



AnyDAC

Getting Started guide

Version 1.4.0

1	Introduction	4
1.1	Overview	4
1.2	Architecture.....	4
1.3	Supported tools	4
1.4	Supported RDBMS's.....	5
1.5	Installation	5
1.6	Roadmap	6
2	Base software.....	7
2.1	Registry entries.....	7
2.2	3d party dependencies.....	7
2.3	Packages	7
2.3.1	Overview.....	7
2.3.2	Installing package binaries into IDE.....	8
2.4	Components.....	9
2.4.1	Main components	9
2.4.2	User interface components	10
2.4.3	Link components	10
2.5	Utilities	10
2.5.1	Administrator	10
2.5.2	Explorer.....	12
2.5.3	Monitor	13
2.5.4	dfmChanger	13
2.5.5	Pump.....	14
2.5.6	Executor.....	15
2.5.7	Unit Tester	16
2.5.8	Speed Tester.....	17
3	Demos.....	19
3.1	Overview	19
3.2	Demo DB	19
3.2.1	Preparing to Demo DB installation	19
3.2.2	Setting up connection definitions	20
3.2.3	Installing Demo DB objects	21
3.3	Running demo applications	21
3.4	Cross RDBMS demos	21
4	Programming	24
4.1	Application used units.....	24
4.2	Go !	25
5	Migrating.....	28
5.1	Overview	28
5.2	AnyDAC analogues of BDE components	28
5.3	BDE aliases migration	29
5.4	Example.....	30

5.5	Additional migration hints	32
6	Where to go further	34
6.1	xRDBMS features.....	34
6.1.1	Data type mapping	34
6.1.2	Macro processor and escape sequences	34
6.2	Deeper into AnyDAC.....	35

1 Introduction

1.1 Overview

AnyDAC is a multy DBMS data access framework for Borland Delphi. The framework name is abbreviation for Any Data Access Components (AnyDAC). In the beginning, it aims to replace BDE data access components in existing Delphi applications with nearly zero effort.

At the moment Borland supplies with their products (Delphi, BCB) the following multy DBMS data access components (hereinafter referred as DAC):

- **BDE:** Since May 2003 BDE and its native SQLLinks have been deprecated and there will be no further development. As result newer RDBMS versions are not supported.
- **ADOExpress.** This library relies on MS ADO and is rather flexible. However it misses many modern DAC features and has not an acceptable performance with some RDBMS (for example, Oracle). As well it does not allow the programmer to use specific database features.
- **DbExpress.** The library is still in an immature state and has many serious limitations and drawbacks.

So, we decided to develop AnyDAC... Our analysis discovered the following set of requirements to DAC:

- **High stability.** Should be able to operate day and night.
- **High performance.** From this point of view, DAC should be “invisible” to the programmers. Programmers will get the ability to implement additional functionality with the same performance.
- **Easy deployment.** DAC should be able to link all the components directly into applications. For some application it is nice feature to have external (DLL or packaged) drivers.
- **Cross RDBMS development.** Many commercial applications will get benefit, if it supports several RDBMS. Usage of multiple DAC's in single application is not an acceptable solution. However a cross RDBMS DAC should also provide programmers with the ability to use maximum RDBMS features in some common ways.
- **Interface compatibility with classic DAC,** such as BDE and DataSnap. It also gets an excellent learning curve and starts to program with new DAC in few minutes after the installation. And most important – migrate existing BDE application with nearly zero effort.

1.2 Architecture

All that has driven AnyDAC architecture to it current state. Also many classic DAC development approaches were reviewed under .Net influence. We are not going to state that ADO.Net is the best DAC, but the mixture of the best ideas from Delphi, .Net worlds and our experience allowed to get this data access framework. The detailed description you can find in [AnyDAC Architecture Guide](#).

1.3 Supported tools

AnyDAC was tested with following compiler tools:

Name	Comment
Borland Delphi 5	This Delphi version has no dbExpress support.
Borland Delphi 6	This Delphi version has dbExpress v 1.0. It has a lot of bugs. So, if you plan to use AnyDAC dbExpress driver, better migrate to Delphi 7. AnyDAC requires Update Pack 2 to be installed.
Borland Delphi 7	

Borland Delphi 2005	Only Win32.
Borland Delphi 2006	Only Win32. dbExpress is not supported. AnyDAC is not really tested with Delphi 2006, because I does not have Delphi 2006.
Borland C++ Builder 5	See Borland Delphi 5.
Borland C++ Builder 6	See Borland Delphi 6.

1.4 Supported RDBMS's

AnyDAC basically should support all RDBMS brands and versions using dbExpress and / or ODBC drivers. But, of course, not all brands and versions have been tested. But it is noted, the best is to use the maximum version of a RDBMS tested with AnyDAC. Maximum, because this version has fixed previous bugs and has extended functionalities (I do not pretend for %100 True, but at our tests is was so).

Possible drivers to use under AnyDAC are "AnyDAC native driver", "AnyDAC dbExpress driver" or "AnyDAC ODBC driver". Through the "AnyDAC dbExpress driver" you can work with any RDBMS, which the dbExpress driver supports. The similar does apply to "AnyDAC ODBC driver". Recommended driver to use is always "AnyDAC native driver". Because it is fine-tuned for target RDBMS. AnyDAC was tested with following RDBMS:

Name	Server and client version	Driver implementation unit	DriverID parameter
MySQL	Server and client 3.21.xx	daADPhysMySQL	MySQL
	Server and client 3.23.57	daADPhysMySQL	MySQL
	Server and client 4.0.13	daADPhysMySQL	MySQL
	Server and client 4.1.0	daADPhysMySQL	MySQL
	Server and client 5.0.1	daADPhysMySQL	MySQL
	Server and client 5.0.6	daADPhysMySQL	MySQL
Oracle	Server and client 8.0.4.3.7	daADPhysOracl	Ora
	Server and client 8.1.6.0.0	daADPhysOracl	Ora
	Server and client 9.2.0.1.0	daADPhysOracl	Ora
MSSQL	Server 2000 (8.00.194) and ODBC driver v 2000.85.1022.00	daADPhysMSSQL	MSSQL2000
	Server 2005 (9.00.1116) and SQL Native Client ODBC driver v 2000.90.1116.00	daADPhysMSSQL	MSSQL2005
MSAccess 2000	Jet 4.0 and ODBC driver v 4.00.6019	daADPhysMSAcc	MSAcc
IBM DB2 UDB	Server and client 8.1	daADPhysDb2	DB2
Sybase ASA	Server and client v 8.0.1	daADPhysASA	ASA

To work with the listed RDBMS, specified client software must be installed. AnyDAC driver has to be linked into application [see chapter 4.1] or install as an existing Delphi package.

1.5 Installation

AnyDAC installer performs the installation of AnyDAC design time packages, tool binaries and Demo DB automatically.

If you choose do not build AnyDAC binaries, then you can install **daADDclXXX** design time package into IDE by your own. For details see Chapter 2.

If you choose do not build AnyDAC Demo DB, then you can install Demo DB objects later by your own. Before this step will be performed, you need to create RDBMS accounts, where Demo DB objects will be installed. For details see Chapter 0.

1.6 Roadmap

- New native drivers:
 - **Interbase and Firebird**
- Delphi.Net port
- Existing drivers support of latest RDBMS version features:
 - **MySQL 5**
 - **Oracle 10**
 - **MSSQL 2005**
- Extending documentation
- Create help file

To be continued ...

2 Base software

2.1 Registry entries

The following registry values have to be added for successful AnyDAC design time usage to Windows Registry under the key HKLM\Software\da-soft\AnyDAC. Actually, installer does that. Note, AnyDAC does not need these registry entries at run time.

Name	Description	Value example
ADHome	Points to AnyDAC installation directory.	C:\Program Files\da-soft\AnyDAC
ConnectionDefFile	Points to system wide AnyDAC connection definition file.	\$(ADHOME)\DB\ADConnectionDefs.ini
DriverFile	Points to system wide AnyDAC driver definition file. It is optional.	\$(ADHOME)\DB\ADDrivers.ini
ExecutorPath	Points to AnyDAC SQL Executor executable file.	\$(ADHOME)\TOOL\Executor\ADEXecutor.exe
ExplorerPath	Points to AnyDAC SQL Explorer executable file.	\$(ADHOME)\TOOL\Explorer\ADExplorer.exe
MonitorPath	Points to AnyDAC SQL Monitor executable file.	\$(ADHOME)\TOOL\Monitor\ADMonitor.exe

2.2 3d party dependencies

AnyDAC uses Indy (Internet Direct - <http://www.indyproject.org>) library installed in the development tool. Supported versions are 8.x, 9.x and 10.x.

At installation time the file \$(ADHOME)\Source\daADIndy.inc is generated. It defines which Indy version to use with particular development tool. If later you will install other Indy version, than you need regenerate it. Go to \$(ADHOME)\Build directory and run command file updateIndy.bat.

To use specific Indy version, edit \$(ADHOME)\SOURCE\daAD.inc file by hands.

```
// Indy
{$I daADIndy.inc}
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// If you are using specific Indy version $DEFINE one of these names and $UNDEF others.
// Or regenerate daADIndy.inc file. For that go to $(ADHOME)\Build directory and run
// command file updateIndy.bat
// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
{$define AnyDAC_MONITOR}
{$undef AnyDAC_INDY8}
{$undef AnyDAC_INDY9}
{$undef AnyDAC_INDY10}
```

Find code showed above. Then "\$DEFINE" required version and remove dot before others "\$UNDEF". Note, only daADStanXXX.dpk package depends on Indy. Due to Indy package naming issues, you may get errors "Required package IndyXXXX does not found". In this case you will need to adjust Indy package name by hands.

2.3 Packages

2.3.1 Overview

All AnyDAC packages are collected together into a Delphi project group – daAD[tool version].bpg. The packages are located in the '\$(ADHOME)\Pack' directory. It is convenient to open the project group, instead of the separate packages. Current version includes the following packages:

Name	Description
daADComl[tool version].dpk	Coml layer, including all interface definition units, parameters, error handling,

	options and DatS layer.
daADStan[tool version].dpk	The Stan layer is a set of common algorithms, routines and constants
daADPhys[tool version].dpk	The Phys layer defines interfaces for physical data access
daADDApt[tool version].dpk	The DApt layer allows automation and fine-tuning of <i>read operation</i> with complex result sets (master-details, nested, ADT, etc) and allows <i>posting back updates</i> to the database system.
daADComp [tool version].dpk	The Comp layer represents the AnyDAC public interfaces as Delphi non-visual components.
daADGUIxForms[tool version].dpk	Delphi forms implementation of GUIx layer.
daADDcl [tool version].dpk	AnyDAC design time package.
daADPhys[driver kind] [tool version].dpk	AnyDAC RDBMS drivers. There are few packages – one for each driver. See later.

