Introduction

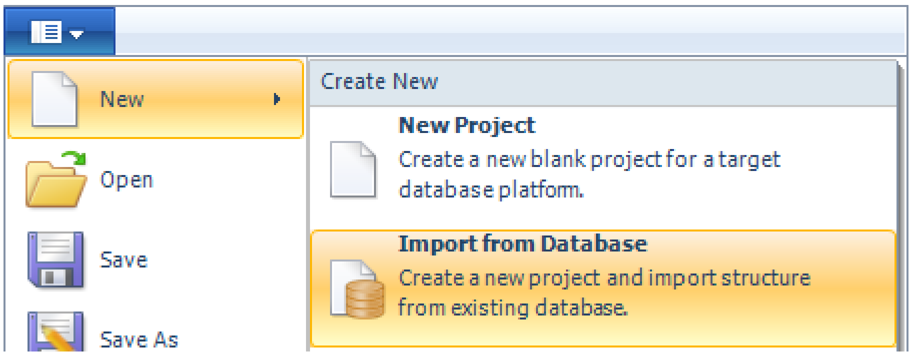
Data modeling is a mandatory requirement throughout the lifetime of a database system: from the initial design of the software, when the first structure is created and modeled, to later on, in system updates, when database structure is modified. There are also situations where manipulation of database structure may be a complicated task, such as when there is a need to get a productive system and to work on a non-documented database, or convert a database from a DBMS to another. There are several tools related to data modeling on the market: some DBMS-specific, others generic; some useful to specific and isolated tasks, others offering a multitude of features (and usually not very cheap). TMS Data Modeler is a tool that provides nothing but essential features for creating and maintaining a database: it integrates database design, modeling, creation and maintenance into a single environment, in a simple and intuitive user interface to manipulate databases efficiently. This article briefly describes the main features of TMS Data Modeler, demonstrating how it may be used to create a project and maintain an existing database.

Data Modeler main features

TMS Data Modeler is a generic tool that allows data modeling independently of the used DBMS, in an easy-to-use interface. The application allows you to start modeling a database from scratch, as well as import the structure of an existing database (reverse engineering). It allows you to generate scripts to create the full database, or upgrade an existing database with update script, through its version control system. In addition, it provides features for conversion from one DBMS to another, consistency check and visualization of entity-relationship diagrams, among others. Data Modeler supports several database management systems, currently: Absolute Database, Firebird 2, MS SQL Server 2000/2005/2008, MySQL 5.1, NexusDB V3 and Oracle 10g.

