
AIMMS Tutorial for Professionals - Linking to the Database

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com

Copyright © 1993–2011 by Paragon Decision Technology B.V. All rights reserved.

Paragon Decision Technology B.V.	Paragon Decision Technology Inc.	Paragon Decision Technology Pte.
Schipholweg 1	500 108th Avenue NE	Ltd.
2034 LS Haarlem	Ste. # 1085	80 Raffles Place
The Netherlands	Bellevue, WA 98004	UOB Plaza 1, Level 36-01
Tel.: +31 23 5511512	USA	Singapore 048624
Fax: +31 23 5511517	Tel.: +1 425 458 4024	Tel.: +65 9640 4182
	Fax: +1 425 458 4025	

Email: info@aimms.com
WWW: www.aimms.com

AIMMS is a registered trademark of Paragon Decision Technology B.V. IBM ILOG CPLEX and sc CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Ziena Optimization, Inc. XPRESS-MP is a registered trademark of FICO Fair Isaac Corporation. MOSEK is a registered trademark of Mosek ApS. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation. \TeX , \LaTeX , and $\AMS-\LaTeX$ are trademarks of the American Mathematical Society. LUCIDA is a registered trademark of Bigelow & Holmes Inc. ACROBAT is a registered trademark of Adobe Systems Inc. Other brands and their products are trademarks of their respective holders.

Information in this document is subject to change without notice and does not represent a commitment on the part of Paragon Decision Technology B.V. The software described in this document is furnished under a license agreement and may only be used and copied in accordance with the terms of the agreement. The documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Paragon Decision Technology B.V.

Paragon Decision Technology B.V. makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Paragon Decision Technology B.V., its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claims for lost profits, fees or expenses of any nature or kind.

In addition to the foregoing, users should recognize that all complex software systems and their documentation contain errors and omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors shall not be responsible under any circumstances for providing information or corrections to errors and omissions discovered at any time in this book or the software it describes, whether or not they are aware of the errors or omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors do not recommend the use of the software described in this book for applications in which errors or omissions could threaten life, injury or significant loss.

This documentation was typeset by Paragon Decision Technology B.V. using \LaTeX and the LUCIDA font family.

Part III

Model Procedures and Functions

Chapter 8

Linking to the Database

In this chapter you will experience how straightforward it is to link your model to a database using the point-and-click database interaction facilities of AIMMS. In addition, the possibility of entering SQL procedures in AIMMS is also illustrated.

This chapter

If you follow the steps in this chapter and you decide that you need to know more about database linkage, please look at the Chapter 'Communicating with Databases' in *The Language Reference*.

Further reading

8.1 Database tables

The linkage between AIMMS and a database relies on either the ODBC (Open DataBase Connectivity) standard, or the OLE DB standard. Almost all commercial database packages support at least one of these standards, including the MS Access database package used in this tutorial. In this tutorial we will connect to MS Access through ODBC.

*ODBC/OLE DB
and MS-Access*

The basic building blocks of a database are database tables containing columns and rows. One or more columns in a particular database table serve as so-called primary key columns. The remaining columns contain data defined over these key columns. The primary key values found in each row uniquely identify that row. For example, the first column in Figure 8.1 is a primary key column and identifies every row uniquely through the name of each location.

*Columns and
rows*

Location	XCoordinate	YCoordinate	UnitStockCost	InitialStockLevel	MinimumStockLevel	MaximumStockLevel	NumberOfInhabitants
Amersfoort	5.377	52.150	0.07	2200	460	2760	123367
Amsterdam	4.00	52.376	0.07	14016	2920	17520	727053
Apeldoorn	5.953	52.213	0.07	2736	570	3420	152060
Arnhem	5.917	51.982	0.07	2252	490	2940	137222
Assen	6.561	53.011	0.07	1056	220	1320	57376
Breda	4.770	51.598	0.07	2920	610	3660	159042
Den Bosch	5.300	51.701	0.07	2200	460	2760	120009
Den Haag	4.303	52.079	0.07	8064	1680	10080	440743
Den Helder	4.754	52.958	0.07	1056	220	1320	59590
Deventer	6.150	52.263	0.07	1400	310	1060	60621
Dordrecht	4.679	51.794	0.07	2160	450	2700	119462
Eindhoven	5.461	51.432	0.105	8330	8330	33320	199077
Emmen	6.805	52.700	0.07	1920	400	2400	105497
Enschede	6.09	52.22	0.07	2704	500	3480	140014
Groningen	6.574	53.226	0.07	3024	630	3780	171193
Haarlem	4.610	52.302	0.105	4310	4310	17240	140262
Leeuwarden	5.702	53.212	0.07	1632	340	2040	80762
Maastricht	5.696	50.857	0.07	2256	470	2020	121479
Nijmegen	5.045	51.04	0.07	2496	520	3120	151064
Rotterdam	4.402	51.929	0.07	10272	2140	12040	592665
Tilburg	5.071	51.568	0.07	3312	690	4140	190559
Utrecht	5.110	52.107	0.07	4080	950	5100	232710
Venlo	6.150	51.374	0.07	1200	250	1500	64580
Vlissingen	3.571	51.458	0.07	760	160	960	44530
Zwolle	6.09	52.522	0.105	2790	2790	11160	104431

