Cathedron®

Manual for the Preview / Field Test Release

14 september 2007

Mattic B.V.
Chatelainestraat 35, 1336 SC Almere, The Netherlands
http://www.mattic.com

Table of contents

1 Introduction

Welcome to Cathedron

1.1 Introducing Cathedron

Cathedron is our award winning environment to design, develop, deploy and maintain data driven systems. Cathedron combines the power of a model driven generator with a high order language and industry standard extendibility. This combination enables you to focus on the concepts (what should be made) instead of the technical details (how it should be made). Shifting your focus to the concepts radically enhances both the speed of development and the quality of the application.

A typical Cathedron application requires only a fraction of the code you would have needed in a 3rd generation language. Your focus stays on the code that matters. Code that directly implements a restriction or a calculation. Code that is almost self describing and can be easily understood in the context of the accompanying model. A model that is always accurate since it is the heart of your application and intensely used by the generator.

The combination of a generator and compact code also means there is little room for bugs. Finally due to the compact and self describing nature of the code, it is easier to share the development with others.

Cathedron offers you a no-nonsense approach. The result is a powerful yet clean environment that is easy to understand and learn. It is this approach that sets us apart from other model driven tools.

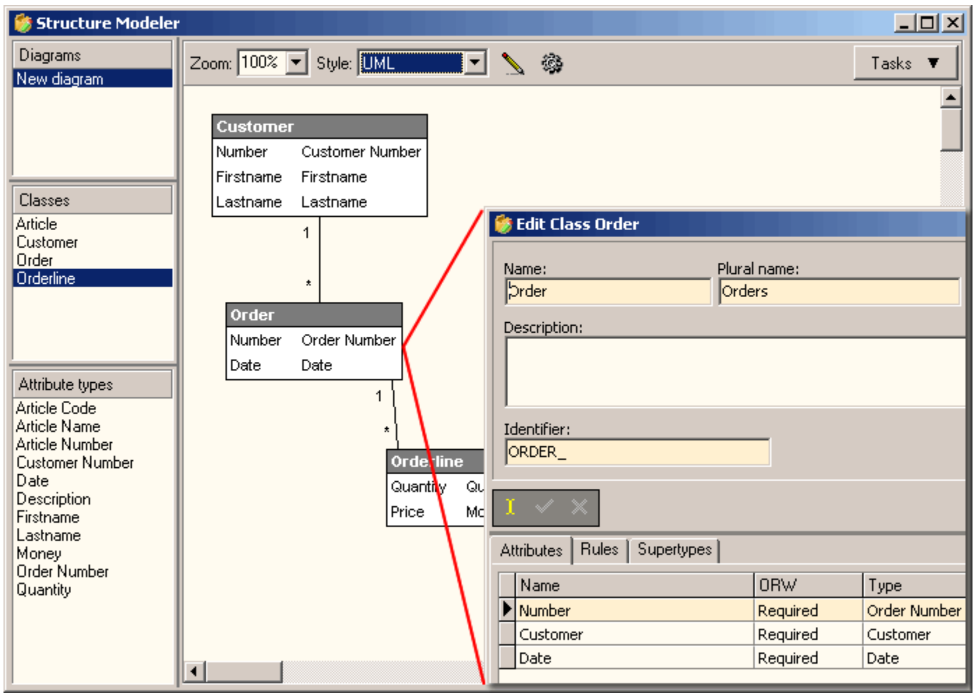
1.2 What is Cathedron?

Cathedron …

- supports the **design**, **development**, **deployment** and **maintenance** of data driven systems (information systems, database systems)
- is a Rapid Application Development (RAD) environment
- is a Model Driven Development (MDD) environment
- offers a two tier (client-server) and a three tier (web application) architecture
- does not generate source code (except SQL-DDL code). It is based on a runtime interpreter that generates the actual application at runtime
- is based on (and limited to) Firebird or InterBase and uses Pascal script as a programming language
- is based on (and limited to) Windows. (Except for the client in the three tier architecture since this can be a common web browser)

Design

Designing an application with Cathedron typically starts with the creation of an information model (data model). This model describes the structure of the information to be stored and processed by the system. The model can be represented using commonly used diagram styles such as UML and ERD.


Structure Modeler

Diagrams
New diagram

Classes
Article
Customer
Order
Orderline

Attribute types
Article Code
Article Name
Article Number
Customer Number
Date
Description
Firstname
Lastname
Money
Order Number
Quantity

Zoom: 100% Style: UML

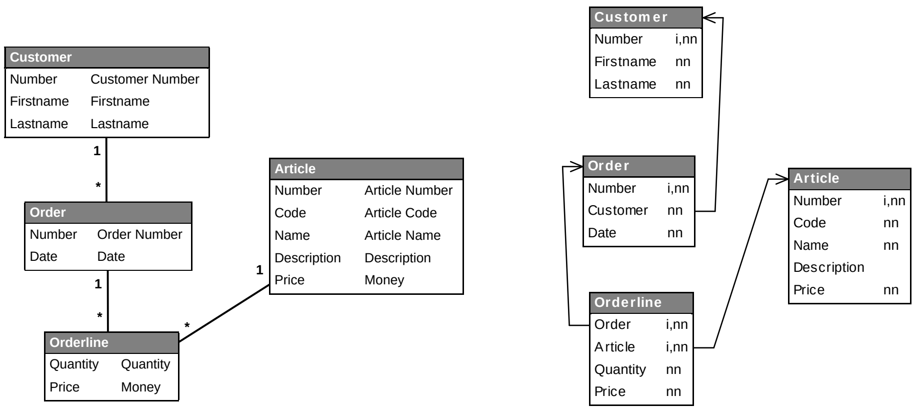
Edit Class Order

Name: Order
Plural name: Orders
Description:
Identifier: ORDER_

Attributes | Rules | Supertypes

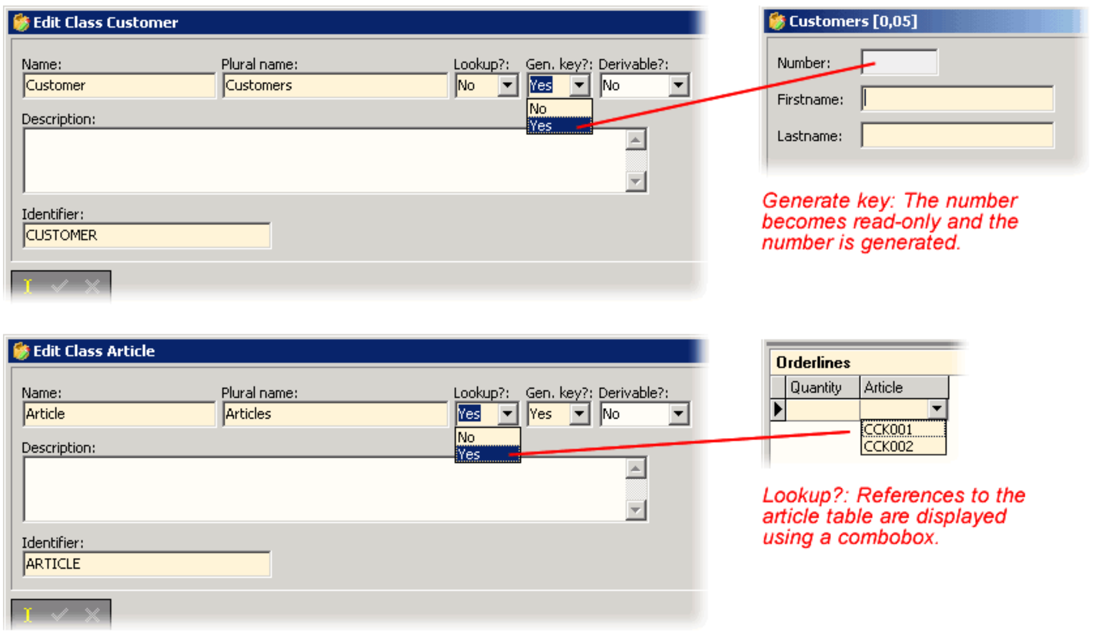
Name	ORW	Type
Number	Required	Order Number
Customer	Required	Customer
Date	Required	Date

Editing the Order class in the Structure modeler.



An example model represented as UML and ERD diagram.

After the basic structure is defined the model can be refined by specifying properties such as generate key, lookup yes/no, short name, edit format etc. Those properties are used to generate a basic interface for the application.



Edit Class Customer

Name:	Plural name:	Lookup?:	Gen. key?:	Derivable?:
Customer	Customers	No	Yes	No

Description:

Identifier: CUSTOMER

Customers [0,05]

Number:

Firstname:

Lastname:

Generate key: The number becomes read-only and the number is generated.

Edit Class Article

Name:	Plural name:	Lookup?:	Gen. key?:	Derivable?:
Article	Articles	Yes	Yes	No

Description:

Identifier: ARTICLE

Orderlines

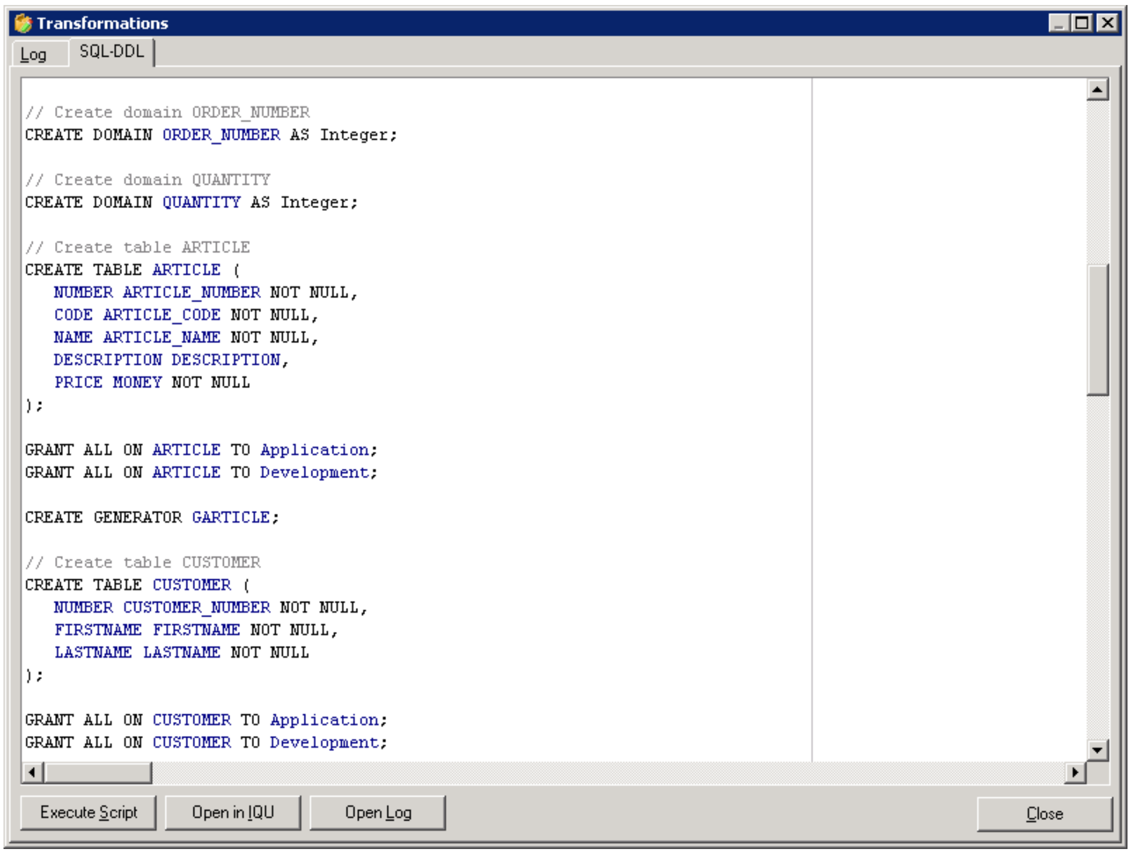
Quantity	Article
	CCK001
	CCK002

Lookup?: References to the article table are displayed using a combobox.

Example of properties and their effects.

Development

After the model is finished you can transform the model into a database. Cathedron will generate the proper SQL-DDL code for Firebird or InterBase. You can now review this code and execute it (within the Cathedron IDE) to create the database.

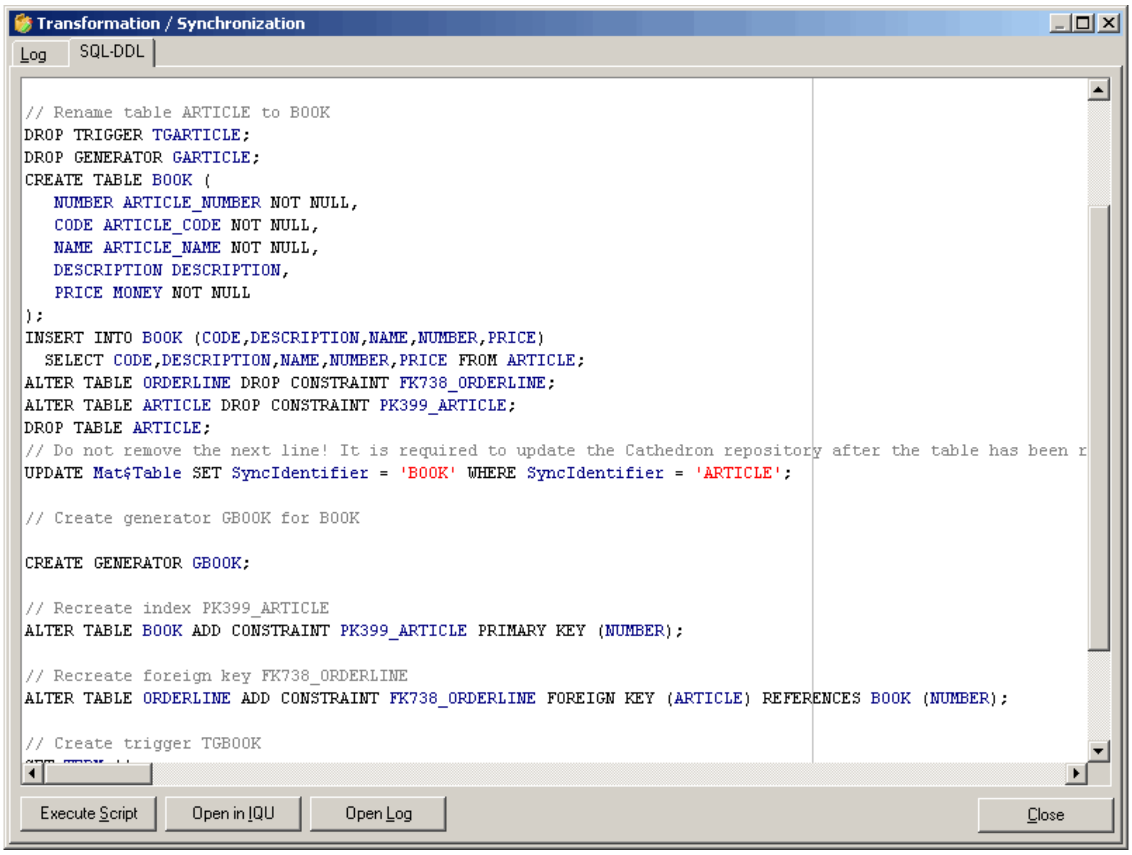


Transformations

```
// Create domain ORDER_NUMBER
CREATE DOMAIN ORDER_NUMBER AS Integer;

// Create domain QUANTITY
CREATE DOMAIN QUANTITY AS Integer;

// Create table ARTICLE
CREATE TABLE ARTICLE (
   NUMBER ARTICLE_NUMBER NOT NULL,
   CODE ARTICLE_CODE NOT NULL,
   NAME ARTICLE_NAME NOT NULL,
   DESCRIPTION DESCRIPTION,
   PRICE MONEY NOT NULL
);

GRANT ALL ON ARTICLE TO Application;
GRANT ALL ON ARTICLE TO Development;

CREATE GENERATOR GARTICLE;

// Create table CUSTOMER
CREATE TABLE CUSTOMER (
   NUMBER CUSTOMER_NUMBER NOT NULL,
   FIRSTNAME FIRSTNAME NOT NULL,
   LASTNAME LASTNAME NOT NULL
);

GRANT ALL ON CUSTOMER TO Application;
GRANT ALL ON CUSTOMER TO Development;
```

Generated SQL-DDL code.

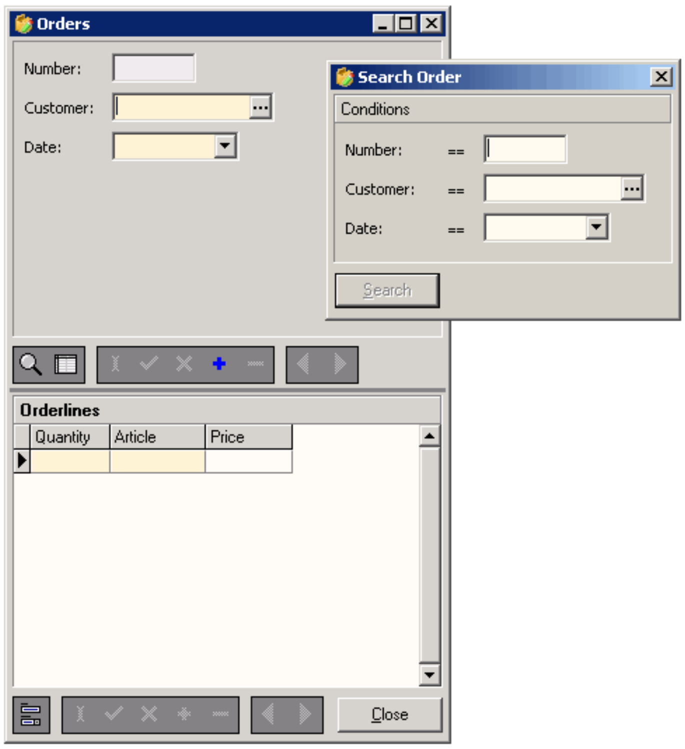
When you have changed your model Cathedron will generate a script to change the database. When possible this script will convert or keep any existing information in the database.



