

Advanced Data Dictionary Architect

User's Guide

Copyright(c) 2003 Afiq Information Systems and Bino Levi International
Investments Ltd.

Contents

About ADDA	3
System Requirements	3
How It Works. Basics	3
Developer Dictionary	3
End User Dictionary	3
Pool	4
Virtual Data Types	4
Upgrade Step By Step – Developer Dictionary	4
Upgrade Step By Step – End User Dictionary	4
Building Dictionary for an Existing Database	4
User Interface	6
Dictionary Designer	6
Main Menu	7
Toolbar	8
Bird's Eye View	8
Table Update	9
Column Update	10
Index Update	11
Stored Code Update	12
Bird's Eye View	13
Database Synchronizer	14
Dictionary FieldBox Add-In for MS Visual Studio	16
Frequently Asked Questions	17

About ADDA

ADDA (stands for: Advanced Data Dictionary Architect) is a light yet powerful tool for .NET platform which manages data upgrade processes in your applications (currently supported backends: MS SQL Server). Its main goal is easy versionless upgrade functionality (that's right, you don't have to run upgrades from version to version). The upgrade process is so easy it doesn't require a DBA or a sophisticated computer user. Furthermore, the data can be customized by the end user to suit his needs and it will be upgraded accordingly while keeping all the user columns. This kind of functionality is especially valuable for off-the-shelf applications.

In addition, ADDA provides convenient overview of your data structure with advanced filtering abilities similar to those in various CASE tools.

To ease the application development, ADDA provides a special plug-in for Visual Studio .NET, which displays all the dictionary data in a convenient manner and enables placing a column name in the clipboard by clicking on it.

ADDA is registered trademark of Afiq Information System and Bino Levi International Investments Ltd. Visit us at: <http://www.power-components.net>

System Requirements

Microsoft .NET Framework 1.1

Microsoft SQL Server 7 or better (other SQL backends in development)

Optional – Microsoft Visual Studio ©

How It Works. Basics

ADDA uses XML-formatted data to store essential information about the table structure as well as stored procedures, views and triggers. (You must ship this XML file with your application.) The upgrading module checks the supplied dictionary data versus the database and generates differential SQL queries which alter table structure, as needed. If needed, tables are created or even completely rebuilt (in case a column cannot be altered or when explicitly asked by the user). Pieces of stored code (procedures, views, triggers) are re-created to match the code stored in the dictionary.

The dictionary which is used to validate the database contains either one or two parts. The essential part is created by developer; the optional one can be designed by the end user using a special graphical interface tool. This tool doesn't allow the end user to modify the data created by developer.

Developer Dictionary

A file stored in XML. To make it usable to the upgrading module and the end user dictionary designer, it must be called "Dict.XML" and it must exist in the same directory as these applications.

End User Dictionary

A file stored in XML. To make it usable to the upgrading module and the end user dictionary designer, it must be called "UserDict.XML" and it must exist in the same directory as these applications. The end user dictionary design is an optional part. It should be shipped only if you want your user to be able to customize the databases used in your applications.

Pool

To ease creating and maintaining columns of the same structure (such as foreign keys), a so-called "pool" table is used. The pool table stores column templates which can be used for quick creation of similar columns and maintaining them later. For example, one can define a pool column called CustomerID as VARCHAR(15). Instead of entering the same comments and description, every time picking VARCHAR as data type, and 15 as length one can simply select it from the pool combo. It is even more powerful when it gets to maintenance – for example, to change the length of **all** the columns linked to the pool column CustomerID, one simply changes the length of the pool column and selects an option to refresh all the linked columns. No longer one-by-one tedious corrections risking omitting a column.

Pool table is a virtual one and therefore, it is omitted when upgrading data.

Virtual Data Types

So-called virtual data types are created to provide correct conversion from legacy data types. Currently there are the following virtual data types:

- CWDate – date as specified by SoftVelocity Clarion (more information is available at <http://www.softvelocity.com>), that is, a 32-bit integer which contains the number of days passed from December 28, 1800. ADDA can convert CWDate to a datetime of equivalent date and 00:00 time.
- CWTime – time as specified by SoftVelocity Clarion, that is, a 32-bit integer which contains a number of hundredths of a second passed from midnight. ADDA can convert CWTime to a date time of "zero" date and equivalent time.