The development *tool version* has to be one of the following:

Name	Description
D5	Delphi 5
D6	Delphi 6
D7	Delphi 7
D9	Delphi 2005 (Win32)
D10	Delphi 2006 (Win32)
BCB5	C++Builder 5
BCB6	C++Builder 6

The *driver kind* has to be one of the following:

Name	Description
DbExp	AnyDAC dbExpress driver. This driver uses dbExpress drivers to communicate with RDBMS. All dbExpress versions are supported.
ODBC	AnyDAC ODBC driver. This driver uses ODBC drivers to communicate with RDBMS. ODBC driver should conform minimum ODBC v2 specification.
MSAcc	AnyDAC Microsoft Access driver. It uses MS JET ODBC driver.
MSSQL	AnyDAC Microsoft SQL Server driver. It uses MSSQL ODBC driver or Native Client driver.
DB2	AnyDAC IBM DB2 driver. It uses DB2 CLI/ODBC driver.
ASA	AnyDAC Sybase ASA driver. It uses ASA ODBC driver.
MySQL	AnyDAC native MySQL driver. The driver supports all MySQL client versions, starting from 3.23.
Oracle	AnyDAC native Oracle driver. The driver supports all Oracle client versions, starting from Oracle 8.0.3

2.3.2 Installing package binaries into IDE

AnyDAC installer will do following steps for you automatically. If you need to rebuild the AnyDAC packages and install design time one by hands, perform following steps:

1. Run Delphi IDE.
2. Delphi prior 2005
 - 2.1. Choose “Tools” → “Environment Options” → “Library” → “Library Path“. Then press the “...” button on the right of the edit box and add path to SOURCE subdirectory of the AnyDAC installation directory. Then press the OK button.
 - 2.2. Choose “File” → “Open“. Set “Files of type” to “Delphi project (*.dpr; *.bpg)“. Then open the file daAD[tool version].bpg in the PACK subdirectory of the AnyDAC installation directory.

IMPORTANT ! You will get message “This package appears to be in an older format. Would you like to convert it to the new format?” There you must answer “No”. The simplest way is to press Escape button and keep it pressed until BPG will be loaded.

- 2.3. Choose “Project” -> “Compile All Projects”. After that AnyDAC packages will be build.
- 2.4. If all passed successful, right click ‘daADDcl[tool version].bpl’ and select “Install”.
3. Delphi 2005 and later
 - 3.1. Choose “Tools” → “Options” → “Library – Win32” → “Library Path”. Then press the “...” button on the right of the edit box and add path to SOURCE subdirectory of the AnyDAC installation directory. Then press the OK button.
 - 3.2. Choose “File” → “Open”. Set “Files of type” to “BDS project group (*.bdsgroup)”. Then open the file daADD9.bdsgroup in the PACK subdirectory of the AnyDAC installation directory.
 - 3.3. Choose “Project” -> “Compile All Projects”. After that AnyDAC packages will be build.
 - 3.4. If all passed successful, right click ‘daADDclD9.bpl’ and select “Install”.
4. You have finished!

2.4 Components







2.4.1 Main components

The “AnyDAC” component palette page contains following non-visual components:

Name	Description
 TADManager	TADManager provides global management of a group of database connections in a AnyDAC application. TADManager (AnyDAC) is similar to TSession (BDE).
 TADConnection	TADConnection provides discrete control over a connection to a single database in a AnyDAC application. The component encapsulates IADPhysConnection interface. TADConnection (AnyDAC) is similar to the TDatabase (BDE).
 TADCommand	TADCommand Command allows execute SQL statement and fetch result set into TADDatSTable. The component encapsulates IADPhysCommand interface.
 TADTableAdaptor	
 TADSchemaAdaptor	
 TADQuery	TADQuery represents a dataset with a result set that is based on an SQL statement. TADQuery (AnyDAC) is similar to TQuery (BDE).
 TADStoredProc	TADStoredProc encapsulates a stored procedure on a database server in a AnyDAC application. TADStoredProc (AnyDAC) is similar to TStoredProc (BDE).
 TADMetaInfoQuery	TADMetaInfoQuery is a dataset, which represents a set of results based on a query of RDBMS meta information. The meta information includes: list of tables, list of columns in a table, list of indexes in a table, etc.
 TADUpdateSQL	TADUpdateSQL extends standard AnyDAC capabilities to post changes from TADQuery or TADStoredProc to RDBMS. The component allows define SQL queries that will be used at Update, Insert or Delete operations. TADUpdateSQL (AnyDAC) is similar to TUpdateSQL (BDE).
 TADDataMove	TADDataMove performs database operations on groups of records. First it can be used to move record set from one data set to another AnyDAC data set. Or to load / unload files with ASCII data. TADDataMove (AnyDAC) is similar to TBatchMove (BDE).
 TADTable	TADTable encapsulates a database table. The component was created to achieve compatibility with BDE DAC. Strongly recommended to use TADQuery instead of TADTable. TADTable (AnyDAC) is similar to TTable (BDE).
 TADClientDataSet	TADClientDataSet represents an in-memory dataset. The component may be used together with AnyDAC adapter components to fetch data or standalone as true in-memory dataset. TADClientDataSet is similar to Borland TClientDataSet, but is not a direct analogue.

2.4.2 User interface components

The “AnyDAC UI” component palette page contains following components:

Name	Description
 TADGUIxFormsQBldrEngine	The component controls the interface of SQL query builder to RDBMS.
 TADGUIxFormsQBldrDialog	The component controls the visual SQL query builder.
 TADGUIxFormsErrorDialog	The component controls the AnyDAC specific error dialog. If application does not use this component, then standard VCL dialog will be used to handle AnyDAC exceptions.
 TADGUIxFormsLoginDialog	The component controls the AnyDAC specific login dialog. If application does not use this component, then user will be not prompted for credential.
 TADGUIxFormsAsyncExecuteDialog	The component controls the AnyDAC asynchronous command execution dialog. If application does not use this component, then there will be no user interaction during asynchronous command execution.
 TADGUIxWaitCursor	The component controls mouse cursor during long running operations.

2.4.3 Link components

The “AnyDAC Links” component palette page contains components linking optional functionality with application. In general there are 2 component kinds:

- TADMoniXXXXClientLink – tracing and monitoring functionality.
- TADPhysXXXXDriverLink – DBMS drivers.

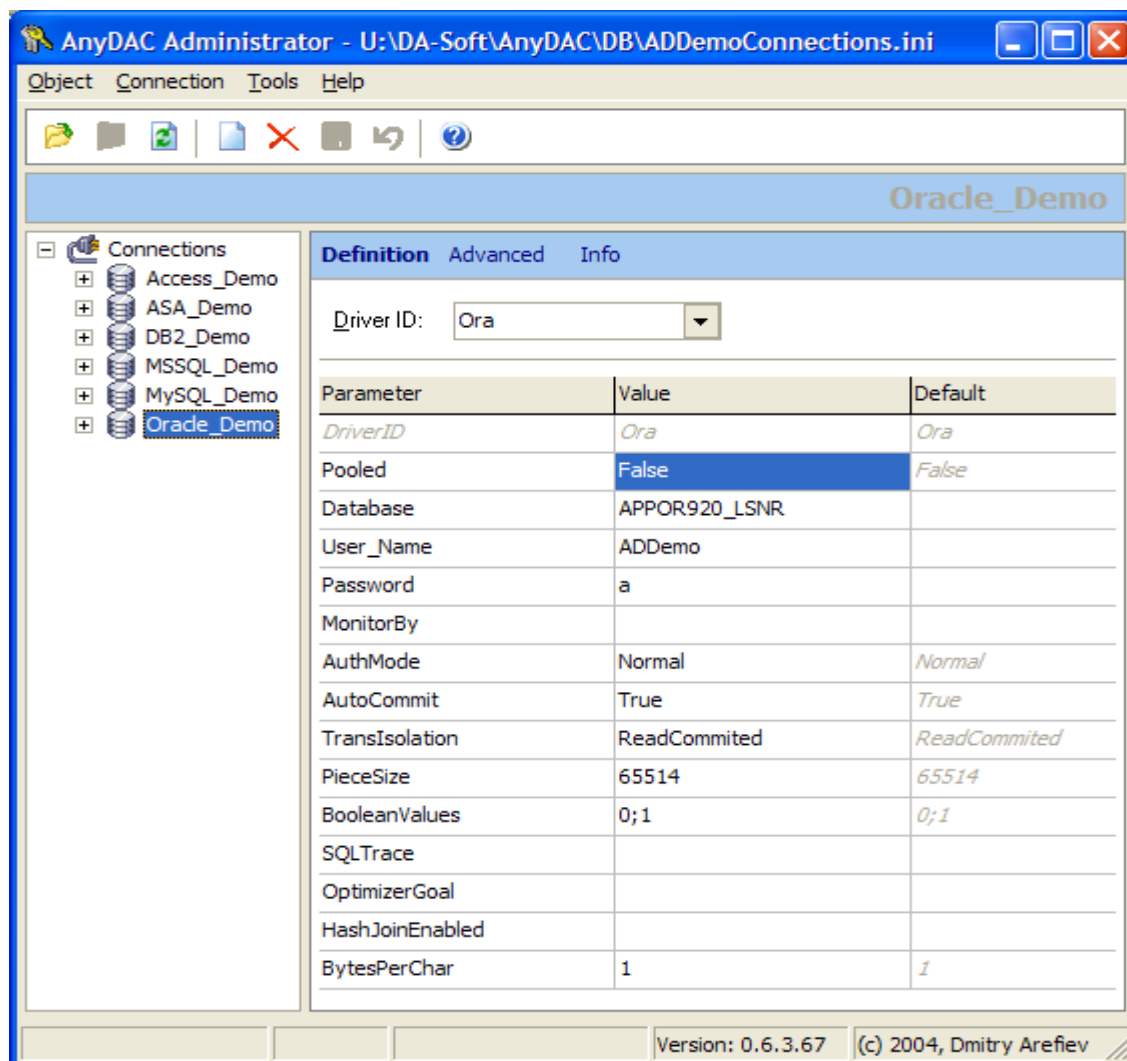
2.5 Utilities

AnyDAC is provided with following set of utilities, located in specified folders:

What	Where
AnyDAC Administrator	\$(ADHOME)\Tool\Administrator
dfmChanger	\$(ADHOME)\Tool\dfmChanger
AnyDAC SQL Script Executor	\$(ADHOME)\Tool\Executor
AnyDAC Explorer	\$(ADHOME)\Tool\Explorer
AnyDAC Debug Monitor	\$(ADHOME)\Tool\Monitor
AnyDAC ASCII Data Pump	\$(ADHOME)\Tool\Pump
AnyDAC Unit Tester	\$(ADHOME)\Tool\QA
AnyDAC Speed Tester	\$(ADHOME)\Tool\Speed

2.5.1 Administrator

The AnyDAC Administrator is the major tool to manage the AnyDAC connections.