Figure 8.1: Contents of the table 'Locations'

The database delivered with this tutorial contains four database tables. The first table contains data that are applicable to both factories and distribution centers (e.g. coordinate data and stock level data). The second table provides data that are needed to configure the factories (e.g. production capacity and cost data). Historical data (e.g. demand values over time) have been placed inside the third table, and will be used to initiate the rolling horizon process. Finally, the fourth database table contains the data that are needed to configure the individual production lines (e.g. production line capacities).

Four database tables


8.1.1 Entering the first database table declaration


You can refer to an external database table within AIMMS by means of a *database table* identifier declaration. As an attribute you can specify the ODBC data source name of the database you want to access, and also the name of the external database table from which you want to read or to which you want to write.

Database table in AIMMS

To declare your first database table in AIMMS, you should perform the following actions:

Creating the LocationTable

- ▶ create a new declaration section named Database Declarations under the Database Link section of the model tree,
- ▶ open the new declaration section,
- ▶ press the **Other...** button  on the toolbar,

- ▶ create a new database table identifier in this new declaration section by selecting the database table icon  in the **Select Type of Identifier** dialog box, and
- ▶ specify 'LocationTable' as its name.

An MS Access database file named 'Softdrink Factory Planning.mdb' has been supplied with this tutorial. Next, you will make this database available to AIMMS by performing the following actions:

Specifying the data source attribute

- ▶ activate the **Data source** wizard in the attribute form of the database table 'LocationTable',
- ▶ choose the **Select File Data Source...** command in the menu that pops up,
- ▶ select the file 'Softdrink Planning.dsn' from the 'Data' subdirectory, and
- ▶ press the **Save** button.

Once you have created the data source, you are now ready and able to select a table from this source. Please, execute the following simple steps:

Specifying the table name attribute

- ▶ activate the **Table name** wizard,
- ▶ choose the **Select Table/Query Name...** command from the pop-up menu,
- ▶ select 'Locations', and
- ▶ press the *OK* button.

If you have not worked with external databases before, it may be of interest to look at the external database table as it appears in the database. For this purpose, you can start MS Access, and inspect the design view of database table Locations as shown in Figure 8.2.