Creating a project in TMS Data Modeler

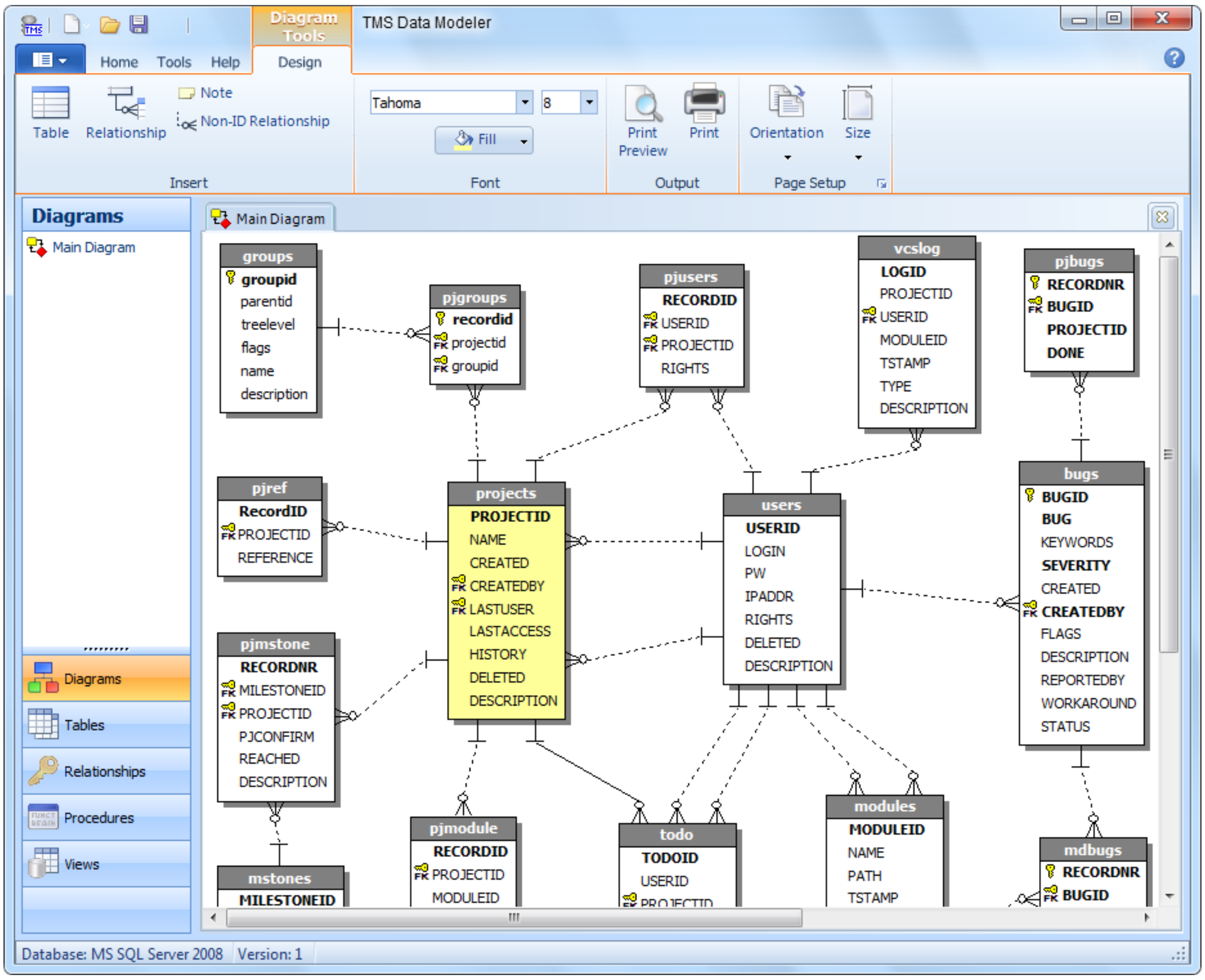
There are two ways to start a project in Data Modeler: creating a new project from scratch or importing data dictionary from an existing database.



New Project
Create a new blank project for a target database platform.

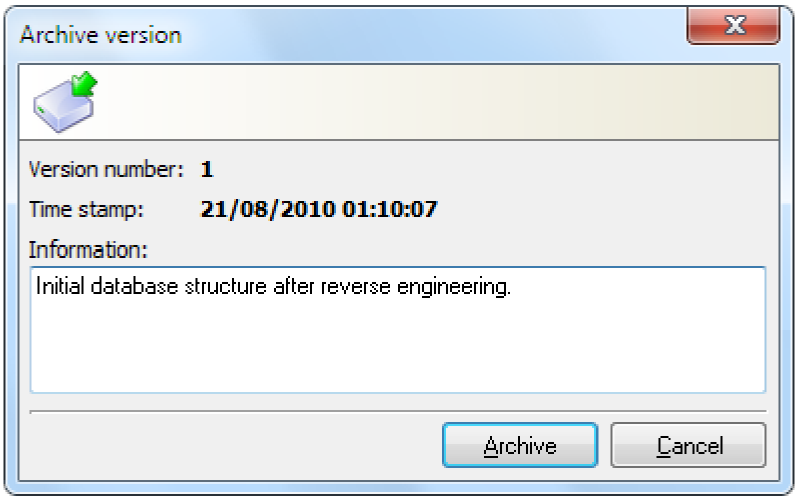
Import from Database
Create a new project and import structure from existing database.

Choosing the "New Project" option, just select the target database and an empty project will be created. By default Data Modeler provides a diagram named "Main Diagram". Tables and relationships between them can be created visually through the diagram. All objects in the database (apart from tables we can have procedures, views, etc.) can be accessed, created and edited through the Project Explorer, located on the left of the screen.
In the "Import from Database" option you need to configure the connection to the database whose structure will be imported. After importing the structure, Data Modeler will hold all the database objects: tables, relationships, triggers, procedures, views, etc. All objects are listed at Project Explorer on the left, in their respective category. For an overview of the imported structure, it is possible to open the "Main Diagram" and select "Add all tables" from the context menu.