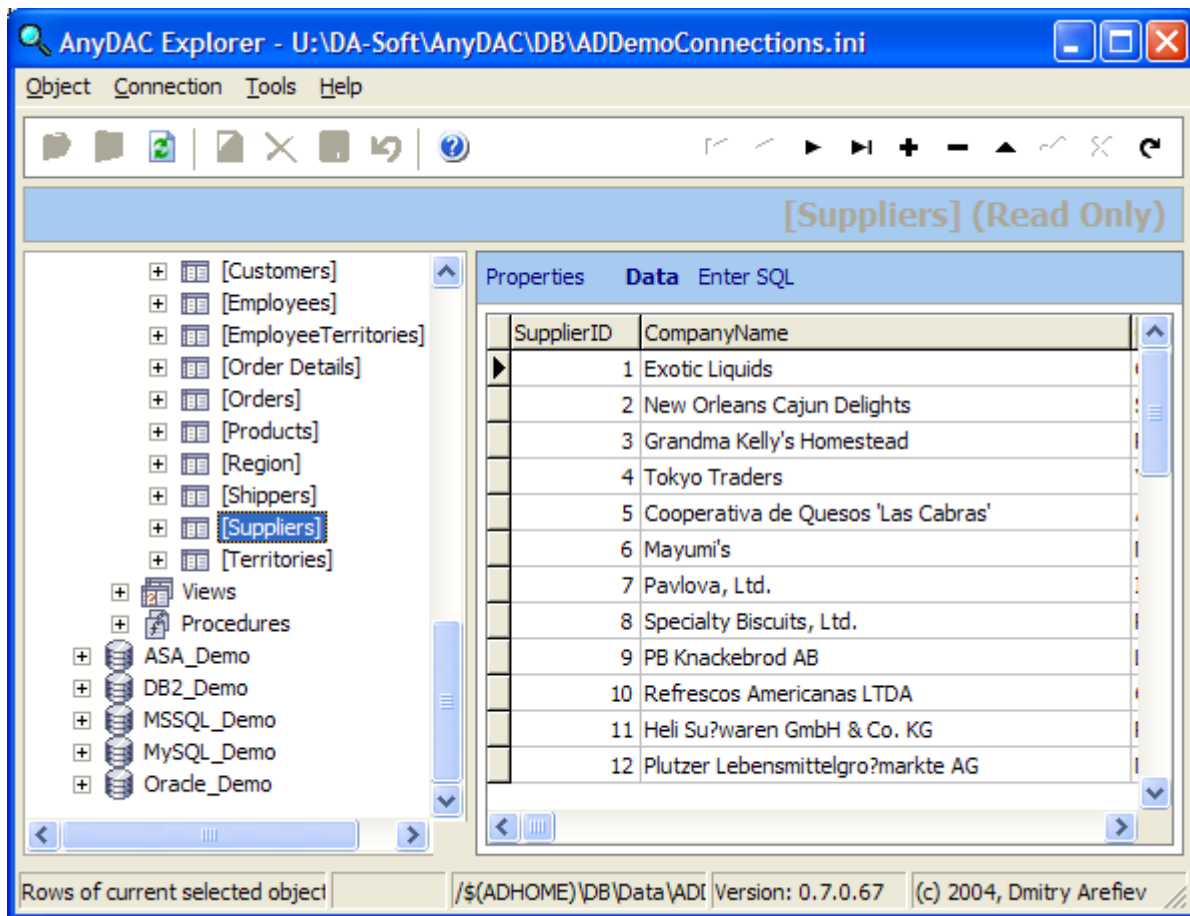
AnyDAC Administrator is graphical tool, analogue to the BDE Administrator. And it is aimed to manage AnyDAC connection definitions.

The connection definition is a named set of parameters defining RDBMS kind to connect, database address, session main property initial values, etc. The set of definitions is stored in the connection definition file. One such file may be marked as "AnyDAC default file".

The "Connections" node allows choose connection definition file to edit in Administrator and optionally to mark it as default file. All other nodes represent connection definitions in the selected file. To edit node, it must be "closed". If it is not, then select node and press "Close" button or menu item.

For more information on connection definitions see **AnyDAC Architecture Guide**. For more information on Administrator see **AnyDAC Explorer Guide**.

2.5.2 Explorer

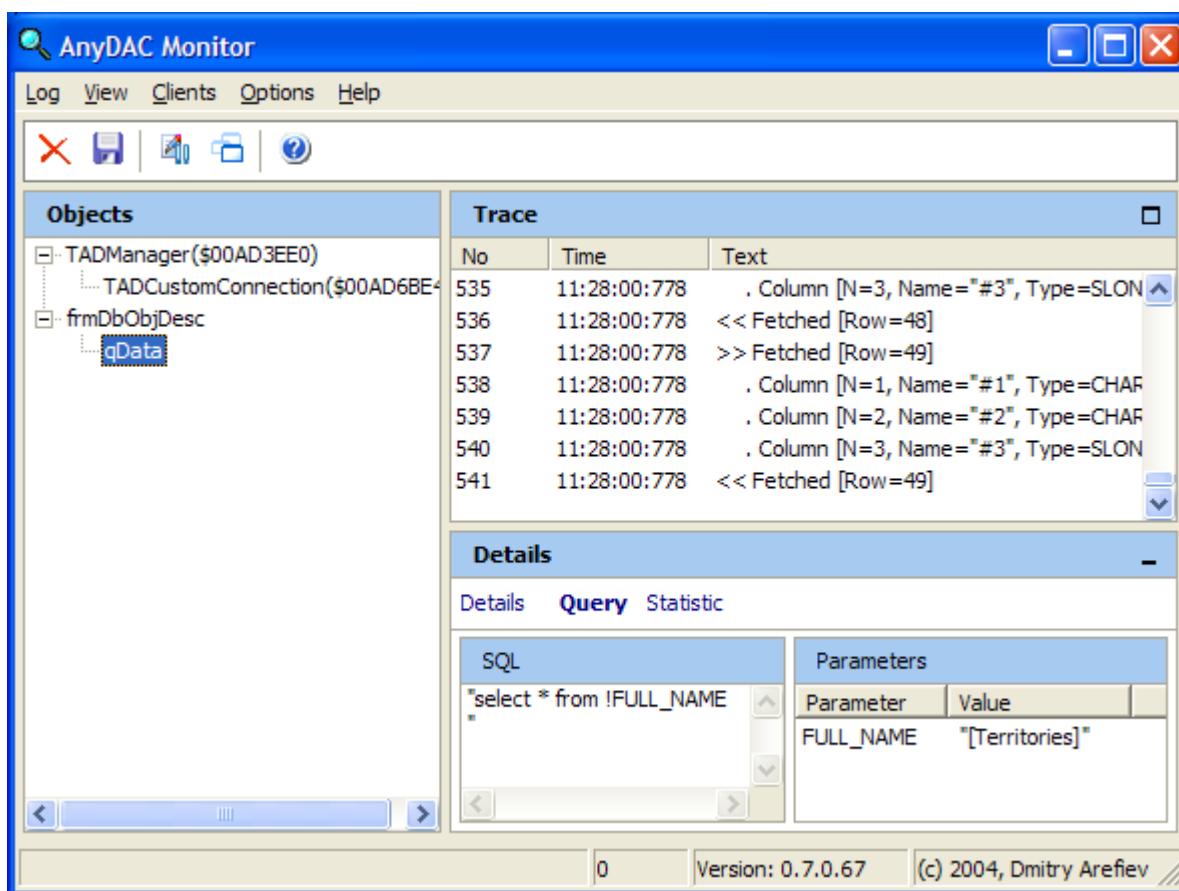


The “AnyDAC Explorer” is a hierarchical database browser with data editing capabilities. Through a persistent connection to a database, the “AnyDAC Explorer” enables you to:

- Browse database server-specific schema objects, including tables, fields, stored procedure definitions, triggers, and indexes.
- Create, view, and edit data in existing tables.
- Create and maintain connection definitions.
- Enter SQL statements to query a database.

“AnyDAC Explorer” is analogue to the “BDE Database Explorer”. For more information on Explorer see ***AnyDAC Explorer Guide***.

2.5.3 Monitor



“AnyDAC Monitor” is a debugging tool for:

- Tracing the communication between an AnyDAC application and the RDBMS.
- Exploring details of commands, parameters, states, execution statistic, etc.

AnyDAC remote monitoring uses TCP/IP as a transport for application output. That means, Monitor may be running as on the same as on the other computer than the application itself. Also single Monitor may receive output from many AnyDAC applications.

To enable monitoring for particular connection definition, set its parameter ‘MonitorBy’ to value ‘Indy’.

For more information on tracing and monitoring see **AnyDAC Architecture Guide**. For more information on Monitor see **AnyDAC Monitor Guide**.

2.5.4 dfmChanger

With this command line tool BDE specific application may be converted into AnyDAC specific application. The idea behind dfmChanger is to replace names of BDE components and properties with appropriate AnyDAC ones. The dfmChanger automatically performs that using conversion rules defined in an external rule file ‘\$(ADHome)\Tool\dfmChanger\BDE2AnyDAC.txt’. This tool is licensed from gs-soft AG.

Run dfmChanger utility without any arguments. It will output following reference text:

```
DFM-Changer 2.2
Copyright (c) 2000-2004 by gs-soft AG, Switzerland (www.gs-soft.com)
All Rights Reserved.

Use: dfmChanger {<filesToProcess>} [-s] [-i] [-a] -f <RuleFile>
```

```
-s - recurse subdirectories
-i - ignore errors
-f - path to filter file
-a - migrate all occurrences. Without, only components will be migrated

Example: dfmChanger x:\myDir\*.pas x:\myDir\*.dfm -s -a -f x:\MyRules.txt
changes all pas and dfm files in myDir incl. subdirs according x:\MyRules.txt
```

In the command line you define where the files to convert are located. The sample below illustrates the migration of all PAS and DFM files in the directory "x:\myDir"

```
$ (ADHome)\Tool\dfmChanger\DFMChanger.exe x:\myDir\*.pas x:\myDir\*.dfm
```

You are able to set different parameter.

```
-s - recurse subdirectories
-i - ignore errors
-f - path to filter file
-a - migrate all occurrences. Without, only components will be migrated

dfmChanger x:\myDir\*.pas x:\myDir\*.dfm -s -a -f x:\MyRules.txt
```

You also have define the location of the rule file.

```
dfmChanger x:\myDir\*.pas x:\myDir\*.dfm -s -a -f x:\MyRules.txt
```

Rule file consist from few lines, each of them looks like:

```
[old name] -> [new name] -> [add this unit to interface USES clause]
```

An example of a translation rule in the rule file is show bellows.

```
[TStoredProc] -> [TADStoredProc] -> [daADCompClient]
```

This will replace TStoredProc with TADStoredProc and add daADCompClient unit to USES clause.

2.5.5 Pump

With this command line tool you can easily import text files into database. The current utility implementation is restricted to input text format, where:

- Values are separated by commas and delimited by double quotas.
- Dates are formatted like a yyyy-mm-dd.
- Times are formatted like a hh:mm:ss.
- Numbers are formatted like a 999999.99.
- First line contains field names.