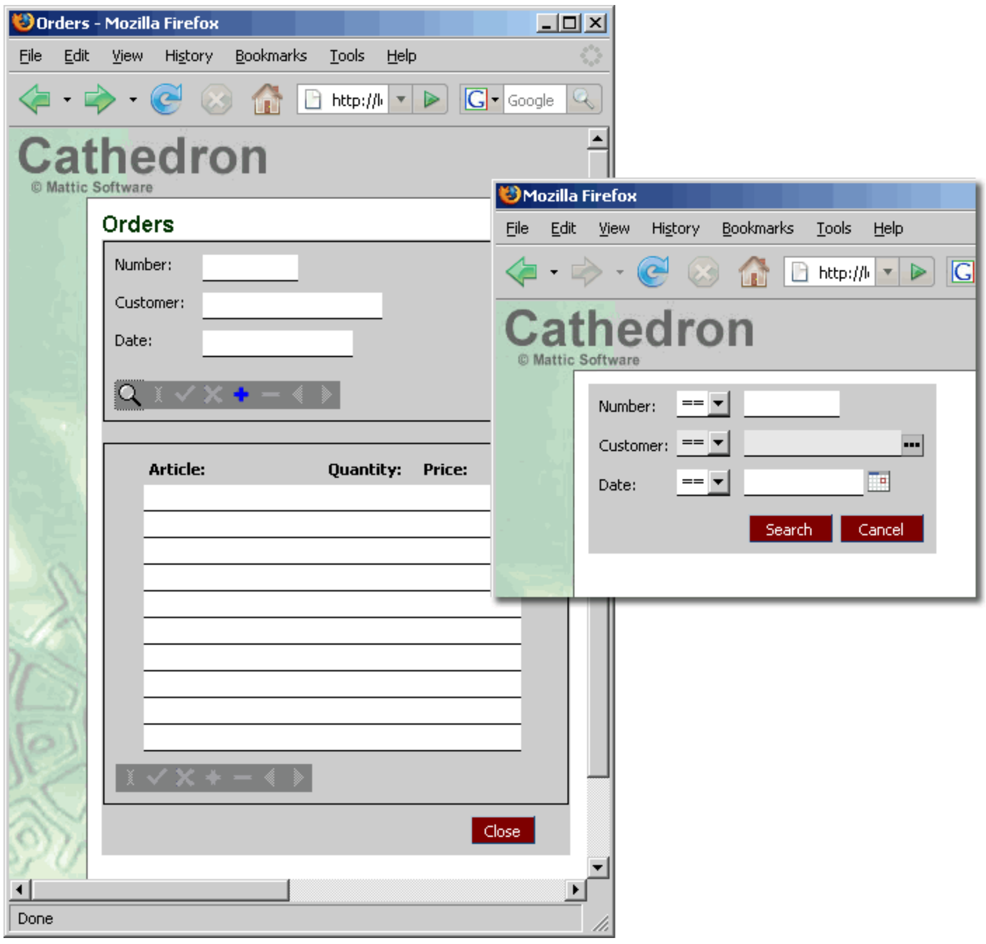
```
// Rename table ARTICLE to BOOK
DROP TRIGGER TGARTICLE;
DROP GENERATOR GARTICLE;
CREATE TABLE BOOK (
   NUMBER ARTICLE_NUMBER NOT NULL,
   CODE ARTICLE_CODE NOT NULL,
   NAME ARTICLE_NAME NOT NULL,
   DESCRIPTION DESCRIPTION,
   PRICE MONEY NOT NULL
);
INSERT INTO BOOK (CODE,DESCRIPTION,NAME,NUMBER,PRICE)
  SELECT CODE,DESCRIPTION,NAME,NUMBER,PRICE FROM ARTICLE;
ALTER TABLE ORDERLINE DROP CONSTRAINT FK738_ORDERLINE;
ALTER TABLE ARTICLE DROP CONSTRAINT PK399_ARTICLE;
DROP TABLE ARTICLE;
// Do not remove the next line! It is required to update the Cathedron repository after the table has been r
UPDATE Mat$Table SET SyncIdentifier = 'BOOK' WHERE SyncIdentifier = 'ARTICLE';

// Create generator GBOOK for BOOK

CREATE GENERATOR GBOOK;

// Recreate index PK399_ARTICLE
ALTER TABLE BOOK ADD CONSTRAINT PK399_ARTICLE PRIMARY KEY (NUMBER);

// Recreate foreign key FK738_ORDERLINE
ALTER TABLE ORDERLINE ADD CONSTRAINT FK738_ORDERLINE FOREIGN KEY (ARTICLE) REFERENCES BOOK (NUMBER);

// Create trigger TGBOOK
```

Generated SQL-DDL code after changing the model.

When the database structure has been generated you can immediately explore your application using the default forms generated by Cathedron.

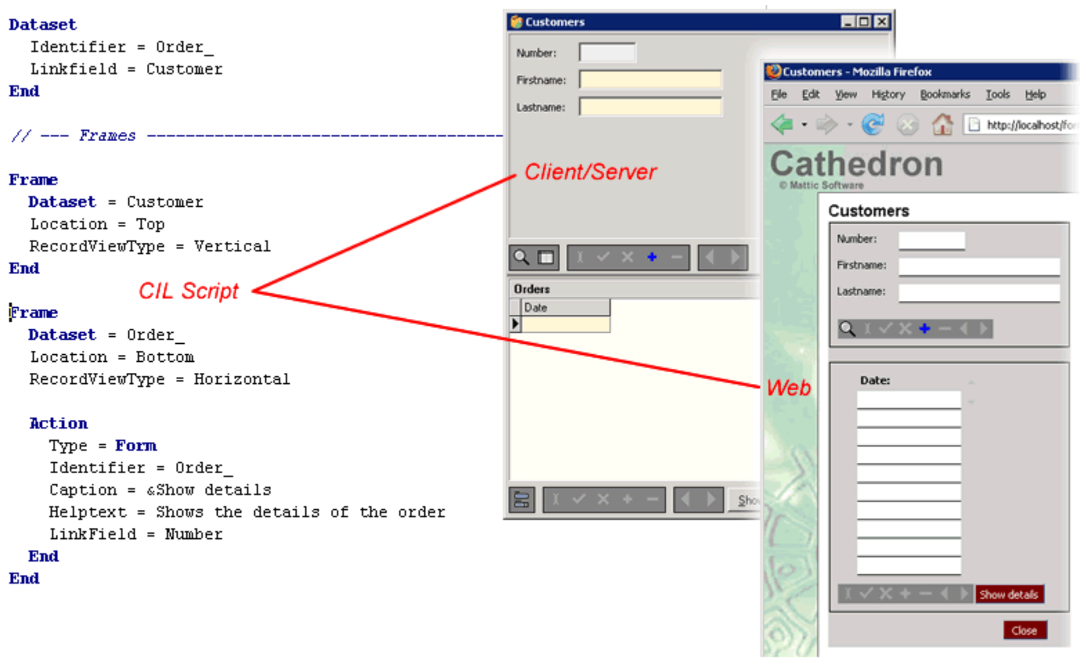


Generated default forms.



Generated default web forms.

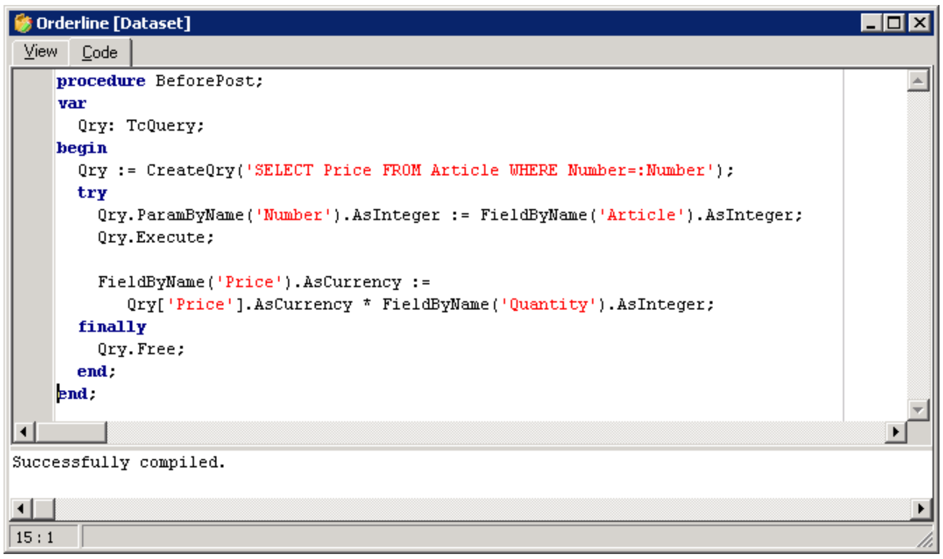
The default forms offers the basic functionality required to manipulate the database. In addition to the default forms, Cathedron has two languages to create a more sophisticated application. The first language is the Cathedron Interface Language (CIL). It is a simple textual property language to specify the structure of the interface.



```
Dataset
  Identifier = Order_
  Linkfield = Customer
End

// --- Frames -----------------------------------

Frame
  Dataset = Customer
  Location = Top
  RecordViewType = Vertical
End

Frame
  Dataset = Order_
  Location = Bottom
  RecordViewType = Horizontal

  Action
    Type = Form
    Identifier = Order_
    Caption = &Show details
    Helptext = Shows the details of the order
    LinkField = Number
  End
End
```

CIL script with resulting Client/Server and Web-forms.

The second language is Pascal script, allowing you to add code to the application.



```
procedure BeforePost;
var
  Qry: TcQuery;
begin
  Qry := CreateQry('SELECT Price FROM Article WHERE Number=:Number');
  try
    Qry.ParamByName('Number').AsInteger := FieldByName('Article').AsInteger;
    Qry.Execute;

    FieldByName('Price').AsCurrency :=
       Qry['Price'].AsCurrency * FieldByName('Quantity').AsInteger;
  finally
    Qry.Free;
  end;
end;
```

```
Successfully compiled.
```

15 : 1

Sample Pascal Script.

Deployment

Cathedron offers several features and utilities to deploy your applications:

- Distribution of your application does not require the distribution of a new executable. All you need is the Cathedron runtime executable.
- The ability to synchronize the development environment with the production environment and vice versa.
- Cathedron comes with tools to automate the distribution of Cathedron releases within a client-server environment[1].

Maintenance

A summary of the maintenance features of Cathedron:

- When you change the model of an existing database Cathedron will generate a delta SQL-DLL script. This script will only contain the SQL-DDL instructions necessary to update your system. Those instructions will attempt to automatically convert all the data in the system [2] [3].
- Cathedron offers the possibility to convert existing Firebird or InterBase databases into a Cathedron compatible database[4].

The Mattic Communication Engine (MCE), described below, offers additional maintenance features.

1.3 The Mattic Communication Engine (MCE)

In addition to Cathedron the optional Mattic Communication Engine (MCE) offers the following features:

- Scheduled execution of tasks on the server. Where a tasks can be:
  - a SQL script
  - a Pascal script
  - an executable
  - backup or sweep of a database
- Sending the result of a task to a mailing list. (For example monthly report in Excel generated by a SQL-script).
- Sending the result code (success or failure) to an administrator.
- Direct (TCP-IP based) communication between clients and clients and the server.
- Logging of exceptions from clients. (A very useful way to discover bugs in your application).

---

[1] Windows XP and Vista distinct between normal users and administrators. A workaround is required to update software on those clients.

[2] There are situations that cannot be automated. For example changing the number of characters of a ‘first name’ from 80 to 40. In those situations Cathedron will generate as much code as possible and offer you the chance to add the appropriate conversion logic.

[3] This feature is not yet entirely functional. Most important omission: changes of primary keys are not yet properly handled.

[4] There are several important restrictions to this feature. These restrictions will be explained in the appropriate sections in this manual.

- Logging of time-consuming queries that where executed on the database. (Optimizing those queries will drastically improve the performance of the entire system).
- A HTTP server. This server can be used to serve HTML pages containing Pascal script.
- The WebCat server. In addition to the features of the HTTP server, this server gives you the possibility to serve the HTML representation of your Cathedron applications to the outside world.

1.4 When to use Cathedron?

Roughly categorized Cathedron can be used in three different ways:

- To build an entirely new system.
- To create a prototype for a system.
- To continue building an existing system.

To build an entirely new system

This option was explained briefly in the introduction. It will be explained in more detail in the ‘Getting started’ chapter.

To create a prototype for a system

Cathedron is very well suited for Rapid Prototyping. The basic difference between a prototype and an entirely new system is the amount of effort to finish the application. The most simple option is to use the modeler. Create the information model and use the default screens to communicate your model to the customer. You could also take your prototype one step further and create custom screens. In most situations you will not write Pascal script to implement business rules and custom behaviour because this is the most time consuming part of the development.

To continue building an existing system

If you have an existing Firebird or InterBase project, it might be an option to convert this project into a Cathedron project. In the future a tutorial will be added to this manual to explain the possibilities and restrictions.

1.5 About the technology

Model Driven Development (MDD)

Cathedron is based on the principle of Model Driven Development:

> *... Model-Driven Development refers to a range of development approaches that are based on the use of software modeling as a primary form of expression. Sometimes models are constructed to a certain level of detail, and then code is written by hand in a separate step. Sometimes complete models are built including executable actions. Code can be*

generated from the models, ranging from system skeletons to complete, deployable products... [Cited from Wikipedia, 2007].

How Cathedron implements Model Driven Development:

- We use an information model as the base for the application.
- The information model can be represented in styles such as UML, ERD or ORM.
- We extend UML with information such as identity to be able to generate primary keys and optional vs. required columns.
- We do not generate code. Cathedron uses a run-time interpreter.
- The model is completed using two languages:
  - The Cathedron Interface Language (CIL)
  - Pascal Script (3rd generation scripting language)
- Cathedron supports regeneration after the model is changed. (While some tools only offer you initial skeleton code).

Benefits of Model Driven Development

Model Driven Development enables you to increase both the speed of development and the quality of your information systems. Surely MDD will not be the end of all your 'development problems', nor will it be the absolute perfect solution. It is however a very fast and reliable method to develop data driven systems.

Model Driven Development offers:

- Rapid Prototyping
  - Easier communication with customers
  - Verified and detailed specifications
- Rapid Development
  - Reduce development costs
  - Easier to calculate and estimate projects
- Compact code requiring less documentation
  - Significant reduction of bugs
  - Easier to share code
- Consistency through generation and architecture
  - Easier to navigate through projects
  - Reduce maintenance costs
  - Short learning curve for customers and developers

Beyond MDD: Combining Technologies

We have never believed a single technique could do the trick. Although generation is very powerful, it is useless if you cannot easily customize the interface and create the application logic you require. For more demanding applications you should even be able to integrate code, screens and components created in a 'normal' 3GL environment. Cathedron combines several techniques to offer a flexible development environment.

Technique	**Description**
Model Driven Development	Default screens and functionality are derived from the information model (structure of your database).
Interface definition language	The high level interface language can be used to customize the look of the default screens.
Integrated scripting language	Use an normal 3GL (Pascal) to write and debug your constraints and logic. The editor includes syntax highlighting, real time information about parameters and methods, macro recording and code completion.
Industry standard extendibility	Extend your system by calling Dynamic Link Libraries, COM/ActiveX objects or external applications.
The options below are currently only available on request. They are not integrated into the standard releases of Cathedron.	
Workflow support	Using the workflow module you can add workflow to your system. In the most simple situation you can use the workflow as an authorization schema. During the design fase you can add descriptions (Use Cases) for each process. If you maintain the descriptions they will become a valuable online manual (Perfect for ISO certification). If you want to create a system that really implements the workflow you can use Cathedron to easily create wizards and statusviews (action lists).
Full integration with Codegear Delphi	Integration with Delphi allows you to expand Cathedron in any way you want. Integrate your code, forms and components. You can expand the scripting language and create your own form templates.

Model Driven Architecture (MDA)

A lot of tools claim to be Model Driven Architecture tools. Compared to the features of those tools, one could say Cathedron is an MDA tool.

Unfortunately if a tool claims to be MDA, you do not know exactly what it is capable of. To be more specific:

- Cathedron uses the so called PIM and PSM models from MDA.
- We do not import or export XMI. (Altough this should be a standard there are numerous of different XMI formats).
- We do not generate code (although this is not explicitly required by MDA).

2 Installation

Installation

2.1 System requirements

Operating system

- Cathedron is in active use on Windows 2000, Windows XP, Windows 2003 Server and Windows Vista.
- Cathedron has been used on Windows NT 4.0, Windows 98 and Windows 95. Current and new releases will not be tested on those operating systems.

Web browsers (used to connect to the Cathedron web server)

- Mozilla based browsers (such as Firefox)
- Internet Explorer

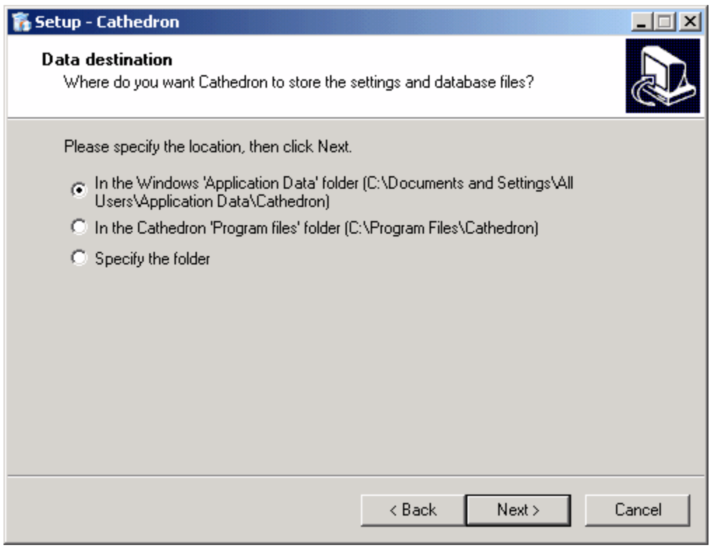
Hardware

Any configuration that will properly run your operating system and database engine will do. Cathedron has no special hardware demands.