Look at the external table

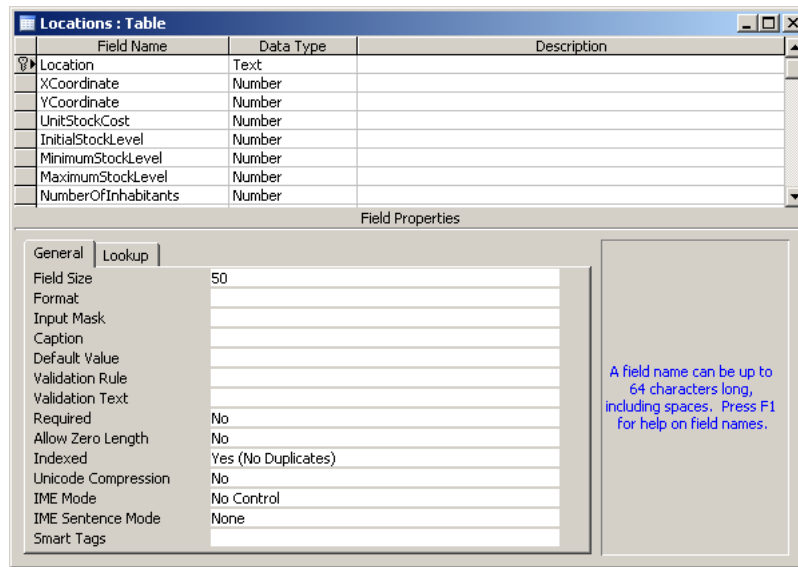




Figure 8.2: The MS Access design view of the Locations table

In general, the naming convention used inside a database table will not be identical to the naming convention used for the corresponding identifiers in AIMMS. That is why a mapping is needed to relate columns in the external database table to identifiers in AIMMS. For example, the mapping between the index identifier 1 in AIMMS and the column named 'Location' in the database can be specified as follows:

Specifying the mapping attribute

- ▶ activate the **Mapping** wizard,
- ▶ select the primary key "Location" from the 'Data Column' drop down list (see Figure 8.3),
- ▶ press the wizard button  to select the index 1 as the 'AIMMS Identifier',
- ▶ press the **Transfer** button  to put the specified mapping into the 'Mappings' list, and
- ▶ press the **OK** button.

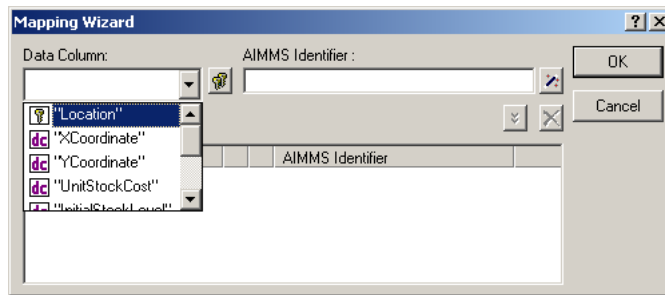


Figure 8.3: The Mapping wizard

Please look at Figure 8.4, and complete the mapping attribute accordingly using the wizard as explained in the previous paragraph.

Completing the mapping

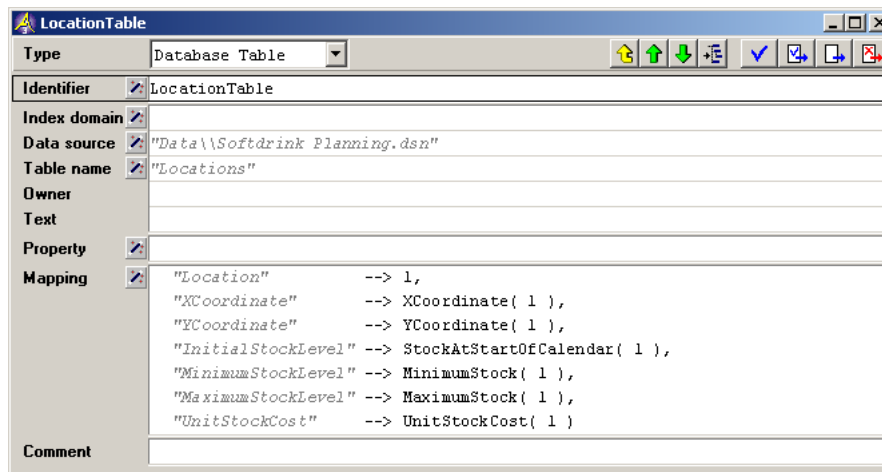


Figure 8.4: Attribute form of the data table 'Locations'

8.1.2 Entering additional database table declarations

Once you have completed your first database table declaration as described in the previous section, you can make the remaining three external database tables available to AIMMS. Before entering the corresponding declarations you need to declare two additional model parameters to store the weekly demand data read from the database.

Weekly demand data

```
PARAMETER:
  identifier : WeeklyDemand
  index domain : (c,w,s)
  unit : hl
```

```

PARAMETER:
  identifier : TotalWeeklyDemand
  index domain : (w,s)
  unit : hl

```

First declare the three additional database table identifiers FactoryTable, CenterTable and ProductionLineTable in the model tree (just below the parameter TotalWeeklyDemand). Then consider the attribute descriptions listed below. Next fill in the three attribute forms accordingly, using the **Data source** wizard, the **Table name** wizard, and the **Mapping** wizard.