Run ADPump utility without any arguments. It will output following reference text:

```
ADPump (Ascii data pump) v 1.0.1
Copyright (c) 1999-2005 by Dmitry Arefiev
All Rights Reserved.

Use: ADPump -d <name> [-n <file name>] [-u <user>] [-w <pwd>]
      [-p <path>] {<text data file>}
-d      - connection definition name
-n      - connection definitions file name
-u      - user name
-w      - password
-p      - path to text data files
```

```
-? or -h - show this text
Comment: The text data file name must have .CSV or .TXT extension.
```

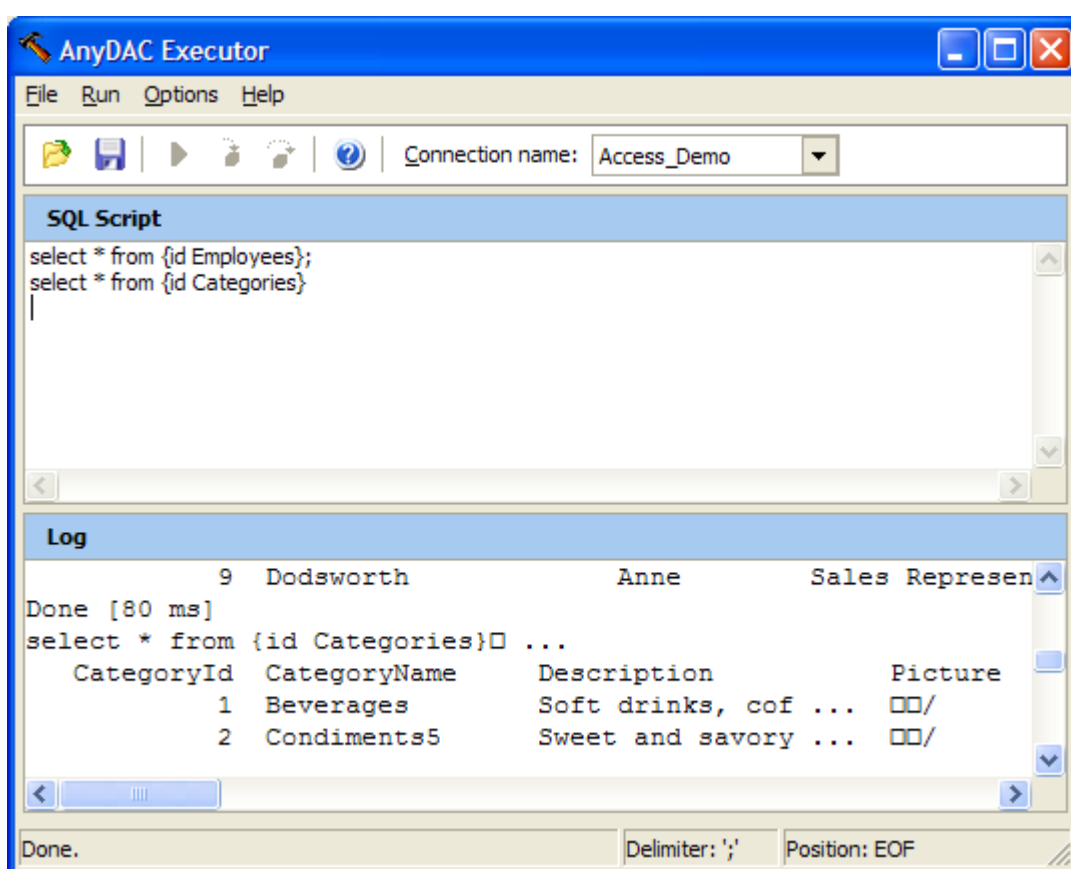
Following is an example of the utility call:

```
ADPump.exe -n $(ADHome)\DB\ADConnectionDefs.ini
-d Oracle_Demo -u ADDemo -w a -p $(ADHome)\DB\Data Territories Suppliers Shippers
```

Here 'Territories', 'Suppliers' and 'Shippers' are the name of the csv files lying in the directory '\$(ADHome)\DB\Data'.

2.5.6 Executor

AnyDAC Executor is the major SQL scripts execution tool.



AnyDAC Executor has dual user interface. One is console UI, it is used when you run Executor with command line parameters specified. Another is graphical UI, as it is on the screen shot. The utility will execute script command by command and will produce output of each command execution.

Run Executor utility with -? argument. It will output following reference text:

```
AnyDAC Executor v 1.0.1 (Build 78)
Copyright (c) 1999-2005, Dmitry Arefiev (http://www.da-soft.com)
All Rights Reserved.

Use: ADExecutor -d <name> [-n <file name>] [-u <user>] [-w <pwd>]
      [-e] [-i] [-s] [-p <path>] {<scripts>}

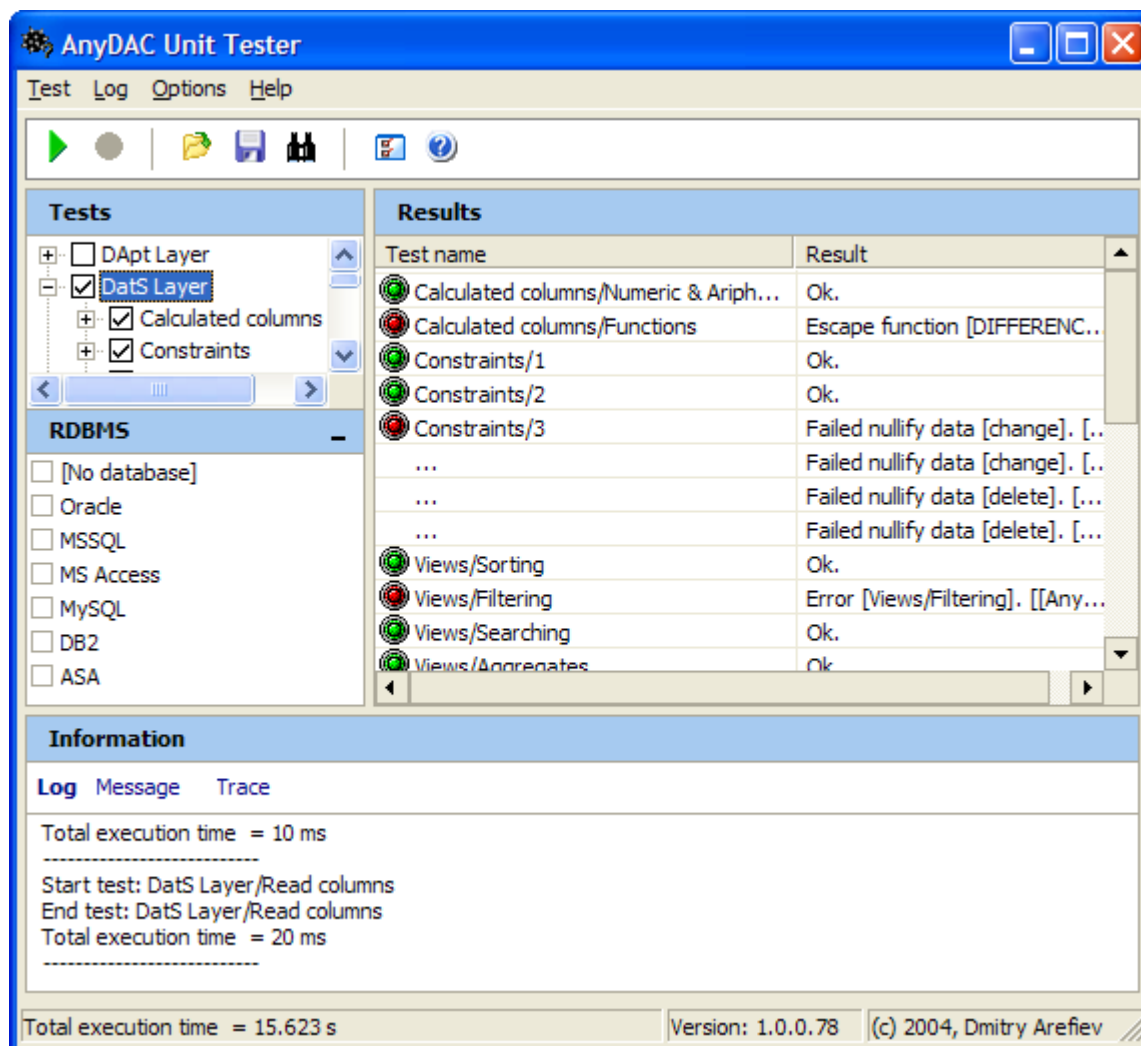
-d      - connection definition name
-n      - connection definitions file name
-u      - user name
-w      - password
-l      - login prompt
```

```
-p      - path to SQL script files
-e      - ignore drop non-existing object error
-i      - continue SQL script execution after error
-s      - do not show messages during SQL script execution
-? or -h - show this text
```

Example: ADExecutor -d Oracle_Demo -i -p x:\MyScripts s1.sql s2.sql
 executes s1.sql and s2.sql scripts in directory x:\MyScripts, using
 connection definition Oracle_Demo, dont stop on error

2.5.7 Unit Tester

AnyDAC Unit Tester is the major AnyDAC QA tool. It is pure GUI tool, as you see on the screen shot:

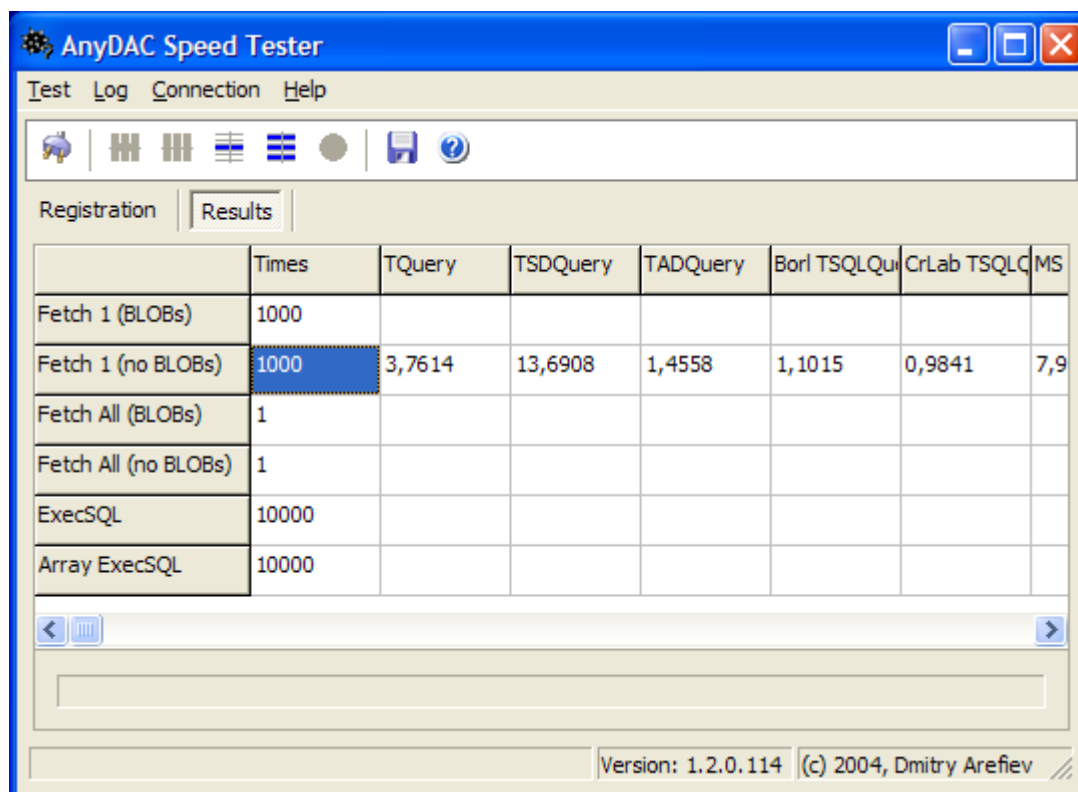


Note, not all tests will be passed with ,Ok', due to:

- RDBMS limitations. And that is most often reason.
- AnyDAC limitations.
- AnyDAC still not fixed issues.