2.2 The Installation Wizard

The installation is a straight forward installation using a wizard. The screenshots below will explain the most important options:

Application Data Folder


Setup - Cathedron

Data destination

Where do you want Cathedron to store the settings and database files?

Please specify the location, then click Next.

[✓ In the Windows 'Application Data' folder (C:\Documents and Settings\All Users\Application Data\Cathedron)]

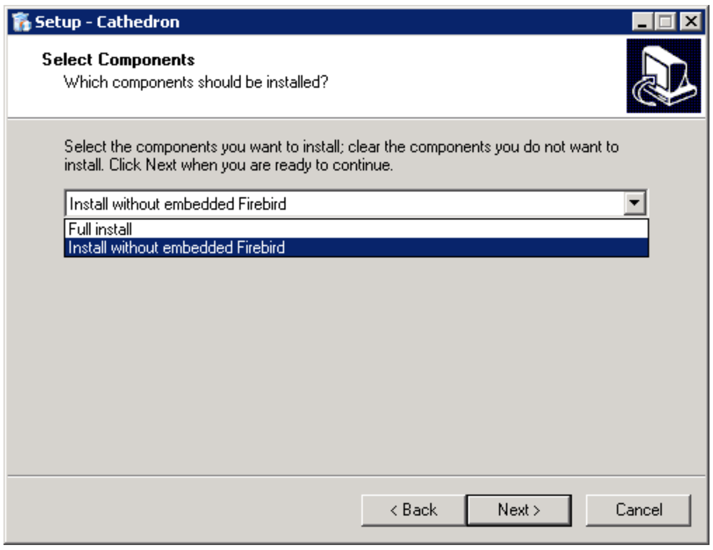
In the Cathedron 'Program files' folder (C:\Program Files\Cathedron)

Specify the folder

In Windows XP and Vista it is not recommended to write data in the Program files folder. XP and Vista recommend to write your data in the Application Data (aka

Program Data) folder. Depending on your personal preference you can choose the Application Data, the Program files or a custom folder.

Installed components



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Install without embedded Firebird

Full install

Install without embedded Firebird

Full install

The option Full install will install the Embedded Firebird database engine. This option is recommended if you want to evaluate the product. The embedded Firebird database has two limitations that are not crucial for most evaluation purposes:

- There is no authorization control.
- Only a single program can access a database at the same time.

This option will not conflict with any existing Firebird or InterBase installations.

Install without embedded Firebird

The second option (Install without embedded Firebird) assumes you already have a working installation of Firebird or InterBase. Choose this option if you are seriously considering development with Cathedron or if you want to use Cathedron with your existing database installation.

Cathedron assumes there is a user with the name ‘dev’ and the password ‘dev’. If you use Cathedron with an existing Firebird or InterBase installation you need to create this user[1].

[1] Appendix 2 describes how to create this user in Firebird.

2.3 About Firebird and InterBase

The current version of Cathedron works only with Firebird 1.5 and up and with Borland InterBase (version 6.5 and up).

Visit: www.firebirdsql.org for more information about the Firebird project.

Visit: www.codegear.com/products/interbase for more information about InterBase.

This release has not yet been tested with InterBase.

2.4 Removing Cathedron

The software can be removed by using the Add/Remove Program option from the Control Panel.

Cathedron performs a clean installation. We do not install files into the Windows or Windows system folders. Cathedron does not create any registry entries.

2.5 What is installed where?

Location	Description
<Program Files>\Cathedron	Main program and libraries
<Program Files>\Cathedron	Embedded version of the Firebird database. (When selected)
<Program Files>\Cathedron\New Project	This folder contains the database ‘New Project.fdb’. This database is simply copied and renamed to quickly start a new project.
<Program Files>\Cathedron\WebCat	Folder with scripts used by the Cathedron web client.
<Data Folder>\Cathedron\Projects	Default location for Cathedron databases.
<User settings>\Application Data\Cathedron\Cache	Automatically created when you run Cathedron. Used to store cache files.

3 Tutorial Information / Data Modeling

Tutorial Information Modeling

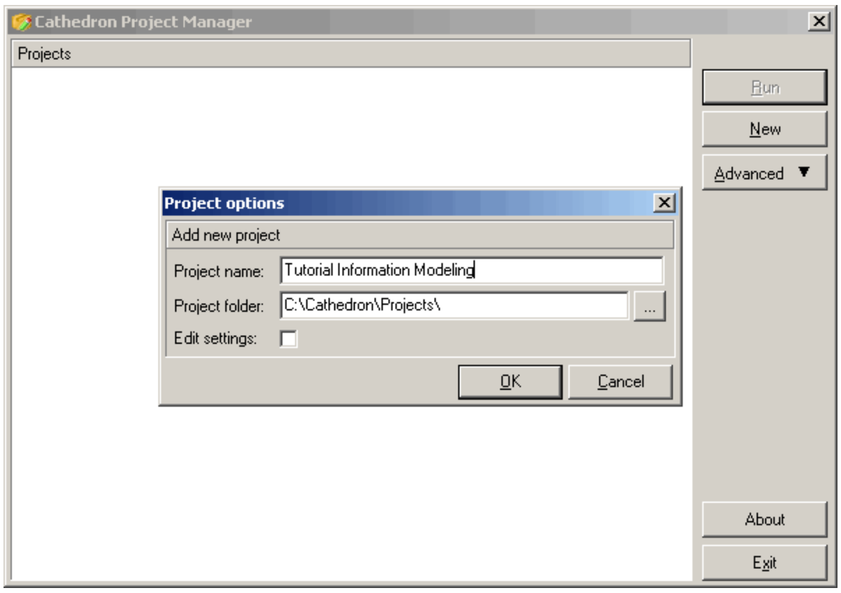
3.1 Before you start

If you have installed Cathedron without the Embedded Firebird database you will have to create an account on your Firebird or InterBase server. Both the name and password of this account should be ‘dev’[1].

There is no need to create this user if you have chosen the Full installation option. This option includes the Embedded Firebird engine that accepts any username/password.

3.2 Getting started

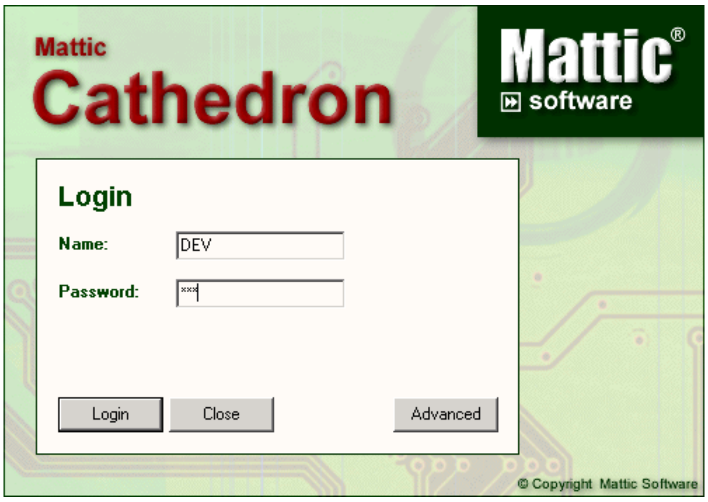
Start the Cathedron Manager from the start menu. The Project manager will open and show a list of projects. Depending on your version of Cathedron, this list is either empty or filled with example or tutorial projects. If the list is empty Cathedron will show the Project options dialog to create a new project. If the list is not empty click New to start this dialog.



Project name: Tutorial Information Modeling
Project folder: C:\Cathedron\Projects\
Edit settings: ☐

Enter ‘Tutorial Information Modeling’ as the title for the project and click OK. When Cathedron asks to run the project choose Yes. The login screen of Cathedron will show.

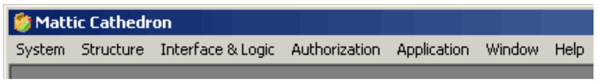
[1] Appendix 2 describes how to create this user in Firebird.



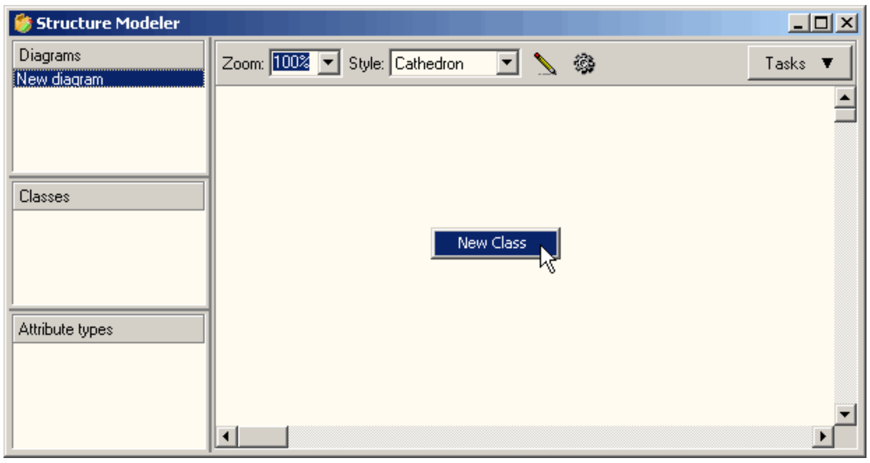
Name: DEV

Password: ***

Login to Cathedron using the default development account. (Name 'dev', password 'dev'). Cathedron opens with an empty screen showing its main menu.



In this tutorial we will focus on the Structure menu. Select the option Modeler from this menu to start the Structure modeler.



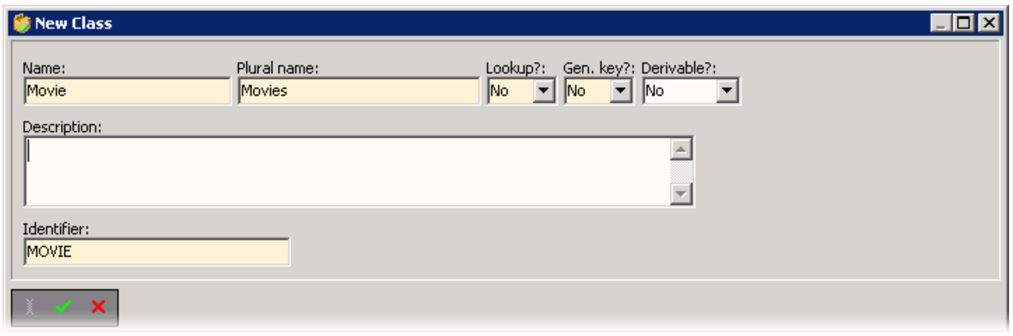
Zoom: 100%

Style: Cathedron

A default diagram named New diagram has been created.

3.3 Creating the first class

Creating our first class is simple: Move your mouse over the canvas, right click and select the option New Class. A dialog will appear to enter the specifications of the class.


New Class

Name: Movie
Plural name: Movies
Lookup?: No
Gen. key?: No
Derivable?: No
Description:
Identifier: MOVIE

The upper part of the dialog is used to enter the specifications of the class.

The example in this tutorial is a basic application for the administration of a movie rental company.

First we will create the Movie class. Enter the name ‘Movie’ and the plural name ‘Movies’ and accept the input by clicking the ✓ button.
If something went wrong you can always correct the mistake by clicking the I button, changing the information and clicking the ✓ button again to accept the changes.

Most buttons on the button bar can be operated using shortcuts. Here is an overview of the buttons and their shortcuts[1]:

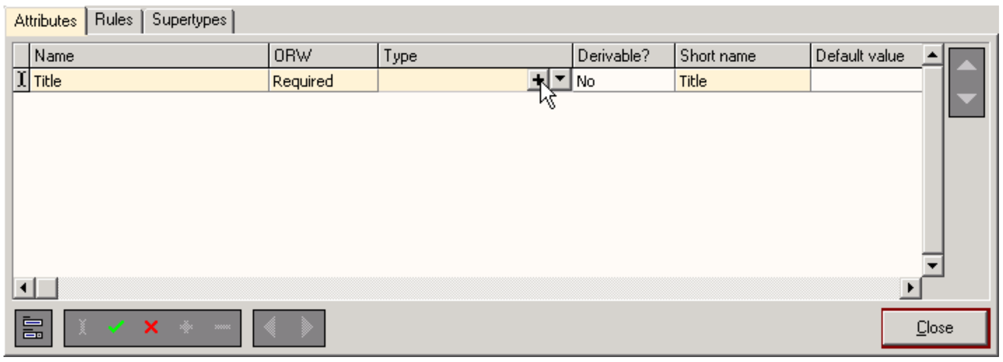
Button	Name	Shortcut	Function
🔍	Search	F5	Opens the search dialog
☰ ▦	Toggle Record / Grid View	F8	Toggles between record and grid view. Record view shows a single record using separate edit controls. Grid view shows multiple records in tabular form.
I	Edit	F2	Enable editing.
✓	Accept	F3	Accept changes.
✗	Cancel	F4	Discard changes.
+	Insert	F9	Insert a new record.
−	Delete	<No shortcut>	Delete the current record.
◀	Previous record	Ctrl ,	Go to the previous record. (When multiple records are retrieved from the database).
▶	Next record	Ctrl .	Go to the next record. (When multiple records

[1] Appendix 1 contains an overview of all shortcuts.

			are retrieved from the database.
▲	Record up	Ctrl ↑	Used in ordered lists. Moves the record up in the list.
▼	Record down	Ctrl ↓	Used in ordered lists. Moves the record down in the list.

Note: Each screen may present a different selection of buttons.

After you have clicked the ✓ button, the focus will shift to the lower part of the screen. In this part we can add the attributes of the class. Since the list of attributes was empty, Cathedron has inserted a new record.



Name	ORW	Type	Derivable?	Short name	Default value
Title	Required		No	Title	

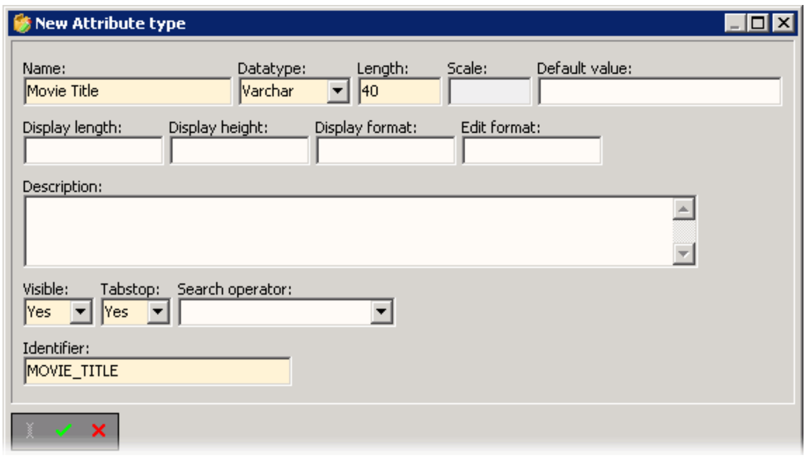
Enter ‘Title’ as the name for the attribute. Set the ORW (Optional, Required, Warning) field to ‘Required’. The third field requires an explanation.

In Cathedron every attribute needs a type. A type can either be a class (created earlier) or a named data type (aka domain). In the example this means we cannot simple select ‘string’ from the type field. Since the model is empty we will have to create a new type. Basically there are three possible types:

Type name	**Explanation**
Movie Title	Choosing the name ‘Movie Title’ assumes titles of movies might differ from other titles. Not only can other titles have a different length, they might also be subjected to different rules (such as all characters upper cased).
Title	Choosing the name ‘Title’ assumes all titles in the model are equal.
String	Choosing the name ‘String’ is a (poor) way to circumvent the explicit typing used by Cathedron. The drawback of this approach: every string will be of the same length!

Option three is clearly not recommended. If you want to play safe you should choose the first option using the most specific name.

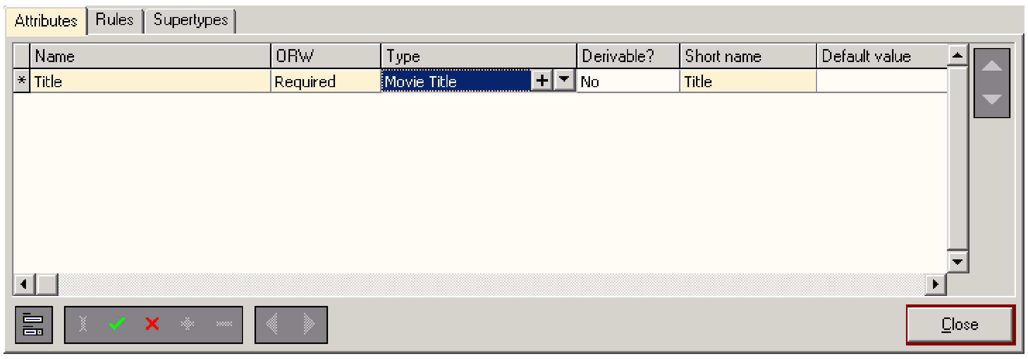
To create a new type click the ⊞ button or press `Ctrl Shift Enter`.