Upgrade Step By Step – Developer Dictionary

1. Make changes to your dictionary, save it to Dict.XML.
2. Ship Dict.XML to the end user.
3. The end user must run the upgrading module.

Upgrade Step By Step – End User Dictionary

1. The end user makes changes to his/her dictionary and saves it to UserDict.XML in the application directory.
2. The end user runs the upgrading module.

Building Dictionary for an Existing Database

Building a new dictionary from scratch can be challenging. It is obvious that importing existing data structures takes a serious burden off the developer. There are a few modes:

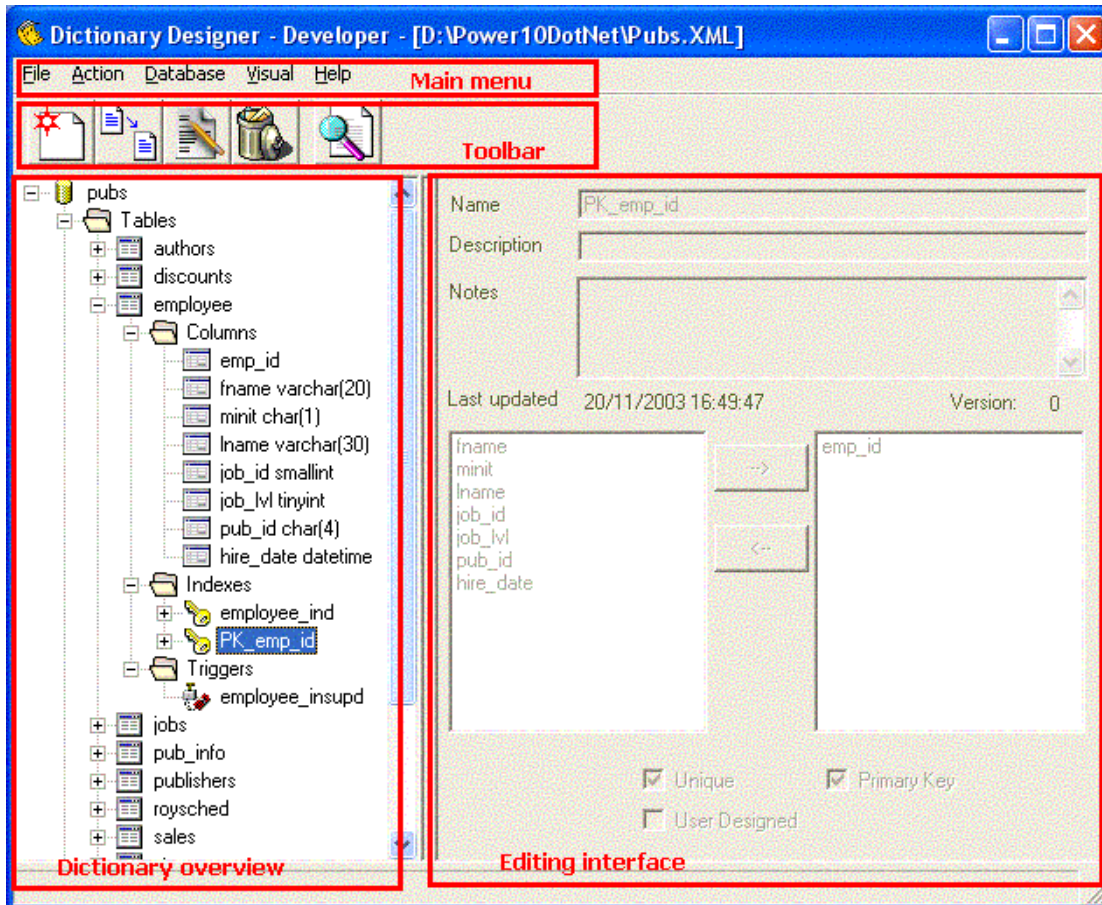
- Importing a complete database
- Differential import of the tables unspecified in the dictionary – available in the end user dictionary designer
- Selective import of specified tables only

In all the modes, data types, triggers, stored procedures, views and when possible column descriptions (`::fn_listextendedproperty` in MS SQL Server, for example) are imported.