Adding the three database tables

```

DATABASE TABLE:
  identifier : FactoryTable
  data source : "Data\\Softdrink Planning.dsn"
  table name : "Factories"
  mapping : "Factory" --> f,
           "UnitProductionCost" --> UnitProductionCost( f ),
           "MaximumTransportCapacity" --> MaximumTransportCapacity( f )

```

```

DATABASE TABLE:
  identifier : CenterTable
  data source : "Data\\Softdrink Planning.dsn"
  table name : "Centers"
  mapping : "Center" --> c,
           "Date" --> w,
           "Scenario" --> s,
           "Demand" --> WeeklyDemand( c, w, s )

```

```

DATABASE TABLE:
  identifier : ProductionLineTable
  data source : "Data\\Softdrink Planning.dsn"
  table name : "ProductionLines"
  mapping :
    "Factory" --> f,
    "ProductionLine" --> p,
    "InitialUsageCount" --> DeteriorationLevelAtStartOfCalendar( f, p ),
    "InitialProductionLevel" --> ProductionLineLevelAtStartOfCalendar( f, p ),
    "MaximumProductionLevel" --> MaximumProductionLineLevel( f, p ),
    "MaximumUsageCount" --> MaximumDeteriorationLevel( f, p )

```

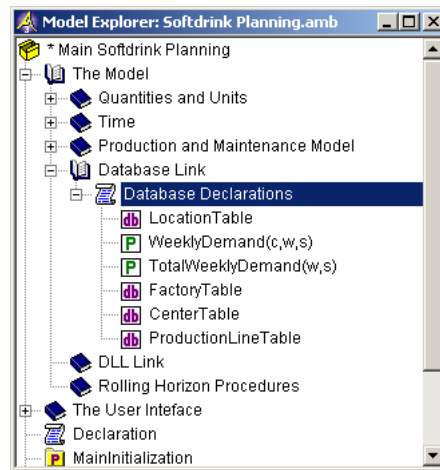


Figure 8.5: The database section of the model tree so far

8.2 Database procedures

When transferring data from, or to, a database table, you may need more sophisticated control over the data link than offered by the standard database table interface. AIMMS offers you this additional control by letting you write and execute *SQL* (Structured Query Language) statements, or providing access to *stored procedures* already available inside the database.

Sophisticated control

8.2.1 SQL queries

It is possible to access data values in a database that are not directly stored in one of its database tables. Consider, for instance, the database table named "ProductionLines" with the two primary key columns "Factory" and "ProductionLine". In this database table, there is no entry for the number of production lines in each factory. However, this information can be obtained from the database through the following *query* using *SQL*.




A first SQL query ...

```
SELECT Factory, COUNT(ProductionLine) AS LineCount
FROM ProductionLines GROUP BY Factory
```

This query temporarily creates a new table inside the database consisting of two columns. The first column is a primary key named 'Factory', while the second column is named 'LineCount' and contains the required totals.

To implement this query in AIMMS, you can create your first database procedure named `NumberOfProductionLinesQuery`. The following steps are required:

... declared in AIMMS

- ▶ close the declaration section named Database Declarations by double clicking on the scroll icon ,
- ▶ press the **Other...** button  on the toolbar,
- ▶ select the database procedure  from the **Select Type of Node** dialog box (see Figure 8.6), and press the **OK** button,
- ▶ enter 'NumberOfProductionLinesQuery' as the name of the database procedure, and
- ▶ press the *Enter* key to register the name.

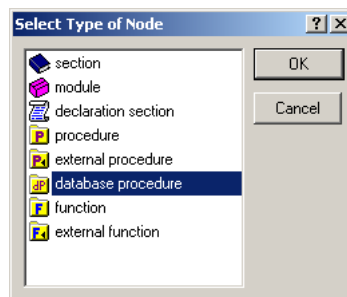


Figure 8.6: The **Select Type of Node** dialog box

After opening the attribute form of the database procedure, please complete it as shown in Figure 8.7. Note that the SQL text must be in double quotes, and can be split over several "quoted" lines using the + operator and the appropriate use of spaces to ensure that consecutive words are not run together. The specified 'UseResultSet' **Property** attribute enables you to use the database procedure as if it were a database table. Without this property, AIMMS does not allow you to specify the **Mapping** attribute, necessary to read data. Note that the **Mapping** wizard is not available for SQL queries.