Name: Movie Title	Datatype: Varchar	Length: 40	Scale:	Default value:
Display length:	Display height:	Display format:	Edit format:	
Description:				
Visible: Yes	Tabstop: Yes	Search operator:		
Identifier: MOVIE_TITLE				

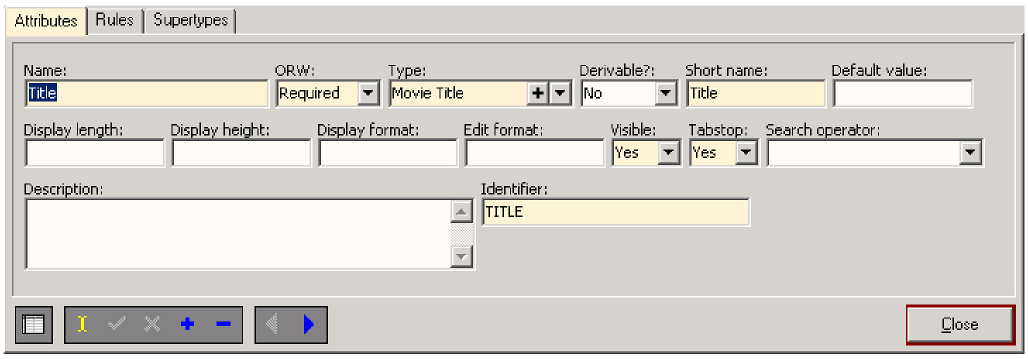
Enter the first three fields as displayed above, accept the data (✓ or F3) and close the dialog.

The attribute frame will now show the type we have just created.



Name	ORW	Type	Derivable?	Short name	Default value
Title	Required	Movie Title	No	Title	

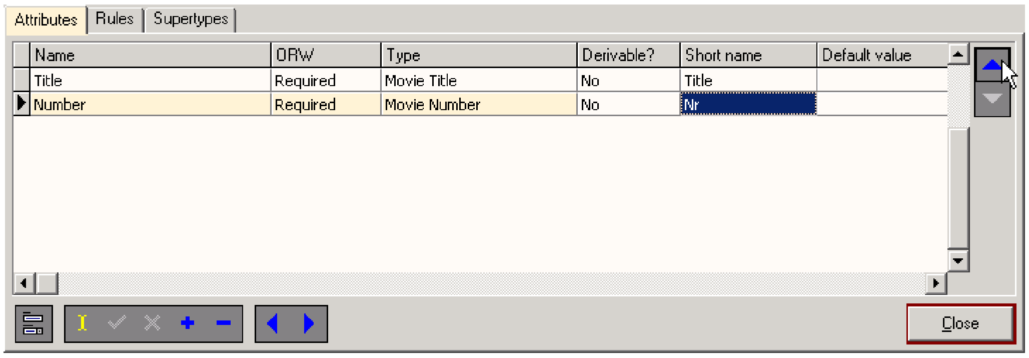
The current attribute frame shows the fields in tabular form. Click the button or press F8 to switch to record view.



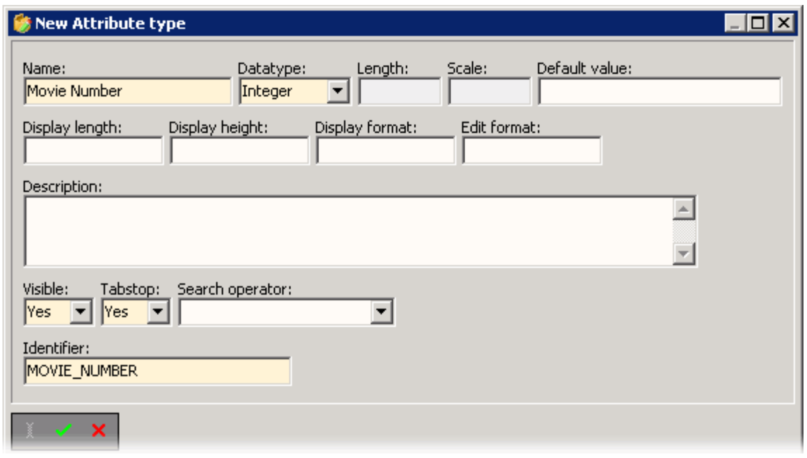
Name: Title	ORW: Required	Type: Movie Title	Derivable?: No	Short name: Title	Default value:	
Display length:	Display height:	Display format:	Edit format:	Visible: Yes	Tabstop: Yes	Search operator:
Description:	Identifier: TITLE					

You might find this view more convenient when entering the details of an attribute. If your screen contains multiple attributes, there is no need to switch back and forward between tabular and record view. You can simply scroll through the records using the ◀ ▶ buttons or the `Ctrl ,` and `Ctrl .` hotkeys.

Accept the changes (F3) and switch back to the tabular view (F8) to add the next attribute.



Name	ORW	Type	Derivable?	Short name	Default value
Title	Required	Movie Title	No	Title	
Number	Required	Movie Number	No	Nr	



New Attribute type

Name: Movie Number	Datatype: Integer	Length:	Scale:	Default value:
Display length:	Display height:	Display format:	Edit format:	

Description:

Visible: Yes
Tabstop: Yes
Search operator:

Identifier: MOVIE_NUMBER

Create the attribute as shown in the first screen. The new type for the field should be created as shown in the second screen.

We have chosen a short name for this attribute. This short name will be used in tabular views when there is not enough room to display the long name of the attributes.

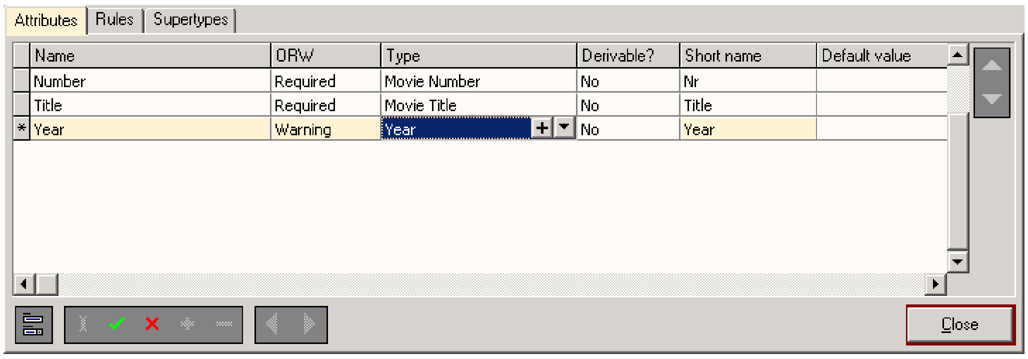
This attribute was created because we know there are different movies with the same title. Therefore we want an additional number to uniquely identify the movie.

It is good practice to put the identifying attribute(s) of a class on the top of the attribute list. We can do this by selecting the 'Number' attribute row and clicking the ▲ button or pressing Ctrl ↑.

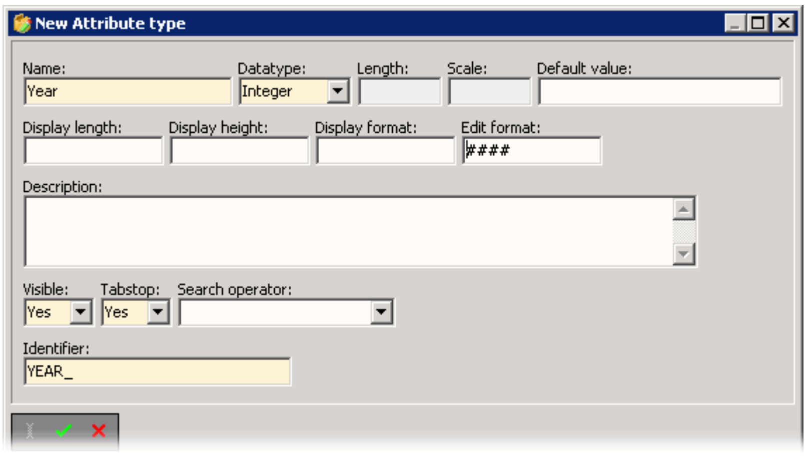
Again one could have a discussion about the best name for the type of the attribute. Should it be 'Movie Number' or 'Number'. We will probably have other classes with a number as the identifying attribute (such as customer).

There are two answers. From a data perspective there is no difference between the number of a movie or the number of a customer. Both are positive integers. From a conceptual perspective both numbers are completely different. Although you can technically add a movie number and a customer number, it makes no sense. In this tutorial we will choose names such as 'Movie Number' and 'Customer Number'.

To complete the class we will add a year attribute. Create the attribute as specified in the screens below.



	Name	ORW	Type	Derivable?	Short name	Default value
	Number	Required	Movie Number	No	Nr	
	Title	Required	Movie Title	No	Title	
*	Year	Warning	Year	No	Year	

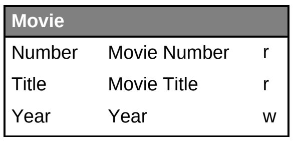


Name:	Year
Datatype:	Integer
Length:	
Scale:	
Default value:	
Display length:	
Display height:	
Display format:	
Edit format:	####
Description:	
Visible:	Yes
Tabstop:	Yes
Search operator:	
Identifier:	YEAR_

Remarks:

- The ORW field is set to Warning. As a result Cathedron will show a warning when the value is omitted. The corresponding database column will still be optional since a user can choose ignore the warning message.
- The name ‘Year’ for the attribute type is a reserved keyword in some databases. Therefore the identifier (technical name) is automatic changed in ‘YEAR_’.
- A simple edit format `####` is used to make sure a year always contains four digits[1].

After you have accepted the data `(F3)` and closed the dialog `(Alt C)` the class will be drawn on the diagram canvas.



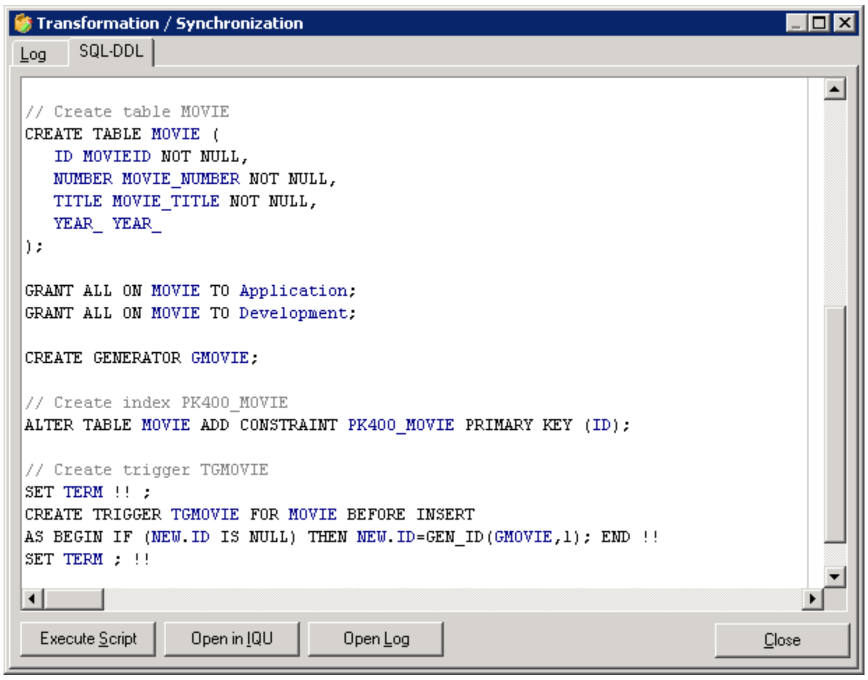
Movie		
Number	Movie Number	r
Title	Movie Title	r
Year	Year	w

3.4 Transforming the model into a database / application

Press the button on the toolbar of the Structure modeler (or press `F9`). The Transformation / synchronization screen will open and Cathedron will make a backup of your project. After the backup has finished Cathedron will transform the information model into a technical model for the database and the application.

[1] The reference chapters 3 and 4 contain more information about edit and display formats.

Cathedron will not immediately create the database. Instead it will present the generated SQL-DDL script for inspection.

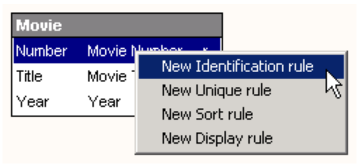


```
// Create table MOVIE
CREATE TABLE MOVIE (
   ID MOVIEID NOT NULL,
   NUMBER MOVIE_NUMBER NOT NULL,
   TITLE MOVIE_TITLE NOT NULL,
   YEAR_ YEAR_
);

GRANT ALL ON MOVIE TO Application;
GRANT ALL ON MOVIE TO Development;

CREATE GENERATOR GMOVIE;

// Create index PK400_MOVIE
ALTER TABLE MOVIE ADD CONSTRAINT PK400_MOVIE PRIMARY KEY (ID);

// Create trigger TGMOVIE
SET TERM !! ;
CREATE TRIGGER TGMOVIE FOR MOVIE BEFORE INSERT
AS BEGIN IF (NEW.ID IS NULL) THEN NEW.ID=GEN_ID(GMOVIE,1); END !!
SET TERM ; !!
```

Scroll through the script to see how the database is created. A closer look at the Movie table shows us four instead of three attributes! An additional ID attribute has been created. Although we have mentioned the Number of a Movie was the identifying attribute, we never specified this in Cathedron.

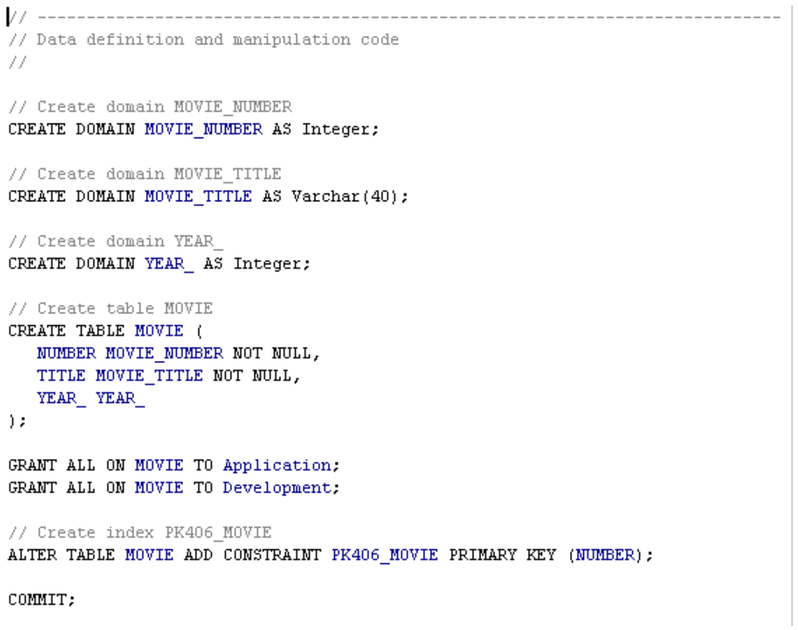
Cancel the execution of the script by closing the dialog.

Note: If you did execute the script, close the dialog and continue as if nothing happened.



Hold the shift key and click on the Number attribute of the Movie class on the diagram. The number attribute should become selected. Now right click the attribute and select New identification rule from the menu. A small ‘i’ will be printed after the Number attribute to indicate it is an identifying attribute. Click on the button `(F9)` to restart the transformation. This time Cathedron will not make a backup. This is only done once during a Cathedron session[1].

[1] A session expires when you logoff or close Cathedron.



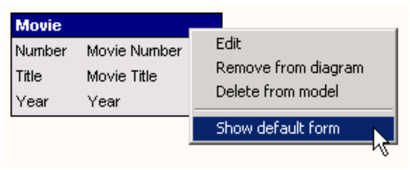
```
// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Create domain MOVIE_NUMBER
CREATE DOMAIN MOVIE_NUMBER AS Integer;

// Create domain MOVIE_TITLE
CREATE DOMAIN MOVIE_TITLE AS Varchar(40);

// Create domain YEAR_
CREATE DOMAIN YEAR_ AS Integer;

// Create table MOVIE
CREATE TABLE MOVIE (
   NUMBER MOVIE_NUMBER NOT NULL,
   TITLE MOVIE_TITLE NOT NULL,
   YEAR_ YEAR_
);

GRANT ALL ON MOVIE TO Application;
GRANT ALL ON MOVIE TO Development;

// Create index PK406_MOVIE
ALTER TABLE MOVIE ADD CONSTRAINT PK406_MOVIE PRIMARY KEY (NUMBER);

COMMIT;
```

Check the generated script and click Execute Script to create the database. After the database has been created Cathedron will recreate the application cache. Close the dialog when the log shows ‘Script executed’.

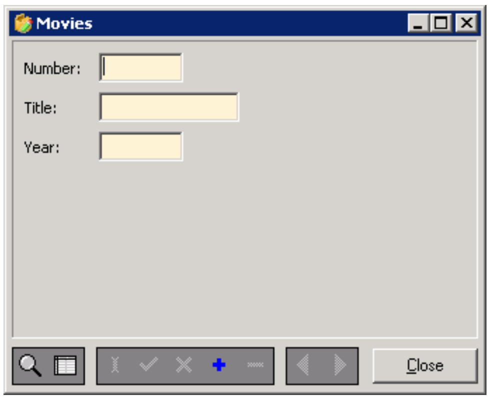
Note: If you had executed the first script, you will see a smaller script, changing the previously created structure. You can safely execute this script and continue.

Note: The numbers of constraints such as foreign keys and uniqueness constraints might differ in your script.

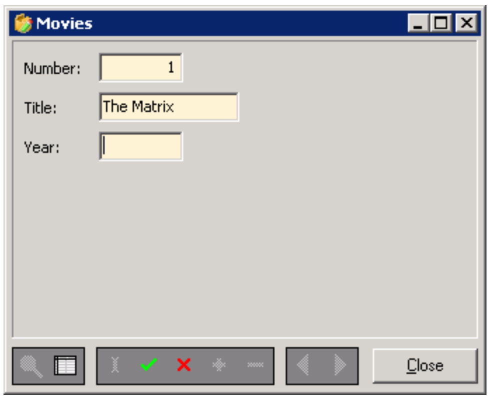
3.5 The first default form



We can now test the class by opening a default form. Simple right click on the title of the class in the diagram and choose Show default form from the menu.