User Interface

Dictionary Designer

Here, the dictionary is composed. There is no visual difference between Dictionary Designer for developers and its counterpart which end users use for data customization, however, the end users are unable to alter/delete the entities created in the developers' Dictionary Designer.



As shown on the figure, the window is split into two panes. The left pane contains an overview of the currently edited dictionary. The right pane changes according to the type of entity to present the relevant editing interface. There is a main menu at the top and toolbar buttons below it.

Main Menu

- *File*
 - *New Dictionary* – discard current dictionary and create a new one (not available in the end user version)
 - *Save Dictionary* – save current dictionary to the same file it was loaded from
 - *Save Dictionary As...* - save current dictionary to a custom location
 - *Load Dictionary* – load dictionary from file
 - *Recent Dictionaries* – load one of the recently edited dictionaries
 - *Exit* – leave the application

- *Action*
 - *Insert* – insert a new entity of the currently active type
 - *Duplicate* – insert a new entity copying the contents of the highlighted entity
 - *Edit* – edit highlighted entity
 - *Delete* – delete highlighted entity
 - *Update Columns with This Pool Name* – update data in the columns linked to the highlighted column from pool
 - *Find...* - find an entity matching the defined criteria. Search starts from the root of the tree.
 - *Find Next* – find an entity matching previously defined criteria. If criteria were not defined yet, the finder dialog will be displayed. Search starts from currently selected entity.

- *Database*
 - *Import All from Database* – discard current dictionary and import all the entities from an OLE DB compatible database. Please check "Allow saving password" or "Use Windows NT integrated security" for proper operation.
 - *Selective Table Import* - import a table or tables from an OLE DB compatible database. Current dictionary will not be discarded. Please check "Allow saving password" or "Use Windows NT integrated security" for proper operation.
 - *Generate SQL Creation Script* – generate an SQL creation script for all the entities in the current dictionary

- *Visual*
 - *Bird's Eye View* – display an overview screen for the current dictionary

- *Help*
 - *About...* - displays brief information about the product

Toolbar

The toolbar duplicates the functionality of the Action sub-menu.



As shown in the figure above:

1. Insert a new entity of the currently active type
2. Insert a new entity copying the contents of the highlighted entity
3. Edit highlighted entity
4. Delete highlighted entity
5. Find an entity matching previously defined criteria. If criteria were not defined yet, the finder dialog would be displayed. Search starts from currently selected entity.

Bird's Eye View

As the name implies, this part of the screen displays the dictionary contents organized in a tree. The root node stores global dictionary properties. Children nodes of the root are folders storing tables, pool columns, views and stored procedures (in this order).

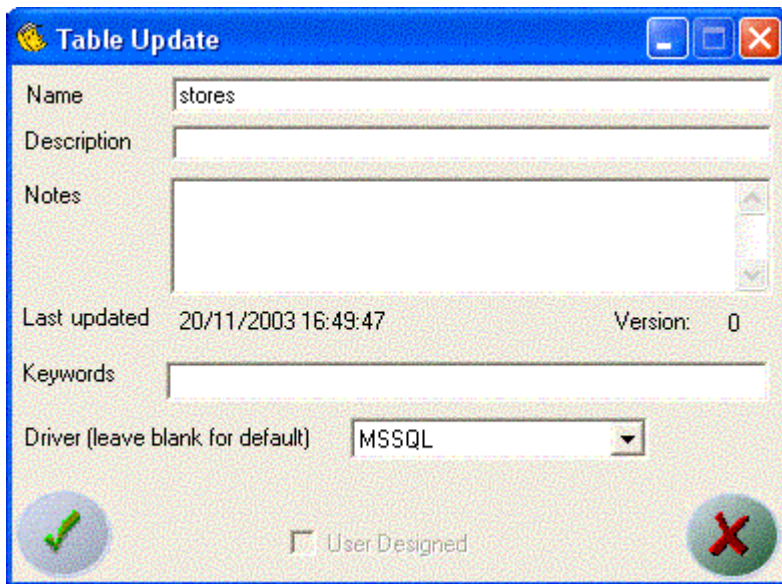
To select an entity, position your mouse cursor over it and click the left button. If you don't have any entities of the type you want to work with, select parent folder instead. Once you've done this, use toolbar buttons, *Action* sub-menu from the main menu or right click for popup menu.