Versioning database

To use the version control features of TMS Data Modeler (comparison and identification of changes, script generation for upgrade) it is necessary to archive versions so they are kept in history. By default any new project starts at version 1, with the status "under construction". In this example, after importing the structure of an existing database, we will archive the first version. Once this version is archived, its status is changed to "closed", and a new version (2) is created with the status "under construction". All changes from now on will be part of the second version.


Archive version

Version number: 1

Time stamp: 21/08/2010 01:10:07

Information:

Initial database structure after reverse engineering.

Creating and editing database objects

The Project Explorer on the left provides access to viewing and editing interfaces to all existing objects, as well as creating new ones through the context menu. Data Modeler's interface allows multiple objects to be opened simultaneously, organized into tabs, which allows easy navigation among them. In this example we will create a new table named "attachments", and later we will relate it to the existing table "projects". We will also create a new field "filesize" in the table "blobs".

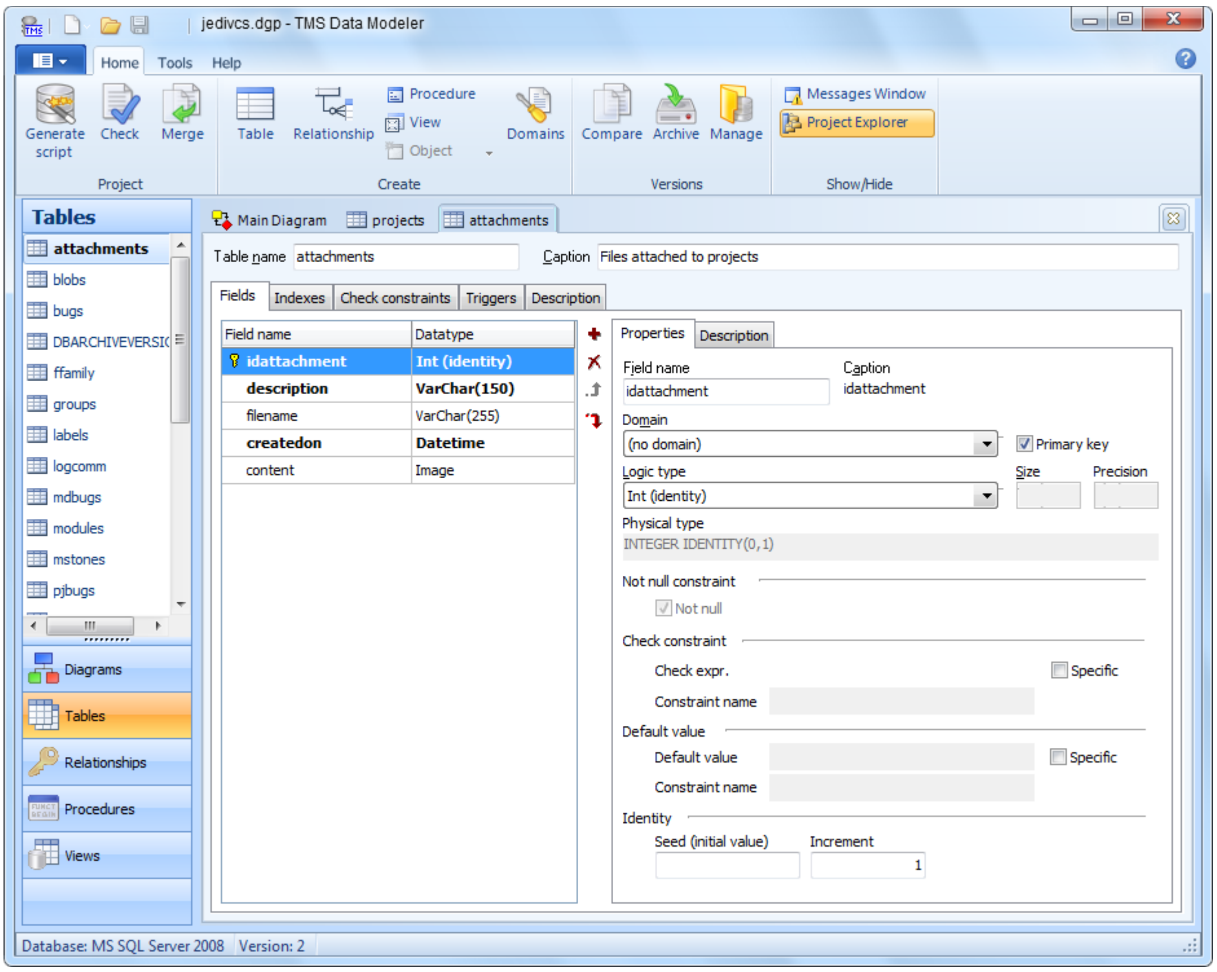


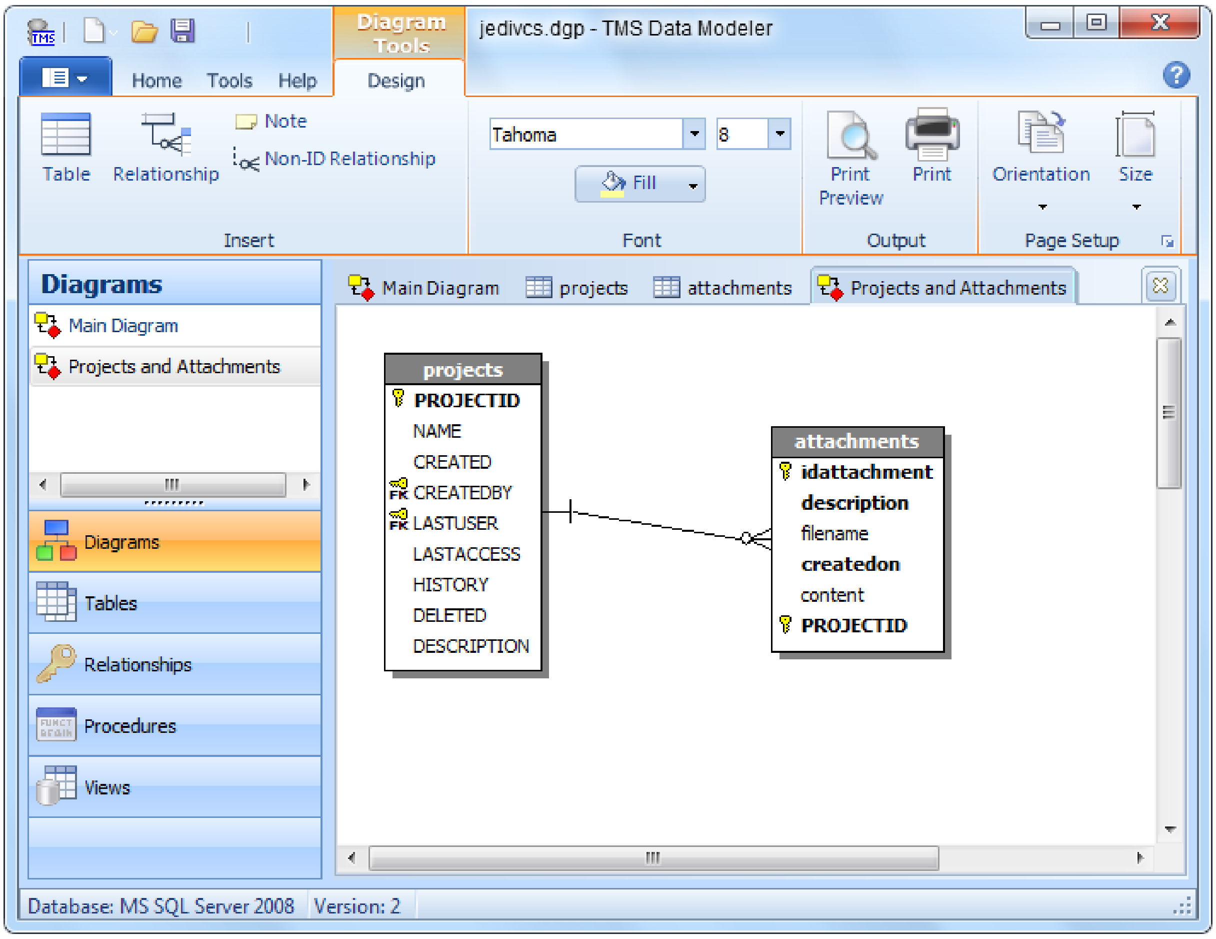
Table name: attachments
Caption: Files attached to projects

Field name	Datatype
idattachment	Int (identity)
description	**VarChar(150)**
filename	VarChar(255)
createdon	**Datetime**
content	Image