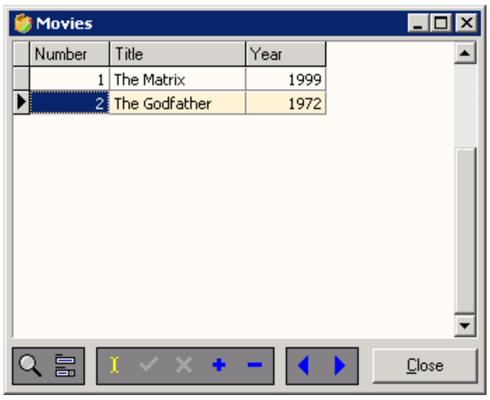
A default screen will be shown. As you can see the button bar on this screen is identical to the button bar used in the dialogs from the Structure modeler. Click the + button or press `F9` to add a new movie.



Number:	1
Title:	The Matrix
Year:	

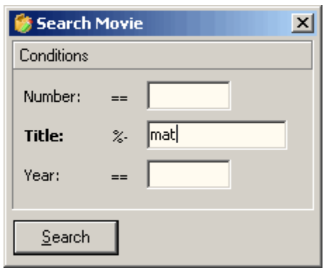
Enter the information as shown and test the warning message on the year field by accepting the data `(F3)`. Act as 'the average user': do not read the message and hit the `Enter` key. You will remain in edit mode and the year field will be focused. Enter '1999' and accept the data.

To test the search capabilities we need to add another record. Click or press `F8` to switch to tabular view and enter the extra record as shown below.



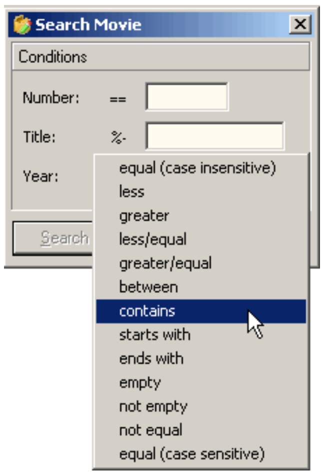
Number	Title	Year
1	The Matrix	1999
2	The Godfather	1972

Click or press `F5` to open the search dialog. Try to find 'The Matrix' by entering 'mat' in the Title field.



Number:	==	
Title:	%-	mat
Year:	==	

Click Search or press `Enter` to start the search. Cathedron will not find any movie matching the criteria. You might have noticed the `%-` before the Title edit control. This symbol indicates Cathedron will look for titles starting with 'mat'. Since the characters 'mat' are not located at the start of the title, the movie cannot be found.

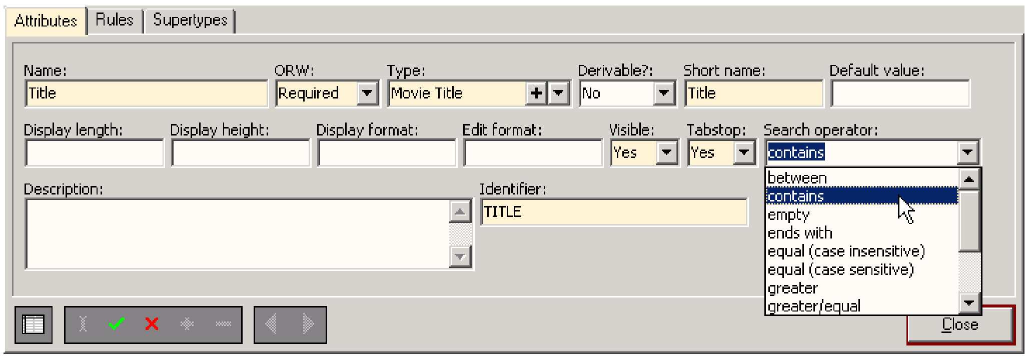


Choose 'contains' from the presented options and search for 'mat' again. This time the movie will be found.

The most frequently used operators can be easily selected using shortcuts. When you open the search dialog and select a field and press one of the letters from the table below, the search operator will be set. This will only work when the edit control is still empty!

Operator	**Symbol**	**Short cut**
Equal (case insensitive)	=	=
Less	<	<
Greater	>	>
Less/Equal	<=	<no shortcut>
Greater/Equal	>=	<no shortcut>
Between	..	.
Contains	%	%
Starts with	%-	-
Ends with	-%	<no shortcut>
Empty	#	#
Not empty	!	!
Not equal	<>	<no shortcut>
Equal (case sensitive)	==	+

It is possible to change the default search operator for a field. Close the movie form, go to the diagram in the Structure modeler, right click on the title of the Movie class and select Edit from the menu.



Name: Title	ORW: Required	Type: Movie Title	Derivable?: No	Short name: Title	Default value:
Display length:	Display height:	Display format:	Edit format:	Visible: Yes	Tabstop: Yes
Search operator: contains	Description:	Identifier: TITLE			

between
contains
empty
ends with
equal (case insensitive)
equal (case sensitive)
greater
greater/equal

Select the Title attribute. Press `F8` to switch to record view and press `F2` to enter edit mode. Set the Search operator field to 'contains' and press `F3` to accept the change. Close the dialog and regenerate the application by clicking the button or pressing `F9`.

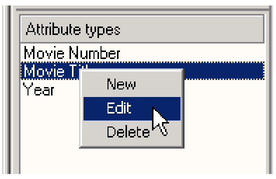
Note: Cathedron will only regenerate its cache. No SQL-DDL is generated.

Right click the Movie class and select Show default form from the menu to reopen the Movie form. Notice forms always open without any data to prevent an unnecessary load of the network and the database server. Check the new default operator setting by opening the search dialog (`F5`).

3.6 Changing the model

Increasing the size of an attribute type

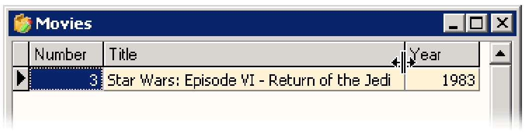
When we try to add the movie 'Star Wars: Episode VI - Return of the Jedi' we run into a limitation of our model. The Movie Title type is not long enough to contain the entire title. Close the Movie form en return to the diagram to change the Movie Title attribute type.



A list of attribute types can be found in the lower left corner of the Structure modeler. Select the Movie Title attribute type and right click it. Select Edit from the popup menu to open the editor. Change the length of the type to 80 characters and close the editor dialog. Synchronize the changes from the model with the database and the application by pressing `F9`.

```
// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Change datatype of domain MOVIE_TITLE
CREATE DOMAIN TMP$MOVIE_TITLE AS Varchar(80);
ALTER TABLE MOVIE ADD TMP$TITLE TMP$MOVIE_TITLE;
UPDATE MOVIE SET TMP$TITLE=TITLE;
ALTER TABLE MOVIE DROP TITLE;
DROP DOMAIN MOVIE_TITLE;
CREATE DOMAIN MOVIE_TITLE AS Varchar(80);
ALTER TABLE MOVIE ADD TITLE MOVIE_TITLE NOT NULL;
UPDATE MOVIE SET TITLE=TMP$TITLE;
ALTER TABLE MOVIE DROP TMP$TITLE;
DROP DOMAIN TMP$MOVIE_TITLE;

// Changing fieldpositions of columns
ALTER TABLE MOVIE ALTER TITLE POSITION 2;
ALTER TABLE MOVIE ALTER YEAR_ POSITION 3;

COMMIT;
```

Cathedron will now compare the model with the existing database. From this comparison it will generate SQL-DDL script to modify the database[1]. As you can see the script uses a temporary column and domain to maintain all the information in the database. You can now execute the script and enter the information about the movie:



Number	Title	Year
3	Star Wars: Episode VI - Return of the Jedi	1983

Decreasing the size of an attribute type

In the upcoming two examples we will show some limitations of this algorithm. First we will reduce the length of the Movie Title.

Close the Movie screen and return to the Structure modeler. Right click the Movie Title attribute type to open the editor and change the length to 50. Close the dialog and synchronize the model with the database and the application `(F9)`. As the header of the script shows the resulting script cannot be executed automatically. Click on the Open in IQU button to edit the script.

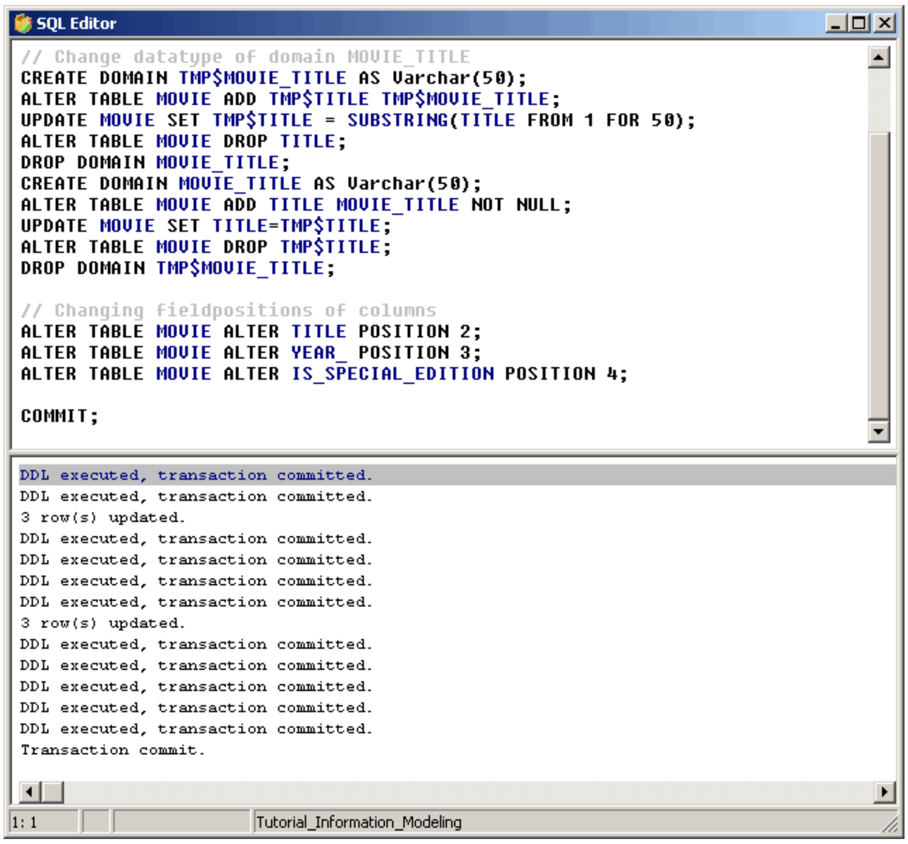
```
// Change datatype of domain MOVIE_TITLE
CREATE DOMAIN TMP$MOVIE_TITLE AS Varchar(50);
** manual ** // Please rewrite the queries below if data conversion is required.
ALTER TABLE MOVIE ADD TMP$TITLE TMP$MOVIE_TITLE;
UPDATE MOVIE SET TMP$TITLE=TITLE;
ALTER TABLE MOVIE DROP TITLE;
DROP DOMAIN MOVIE_TITLE;
CREATE DOMAIN MOVIE_TITLE AS Varchar(50);
ALTER TABLE MOVIE ADD TITLE MOVIE_TITLE NOT NULL;
UPDATE MOVIE SET TITLE=TMP$TITLE;
ALTER TABLE MOVIE DROP TMP$TITLE;
DROP DOMAIN TMP$MOVIE_TITLE;
```

Remove the warning header and proceed to the line marked with `** manual **`. As this line states we should change the update queries below this marker, if data

[1] Cathedron does not use the ALTER DOMAIN statement since this statement has some limitations. We have chosen a single solution for as many situations as possible.

conversion is required. Change the script as specified below and execute it by pressing `Ctrl Enter`.

Note: The Query Editor is an embedded version of our standalone utility IQU (Interactive Query Utility). More information about the use of IQU and the SQL Editor can be found on www.mattic.com/iqu. A basic manual can be found in reference chapter 1.



```
// Change datatype of domain MOVIE_TITLE
CREATE DOMAIN TMP$MOVIE_TITLE AS Varchar(50);
ALTER TABLE MOVIE ADD TMP$TITLE TMP$MOVIE_TITLE;
UPDATE MOVIE SET TMP$TITLE = SUBSTRING(TITLE FROM 1 FOR 50);
ALTER TABLE MOVIE DROP TITLE;
DROP DOMAIN MOVIE_TITLE;
CREATE DOMAIN MOVIE_TITLE AS Varchar(50);
ALTER TABLE MOVIE ADD TITLE MOVIE_TITLE NOT NULL;
UPDATE MOVIE SET TITLE=TMP$TITLE;
ALTER TABLE MOVIE DROP TMP$TITLE;
DROP DOMAIN TMP$MOVIE_TITLE;

// Changing fieldpositions of columns
ALTER TABLE MOVIE ALTER TITLE POSITION 2;
ALTER TABLE MOVIE ALTER YEAR_ POSITION 3;
ALTER TABLE MOVIE ALTER IS_SPECIAL_EDITION POSITION 4;

COMMIT;
```

```
DDL executed, transaction committed.
DDL executed, transaction committed.
3 row(s) updated.
DDL executed, transaction committed.
DDL executed, transaction committed.
DDL executed, transaction committed.
DDL executed, transaction committed.
3 row(s) updated.
DDL executed, transaction committed.
DDL executed, transaction committed.
DDL executed, transaction committed.
DDL executed, transaction committed.
DDL executed, transaction committed.
Transaction commit.
```

Close the SQL Editor and close the Transformation / Synchronization dialog. Cathedron will now recreate the application cache.

Remarks:

- Cathedron has no idea what SQL has been executed. It simply assumes you have executed the script. If you have made any mistakes it will try to correct these the next time you synchronize the model with the database.
- Please be aware that duplicate values can be created by simply trimming the title field. If the title field would have been covered by a primary key or uniqueness constraint the new data might have violate these constraints.

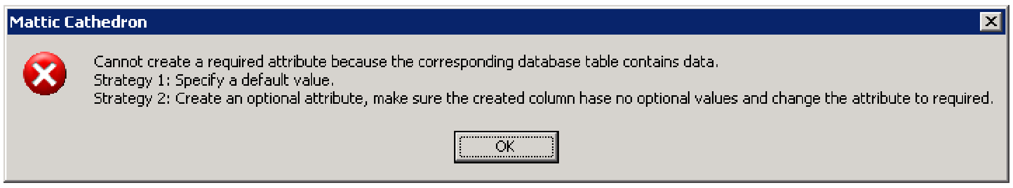
Adding a required attribute

In the last example of this paragraph we will extend the movie class with an attribute that has a fixed list of options.

Return to the Structure modeler and right click the Movie class to edit the class. Press `Alt 1` to switch to the first tab (attributes) in the lower frame of the form.

Note: You can use `Alt 1 - 9` to switch between the tabs at the bottom and `Ctrl 1` to return the focus to the upper frame.

Create a new attribute pressing `F9`. Enter ‘Is Special Edition’ as the title. Next set the ORW field to Required and press `Tab` to proceed to the next field. Cathedron will now show an error message:



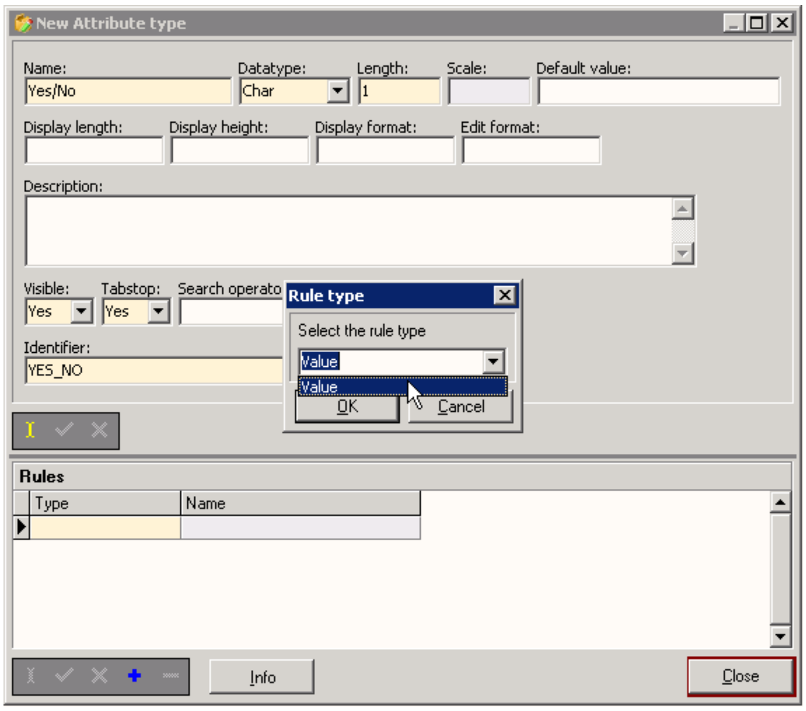
Mattic Cathedron

Cannot create a required attribute because the corresponding database table contains data.
Strategy 1: Specify a default value.
Strategy 2: Create an optional attribute, make sure the created column hase no optional values and change the attribute to required.

As you can see Cathedron cannot simple create a new required attribute. You have two options:

- Specify a default value that is used to complete the missing values.
- Make the attribute optional. Complete the missing values yourself. After you have completed the values you can return to the modeler to make the attribute required. (Cathedron actually checks the data in the table before giving this warning).