2.5.8 Speed Tester

AnyDAC Unit Tester is the major AnyDAC benchmark tool. It is pure GUI tool, as you see on the screen shot:



The rows are tests and columns are DAC products. The “Times” column allows specifying how much time to run test. After run of selected tests, the cells will show run time in seconds of DAC product on test, if test is finished successful. If test failed, then the cell will show error text.

Current ADSPeed version supports following DBMS's and DAC products for each DBMS:

- Oracle

DAC Name	Access path
AnyDAC v 1.2.0	AnyDAC dbExpress driver -> Borland dbExpress Oracle driver -> OCI8
	AnyDAC direct Oracle driver -> OCI8
BDE v 5.2	SQLLink for Oracle -> OCI8
DbGo (Delphi7)	Oracle OLEDB Oracle provider -> OCI8
	MS OLEDB Oracle provider -> OCI7
DOA v 4.0.0	OCI8
NCOCI8 v 1.0.3	OCI8
ODAC v 3.60	OCI8
ODBC Express v 5.07	Oracle ODBC Oracle driver -> OCI8
	MS ODBC Oracle driver -> OCI8
SQL Direct v 3.0.0	SQL Direct direct Oracle driver -> OCI8
SQL Express (Delphi7)	Borland dbExpress Oracle driver -> OCI8
	CoreLab dbExpress Oracle driver -> OCI8
	CoreLab NET dbExpress Oracle driver
	CDS -> CoreLab dbExpress Oracle driver -> OCI8

- MSSQL

DAC Name	Access path
AnyDAC v1.2.0	AnyDAC direct MSSQL driver -> MS ODBC MSSQL driver
BDE v 5.2	SQLLink for MSSQL -> DBLIB
DbGo (Delphi7)	ADO -> MS OLEDB MSSQL provider
ODBC Express v 5.07	MS ODBC MSSQL driver
SDAC v 3.00	MS OLEDB MSSQL provider
SQL Direct v 3.0.0	SQL Direct direct MSSQL driver -> DBLIB
SQL Express (Delphi7)	Borland dbExpress MSSQL driver -> MS OLEDB MSSQL provider
	CoreLab dbExpress MSSQL driver -> MS OLEDB MSSQL provider

ADSpeed may be compiled only for one target DBMS. To choose DBMS, open ADSpeed.inc file and define one name:

Name	Target DBMS
AnyDACSPEED_ORA	Oracle
AnyDACSPEED_MSSQL	MSSQL

To compile ADSpeed you should have specified DAC products installed. If you does not have some DAC product installed, then exclude this DAC from ADSpeed on your own.

AnyDAC Speed Tester database object will be created as part of Demo DB installation. To fill DB tables, you should run ADSpeed and choose Connection -> Generate Data. The default number of records to generate will be sufficient for most of installations. ADSpeed will take connection definitions from default AnyDAC connection definition file.

Also see AnyDAC License.txt file for additional conditions of AnyDAC Speed Tester application usage.

3 Demos

3.1 Overview

AnyDAC uses Northwind DB as demo DB. Also few additional tables were created for AnyDAC QA. AnyDAC has scripts for all supported RDBMS's to create Northwind DB and QA objects. DEMO software consists of:

What	Where
Demo applications	\$(ADHOME)\Demo
Demo connection definition file	\$(ADHOME)\DB\ADConnectionDefs.ini
SQL scripts to create demo DB objects	\$(ADHOME)\DB\SQL
ASCII CSV files with demo data	\$(ADHOME)\DB\Data
Command files to automate creation of DB objects and data loading	\$(ADHOME)\DB\Install

3.2 Demo DB

This chapter describe steps performed by hands to prepare demos to run. If at installation you asked installer to setup connection definitions and build demo DB's, then you can miss this chapter.

3.2.1 Preparing to Demo DB installation

3.2.1.1 MS Access

In the case of MS Access we supply existing database file ADDemo.mdb that is ready to use. It was made due to Access VB script limitations (mainly due to impossibility of correct working with SQLNumeric data type). So, you don't need to create Access Demo DB.

3.2.1.2 MSSQL

In the case of MSSQL, you may have Northwind DB already installed on your server. So, you may use it, and only AnyDAC QA objects need to be installed into existent Northwind DB. If you have not Northwind DB on your server, you need to create new database on your server. For example:

```
isql.exe -U sa -S DA-NB\DAMS
1> create database addemo
2> go
```

Then you need to adjust **mssqlXXXX.bat** file. The file \$(ADHOME)\DB\Install\ **mssqlXXXX.bat** creates AnyDAC DB objects for MSSQL:

```
@_createDB.bat MSSQL MSSQLXXXX_Demo False
```

The last parameter (by default False) controls, will be Northwind DB tables created (True) or not (False).

3.2.1.3 MySQL

In the case of MySQL you have to create new DB. For example:

```
mysql.exe -u root
> create database addemo;
```

3.2.1.4 IBM DB2

In the case of DB IBM DB2 you have to create new DB with graphic types support. It is because Northwind DB tables include columns of graphic and vargraphic types.

3.2.1.5 Oracle

In the case of Oracle you have to create a new user. For example:

```
> connect system/manager
> create user addemo identified by a default tablespace user temporary tablespace temp quota
unlimited on user;
> grant connect, resource to addemo;
```

3.2.2 Setting up connection definitions

The new empty demo database or user you had to create at the Preparation section. Demo applications use specified "connection definition file" to connect to your demo DB's. The connection definitions file has 6 demo definitions; one for each AnyDAC supported RDBMS type:

Connection definition name	Main parameters	Description
MySQL_Demo	DriverID=MySQL Host=app Port=3306 Database=ADDemo User_Name=root Password=	AnyDAC MySQL driver ID. MySQL server host. Should be not empty for nonlocal server. MySQL server port. Should be specified if not default. MySQL database name. MySQL user name. User password.
Oracle_Demo	DriverID=Ora DataBase= LSNR_name User_Name=ADDemo Password=d	AnyDAC Oracle driver ID. TNS names service name. Oracle user name. User password.
Access_Demo	DriverID=MSAcc Database=\$(ADHOME)\DB\Data\ADDemo.mdb	AnyDAC MSAccess driver ID. MSAccess database file path.
MSSQL2000_Demo	DriverID=MSSQL2000 Server=Server_name Database=Northwind; User_Name=sa Password= MetaDefSchema=dbo MetaDefCatalog=Northwind	AnyDAC MSSQL 2000 driver ID. MSSQL server name. MSSQL database name. MSSQL user name. User password. Default schema for metadata retrieval. Default catalog for metadata retrieval.
MSSQL2005_Demo	DriverID=MSSQL2005 Server=Server_name Database=Northwind; User_Name=sa Password= MetaDefSchema=dbo MetaDefCatalog=Northwind	AnyDAC MSSQL 2005 driver ID. MSSQL server name. MSSQL database name. MSSQL user name. User password. Default schema for metadata retrieval. Default catalog for metadata retrieval.
DB2_Demo	DriverID=DB2 Database=DEDEMO User_Name=db2admin Password=d MetaDefSchema=db2admin	AnyDAC DB2 driver ID. IBM DB2 database alias. DB2 user name. User password. Default schema for metadata retrieval.
ASA_Demo	DriverID=ASA Database=addemo User_Name=dba Password=sql EngineName=Server_name MetaDefSchema=dba	AnyDAC ASA driver ID. ASA database name. ASA user name. User password. ASA server name. Default schema for metadata retrieval.

You must set parameter values appropriate for your environment (e.g. Server, User_Name, Password etc.) either by hands or using AnyDAC Explorer. After setting all needed parameters test your demo connection definition with ADExplorer tool.

3.2.3 Installing Demo DB objects

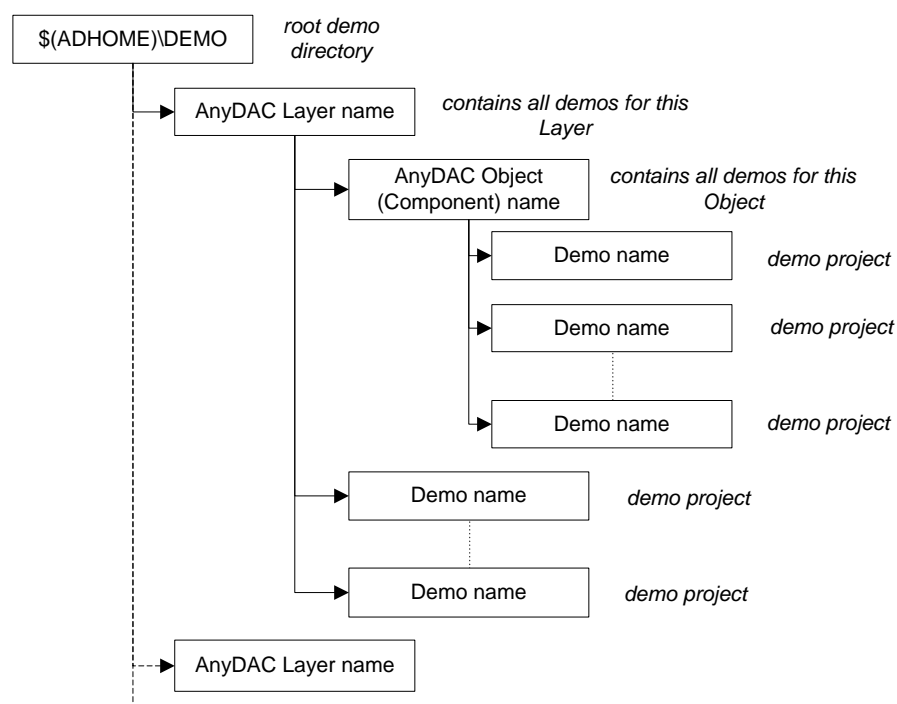
Before this section you had to create new DB or user and test your demo connection definition. To install Northwind DB and AnyDAC QA objects, run one of files in \$(ADHOME)\DB\Install directory:

For RDBMS	File name
MS Access	access.bat
MSSQL 2000	mssql2000.bat
MSSQL 2005	mssql2005.bat
MySQL	mysql.bat
Oracle	oracle.bat
IBM DB2	db2.bat
ASA	asa.bat