To change the order of columns or indexes in the same table, drag the entity you want to reorder and drop it on the destination entity. You can also copy columns to other tables or pool by dragging and dropping.

There is a number of screens used for updates, depending on the type of the edited entity.

Warning: changing an entity's name is supported only for columns. Otherwise, the system won't recognize the entity with a different name.

Table Update



The screenshot shows a 'Table Update' dialog box with the following fields and values:

- Name: stores
- Description: (empty)
- Notes: (empty text area)
- Last updated: 20/11/2003 16:49:47
- Version: 0
- Keywords: (empty)
- Driver (leave blank for default): MSSQL

At the bottom, there is a green checkmark icon on the left, a checkbox labeled 'User Designed' in the center, and a red X icon on the right.

Contains details of the currently edited table:

- *Name* – the table's name
- *Description* – the table's description
- *Notes* – designer's notes about the table
- *Last updated* – date and time when the details of the table were last updated
- *Version* – every time changes are made to one or more fields and the record is saved, the version is incremented
- *Keywords* – an overview screen enables search by the string of keywords. It is recommended to include some kind of criteria in the string of keywords, for example, all the tables related to customer management, mark with "CUST", all the tables related to stock management, mark with "STOCK", etc.
- *Driver* backend driver of the table. Currently used multiple drivers in the same dictionary is disabled.

Column Update

The screenshot shows the 'Column Update' dialog box with the following fields and controls:

- Name: fname
- Description: (empty)
- Notes: (empty)
- Last updated: 20/11/2003 16:49:47
- Version: 0
- Pool Name: (empty dropdown)
- Datatype: varchar
- Length: 20
- User Designed:
- Identity:
- Old Names List: (empty list)

Contains details of the currently edited column:

- *Name* – the column's name
- *Description* – the column's description
- *Notes* – the designer's notes about the column
- *Last updated* – the date and time when the column details were last updated
- *Version* – every time changes are made and the record is saved, the version is incremented
- *Pool name* – reference to a column from the pool table (optional). If selected, data type and length (if needed) are updated according to the pool column data. Description and notes also are updated if they are empty.
- *Datatype* – backend data type
- *Length* – maximal length, if the data type must be specified with length (such as strings of various types)
- *User Designed* – whether the column is designed by the end-user
- *Identity* – whether the column is an identity (auto-incremented) column
- *Old Names list* – if the column name was changed, ADDA keeps the old name to convert the legacy data. The legacy data type is stored as well. It may be changed by the user, for example, to provide proper conversion for virtual data types. When the right mouse button is clicked over the list, a context menu appears:
 - *Edit* – edit existing old name or force a different data type. Currently, there are two virtual types added to the existing backend data types: "cwdate" and "cwtime".
 - *Delete* – delete existing old name. Useful when making a few changes on the same session.

Index Update

The screenshot shows the 'Index Form' dialog box. The 'Name' field is filled with 'UPK_storeid'. The 'Last updated' field shows '20/11/2003 16:49:47' and the 'Version' field shows '0'. The 'Component Selector' has two lists: the left list contains 'stor_name', 'stor_address', 'city', 'state', and 'zip'; the right list contains 'stor_id'. Below the lists are two arrows: a right-pointing arrow (--) and a left-pointing arrow (<--). At the bottom, there are three checkboxes: 'Unique' (checked), 'Primary Key' (checked), and 'User Designed' (unchecked). A green checkmark icon is on the left and a red X icon is on the right.

Contains details of the currently edited index:

- *Name* – the index's name
- *Description* – the index 's description
- *Notes* – designer's notes about the index
- *Last updated* – date and time when the index details were last updated
- *Version* – every time changes are made and the record is saved, the version is incremented
- *User Designed* – whether the index was designed by the end user
- **Component selector** – contains two lists of the columns from the selected table. Selected components are listed in the right list. Use left arrow to remove components, right arrow to add components.
- *Unique* – whether the index is unique
- *Primary Key* – whether the index is a primary key. Only one primary key can be specified per table.

Stored Code Update

Stored code update interface handles update of triggers, views and stored procedures.