Field name: idattachment
Caption: idattachment
Domain: (no domain)
[✓] Primary key
Logic type: Int (identity)
Size:
Precision:
Physical type: INTEGER IDENTITY(0,1)
Not null constraint
[✓] Not null
Check constraint
Check expr.:
[ ] Specific
Constraint name:
Default value
Default value:
[ ] Specific
Constraint name:
Identity
Seed (initial value):
Increment: 1

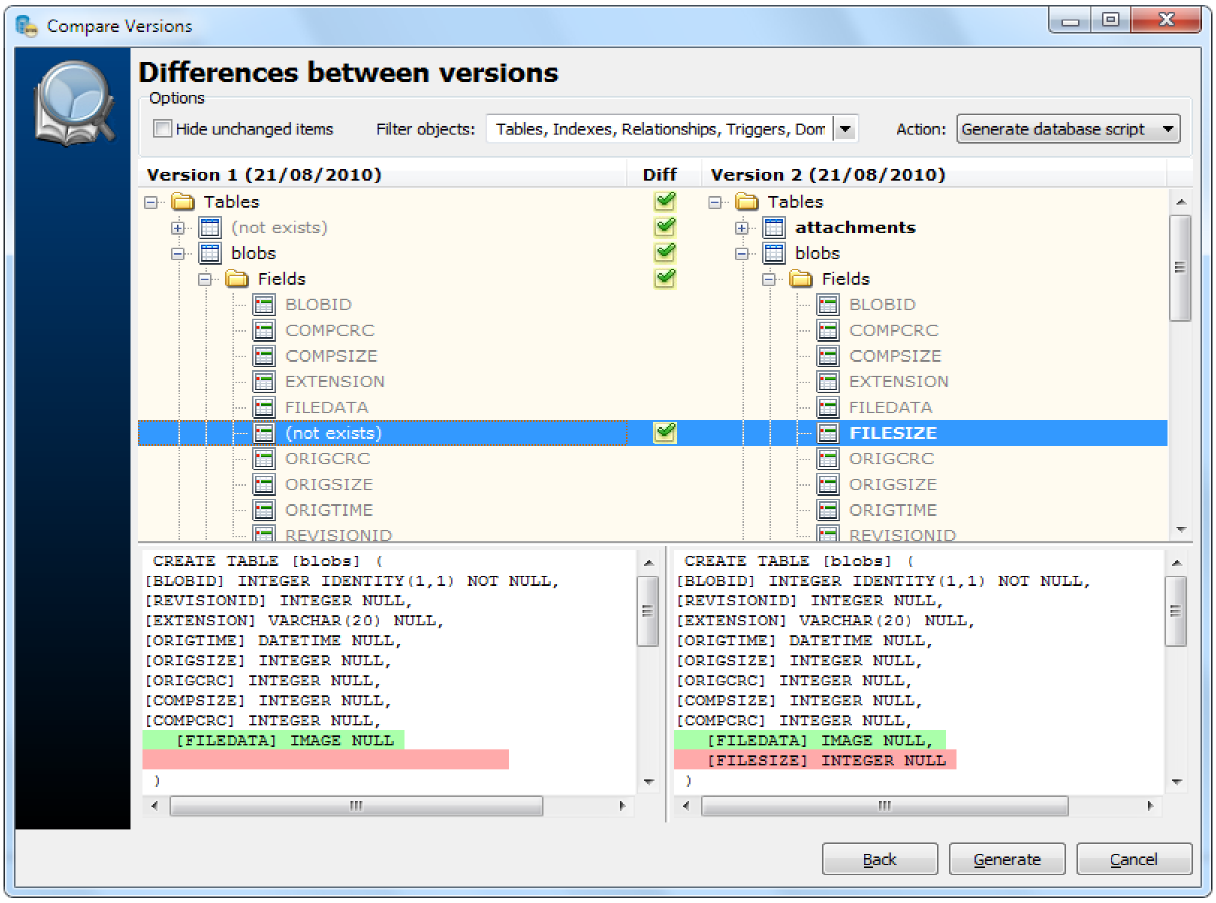
Database: MS SQL Server 2008 Version: 2

Data Modeler allows you to create different diagrams by dragging the desired tables from Project Explorer to the diagram area; related tables are automatically linked. As a result, it is possible to see different sets of tables, separated by module or system context.
To relate the two tables visually, through the diagram, insert a new diagram to make viewing easier, and drag the "projects" and "attachments" tables from Project Explorer to diagram area. Selecting the "Relationship" button on the toolbar, just click on the parent table and drag the mouse to the child table. This will display a window for inserting a new relationship, in which we can set its name, keys and other options. After confirmation, the relationship between the tables is created and displayed immediately in the diagram. Note that a field "ProjectID" was automatically created in the "attachments" table, related to the primary key in the "projects" table.