After command file finishes, Northwind DB and AnyDAC QA objects will be created. After installation your demo database must contains the next tables:

3.3 Running demo applications

All AnyDAC demos are placed in the \$(ADHOME)\DEMO catalog with following structure:



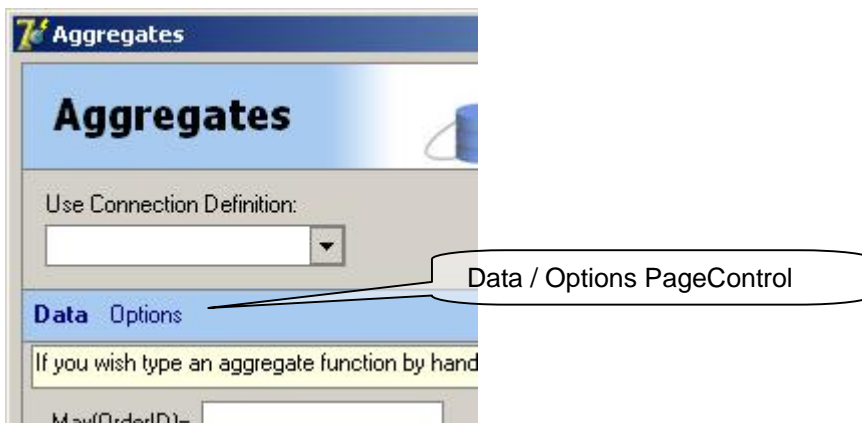
For more details about running AnyDAC demo applications have a look to document AnyDAC_HowToRunDemoApplications.doc.

3.4 Cross RDBMS demos

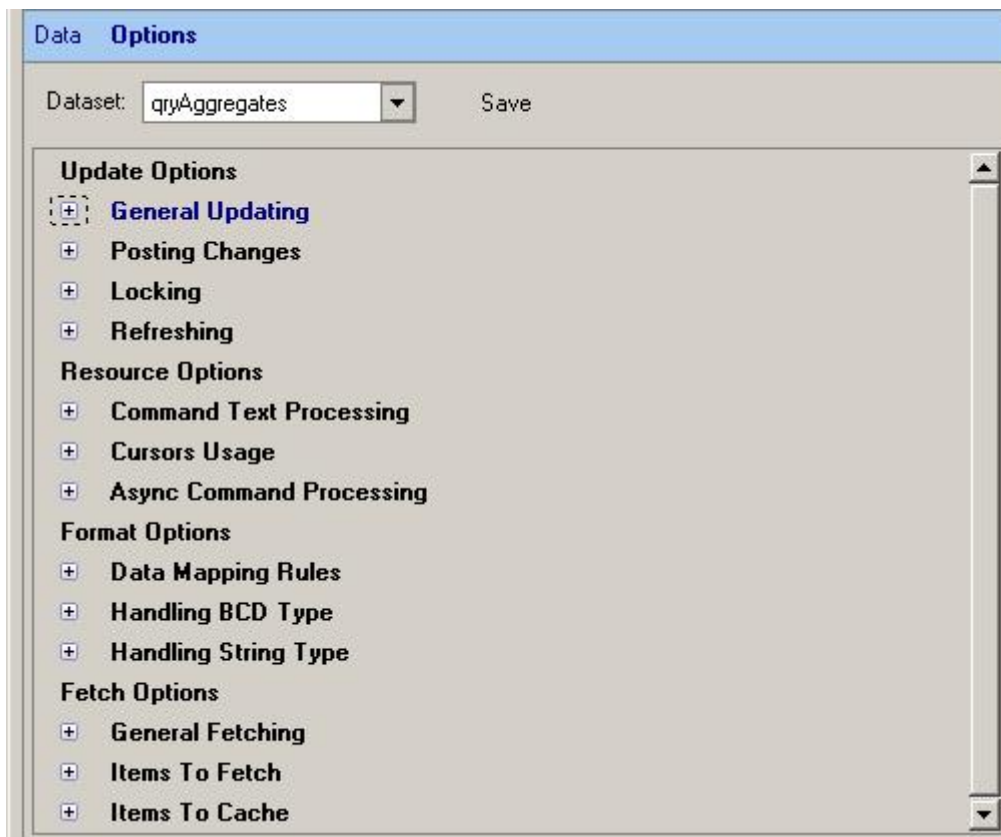
All cross RDBMS demos have similar user interface. Below the application title there is a combo box with RDBMS list from connection definition file. To connect to the desired DB you have select the name of the one in the combo box:



Some demos contain the page control with two pages.



The Data page actually contains user interface of demo application. And the Options page contains settings of AnyDAC objects. In the Dataset's list you may select the component, so all changed settings on the Options page will be applied to the selected component. All settings are divided into four groups. To expand/collapse the items of the groups click on the plus/minus button. To save the settings click Save button.



Data Options

Dataset: Save

Update Options

- + **General Updating**
- + Posting Changes
- + Locking
- + Refreshing

Resource Options

- + Command Text Processing
- + Cursors Usage
- + Async Command Processing

Format Options

- + Data Mapping Rules
- + Handling BCD Type
- + Handling String Type

Fetch Options

- + General Fetching
- + Items To Fetch
- + Items To Cache

The bottom of Demo form contains Close button and optional hyperlink to Demo description.



[Get more](#) Close

4 Programming

4.1 Application used units

For the detailed explanation of issues described in this chapter, see *AnyDAC Architecture*.

The general AnyDAC application consists from many components. Each component implements some interface and requires other interfaces to be implemented. Many interfaces are defined as COM interfaces. So, for AnyDAC application to work, interface-implementing units must be linked into the application. That will register appropriate interface factories with AnyDAC internal COM server. If some required interface implementation is missed, and application will ask for this interface, then COM server will raise exception “*Object factory for class {GUID} missing*”.

Some interface implementations are mandatory and some are optional. The following is part of USES clause of *daADCompClient* unit and actually list units containing mandatory interface implementations:

```
daADStanDef, daADStanAsync, daADStanPool, daADStanExpr, // Stan layer
daADMoniIndyClient, daADMoniFlatFile, // Moni layer
daADDaptManager, // DApt layer
daADPhysManager, // Phys layer
```

If your application uses AnyDAC components (*daADCompClient* unit), then these units are already automatically referenced by any of non-visual component.

But *GUIx* layer interface implementations must be referenced explicitly by application, and differently for console and GUI application. Only *IADGUIxWaitCursor* interface implementation is mandatory. If you are building GUI application, you have to add reference to the unit *daADGUIxFormsWait*. If you are building console application- *daADGUIxConsoleWait*. Note, that all other *GUIx* interfaces are optional. If, for example, login dialog (*IADGUIxLoginDialog*) implementation is not linked into the application, then user will be not asked for credentials.

Also all AnyDAC RDBMS drivers are optional. So, if you want to link AnyDAC drivers into your application, add the units listed in chapter 1.4.

The following code shows, how to bind AnyDAC into a GUI application, which uses AnyDAC components and works with Oracle only:

```
program MyGUIApplication;

uses
  Forms,
  daADGUIxFormsWait, // links GUIx layer wait cursor GUI implementation
  daADPhysOracl, // links Oracle driver
```

If your application does not use AnyDAC components, but works with layers directly, then you should link managers by hands. The following code shows, how to bind AnyDAC into a GUI application working with MSSQL and MSAccess RDBMS's:

```
program MyGUIApplication;

uses
  Forms,
  daADStanDef, daADStanAsync, daADStanPool, daADStanExpr, // Stan layer
  daADMoniIndyClient, daADMoniFlatFile, // Moni layer
  daADDaptManager, // DApt layer
  daADGUIxFormsWait, // links GUIx layer wait cursor GUI implementation
  daADPhysManager, // links Phys layer
  daADPhysMSSQL, // links AnyDAC MSSQL driver
```



```
daADPhysMSAcc, // links AnyDAC MSAccess driver
```

The following code shows, how to bind AnyDAC into a console application working with DB2 and just fetching data from DB:

```
program MyConsoleApplication;

{$APPTYPE CONSOLE}

uses
  daADStanDef, // Stan layer
  daADGUIxConsoleWait, // links GUIx layer wait cursor console implementation
  daADPhysManager, // links Phys layer
  daADPhysDB2, // links AnyDAC DB2 driver
```

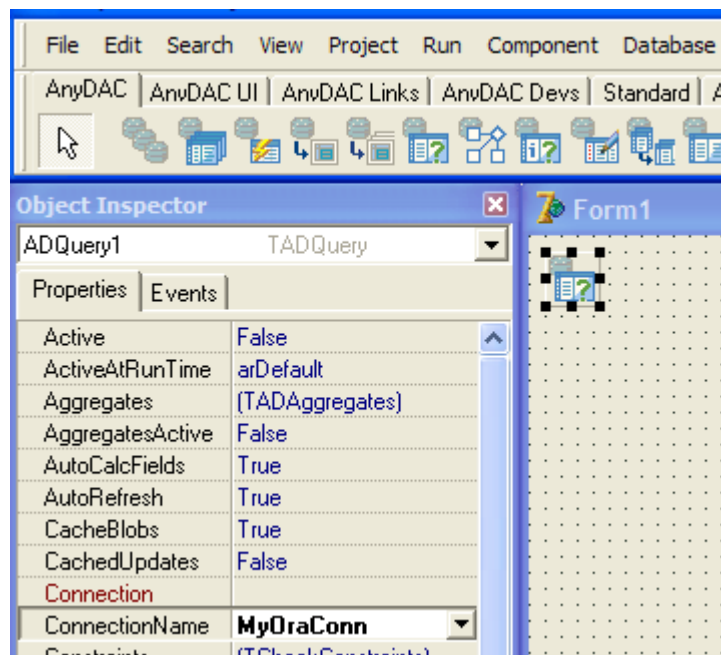
4.2 Go !

To start use your BDE DAC knowledge, if you have been working with the BDE components before. Otherwise – you teach yourself quickly!