Stored Code Update Form

Name: employee_insupd

Description:

Notes:

Last updated: 20/11/2003 16:49:47

Version: 0

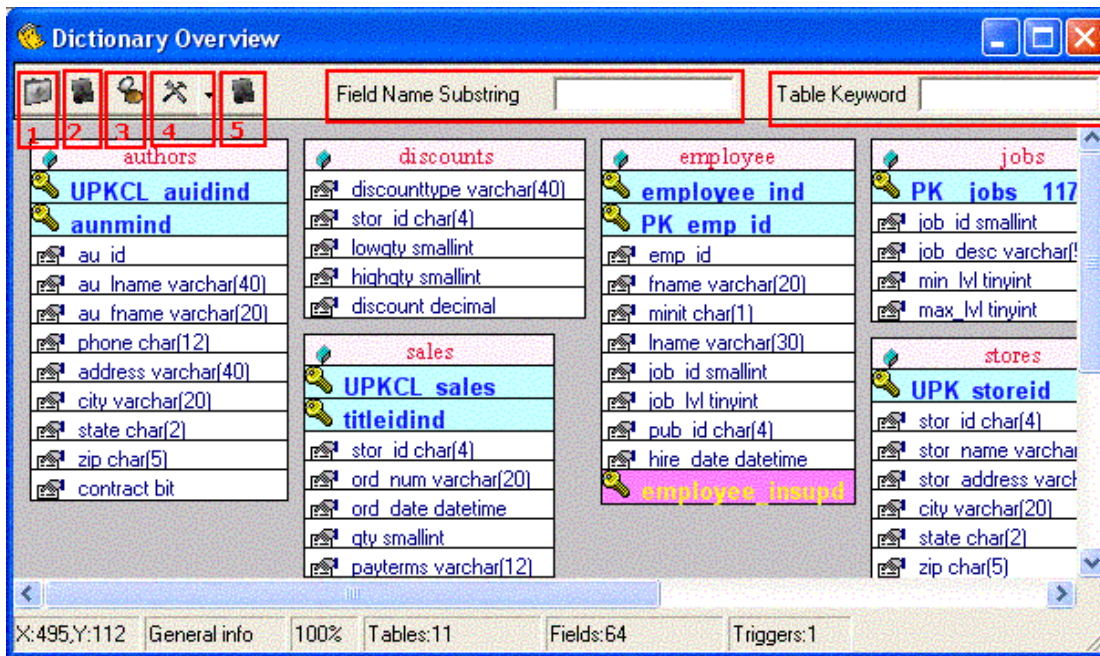
```
CREATE TRIGGER employee_insupd
ON employee
FOR insert, UPDATE
AS
--Get the range of level for this job type from the jobs table.
declare @min_lvl tinyint,
        @max_lvl tinyint,
        @emp_lvl tinyint,
        @job_id smallint
select @min_lvl = min_lvl,
       @max_lvl = max_lvl,
       @emp_lvl = i.job_lvl,
       @job_id = i.job_id
```

User Designed

Code

- *Name* – stored code entity's name
- *Description* – stored code entity's description
- *Notes* – designer's notes about the stored code entity
- *Last updated* – date and time when the stored code entity details were last updated
- *Version* – every time changes are made and the record is saved, the version is incremented
- *Code* – holds the entity's code, or, being more specific, proper CREATE statement.

Bird's Eye View



This screen displays filter-enabled overview of the currently edited dictionary.

