimms.com/2014/10/27/installing-32-bit-and-64-bit-microsoft-access-drivers-next-to-each-other/>
  - <http://techblog.aimms.com/2012/01/20/inspecting-sql-statements-created-by-aimms/>



## Announcement of next webinar

---

- The next webinar in this series, titled “Catch the errors with the AIMMS error handler”, will be presented by Haraldur Haraldsson, AIMMS Specialist.

Join us – December 17, 2014:  
10 AM CET and  
8AM PDT/11AM EDT

# Questions?

---