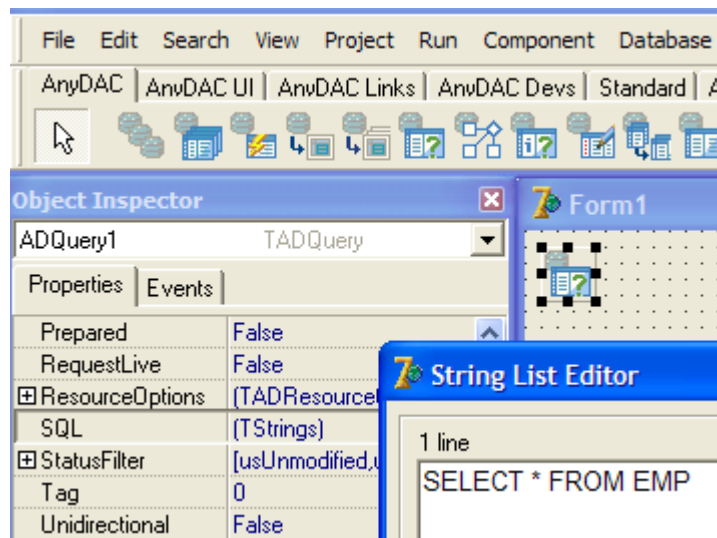
1. Run Delphi and create a new application.
2. Edit `$(ADHOME)\DBVADConnectionDefs.ini` file to add a new connection definition. The simplest AnyDAC connection definition to connect to Oracle RDBMS will look like:

```
[MyOraConn]
ServerID=Ora
DataBase=<your TNS (dtabase) name here>
```

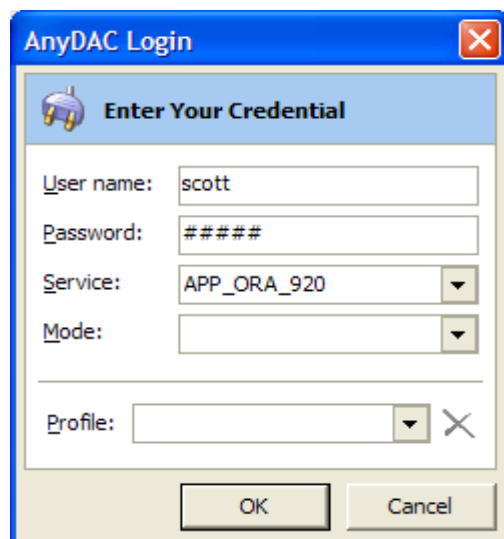
3. Drop a *TADQuery* component on to the form and set its *ConnectionName* property to *MyOraConn*:



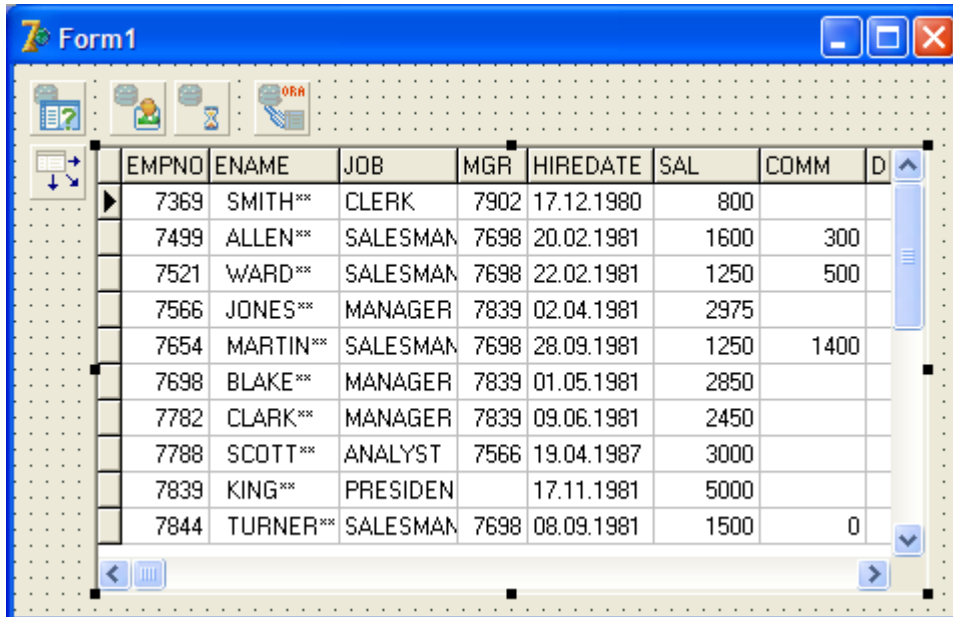
4. Set SQL property to `SELECT * FROM EMP` and press the OK button.



- Set *Active* property to true, enter your login information and press the OK button:



- Now drop a *TDataSource* component on to the form and set its *DataSet* property to *ADQuery1*.
- Drop a *TDBGrid* control on to the form and set its *DataSource* property to *DataSource1*.
- Now you see the data is fetched from Oracle database and you are also able to edit the data in runtime mode:



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	D
7369	SMITH**	CLERK	7902	17.12.1980	800		
7499	ALLEN**	SALESMAN	7698	20.02.1981	1600	300	
7521	WARD**	SALESMAN	7698	22.02.1981	1250	500	
7566	JONES**	MANAGER	7839	02.04.1981	2975		
7654	MARTIN**	SALESMAN	7698	28.09.1981	1250	1400	
7698	BLAKE**	MANAGER	7839	01.05.1981	2850		
7782	CLARK**	MANAGER	7839	09.06.1981	2450		
7788	SCOTT**	ANALYST	7566	19.04.1987	3000		
7839	KING**	PRESIDEN		17.11.1981	5000		
7844	TURNER**	SALESMAN	7698	08.09.1981	1500	0	

9. Add *daADPhysOracI* and *daADGUixFormsWait* unit references to your interface uses clause. Most simple way is to drop *TADGUixFormsLoginDialog* component from AnyDAC UI component page and *TADPhysOracIDriverLink* from AnyDAC Links page to your form.

10. Run your application.

Actually, all the described steps are exactly the same as in BDE DAC! Further you will see similarities between BDE DAC and AnyDAC. In the mean time AnyDAC is offering you many other powerful features.

5 Migrating

5.1 Overview

In general AnyDAC components have a high compatibility level with BDE DAC. This means a syntax and semantic of the properties and the methods in AnyDAC and BDE. But some parts are different:

- BDE and AnyDAC have components with different names.
E.g. BDE – TQuery, AnyDAC – TADQuery.
- BDE and AnyDAC have different alias systems. BDE stores aliases in the binary file *IDAPI.CFG*; on the other hand AnyDAC stores connection definition in the file *ADConnectionDefs.ini*. The parameters for BDE SQLLink's and AnyDAC drivers are different.
- BDE and AnyDAC may have different data type mapping for the same RDBMS. AnyDAC follows more close to Borland dbExpress data type mapping. However AnyDAC has more powerful capabilities to adjust the data type mapping.

That is nearly all of what should be changed at migration (BDE → AnyDAC).

After the migration you are able to review your application for:

- using extended AnyDAC functionality to simplify your application.
- using extended AnyDAC functionality to extend the functionality of your application.
- using AnyDAC options to fine tune your application and speed it up.

5.2 AnyDAC analogues of BDE components

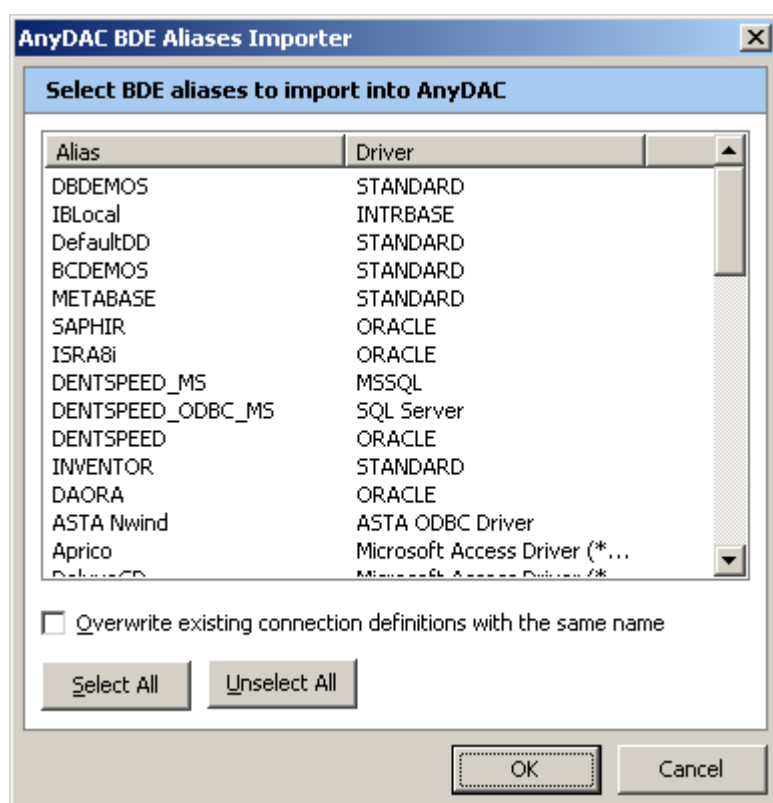
The following table shows the AnyDAC terms and its synonym BDE terms:

BDE	AnyDAC	AnyDAC unit
TSession	TADManager	daADCompClient
TDatabase	TADConnection	daADCompClient
[...] Alias [...]	[...] ConnectionDef [...]	
[...] Database [...]	[...] Connection [...]	
SessionName		
Session	ADManager	daADCompClient
PrivateDir		
TQuery	TADQuery	daADCompClient
TStoredProc	TADStoredProc	daADCompClient
TTable	TADTable	daADCompClient
TUpdateSQL	TADUpdateSQL	daADCompClient
TParam	TADParam	daADStanParam
TParams	TADParams	daADStanParam
TBlobStream	TADBlobStream	daADCompDataSet
TBatchMove	TADDataMove	daADCompDataMove
TBDEDataSet, TBDEDataSet	TADRDBMSDataSet	daADCompClient
EDBEngineError	EADDBEngineException	daADStanError
TTransIsolation	TADTxIsolation	daADStanOption
tiXXXX	xiXXXX	daADStanOption
TUpdateKind	TADPhysUpdateRequest	daADPhysIntf
ukModify	arUpdate	daADPhysIntf
ukInsert	arInsert	daADPhysIntf
ukDelete	arDelete	daADPhysIntf
TUpdateAction	TADErrorAction	daADStanIntf
uaFail	eaFail	daADStanIntf

uaAbort	eaExitFailure	daADStanIntf
uaSkip	eaSkip	daADStanIntf
uaRetry	eaRetry	daADStanIntf
uaApplied	eaApplied	daADStanIntf
TBatchMode	TADDataMoveMode	daADCompDataMove
batAppend	dmAppend	daADCompDataMove
batUpdate	dmUpdate	daADCompDataMove
batAppendUpdate	dmAppendUpdate	daADCompDataMove
batDelete	dmDelete	daADCompDataMove
batCopy	dmAlwaysInsert	daADCompDataMove

5.3 BDE aliases migration

AnyDAC Explorer and *AnyDAC Administrator* have both the BDE aliases migration function. Run one of the application and choose “BDE” → “Import Aliases”:



This dialog shows all your BDE aliases. Select the ones you like to migrate to AnyDAC. Click “Overwrite existing connection definitions ...”. The existing AnyDAC alias definitions with the same name as BDE aliases will be overwritten. In case you do not like to overwrite existing aliases the duplicates become a name like “DBDEMOS_1”. After pressing OK, AnyDAC imports all BDE RDBMS’s aliases supported by AnyDAC.

AnyDAC imports for each old BDE alias a new connection definition in AnyDAC. They will be placed into the connection definitions file.