In this example we will chose the first option. Click OK to close the message. Change the ORW column into Optional. Go to the Default value field and enter ‘N’ (without the quotes) as the default value. Switch back to the ORW field and change it to Required. Create a new attribute type with the name ‘Yes/No’ as shown below:



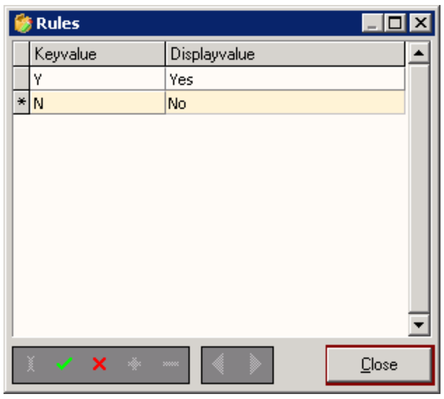
New Attribute type

Name: Yes/No
Datatype: Char
Length: 1
Scale:
Default value:
Display length:
Display height:
Display format:
Edit format:
Description:
Visible: Yes
Tabstop: Yes
Search operato
Identifier: YES_NO

Rule type
Select the rule type
Value

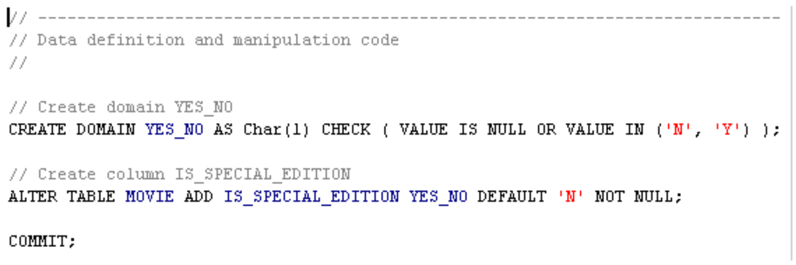
Type	Name

When you have entered the data in the upper frame of the form, click on the + button in the lower frame to add a value rule. Select ‘Value’ from the dialog that appears.



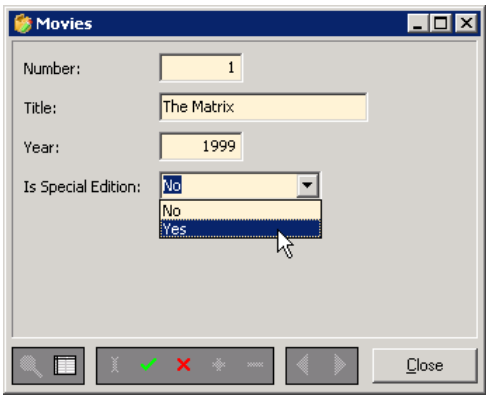
	Keyvalue	Displayvalue
	Y	Yes
*	N	No

Enter the data as displayed above. When finished, close all dialogs and synchronize the model with the database and application (`F9`). Cathedron will generate a SQL-DDL script below to change the database:



```
// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Create domain YES_NO
CREATE DOMAIN YES_NO AS Char(1) CHECK ( VALUE IS NULL OR VALUE IN ('N', 'Y') );

// Create column IS_SPECIAL_EDITION
ALTER TABLE MOVIE ADD IS_SPECIAL_EDITION YES_NO DEFAULT 'N' NOT NULL;

COMMIT;
```

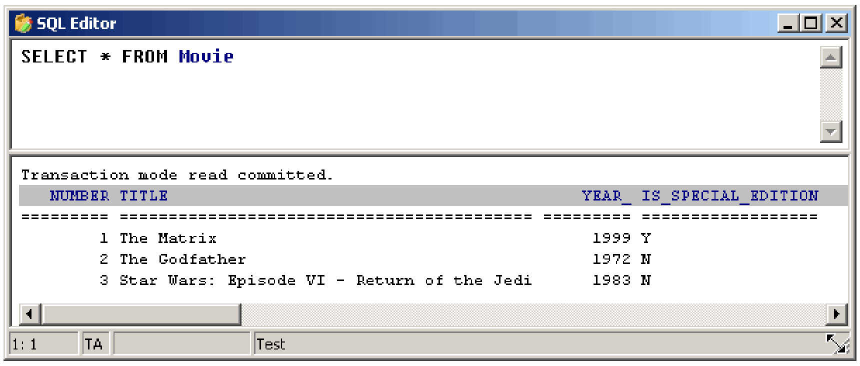
Execute the script, close the dialog and open the default form for the Movie table.



Number:	1
Title:	The Matrix
Year:	1999
Is Special Edition:	No

Open the search dialog (`F5`) and search The Matrix. Press `F2` to edit the information and select Yes from the Is Special Edition field.

To examine the actual data stored in the database select the SQL Editor option from the Structure menu. Enter the query specified below and press `Ctrl Enter` to execute it.



```
SELECT * FROM Movie
```

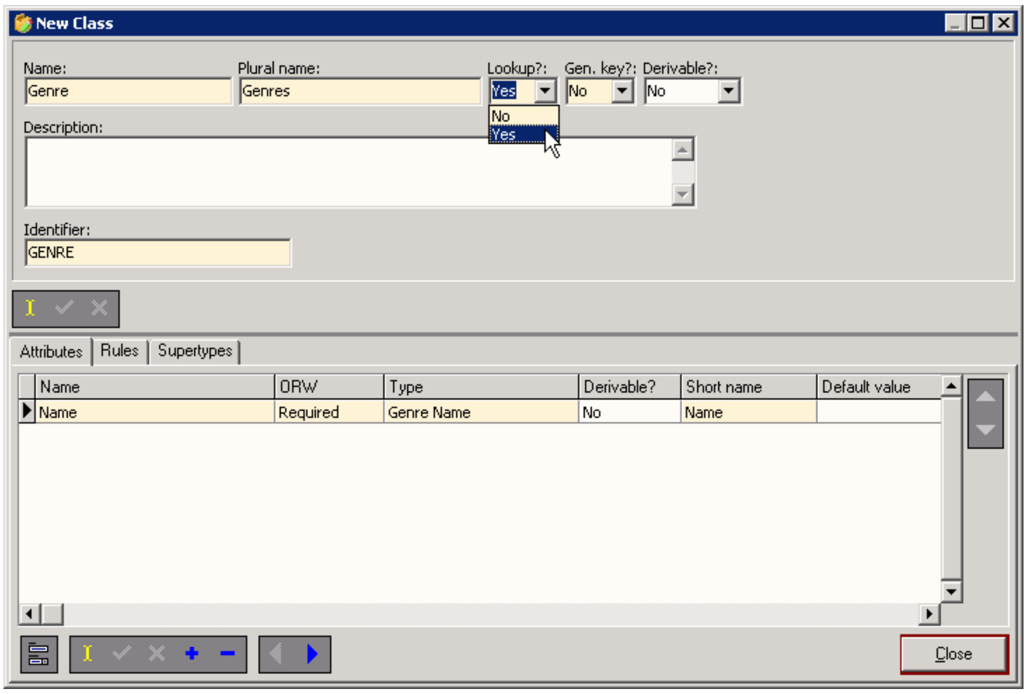
```
Transaction mode read committed.
   NUMBER TITLE                                       YEAR_  IS_SPECIAL_EDITION
========= ========================================== ========= ==================
        1 The Matrix                                      1999 Y
        2 The Godfather                                   1972 N
        3 Star Wars: Episode VI - Return of the Jedi      1983 N
```

As you can see the data is stored in a Char(1) field while the interface displays a list with ‘Yes’ and ‘No’.

3.7 Extending the model

Relations between classes

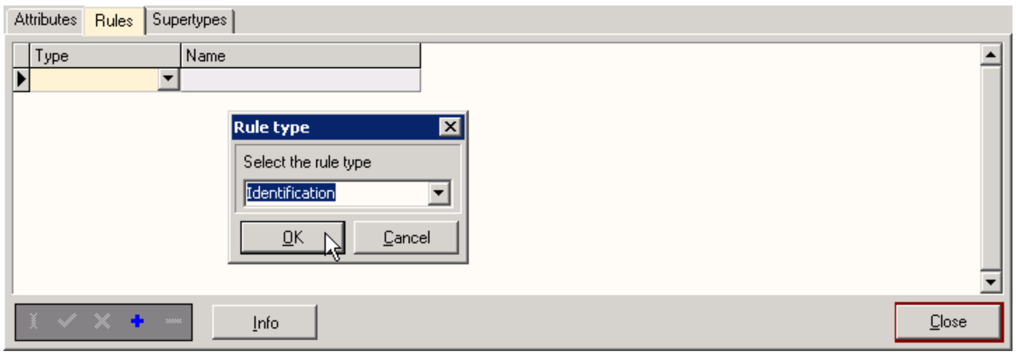
In this example we will add information about genres to our Movie class. Close all forms and return to the Structure modeler. Create a new class on the right side of the Movie class. Right click the canvas and select New from the menu to show the New Class dialog.



Name: Genre
Plural name: Genres
Lookup?: Yes (No / Yes)
Gen. key?: No
Derivable?: No
Description:
Identifier: GENRE

Name	ORW	Type	Derivable?	Short name	Default value
Name	Required	Genre Name	No	Name	

Create the Genre class as specified above. Notice the Lookup field is set to Yes. The Genre Name attribute should be a Varchar with length 20.



Type	Name

Rule type

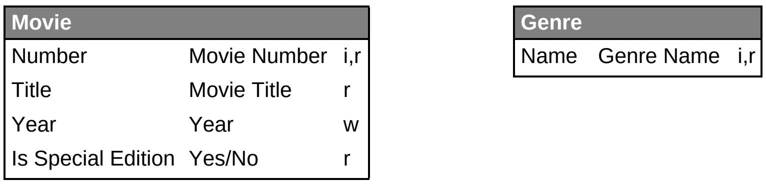
Select the rule type

Identification

OK Cancel

Switch to the second tab in the bottom frame `(Alt 2)` to add an identification rule for the class. Press `F9` to add a new rule and select Identification as the type. A dialog will appear to specify the attributes covered by the identity rule. Add the Name attribute to this list, accept the data and close the dialogs. (When we created the movie class we used the mouse and the popup menu to add the identification rule).

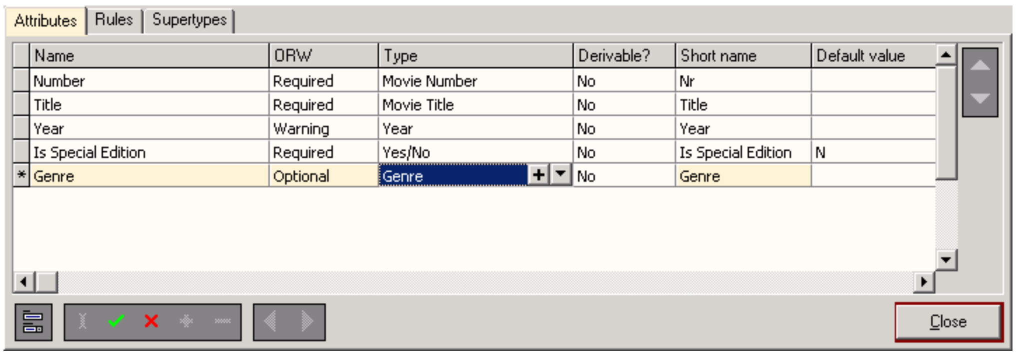
You diagram should now look like this:



Movie		
Number	Movie Number	i,r
Title	Movie Title	r
Year	Year	w
Is Special Edition	Yes/No	r

Genre		
Name	Genre Name	i,r

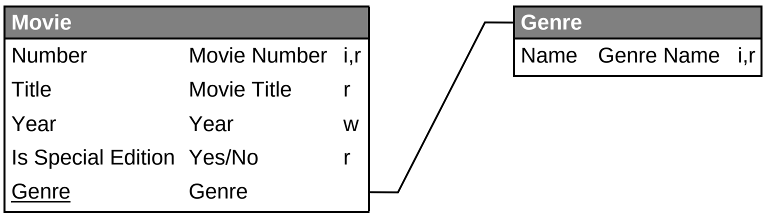
In the next step we will add a relation between the two classes. Right click the title of the Movie class and select Edit from the popup menu. Press `Alt 1` to focus the attribute frame. Press `F9` to add a new attribute.



	Name	ORW	Type	Derivable?	Short name	Default value
	Number	Required	Movie Number	No	Nr	
	Title	Required	Movie Title	No	Title	
	Year	Warning	Year	No	Year	
	Is Special Edition	Required	Yes/No	No	Is Special Edition	N
*	Genre	Optional	Genre	No	Genre	

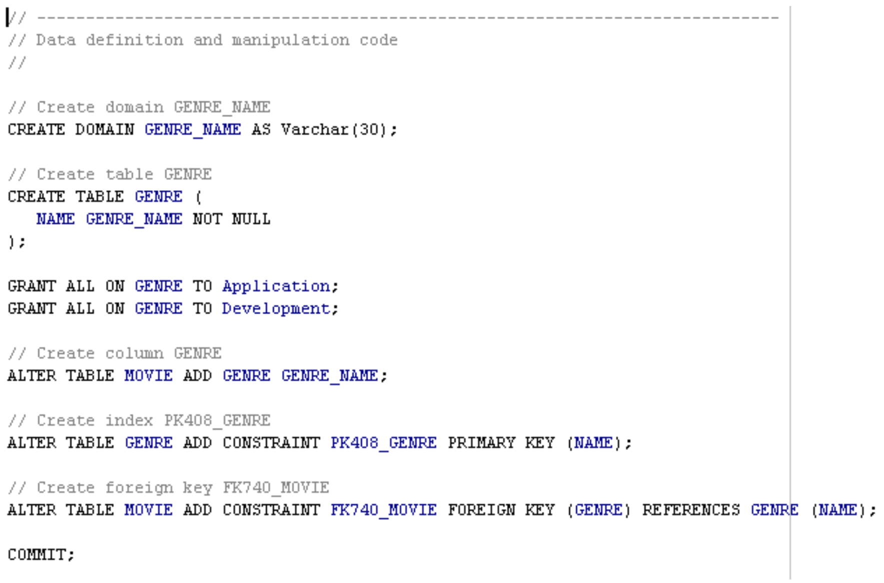
Enter ‘Genre’ as the name for the attribute. Enter Optional in the ORW field.

Notice Cathedron has already selected a default value for the type. The Genre class we have just created has been added to the list of types. Cathedron assumed you wanted to make a reference to this Class. (The attribute you are creating is a so called referring or referencing attribute). Accept the proposal and complete the new attribute by pressing `F3` and close the dialog. Your diagram should now look like this:



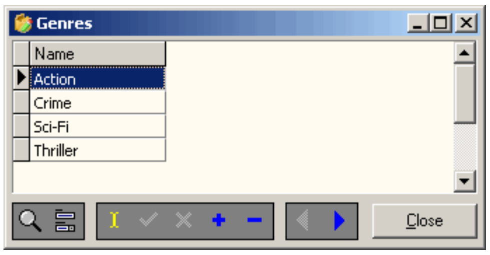
Notice we have created an optional attribute. We can only add a required attribute by specifying a default value. Since we cannot determine a valid default value, we had to create an optional attribute.

Press `F9` to synchronize the changes in the model with the database and the application.



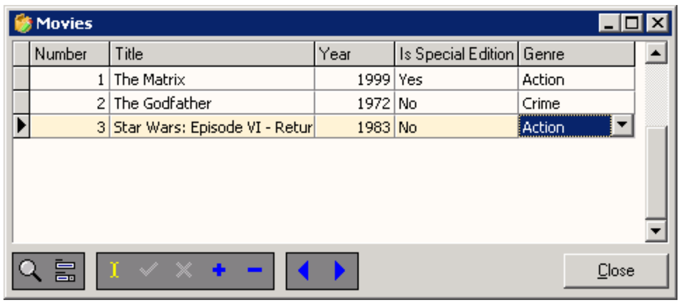
```
// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Create domain GENRE_NAME
CREATE DOMAIN GENRE_NAME AS Varchar(30);

// Create table GENRE
CREATE TABLE GENRE (
   NAME GENRE_NAME NOT NULL
);

GRANT ALL ON GENRE TO Application;
GRANT ALL ON GENRE TO Development;

// Create column GENRE
ALTER TABLE MOVIE ADD GENRE GENRE_NAME;

// Create index PK408_GENRE
ALTER TABLE GENRE ADD CONSTRAINT PK408_GENRE PRIMARY KEY (NAME);

// Create foreign key FK740_MOVIE
ALTER TABLE MOVIE ADD CONSTRAINT FK740_MOVIE FOREIGN KEY (GENRE) REFERENCES GENRE (NAME);

COMMIT;
```

Execute the generated script and test the application by opening the default form for Genre.



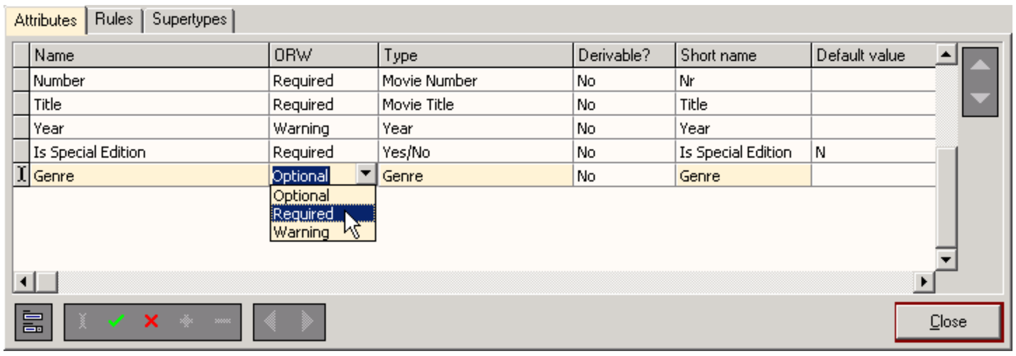
Name
Action
Crime
Sci-Fi
Thriller

Add the genres as specified above. Close the form and open the Movie form. Press `F5` to open the search dialog. Press the `!` to change the search operator into `Not empty` and press `Enter` to start the search. When the Search result dialog appears click Select all to transfer all the movies to the form.