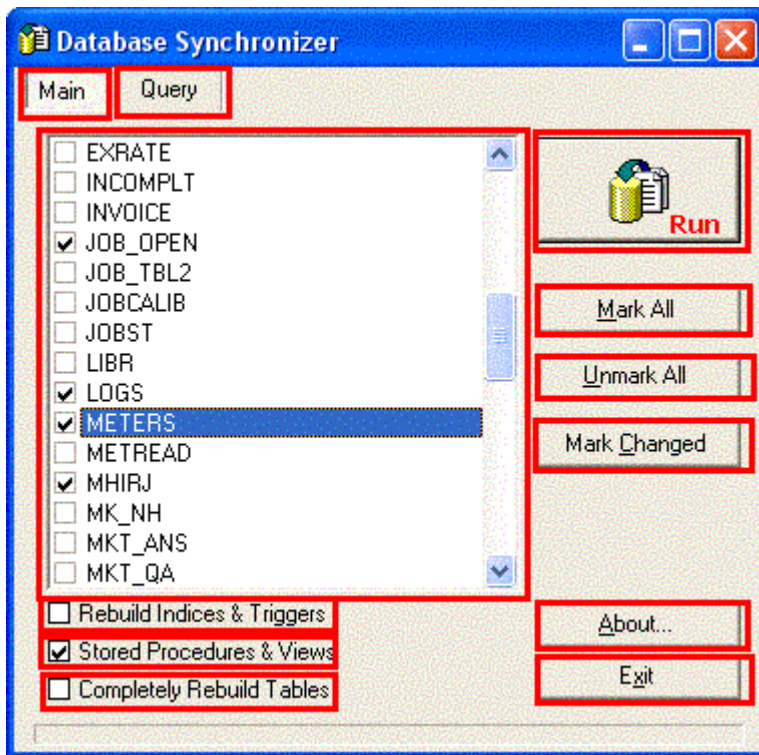
Controls:

- *Field name substring* – a substring of the field name
- *Table keyword* – a substring of the tables' "keyword" field
- *1 (Zoom in)* – zoom overview in
- *2 (Zoom out)* – zoom overview out
- *3 (Is filter active)* – turn filtering by name substring / table keyword on/off
- *4 (Select zoom)* – select overview zoom
- *5 (Print active view)* – print currently displayed overview

Database Synchronizer

Database Synchronizer restructures the target database to fit the structure specified in the dictionary. It does so by executing differential SQL scripts generated on-the-fly by comparison between the dictionary and the database.

When an end user first loads Database Synchronizer, a dialog appears prompting the user to specify the connection details. Once selected, the connection details are saved in POWER10.CFG file in the run directory.



Tables from the dictionary are listed. Checkbox near every table defines whether the table is to be included in the upgrading process. If it isn't checked, then, no script will be generated for it, even if the structures don't match. Normally, if the structures match, then no script generated; otherwise, ALTER TABLE statements create/rename columns whenever needed, or CREATE TABLE is run to build a new table. The *Query* tab displays the query which will be executed. The end user can copy it to the clipboard if needed.

Controls:

- *Rebuild Indices & Triggers* – re-creates all indexes and triggers for selected table even if the structures are up-to-date
- *Stored Procedures & Views* – re-creates stored procedures & views from the dictionary
- *Completely Rebuild Tables* – re-creates all tables (preserving the data, of course). The physical ordering of columns in the table makes little to no difference, but if the user wants to adhere to standards, this is the way to reorder the columns according to the dictionary.
- *Mark All* – mark all tables for upgrade (default)
- *Unmark All* – clear mark from all tables
- *Mark Changed* – mark only those tables in which the structure is not up-to-date
- *About* – display brief information about the product
- *Exit* – exit the application without performing the upgrade
- *Run* – perform the upgrade and exit

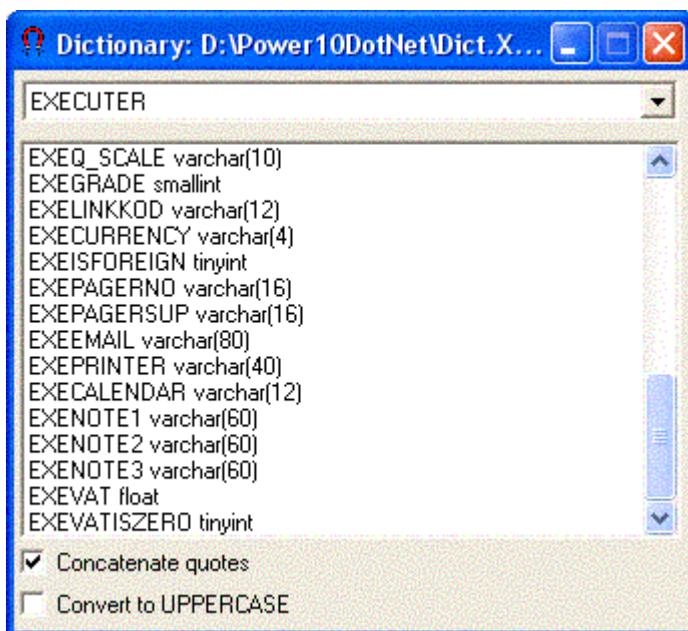
- *Query* – view the query to be run
- *Main* – return to the main tab page