Version upgrade

After the changes are made in the project, version 2 becomes different from version 1, which was archived right after importing the structure from the database. Using the comparing versions tool of Data Modeler, we can view the structure of each version side by side, with their differences highlighted (created, removed or changed objects), and the creation script of selected objects. On the same screen we can select the changes to generate a script to update the database.


Compare Versions
Differences between versions
Options
Hide unchanged items
Filter objects: Tables, Indexes, Relationships, Triggers, Dom
Action: Generate database script
Version 1 (21/08/2010)
Diff
Version 2 (21/08/2010)
Back
Generate
Cancel

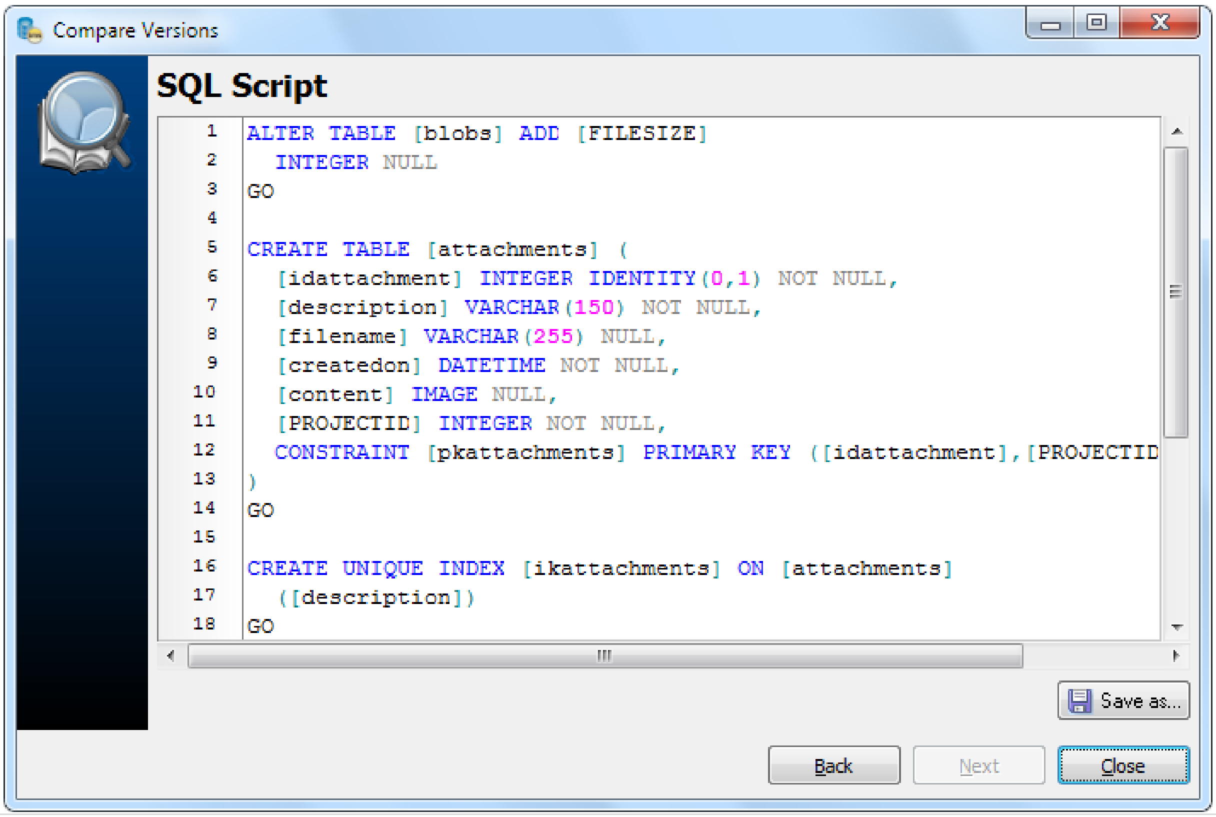
Version 1 (21/08/2010)	Diff	Version 2 (21/08/2010)
Tables	✓	Tables
(not exists)	✓	attachments
blobs	✓	blobs
Fields	✓	Fields
BLOBID		BLOBID
COMPCRC		COMPCRC
COMPSIZE		COMPSIZE
EXTENSION		EXTENSION
FILEDATA		FILEDATA
(not exists)	✓	FILESIZE
ORIGCRC		ORIGCRC
ORIGSIZE		ORIGSIZE
ORIGTIME		ORIGTIME
REVISIONID		REVISIONID

```
 CREATE TABLE [blobs] (
[BLOBID] INTEGER IDENTITY(1,1) NOT NULL,
[REVISIONID] INTEGER NULL,
[EXTENSION] VARCHAR(20) NULL,
[ORIGTIME] DATETIME NULL,
[ORIGSIZE] INTEGER NULL,
[ORIGCRC] INTEGER NULL,
[COMPSIZE] INTEGER NULL,
[COMPCRC] INTEGER NULL,
   [FILEDATA] IMAGE NULL

 )
```

```
 CREATE TABLE [blobs] (
[BLOBID] INTEGER IDENTITY(1,1) NOT NULL,
[REVISIONID] INTEGER NULL,
[EXTENSION] VARCHAR(20) NULL,
[ORIGTIME] DATETIME NULL,
[ORIGSIZE] INTEGER NULL,
[ORIGCRC] INTEGER NULL,
[COMPSIZE] INTEGER NULL,
[COMPCRC] INTEGER NULL,
   [FILEDATA] IMAGE NULL,
   [FILESIZE] INTEGER NULL
 )
```