Specifying the database procedure attributes

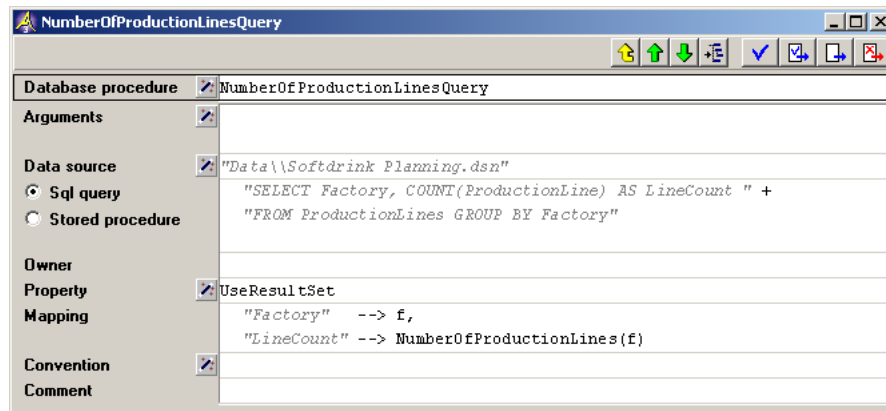


Figure 8.7: A database procedure to execute an SQL command

8.2.2 Stored procedures

In the previous subsection, you placed your own SQL query inside an AIMMS database procedure. In this subsection, you will consider a query that already resides inside the database, and that you can also access from within an AIMMS database procedure.


Procedures inside the database ...

A stored procedure can have one or more arguments, and it is straightforward to specify these arguments in an AIMMS database procedure. In this tutorial, however, the stored procedure named `TotalDemand` and `AllCenters` are used, and these procedures happen not to have arguments.

... with or without arguments

To declare your second database procedure, please execute the following actions:

Declaring the database procedures

- ▶ insert a new database procedure in the model tree, and specify 'TotalDemandQuery' as its name,
- ▶ open its attribute form,
- ▶ use the **Data source** wizard to select 'Softdrink Planning.dsn' as its **Data source** attribute,
- ▶ press the radio button in front of the **Stored procedure** attribute,
- ▶ activate the **Stored procedure** wizard,
- ▶ choose the **Select Stored Procedure Name...** command in the menu that pops up,
- ▶ select 'TotalDemand' as the **Stored procedure** attribute,
- ▶ complete the attribute form as shown in Figure 8.8, and
- ▶ close the attribute form using the **Check, commit and close** button 

Database procedure	TotalDemandQuery
Arguments	
Data source	"Data\Softdrink Planning.dsn"
Sql query	"TotalDemand"
Stored procedure	
Owner	
Property	UseResultSet
Mapping	"Date" --> w, "Scenario" --> s, "TotalDemand" --> TotalWeeklyDemand(w, s)
Convention	
Comment	

Figure 8.8: The completed attribute form of the database procedure TotalDemandQuery

And to declare your third database procedure with 'AllCentersQuery' as its name, please perform similar steps as mentioned above, only this time select 'AllCenters' as the **Stored procedure** attribute. The completed attribute form should look like the one in Figure 8.9).

Database procedure	AllCentersQuery
Arguments	
Data source	"Data\Softdrink Planning.dsn"
Sql query	"AllCenters"
Stored procedure	
Owner	
Property	UseResultSet
Mapping	"Center" --> c
Convention	
Comment	

Figure 8.9: The completed attribute form of the database procedure AllCentersQuery

The part of the model tree describing the database link is shown in Figure 8.10. *Database declarations so far*

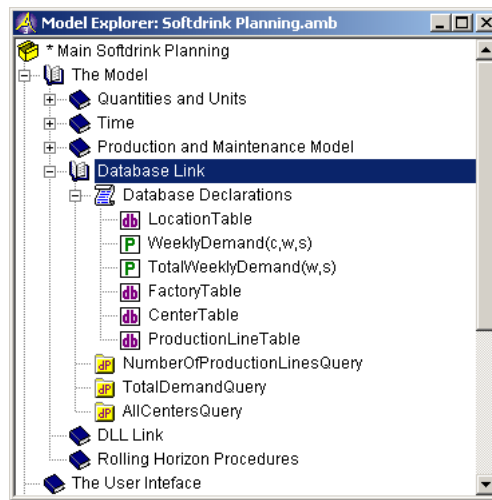


Figure 8.10: An intermediate model tree showing all database identifiers and procedures