Dictionary FieldBox Add-In for MS Visual Studio

Digging up field names can be an annoying task, especially if the dictionary is vast and the code is database-intensive, such as business rules. The Server Explorer included in MS Visual Studio provides online information only for existing databases. Dictionary FieldBox enables a quick peek at the dictionary contents and handy transfer of the field name to the clipboard by double-clicking the field in the list.

To take advantage of this add-in, simply copy the dictionary you've created to the solution root directory naming it DICT.XML. When the solution opens, the floating FieldBox window will open. When closing the solution, the window will also be closed.

To reactivate the window after closing it while still in the same solution, select Tools -> Add-in Manager from the Visual Studio menu, then make sure Dictionary FieldBox is checked for use and press OK. A message box will ask whether you want to reactivate the FieldBox. Answer Yes.



Frequently Asked Questions

Q: There are numerous packages like ADDA which offer automatic upgrades, why should I use yours?

A: Our database upgrades are versionless. To go, for example, from version 1 to 10, usually long scripts must be run or manual selection is involved. A simple unsophisticated user can have a difficult time accomplishing this. ADDA analyzes the user's database structure and makes essential changes running differential queries. Re-distribution is easy and simple; no writing to registry or other Windows-specific operations are involved so you can include it in your setup or patching utility. Another distinctive advantage is the full support of end-user's database customization (enterprise version).

Q: How do I start working with ADDA?

A: After you install the package (which I believe you already did), open Dictionary Designer for the developer. If you have a database to start from, select Database->Import All from Database from the main menu. Please check "Allow saving password" or "Use Windows NT integrated security" for proper operation. Edit the dictionary as needed, then save it under name DICT.XML. Distribute it to your users with the Database Synchronizer package. When making changes to DICT.XML, re-distribute it to the users along with the updated version (simply overwrite the old copy).

Q: What do I need to redistribute to the end user?

A: Database Synchronizer and, optionally, Dictionary Designer for end users if you want to allow customization of the users' databases. There is a sub-directory named *Redistribute* under the root folder of the application. It contains a Setup utility and the same pack of files as they are under sub-directory named *Raw*, in case you want to distribute them in your package. In the enterprise edition of ADDA there are two options: to supply Database Synchronizer only or both Database Synchronizer and Dictionary Designer for end users. The provided installation utility contains a selection screen.

Q: I cannot connect to my database! I get exceptions saying the system is unable to logon.

A: When defining connection in the Data Link dialogue, please check "Allow saving password" or "Use Windows NT integrated security". There is a problem resolving this for now.

Q: I closed Dictionary FieldBox by clicking on X button. How do I reactivate it?

A: Simple. Select Tools -> Add-in Manager from the Visual Studio menu; make sure Dictionary FieldBox is checked for use and press OK. A message box will ask you whether you want to reactivate the FieldBox. Answer Yes.

Q: What ADDA stands for?

A: ADDA stands for Advanced Data Dictionary Architect.

Q: I develop in Delphi / C# Builder / other non-MS .NET tool. Can I use ADDA?

A: Yes you can. However, you won't be able to use FieldBox add-in for MS Visual Studio ©.

Q: ADDA sounds great, but I use Oracle / MySQL / PostgreSQL / whatever. Do you support this backend?

A: Not currently, however, it certainly is within the scope of our design. That's why we have this driver selection combo. Once we implement it, you'll be able to switch to it on-the-fly with the automatic redefinition of all the field types. However, cross-database automatic upgrades are questionable.

Q: I liked your length entry control. Where does it come from?

A: I'm so glad you've asked☺. It is our homebrew masked edit control. It enables convenient fool-proof data entry of numbers, patterns, dates, time and forcing input language and configures itself according to the selected mode. Except for Entry, we also have a bulk lookup control, data grid with smart paging mechanism loading small portions of the table, automatic update form, and lots of other possibilities. Be advised that we are going to release it all in a separate package.