Clicking on "Generate", we have our script ready to update the database from version 1 to version 2, containing all alterations.



```
ALTER TABLE [blobs] ADD [FILESIZE]
  INTEGER NULL
GO

CREATE TABLE [attachments] (
  [idattachment] INTEGER IDENTITY(0,1) NOT NULL,
  [description] VARCHAR(150) NOT NULL,
  [filename] VARCHAR(255) NULL,
  [createdon] DATETIME NOT NULL,
  [content] IMAGE NULL,
  [PROJECTID] INTEGER NOT NULL,
  CONSTRAINT [pkattachments] PRIMARY KEY ([idattachment],[PROJECTID
)
GO

CREATE UNIQUE INDEX [ikattachments] ON [attachments]
  ([description])
GO
```

Reading the structure of a Data Modeler project from your application

TMS Software offers a free library, Data Modeler Library (DMLib), which allows access to the structure of a database stored in a TMS Data Modeler project, from any application in Delphi or C++Builder. It is a collection of read-only classes containing clear methods and properties for obtaining information about all objects from the data dictionary. Here is a small example of how to use DMLib for getting the fields and their data types from a specific table in the data dictionary:

```
program DMread;
uses
  SysUtils, uAppMetaData, uGDAO;

var
  amd: TAppMetaData;
  table: TGDAOTable;
  field: TGDAOField;
  i: integer;

begin
  amd := TAppMetaData.LoadFromFile('C:\tmssoftware\dmlib\jedivcs.dgp');
  try
    table := amd.DataDictionary.TableByName('attachments');
    if table <> nil then
    begin
      for i := 0 to table.Fields.Count-1 do
      begin
        field := table.Fields[i];
        Writeln(Format('Field %d: %s [%s]',
          [i + 1, field.FieldName, field.DataType.Name]));
      end;
    end
    else
      Writeln('Table not found.');
  finally
    amd.Free;
  end;
end.
```

Output:

```
Field 1: idattachment [Int (identity)]
Field 2: description [VarChar]
Field 3: filename [VarChar]
Field 4: createdon [Datetime]
Field 5: content [Image]
Field 6: PROJECTID [Int]
```

Conclusion

In this article only the main features of TMS Data Modeler have been described, with examples of how to create a project and maintain a database using this tool. TMS Data Modeler offers many other features for modeling and maintaining databases, and comes with complete manual reference and online help. Access our website for details on further application features and benefits. For more information and a free trial download of TMS Data Modeler, visit:
http://www.tmssoftware.com/site/tmsdm.asp

TMS Data Modeler Library is available for free at:
http://www.tmssoftware.com/site/tmsdmlib.asp