	Number	Title	Year	Is Special Edition	Genre
	1	The Matrix	1999	Yes	Action
	2	The Godfather	1972	No	Crime
▶	3	Star Wars: Episode VI - Retur	1983	No	Action

Edit the movies to set the genres as show above. Since every movie has a genre we can now change the Genre attribute of the Movie class into a required attribute. To do this: Close the Movie form, return to the Structure modeler and edit the Movie class as shown below.



Name	ORW	Type	Derivable?	Short name	Default value
Number	Required	Movie Number	No	Nr	
Title	Required	Movie Title	No	Title	
Year	Warning	Year	No	Year	
Is Special Edition	Required	Yes/No	No	Is Special Edition	N
Genre	Optional	Genre	No	Genre	

Close the dialog and synchronize the model (`F9`) and execute the generated script.

```
// ---------------------------------------------------------------------------
// Checks
//

SET SUPPRESS_ERROR ON;

// Change: Column GENRE becomes required.
// Check: If column GENRE does not yet exists, table MOVIE should be empty.
CHECK SELECT FIRST 1 * FROM MOVIE WHERE NOT EXISTS (SELECT RDB$Field_Name FROM RDB$Relation_fields WHERE RDB
// Check: If column GENRE exists, every record of table MOVIE should have a value for this column.
CHECK SELECT FIRST 1 * FROM MOVIE WHERE GENRE IS NULL AND EXISTS (SELECT RDB$Field_Name FROM RDB$Relation_fi

SET SUPPRESS_ERROR OFF;

// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Changing NOT NULL rule of column GENRE
ALTER TABLE MOVIE ADD TMP$GENRE GENRE_NAME NOT NULL;
UPDATE MOVIE SET TMP$GENRE=GENRE;
ALTER TABLE MOVIE DROP CONSTRAINT FK740_MOVIE;
ALTER TABLE MOVIE DROP GENRE;
ALTER TABLE MOVIE ADD GENRE GENRE_NAME NOT NULL;
UPDATE MOVIE SET GENRE=TMP$GENRE;
ALTER TABLE MOVIE DROP TMP$GENRE;

// Recreate foreign key FK740_MOVIE
ALTER TABLE MOVIE ADD CONSTRAINT FK740_MOVIE FOREIGN KEY (GENRE) REFERENCES GENRE (NAME);

COMMIT;
```

Deploying changes in a production environment

The generated scripts started with a section that checks the database to see if we had actually entered values for every row in the Genre column of the Movie table. This section was added for situations where you have separated the development environment from the actual production/live environment.

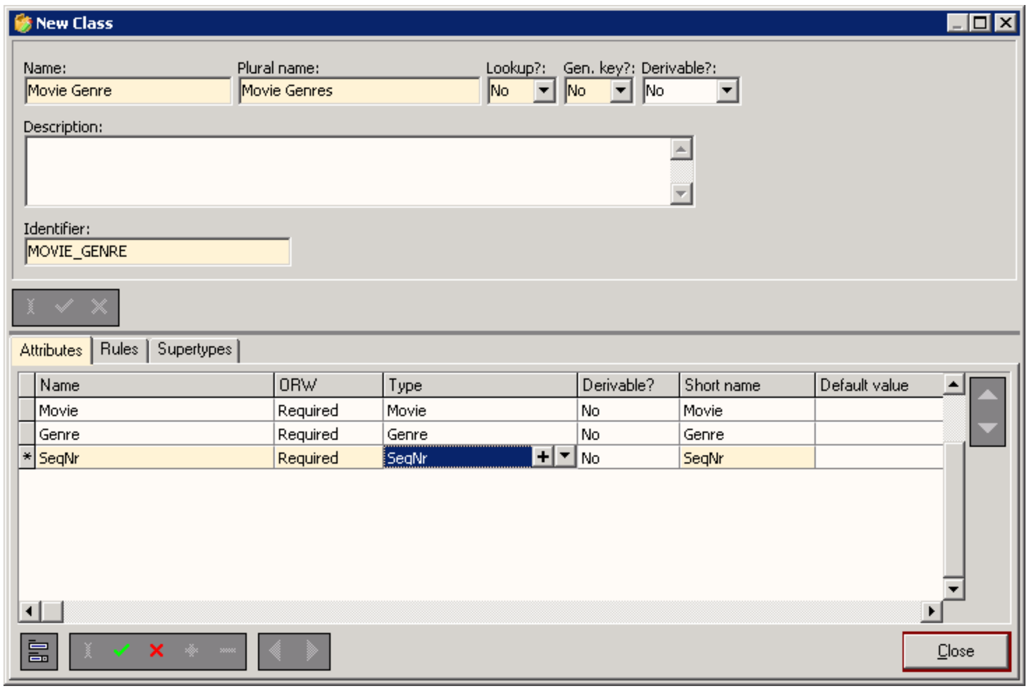
Although Cathedron has checked the data in the current database, it cannot check the data in the production environment. When you deploy the generated script on the production environment, it will stop the script in case one of the checks fail.

All generated scripts are stored in a log. You can view this log by selecting the Development Log option from the Structure menu.

Adding a n:m relation / Intersection classes

Websites such as www.imdb.com use multiple genres to categorize a movie. We will now the structure to add an ordered n:m relation to the model.

Close all dialogs and return to the modeler. Right click the canvas below Movie and Genre to create a new class. Choose New from the popup menu and enter the data as shown below.

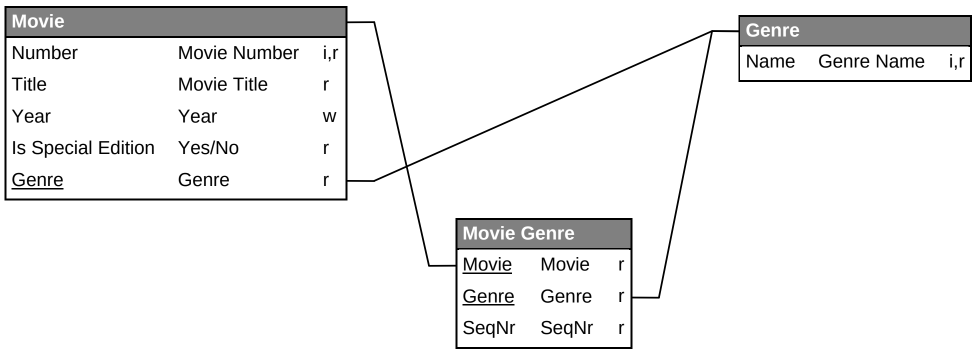


New Class

Name: Movie Genre
Plural name: Movie Genres
Lookup?: No
Gen. key?: No
Derivable?: No
Description:
Identifier: MOVIE_GENRE

	Name	ORW	Type	Derivable?	Short name	Default value
	Movie	Required	Movie	No	Movie	
	Genre	Required	Genre	No	Genre	
*	SeqNr	Required	SeqNr	No	SeqNr	

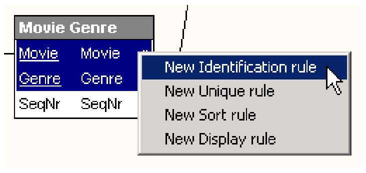
The attributes Movie and Genre refer to their respective classes. The attribute SeqNr needs a new type named ‘SeqNr’. The data type for the SeqNr is ‘Integer’. The sequence number is an additional attribute used to order the genres for each movie. After closing the dialog your diagram should look like this:



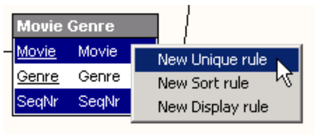
Cathedron does not display n:m relations as lines between classes. A n:m relation is always created and displayed using an so called intersection class. As a result every line in any diagram style is always a 0 or 1 to many relation. In the ERD and Cathedron style such lines always start at the referencing attribute and end at the title of the class they are referring to. The keep the diagrams clear the Cathedron style does not display the cardinality symbols for standard 0:n relations[1].

Before synchronizing the model we should add the identification rule. We will do this using the diagrams popup menu's. (We could just as well have done this from the New Class dialog as shown earlier).

Hold the `Shift` key and click on the Movie attribute. Than hold both the `shift` and the `Ctrl` key and click Genre. Now right click one of the selected attributes and select New identification rule from the menu.

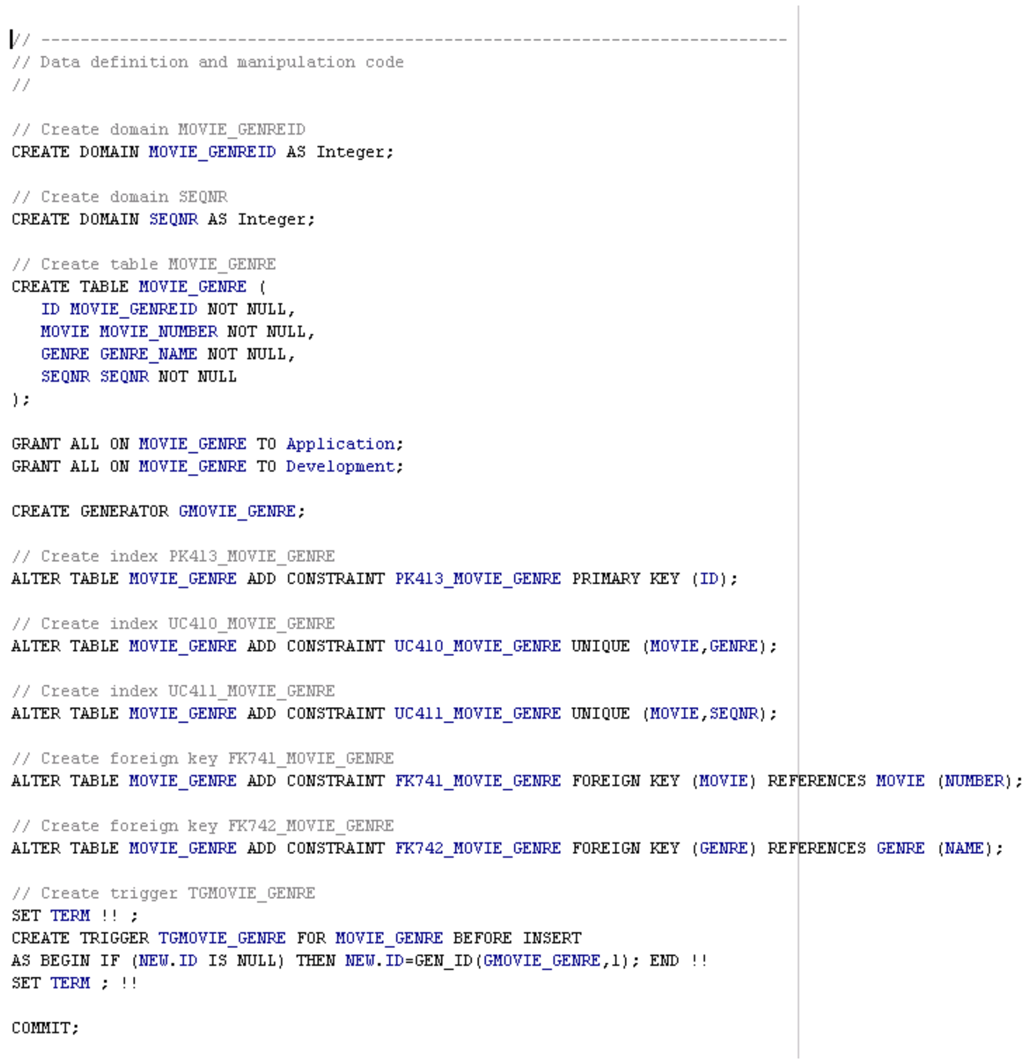


Formally there is another rule stating each sequence number should only occur once for each movie. To add this rule select the Movie and SeqNr attributes and select New Unique rule from the popup menu.



Remark: You can also use the diagram to delete rules. To do this select an attribute and select the Delete … rule option from the menu. After synchronizing the model Cathedron will generate this script:

[1] Reference chapter 2 contains more information on the different diagram styles.

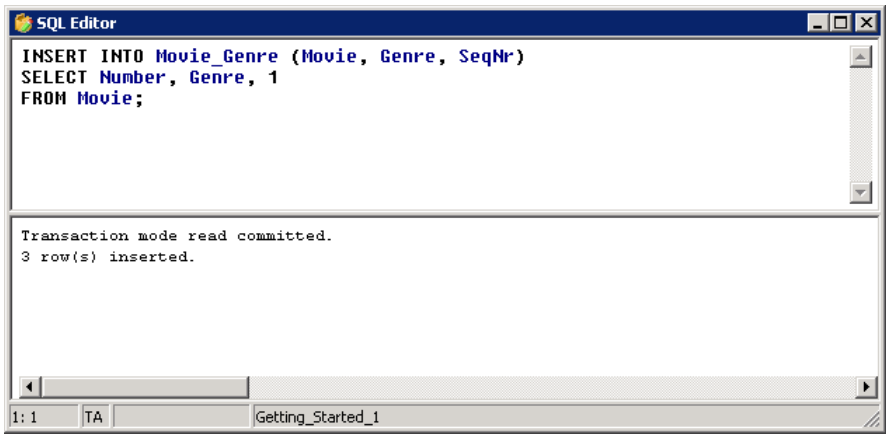


```
// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Create domain MOVIE_GENREID
CREATE DOMAIN MOVIE_GENREID AS Integer;

// Create domain SEQNR
CREATE DOMAIN SEQNR AS Integer;

// Create table MOVIE_GENRE
CREATE TABLE MOVIE_GENRE (
   ID MOVIE_GENREID NOT NULL,
   MOVIE MOVIE_NUMBER NOT NULL,
   GENRE GENRE_NAME NOT NULL,
   SEQNR SEQNR NOT NULL
);

GRANT ALL ON MOVIE_GENRE TO Application;
GRANT ALL ON MOVIE_GENRE TO Development;

CREATE GENERATOR GMOVIE_GENRE;

// Create index PK413_MOVIE_GENRE
ALTER TABLE MOVIE_GENRE ADD CONSTRAINT PK413_MOVIE_GENRE PRIMARY KEY (ID);

// Create index UC410_MOVIE_GENRE
ALTER TABLE MOVIE_GENRE ADD CONSTRAINT UC410_MOVIE_GENRE UNIQUE (MOVIE,GENRE);

// Create index UC411_MOVIE_GENRE
ALTER TABLE MOVIE_GENRE ADD CONSTRAINT UC411_MOVIE_GENRE UNIQUE (MOVIE,SEQNR);

// Create foreign key FK741_MOVIE_GENRE
ALTER TABLE MOVIE_GENRE ADD CONSTRAINT FK741_MOVIE_GENRE FOREIGN KEY (MOVIE) REFERENCES MOVIE (NUMBER);

// Create foreign key FK742_MOVIE_GENRE
ALTER TABLE MOVIE_GENRE ADD CONSTRAINT FK742_MOVIE_GENRE FOREIGN KEY (GENRE) REFERENCES GENRE (NAME);

// Create trigger TGMOVIE_GENRE
SET TERM !! ;
CREATE TRIGGER TGMOVIE_GENRE FOR MOVIE_GENRE BEFORE INSERT
AS BEGIN IF (NEW.ID IS NULL) THEN NEW.ID=GEN_ID(GMOVIE_GENRE,1); END !!
SET TERM ; !!

COMMIT;
```

A closer look at the script shows us that Cathedron has generated an ID column for the Movie_genre table. Cathedron demands that every table has a primary key covering only a single column. When Cathedron transformed the Movie Genre class into a table, it degraded the identity rule on Movie and Genre to a uniqueness constraint in the database. A new column ID covered by the table's primary was generated. Finally a trigger has been generated to automatically generate numbers for the ID column.

Suppose we have a database with hundreds of movies and wanted to copy the genre information from those movies into the new table. To do this we need to create the new structure and write a query to copy the data from the new to the old structure.

Execute the script and close the dialog. Open the SQL Editor from the Structure menu and enter the query as shown below to copy the data:

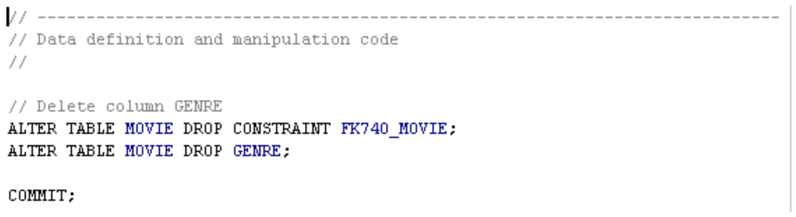


```
INSERT INTO Movie_Genre (Movie, Genre, SeqNr)
SELECT Number, Genre, 1
FROM Movie;
```

```
Transaction mode read committed.
3 row(s) inserted.
```

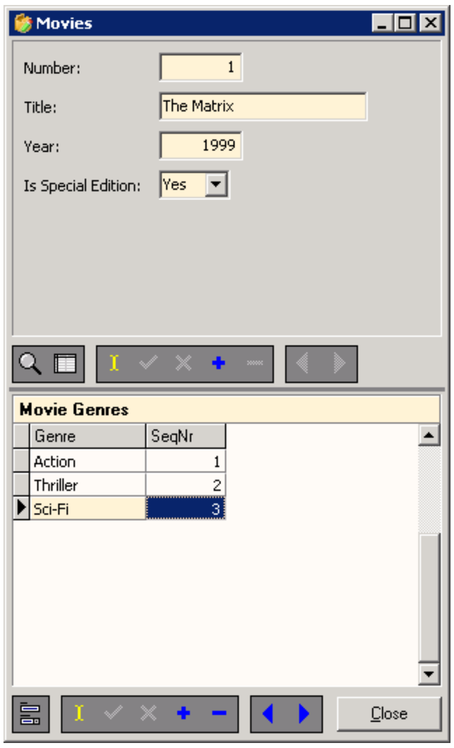
Execute the query by pressing `Ctrl Enter`. Validate the result by checking the output. The message `3 row(s) inserted` corresponds to the actual number of rows in the table so we can safely close the editor (and commit the data when asked!).