5.4 Example

This example shows how the Borland demo application *MastApp* is transferred to AnyDAC. The following steps are required:

1. Create new directory (for example, *ADMastApp*). Then copy *MastApp* source files from Delphi Demos directory to new one created.
2. Open a cmd window and change to the *ADMastApp* directory. Run the AnyDAC transformation tool **dfmChanger** in order to replace the BDE terms with its AnyDAC synonyms.

```
<ADHome>\TOOL\dfmChanger\dfmChanger.exe *.pas *.dfm -a -f <ADHome>\TOOL\dfmChanger\gsSoftBDE2AnyDAC.txt
```

3. Now you should search in PAS and DFM for all marked for removal properties and methods. Simplest way to do so is to search for string 'removed>'. In *MastApp* the single place is *DataMod.dfm*:

```
object Connection: TADConnection
.....
  <SessionName removed> = 'Default'
.....
end
```

Just remove this line using any text editor.

4. Create AnyDAC connection definition with AnyDAC Explorer. If the application *MastApp* already contains a BDE alias check for an equal AnyDAC alias. If it does not exist migrate the BDE alias to a AnyDAC alias as described above. If no BDE alias exists create a new one with any name. Then setup connection definition parameters. This is the simplest created connection definition for *MastApp*:

```
[MASTSQL]
DriverID=Ora
DataBase=APPOR920_LSNR
User name=MastApp_DB
```

NOTE: This implies, you have created Oracle user *MastApp_DB* and loaded DBDEMOS database (Paradox or Interbase) into Oracle *MastApp_DB* user schema. For example, you can use BDE's DataPump. Although you will need to adjust identifier registers – they must be UPPER CASE in Oracle.

5. Add RDBMS specific AnyDAC driver to the application. Add one of *daADPhys[RDBMS driver ID]* units to your project. E.g. if you use oracle add *daADPhysOracl* to the project. Then add *daADGUIxFormsWait* to your project:

```
program Mastapp;

uses
  Forms,
  daADPhysOracl,
  daADGUIxFormsWait,
  MAIN in 'MAIN.PAS' {MainForm},
  .....
```

6. Now open *DataMod.dfm* file with a text editor. Setup *TADConnection* component in data module. Actually all properties are well, excluding *Params* – there is user name and password for Interbase version of application. So, we will replace with following:

```
Params.Strings = (
  'USER NAME=MastApp_DB'
  'PASSWORD=m' )
```

7. Interbase SQLLink and AnyDAC Oracle driver have different data type mappings. So, we should adjust that. To setup data mapping we should fill up collection property *TADConnection.DataFormat.MapRules*. Lets do it in *DataMod.DFM*:

```
object Database: TADConnection
.....
  FormatOptions.OwnMapRules = True
  FormatOptions.MapRules = <
    item
      PrecMax = 10
      PrecMin = 0
      ScaleMax = 0
      ScaleMin = 0
      SourceDataType = dtFmtBCD
      TargetDataType = dtInt32
    end
    item
      SourceDataType = dtFmtBCD
      TargetDataType = dtDouble
    end
    item
      SourceDataType = dtDateTimeStamp
      TargetDataType = dtDateTime
    end>
  .....
end
```

8. AnyDAC does not support desktop DB's, like Paradox or Dbase. So, we have removed all desktop DB (Paradox, Dbase) code from application:
- method *TMastData.UseLocalData* in *DataMod.pas*.
 - method *TMainForm.ViewLocalClick* in *Main.pas*
 - TField.Origin* property in *DataMod.dfm* has form: "TABLE.DB".FieldName. We will change that always to "TABLE".FieldName.
9. Now we should adjust application source code to Oracle:
- In method *TMainForm.ViewRemoteClick* in *Main.pas*, replace string ' (Local Interbase)' with ' (Oracle)'
 - Remove handler *TMainForm.ViewMenuClick* in *Main.pas*
 - Remove method *TMastData.DataDirectory* in *DataMod.pas*
 - Application creates DB connection definition on fly, if it does not exist. Change *TMastData.UseRemoteData* in *DataMod.pas* to:

```
procedure TMastData.UseRemoteData;
var
  Params: TStringList;
begin
  { See if the ConnectionDef exists. if not then add it. }
  if not ADManager.IsConnectionDef('MASTSQL') then
    begin
      Params := TStringList.create;
      try
        Params.Values['DATABASE'] := 'APPOR920_LSNR';
        Params.Values['USER NAME'] := 'MASTAPP_DB';
```

```
ADManager.AddConnectionDef('MASTSQL', 'ORA', Params);
finally
    Params.Free;
end;
end;
SetDatabaseConnectionDef('MASTSQL');
end;
```

- e. We have to adjust SQL commands used by applications. In *DataMod* query *CustByLastInvQuery* has DESCENDING key word, but Oracle uses DESC.

5.5 Additional migration hints

1. Some properties should be removed at all, because AnyDAC does not have analogues:
 - o SessionName should be removed completely.
 - o PrivateDir should be removed completely.
2. Although AnyDAC has analogue of BDE's TTable named TADTable, it is not fully compatible. And TADTable is here rather as temporary migration solution. So, it is good idea to replace all TTable's with TADQuery's.
3. Constructions, like a:

```
Screen.Cursor := crSQLWait;
try
    .....
finally
    Screen.Cursor := crDefault;
end;
```

Should be replaced with:

```
uses
    daADStanFactory, daADGUIxIntf;
.....
var
    oWait: IADGUIxWaitCursor;
.....
ADCreateInterface(IADGUIxWaitCursor, oWait);
oWait.StartWait;
try
    .....
finally
    oWait.StopWait;
end;
```

4. AnyDAC does not have a BDE API analogues. So, everything using directly BDE API should be recoded.
5. Many 3d party products, like a reporting or m-tier libraries, requires DAC adapter unit. In most cases it exists for BDE and does not exists for AnyDAC. So, either develop by your self (taking BDE adapter as template), either contact us.
6. EDBEngineError is BDE specific exception class. AnyDAC has analogue – EADDBEngineException class. When handling BDE exceptions, programmer uses ErrorCode property to get error kind. AnyDAC has property Kind, which returns enumerated value.

For example, change following code:


```
if E is EDBEngineError then
begin
  case EDBEngineError(E).Errors[0].ErrorCode of
    DBIERR_KEYVIOL: MetaBaseDBError(SMb_DataSetInvalidPKeyValue, E);
  end;
```

into:

```
if E is EADDBEngineException then
begin
  case EADDBEngineException(E).Kind of
    ekUKViolated: MetaBaseDBError(SMb_DataSetInvalidPKeyValue, E);
  end;
```

7. TDataMove and TADDataMove are very different in many ways. So, you will need seriously rework any advanced code using TDataMove.
8. TADConnection.OnLogin has incompatible with TDatabase.OnLogin parameters list. So, for example, you will need to replace following handler:

```
procedure TMyDataModule.dbLogin(Connection: TDEConnection; LoginParams:
TStrings);
begin
  LoginParams.Values['USER NAME'] := 'me';
  LoginParams.Values['PASSWORD'] := 'pwd';
end;
```

with this one:

```
procedure TMyDataModule.dbLogin (AConnection: TADCustomConnection;
  const AConnectionDef: IADStanConnectionDef);
begin
  AConnectionDef.UserName := 'me';
  AConnectionDef.Password := 'pwd';
end;
```

6 Where to go further

After you have worked through some demo applications and got a feeling you need to know more, following sections will get few hints about, what you can find useful in AnyDAC. After that will be a time to read ***AnyDAC Architecture guide***.

6.1 xRDBMS features

In general this features we can split into 3 main areas:

- Flexible data type mapping system.
- AnyDAC command text macro processor.
- Consistent behaviour across all supported RDBMS's.

6.1.1 Data type mapping

It allows setup any suitable for your task mapping of RDBMS data types into AnyDAC type system. As result, you can setup unified type system across all RDBMS's supported by your application.

Many AnyDAC objects have **FormatOptions** property. And it has property **MapRules**, which contains user defined data type mapping rules. For example, to map **DECIMAL(<= 9, 0)** type into **dtInt32** specify following:

```
with FormatOptions do begin
  OwnMapRules := True;
  with MapRules.Add do begin
    SourceDataType := dtBCD;
    PrecMax := 9;
    PrecMin := 0;
    ScaleMax := 0;
    ScaleMin := 0;
    TargetDataType := dtInt32;
  end;
end;
```

6.1.2 Macro processor and escape sequences

The AnyDAC macro processor transforms command text with macroinstructions into a form understood by the RDBMS. This means, the macroinstructions are not visible to the RDBMS. AnyDAC supports two kinds of macroinstructions:

- Substitution variables. They allow substitution to put parameters in command text. This is to extend the use of parameters. For example, to parameterise a table name in FROM clauses or column names in SELECT clauses substitution variables can be used but parameters are of no use.
- Escape sequences. They allow writing RDBMS independent SQL commands. For details see Architecture Guide "Macro processing" chapter. AnyDAC has 5 kinds of escape sequences:
 - Allowing constant substitution. For example, **{d 2004-08-30}** will be expanded into RDBMS specific date constant.
 - Allowing identifier substitution. For example, **{id My Table}** will be expanded into RDBMS specific quoted table name.
 - Conditional substitution. For example, **{if (OracI, TO_CHAR, MSSQL, CONVERT)}** will be expanded into TO_CHAR on Oracle and CONVERT on MSSQL.
 - LIKE operator escape sequence.

- Scalar functions. For details see Architecture Guide “Appendix 2 - Macro functions”.

6.2 Deeper into AnyDAC

After getting familiar with AnyDAC high-level components, you may want to use other AnyDAC layers directly. There are 3 main layers – DatS, Phys and Dapt. These layers should be used, if you need to achieve full control and full speed on your data access. The following example, shows how to establish a connection to an Oracle RDBMS and fetch the data from Scott.Emp table into DatS table object and display it.

```
procedure TForm1.FormCreate(Sender: TObject);  
var  
    oConn: IADPhysConnection;  
    oCmd: IADPhysCommand;  
    oTab: TADDatSTable;  
    I, J: Integer;  
begin  
    // 1. open AnyDAC's ADPhysManager  
    ADPhysManager.ConnectionDefs.Storage.FileName:= $(ADHOME)\DB\ADConnectionDefs.ini';  
    ADPhysManager.Open;  
  
    // 2. create connection  
    ADPhysManager.CreateConnection('MyOra', oConn);  
    oConn.Open;  
  
    // 3. create command and prepare SELECT statement  
    oConn.CreateCommand(oCmd);  
    oCmd.Prepare('select * from Emp');  
  
    // 4. fetch rows from RDBMS to DatSTable  
    oTab := TADDatSTable.Create;  
    oCmd.Fetch(oTab);  
  
    // 5. print rows  
    mmLog.Clear;  
    for I := 0 to oTab.Rows.Count - 1 do  
    begin  
        s := '';  
        for J := 0 to oTab.Columns.Count - 1 do  
            s := s + oTab.Rows[I].GetData(J) + ' ';  
        mmLog.Add(s);  
    end;  
  
    // 6. free resources  
    oTab.Free;  
end;
```