We can now edit the Movie class and remove the Genre attribute by clicking the − button in the Attribute frame. There is no shortcut for this option to prevent accidental deletion of data. Close the dialog and synchronize the model.



```
// ---------------------------------------------------------------------------
// Data definition and manipulation code
//

// Delete column GENRE
ALTER TABLE MOVIE DROP CONSTRAINT FK740_MOVIE;
ALTER TABLE MOVIE DROP GENRE;

COMMIT;
```

The script will simply drop the genre column from the movie table. Execute the script and open the default Movie form.


Movies

Number:	1
Title:	The Matrix
Year:	1999
Is Special Edition:	Yes

Movie Genres

Genre	SeqNr
Action	1
Thriller	2
Sci-Fi	3

The Genre attribute has disappeared from the top frame. A lower frame has been created to show and maintain multiple genres for each movie. The tutorial about the Cathedron Interface Language (CIL) will show how to use and hide the SeqNr field.

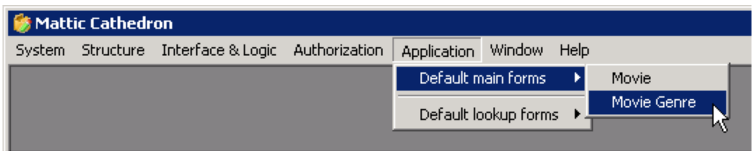
Notice the generated ID field of Movie Genre was not displayed. It is maintained automatically and is assumed not to be interesting from the user's point of view.

3.8 Testing the application

So far we have been working in the development environment of Cathedron. To test the application environment we have three options.

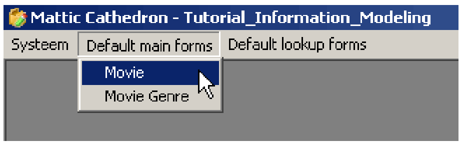
Examining the application menu from within the development environment

Open the Application menu to see the default application menu.



You can open any default form you whish to inspect from this menu.

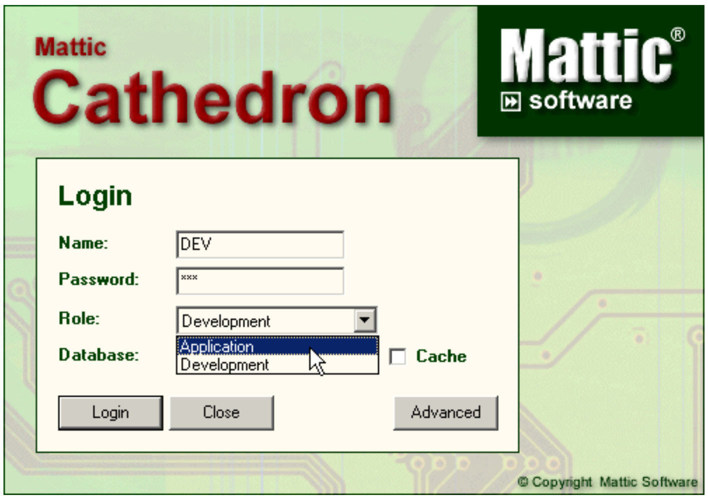
Switching to the application menu



A second option is to switch to the application menu by selecting the Application menu option from the System menu.

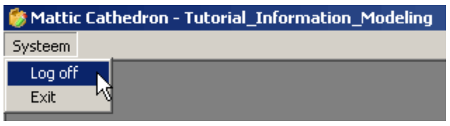
Logging in as a user

Finally we can login as a user. Select the Log off option from the System menu. When the login dialog appears, click the Advanced button and change the Role field to 'Application'.



Name: DEV
Password: ***
Role: Development
Application
Development
Database:

Click Login to login as an application user.

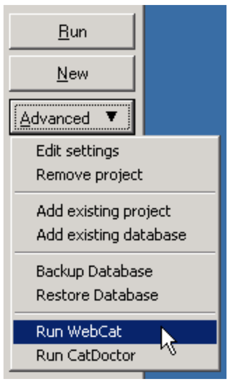


You will find yourself in an empty Cathedron environment with no other options than Log off and Exit. This is done for security reasons. If we would show the default menu we might have exposed screens to users they are not allowed to see.

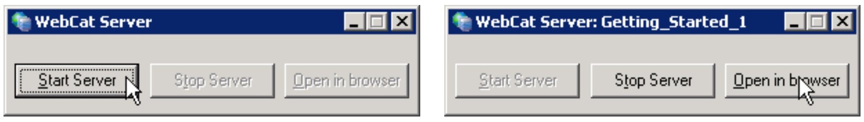
Before users can work with the application you will need to learn how to create a custom menu. This is explained in the Cathedron Interface Language (CIL) tutorial.

3.9 Testing the application from a web browser

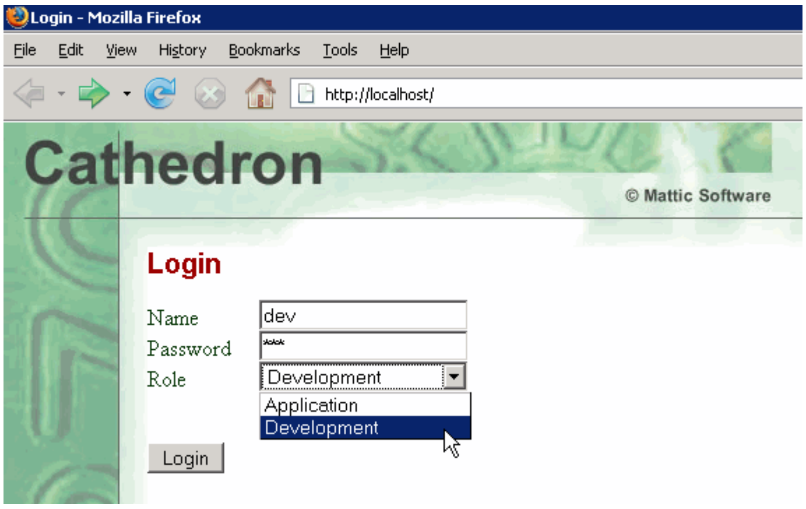
To see the application from a web browser you need to close Cathedron and switch back to the Project manager[1].



Click the Advanced button and select Run WebCat from the menu. A small window will appear to control a development version of the WebCat Server.



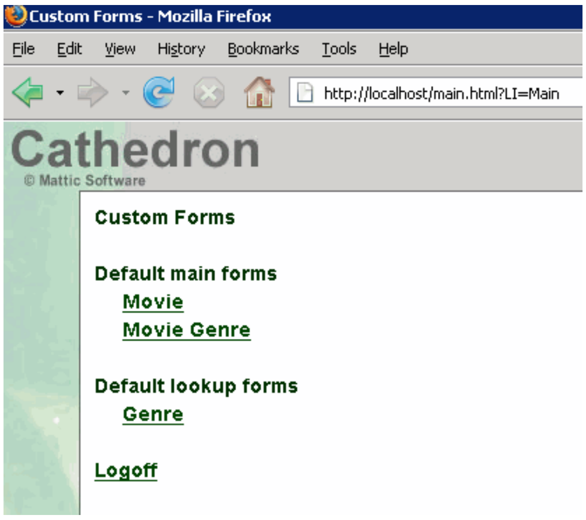
First click Start Server, then click Open in browser. The WebCat Server will start and your standard web browser will open.



Name: dev
Password: ***
Role: Development

[1] Since the Embedded Firebird Database does not allow multiple connections to a database you have to close Cathedron. If you are using the normal version of Firebird or InterBase there is no need to close Cathedron.

As you can see from the address bar the browser will connect to the http port (80) on your machine. Login to the server with the default development account (dev/dev) and change your role to Development. (If you forget this, you will not see a menu).


Custom Forms - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/main.html?LI=Main

Cathedron
© Mattic Software

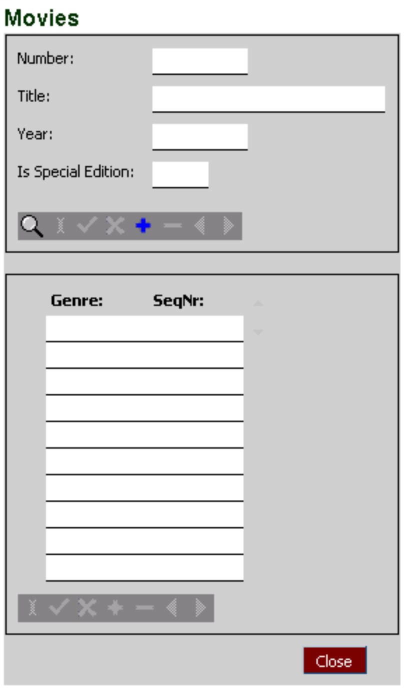
Custom Forms

Default main forms
- Movie
- Movie Genre

Default lookup forms
- Genre

Logoff

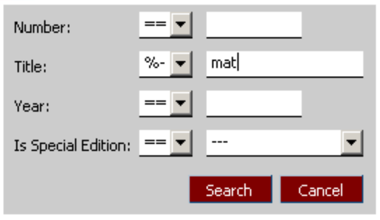
The browser will now show the default menu. Click on the link to the Movie form to open it.


Movies

Number:	
Title:	
Year:	
Is Special Edition:	

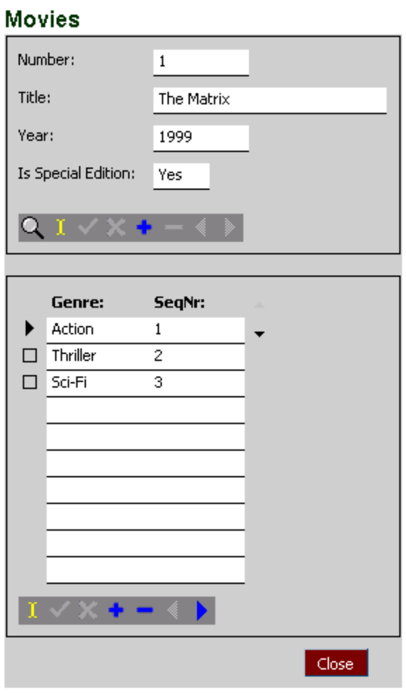
Genre:	SeqNr:

Click on the button to open the search form.



Number:	==	
Title:	%-	mat
Year:	==	
Is Special Edition:	==	---

Change the search operator to % (contains) and enter ‘mat’ as the search criteria. Click on the Search button to start the search.


Movies

Number:	1
Title:	The Matrix
Year:	1999
Is Special Edition:	Yes

Genre:	SeqNr:
Action	1
Thriller	2
Sci-Fi	3

The web server will now show the Movie form containing the Matrix.

Remark: The WebCat Server serves plain HTML forms. Data is validated on the server and no attempts were made to validate the data within the browser.

This ends the Information Modeling Tutorial. You can now continue with the Cathedron Interface Language tutorial for more information about creating customized interfaces.

4 Tutorial
Cathedron Interface Language (CIL)

Tutorial Cathedron Interface Language (CIL)

This part of the manual was not yet finished at the time this Cathedron distribution was produced. We kindly request you to check the download page of Cathedron on www.mattic.com for a new version of this manual.

5 Tutorial Pascal Script

Tutorial Pascal Script

This part of the manual was not yet finished at the time this Cathedron distribution was produced. We kindly request you to check the download page of Cathedron on www.mattic.com for a new version of this manual.

R Reference Guide

R1 SQL Editor / IQU

This part of the manual was not yet finished at the time this Cathedron distribution was produced. We kindly request you to check the download page of Cathedron on www.mattic.com for a new version of this manual.

R2 Structure Diagram Styles

This part of the manual was not yet finished at the time this Cathedron distribution was produced. We kindly request you to check the download page of Cathedron on www.mattic.com for a new version of this manual.

R3 Edit Formats

This part of the manual was not yet finished at the time this Cathedron distribution was produced. We kindly request you to check the download page of Cathedron on www.mattic.com for a new version of this manual.

R4 Display Formats

This part of the manual was not yet finished at the time this Cathedron distribution was produced. We kindly request you to check the download page of Cathedron on www.mattic.com for a new version of this manual.

A Appendices

A1 Shortcuts / Hotkeys

Cathedron

Shortcut	Name	Function
		Global shortcuts are listed behind the options in the main menu.
`Alt 0`	Window list	Shows a list of open windows.
`Pause`	Lock	Locks Cathedron by showing the login screen. Forms will be hidden, not closed. You remain the state of your session.

Forms, wizards and statusviews

Shortcut	Name	Function
`F1`	Help	Shows the help screen.
`F11`	Refresh	Refresh all information. The information will be re-fetched from the database.
`F12`	Close dialog Next page	Close dialog. Next page in a wizard or close wizard.
`Ctrl F12`	Previous page	Previous page in a wizard.

`Ctrl 1`
`Ctrl 2`
…
`Ctrl 9`
	Select upper frame x	Most forms have one top frame. You can quickly move the focus to this frame by pressing `Ctrl 1`.
		In case you every come across a form with multiple frames on the top (each with it's own tab) you can select these frame by holding the `Ctrl` key and pressing the number of the frame.

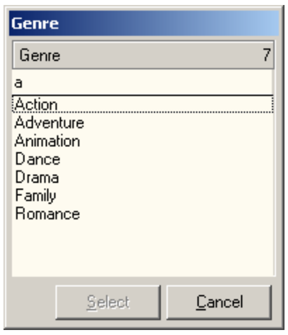
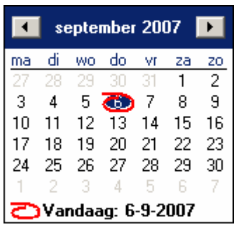
`Alt 1`
`Alt 2`
…
`Alt 9`
	Select lower frame x	If a form has lower frames you can select those frames by holding the `Alt` key and pressing the number of the frame.

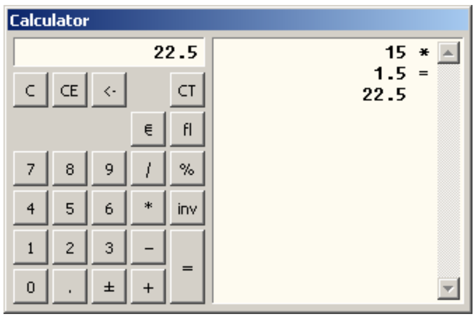
Frames within a form, wizard or statusview

Shortcut	Button	Name	Function
`F5`	🔍	Search	Opens the search dialog
`F8`	▤ ▦	Toggle Record / Grid View	Toggles between record and grid view. Record view shows a single record using separate edit controls. Grid view shows multiple records in tabular form.
`F2`	I	Edit	Enable editing.
`F3`	✓	Accept	Accept changes.

`F4`	✕	Cancel	Discard changes.
`F9`	+	Insert	Insert a new record.
	−	Delete	Delete the current record.
`Ctrl ,`	◀	Previous record	Go to the previous record. (When multiple records are retrieved from the database).
`Ctrl .`	▶	Next record	Go to the next record. (When multiple records are retrieved from the database.
`Ctrl ↑`	▲	Record up	Used in ordered lists. Moves the record up in the list.
`Ctrl ↓`	▼	Record down	Used in ordered lists. Moves the record down in the list.

Controls within a frame

Applies to	Shortcut	Name	Function
Combo boxes.	`Alt ↓`	Open Combo	Opens a combo box. Note: Instead of opening a combo box you can also search an item by entering the first characters of the item.
	`Ctrl Enter`	Open Quick Search	Opens the Quick Search dialog to easily search in the values of a combo box.  Genre Genre 7 a Action Adventure Animation Dance Drama Family Romance Just type some letters of the value and press `Enter` to search. Use the arrow keys to select the option and press `Enter` to accept it.
Edit controls with a + button.	`Ctrl Shift Enter`	Add item	Opens the dialog to add an item to a list.
Date/Time fields. 6-9-2007	`Alt ↓`	Open Editor	Shows the date editor.  september 2007 ma di wo do vr za zo 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 Vandaag: 6-9-2007
	`Space`	Current Date	Enters the current date into the control.

	`Del`	Erase Date	Erases the date.
Numeric fields.	`Alt ↓`	Open Calculator	Opens the calculator to calculate values.  Calculator 22.5 15 * 1.5 = 22.5 C CE <- CT € fl 7 8 9 / % 4 5 6 * inv 1 2 3 – = 0 . ± + Enter the calculation, check the results and press `Ctrl Enter` to transfer the result to the edit control. Press `Esc` to ignore the calculation. Press `Ctrl Del` to start a new calculation within the calculator.

Search Operators

Operator	Symbol	Shortcut
Equal (case insensitive)	=	=
Less	<	<
Greater	>	>
Less/Equal	<=	<no shortcut>
Greater/Equal	>=	<no shortcut>
Between	..	.
Contains	%	%
Starts with	%-	<no shortcut>
Ends with	-%	<no shortcut>
Empty	#	#
Not empty	!	!
Not equal	<>	<no shortcut>
Equal (case sensitive)	==	+

A2 Adding the default development user in Firebird

To create the development user:

- Select Run from the start menu.
- Run cmd to open a command line window.
- Change to the Firebird binary folder.
  ```
  cd c:\Program files\Firebird\Firebird_2_0\bin
  ```
- Run gsec using this command line:
  ```
  gsec -user sysdba -pass masterkey -add dev2 -pw dev
  ```
- Close the command line window.

For more information on gsec you can invoke the built in help using the `gsec ?` command or consult the Firebird Quick Start Guide. In the chapter about ‘Server configuration and management’ you will find the section ‘User management: gsec’.